

Procédure de mise en service d'un MMDVM en partant de zéro

*Il existe peut-être une révision plus récente de ce document. Avant toute chose, pensez à [télécharger la dernière révision](#).
Tous les commentaires sont les bienvenus. Vous pouvez les déposer sur mon blog [en cliquant ici](#).*

Ce document recense l'intégralité des actions réalisées pour mettre en service mes relais MMDVM. Cependant, malgré toute l'attention et le temps passé à le rédiger, il peut ne plus être totalement applicable lorsque vous le consulterez. J'ai par exemple dû procéder à une importante mise à jour lorsque j'ai voulu mettre un nouveau MMDVM en service, en 2022. Le système d'exploitation de la carte **Raspberry Pi** ainsi que le logiciel chargé dans la carte **Nucleo STM32F446** et l'écran **Nextion** évoluent, ce qui donne lieu à des incompatibilités ou contraint à utiliser d'autres méthodes pour mettre l'ensemble en œuvre.

Ne soyez pas rebutés par la quantité de choses à effectuer. Dites-vous qu'il ne vous reste presque plus qu'à lire et appliquer ce qui est écrit. Imaginez le temps que ça vous prendrait en partant de zéro, sans la moindre information.

L'interface utilisée dans mon cas est celle publiée par F5UII / Christian : <https://www.f5uii.net/mmdvm-nucleo-stm32-pcb-shield/>
Je dispose de quelques circuits imprimés inutilisés. Si ça vous intéresse [laissez-moi un message sur mon blog](#).

On trouve quelques cartes **Nucleo STM32F446** à partir d'une trentaine d'euros sur Aliexpress, mais cela reste assez rare. Le prix le plus couramment pratiqué est plutôt autour de 50€. Quelqu'un a dû se rendre compte qu'il n'était pas nécessaire de les brader, ou la pénurie mondiale de composants est passée par là, car les deux cartes que j'ai achetées la-bas en 2019 et 2021 m'ont coûté un peu moins de 21€ chacune !

Ce document a été mis à jour en Avril 2024 suite à l'installation complète réalisée à partir des dernières versions en date de tous les logiciels mentionnés ci-après.

Présentation rapide du concept « MMDVM »

Le terme MMDVM est l'acronyme de Multi Mode Digital Voice Modem : Modulateur Démodulateur Multi-Modes Vocaux Numériques. La force de ce système est de permettre l'utilisation d'équipements radio fonctionnant initialement uniquement en FM, afin de réaliser un relais radio supportant de nombreux modes numériques : DMR, C4FM, D-STAR, NXDN, DAPNET (POCSAG) et évidemment FM.

Un MMDVM est composé des éléments suivants :

- Un équipement radio dédié à la réception des transmissions distantes, pouvant fournir à une interface externe un signal analogique non filtré véhiculant le signal démodulé reçu* ;
- Un équipement radio dédié à l'émission, à destination des équipements radio distants, capable de transmettre un signal audio analogique « large bande » via son entrée non filtrée* ;
- Une interface analogique placée entre les équipements radio et la carte processeur :
 - Amplification ou atténuation des signaux audio analogiques (celui reçu et celui émis), ainsi que leur remise en forme ;
 - Interface avec les signaux TOR (Tout Ou Rien) des équipements radio : PTT, indication de détection de porteuse, signal analogique indiquant la force du signal reçu (RSSI : Received Signal Strength Indication) ;
 - Indications visuelles par l'intermédiaire de LED et éventuellement d'un afficheur de type alphanumérique ou graphique.
 - Microcontrôleur (carte Arduino Mega) ou processeur (carte STM32) assurant la conversion analogique/numérique et numérique/analogique.
- Une carte **Raspberry Pi** assurant le traitement des signaux audio numériques, les fonctions de base d'un relais (balise en morse par exemple) et l'interface avec les serveurs Internet d'interconnexion des relais MMDVM.

** Si le relais est utilisé en mode « Hotspot » (simplex), on n'utilise qu'une seule radio en tant qu'émetteur et récepteur.*

Des équipements intégrant sur une même carte électronique la majorité de ces fonctions sont aussi disponibles. Leur faible puissance d'émission restreint cependant leur périmètre de couverture radio au sein du domicile de l'opérateur.

Il n'est pas possible d'utiliser la suite de logiciels MMDVM en tant que convertisseur DMR vers FM analogique par exemple. Lorsqu'un relais MMDVM est utilisé en mode DMR, il reçoit une émission DMR, la traite en interne, et la retransmet uniquement en DMR. La fonction « Multi Modes » est donc assurée de façon exclusive, soit un mode à la fois, sans conversion d'un mode à l'autre. Certains serveurs d'interconnexion de relais MMDVM offrent des interfaces entre modes numériques, mais le relais MMDVM lui-même ne gère qu'un des modes à la fois sur ses interfaces de réception et d'émission.

Installation du système d'exploitation de la carte Raspberry Pi

La carte Raspberry Pi nécessite un système d'exploitation pour fonctionner, comme tout ordinateur. Le choix de celui-ci est laissé libre tant qu'il est basé sur le système Linux, car les logiciels permettant la fonction MMDVM seront compilés et installés par la suite. Afin de nous simplifier la vie ici, nous allons utiliser un des variants du système d'exploitation mis à disposition par la fondation Raspberry Pi. Nous allons nous orienter vers une version sans interface graphique, puisque toutes les actions seront réalisées sous forme de lignes de commande à saisir au clavier sur une interface « terminal ». Ne soyez pas rebutés de suite par cette façon de faire, tout sera expliqué en détails dans ce document. Ce choix de système d'exploitation léger permet d'utiliser une génération de carte Raspberry Pi ancienne, et une carte mémoire SD de capacité relativement faible.

Chargement du système d'exploitation

Comme indiqué précédemment, nous utiliserons **Raspberry Pi OS**, qui est conçu par l'équipe qui a créé la carte Raspberry Pi. Son téléchargement et le stockage sur la carte mémoire SD sont simplifiés grâce à l'outil mis à disposition par cette équipe, disponible pour Windows, macOS et Linux, et accessible à l'adresse suivante : <https://www.raspberrypi.com/software/>.

Si vous avez déjà installé le système d'exploitation **Raspbian** dans le passé, vous avez pour habitude d'utiliser le compte utilisateur défini par défaut, à savoir l'utilisateur **pi**, avec le mot de passe **raspberry**.

La sécurité est maintenant accrue, afin que les cartes **Raspberry Pi** accessibles depuis Internet ne puissent plus être « piratées » facilement en utilisant ce compte défini par défaut. Avec les versions récentes du système d'exploitation dédié à la **Raspberry Pi**, il appartient à l'utilisateur de définir lui-même l'identifiant et le mot de passe. Il n'existe donc plus de compte prédéfini par défaut. Comme nous utiliserons cette carte sans y connecter d'écran, de clavier et de souris, il sera impossible de définir ce compte utilisateur si celui-ci n'a pas été choisi avant le chargement du système d'exploitation sur la carte SD, via le logiciel **Raspberry Pi Imager**. Suivez donc avec la plus grande attention les étapes décrites ci-dessous.

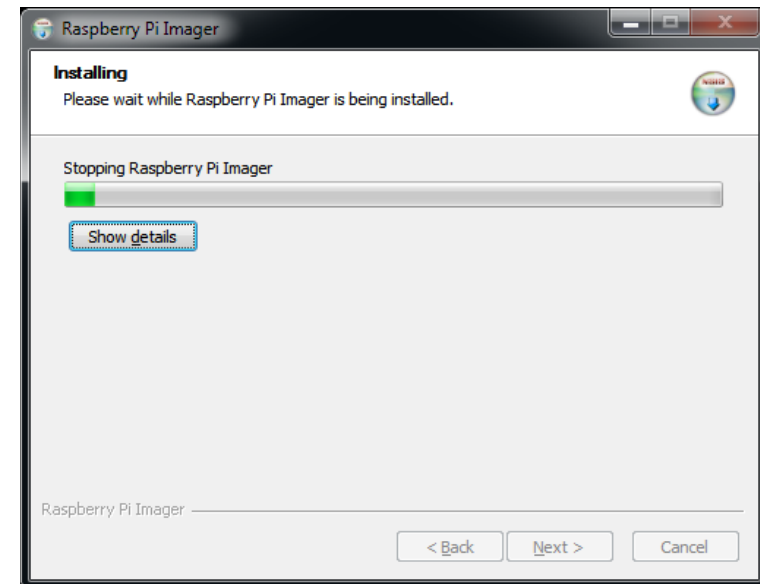
Installation de Raspberry Pi Imager sous Windows

Rien de bien compliqué ici, il suffit de confirmer l'installation et de valider toutes les étapes :

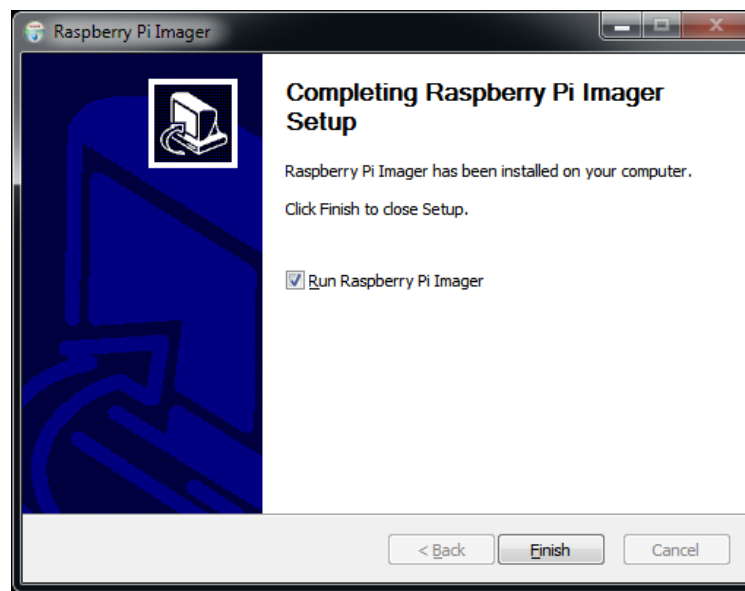
Cliquer sur **Install** :



L'installation s'exécute :



À l'issue de celle-ci, une nouvelle fenêtre s'affiche. Laisser la case **Run Raspberry Pi Imager** cochée et cliquer sur **Finish** :



Paramétrage de Raspberry Pi Imager et chargement du système d'exploitation sur la carte SD

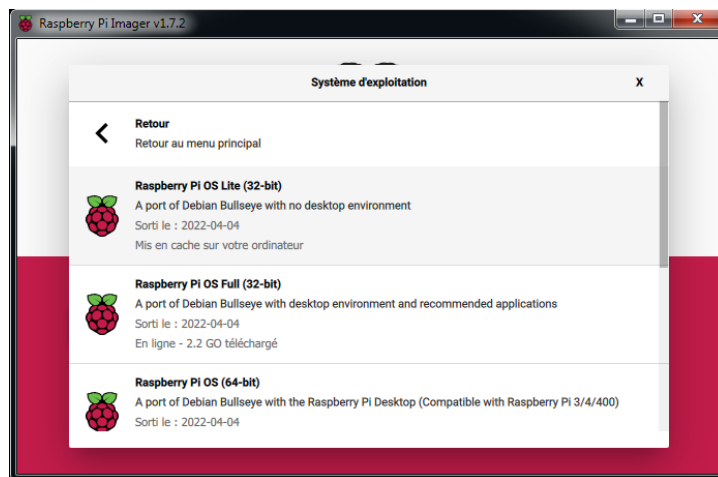
Dans ces étapes, je définis les identifiants qui étaient jusqu'ici ceux par défaut sur ce système d'exploitation, car certaines étapes décrites dans ce document sont antérieures à cette nouvelle règle de sécurité. Cela me permet d'éditer uniquement les nouvelles actions à effectuer, sans procéder à des modifications importantes de ce document qui dépasse les 100 pages. De plus, mon MMDVM est connecté à un modem 4G dont l'abonnement pour particuliers ne permet pas d'ouvrir de port pour donner un accès à la console Linux de ma carte Raspberry Pi depuis Internet. C'est une bonne sécurité, mais cela rend le MMDVM totalement incontrôlable à distance s'il est installé sur un point haut autre que le domicile du responsable.

Si vous définissez d'autres identifiants pour l'utilisateur, vous devrez saisir ceux-ci à la place de **pi** et **raspberry** pour les étapes décrites dans le reste de ce document.

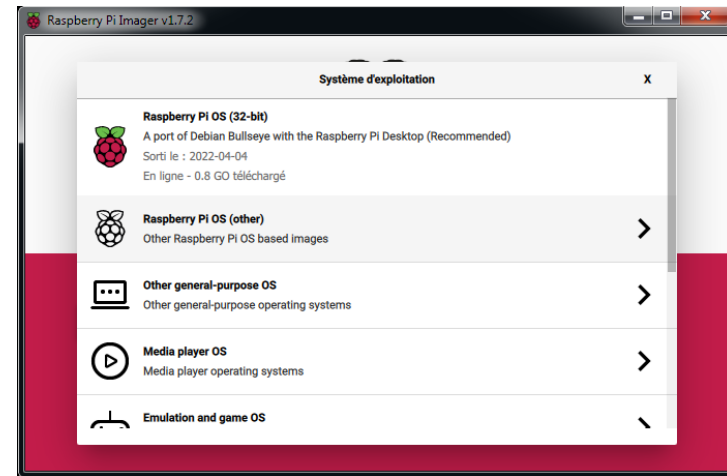
Le logiciel **Raspberry Pi Imager** est exécuté automatiquement :



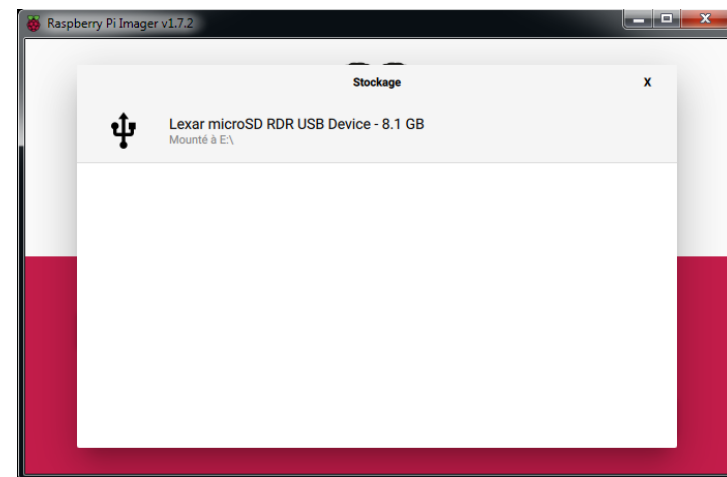
Cliquer sur **Raspberry Pi OS Lite (32-bit)** :



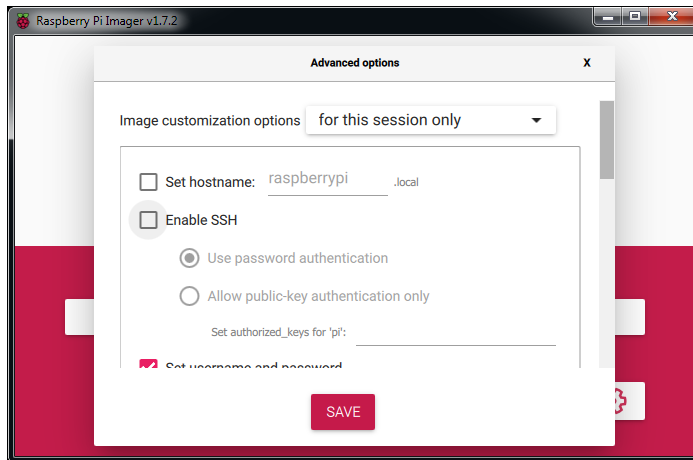
Cliquer sur **CHOISISSEZ L'OS**, puis sur **Raspberry Pi OS (other)** :



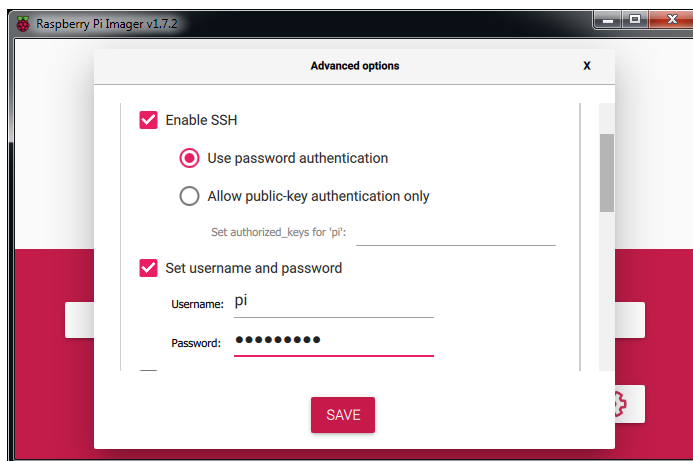
Cliquer sur le bouton **CHOISISSEZ LE...** pour définir la lettre du disque affectée à votre carte SD :



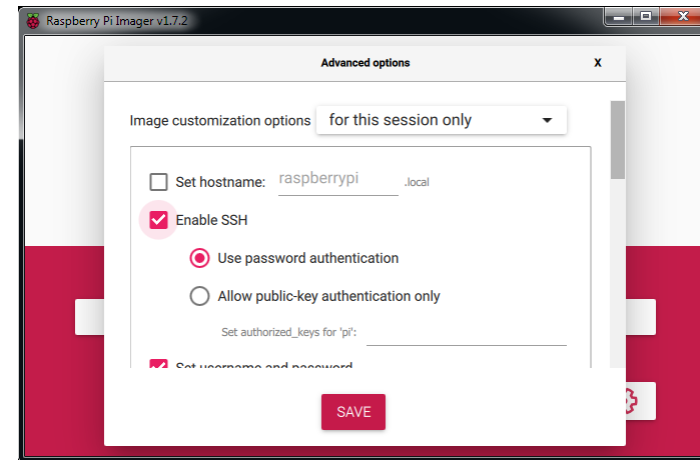
Cliquer sur le bouton de paramètres tout en bas à droite ⚙️ :



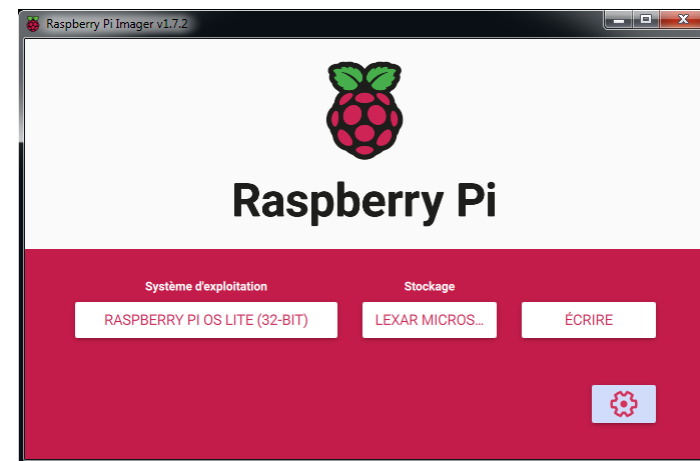
Faire descendre la barre de défilement vertical afin de définir les identifiants utilisateur (ici le mot de passe sera **raspberry**) :



Cocher la case **Enable SSH** afin d'autoriser les connexions IP à la console Linux :



Cliquer sur **SAVE** pour quitter l'édition des paramètres, puis sur **ÉCRIRE** pour lancer l'installation sur la carte SD :



Accepter la mise en garde concernant la perte des données :



Attendre la fin de l'écriture des données sur la carte SD :



Une fois le chargement terminé sur la carte SD, cliquer sur **CONTINUER** et la placer dans le support de la carte Raspberry PI

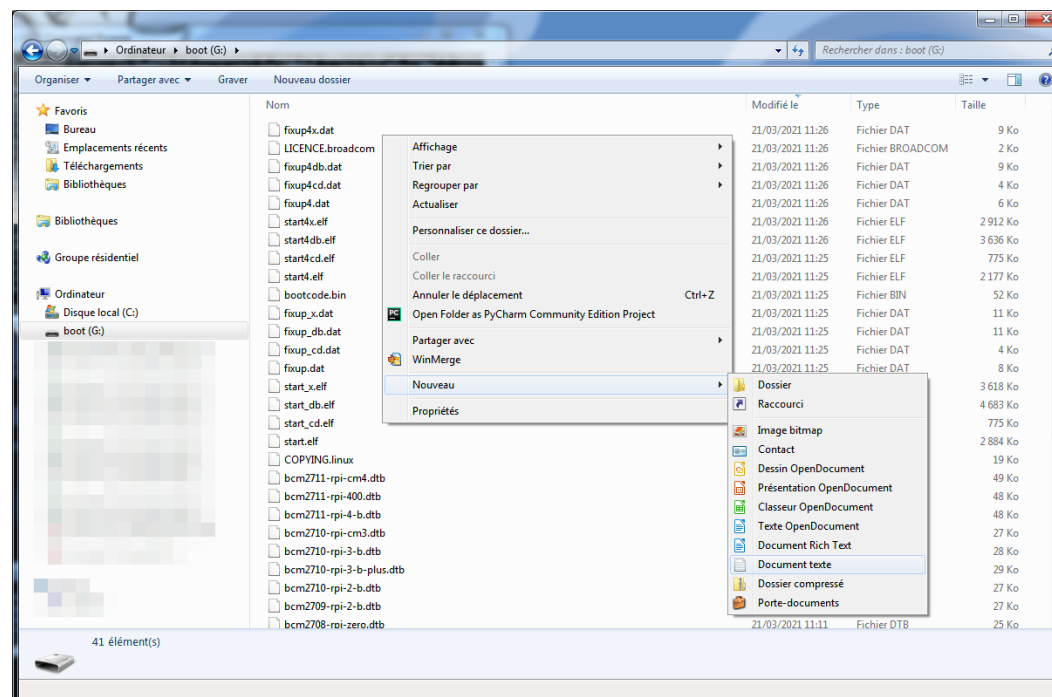


Avant de commencer

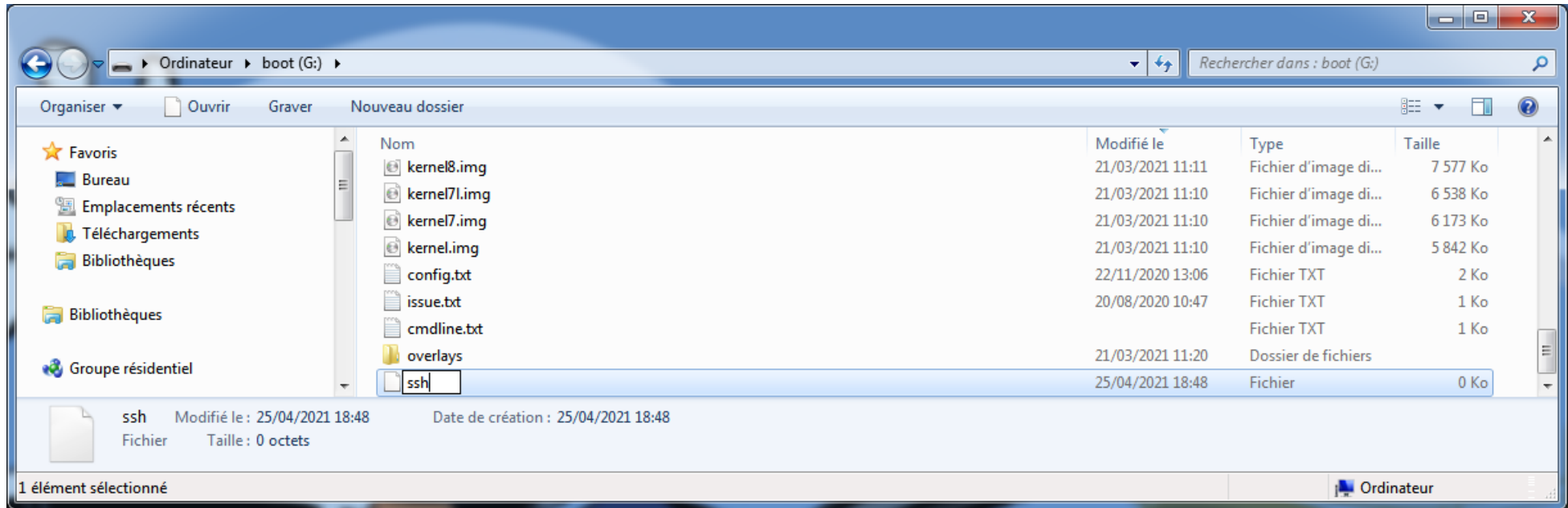
Activation de l'accès à la console par IP via le protocole SSH

La majorité des étapes décrites ci-dessous est effectuée en utilisant un client SSH. Ce logiciel permet d'exécuter des commandes sur la carte Raspberry Pi par l'intermédiaire de son lien réseau. Pour des raisons de sécurité, l'accès SSH n'est pas activé par défaut sur le système d'exploitation Raspbian. Si l'activation n'a pas été faite lors du chargement du système d'exploitation décrit dans le paragraphe précédent, il vous faudra créer un fichier vierge nommé **ssh** dans la racine de la carte SD, à partir de Windows.

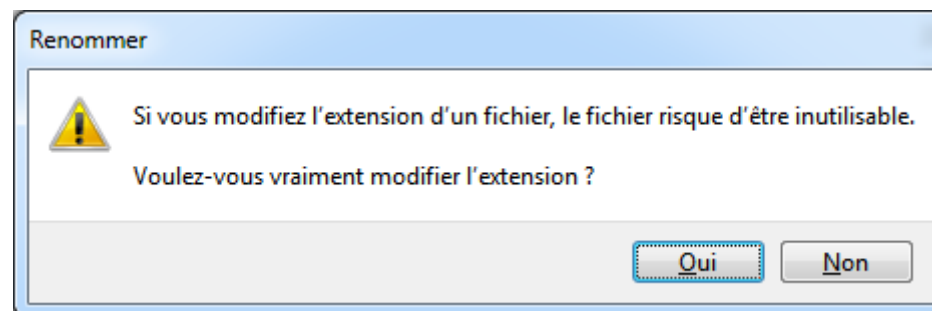
Depuis la racine de la carte SD, faire un clic droit et aller sur **Nouveau** dans le menu déroulant, puis cliquer sur **Document texte** :



Remplacer le nom du fichier par **ssh**, sans aucune extension de fichier, et valider en appuyant sur la touche **Entrée** :



Validez cette action en cliquant sur **Oui** :



Éjectez proprement le disque SD depuis la barre des tâches de Windows, insérez à nouveau la carte SD dans le lecteur de la Raspberry Pi et mettez-la sous tension.

Retrouvez l'adresse IP de la carte Raspberry Pi depuis l'interface web de votre routeur ou de votre box Internet. Cette étape est très spécifique compte tenu de la multitude d'équipements réseau pouvant attribuer une adresse IP, et ne sera donc pas détaillée dans ce document. Questionnez votre moteur de recherche favori en incluant la référence de votre équipement dans les mots clés de recherche. Il est par ailleurs très vivement conseillé de définir un bail DHCP permanent pour la carte Raspberry Pi, afin que son adresse IP ne change pas à chaque mise sous tension. Ici encore, tout dépend de votre routeur ou votre box Internet.

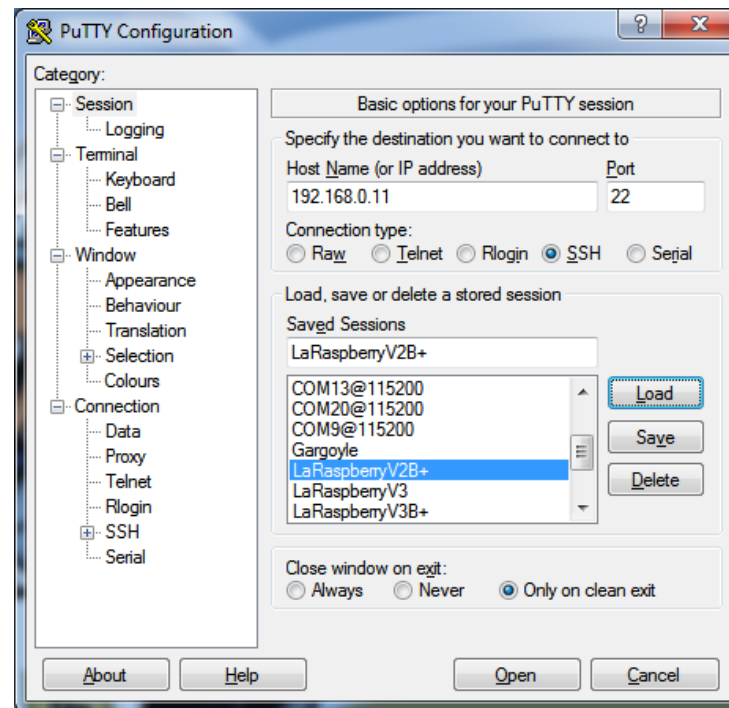
Connexion à la console

Une fois l'adresse IP connue, il faudra vous y connecter en utilisant un client SSH. J'utilise « Putty », qui est disponible gratuitement à cette adresse : <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>

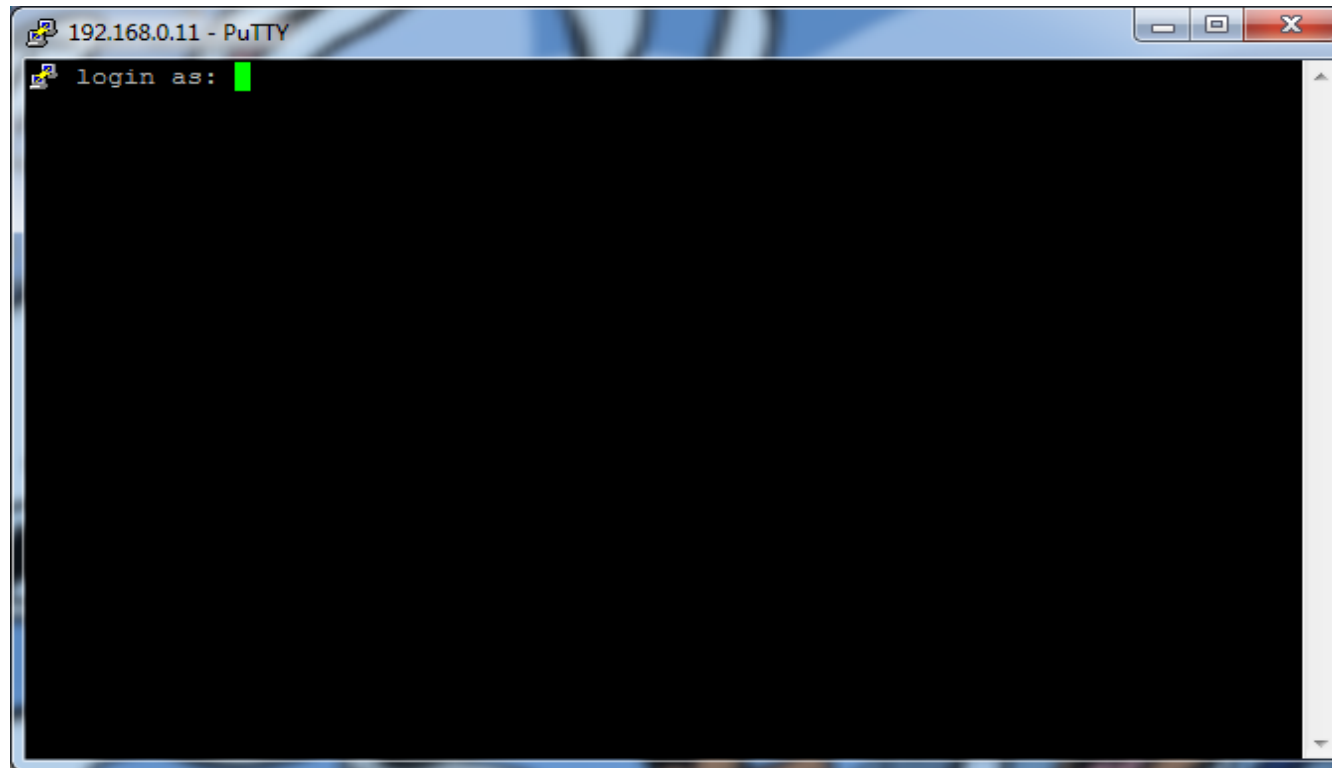
Une fois installé, utilisez l'icône disponible sur le bureau de Windows pour lancer le logiciel :



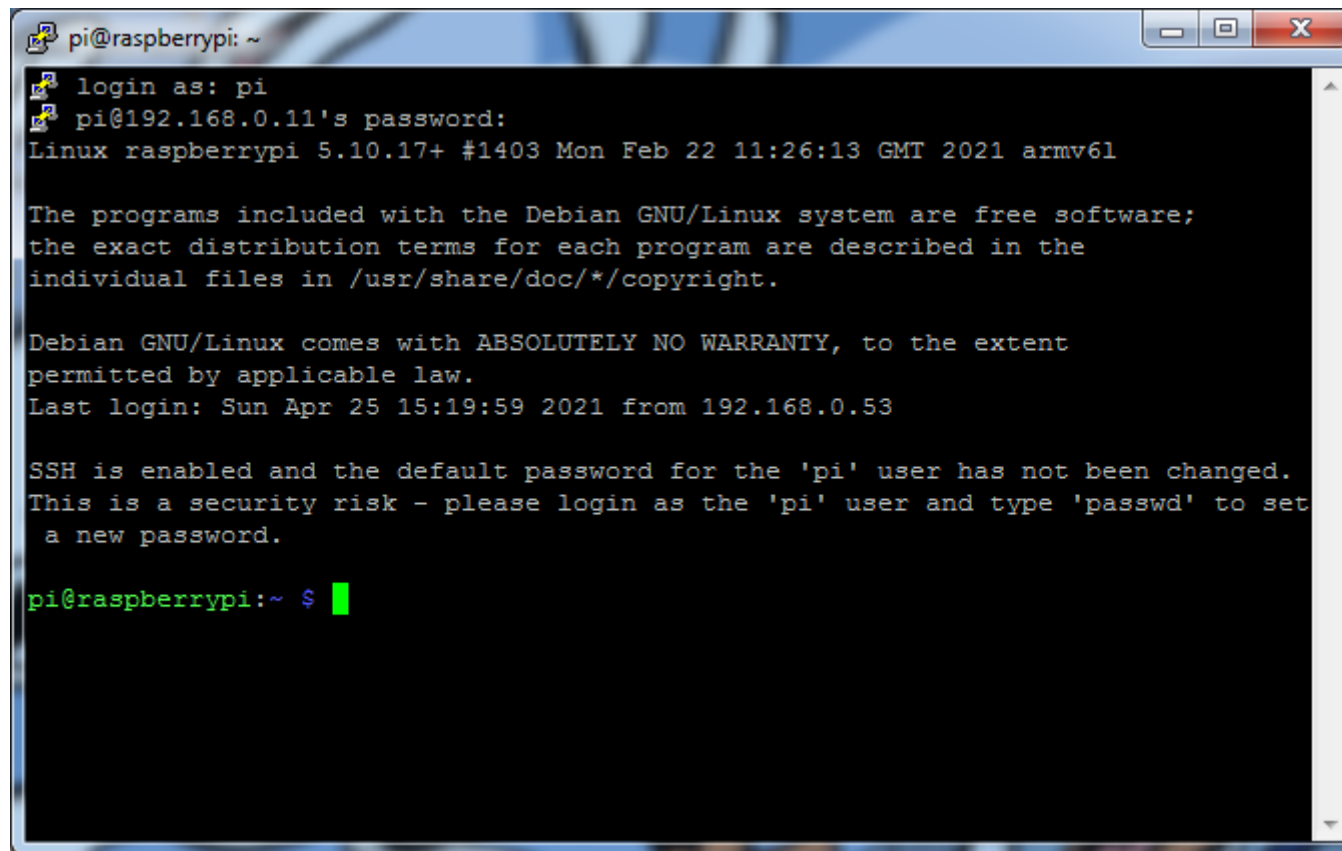
Sélectionnez le type de connexion **SSH**, saisissez l'adresse IP de la carte Raspberry Pi, laissez le port à 22, donnez un nom à cette connexion dans la case **Saved Sessions**, et cliquez sur **Save**. Vous accéderez ainsi rapidement à la Raspberry Pi lors des connexions suivantes :



On accède ensuite à la console SSH en double-cliquant sur le nom donné à la connexion :



Le système d'exploitation vous invite à vous identifier. Tapez **pi** à la suite de « login as : » et validez avec la touche **Entrée**. Lorsque la ligne « Password » s'affiche, tapez **raspberry**. Rien ne s'affiche à l'écran, mais c'est normal. Validez ensuite avec la touche **Entrée**.

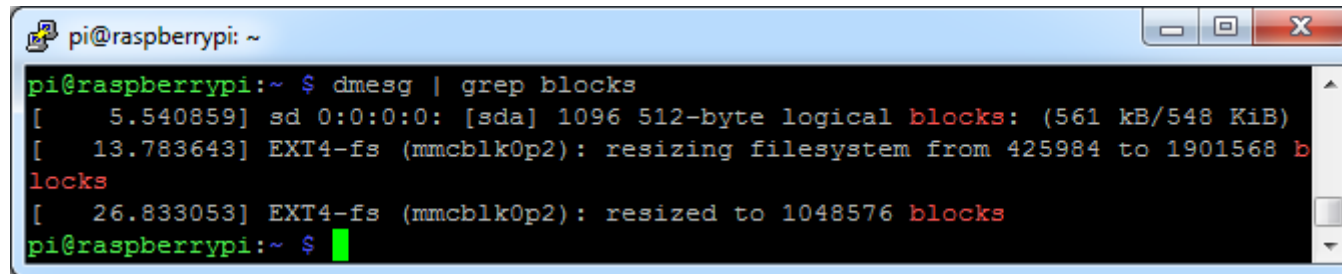


```
pi@raspberrypi: ~  
login as: pi  
pi@192.168.0.11's password:  
Linux raspberrypi 5.10.17+ #1403 Mon Feb 22 11:26:13 GMT 2021 armv6l  
  
The programs included with the Debian GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
Last login: Sun Apr 25 15:19:59 2021 from 192.168.0.53  
  
SSH is enabled and the default password for the 'pi' user has not been changed.  
This is a security risk - please login as the 'pi' user and type 'passwd' to set  
a new password.  
  
pi@raspberrypi:~ $
```

Vous voilà connecté à la console Linux. C'est à partir de ce moment que vous pouvez exécuter toutes les commandes listées en italique dans ce document.

Extension du système de fichiers

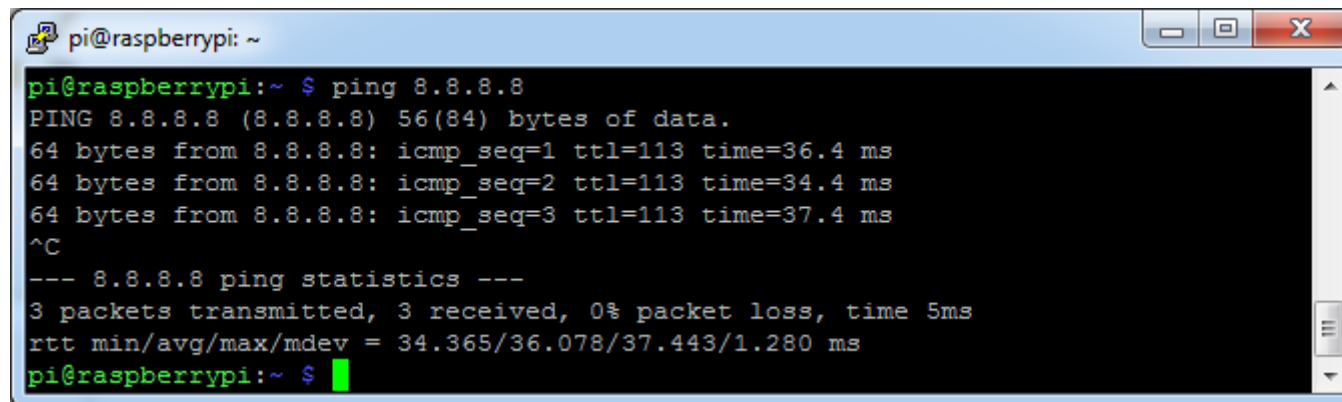
Les systèmes d'exploitation Raspberry OS récents se chargent automatiquement de redimensionner les partitions, afin d'utiliser la totalité de l'espace disponible sur la carte SD. Avant, il fallait utiliser la commande *sudo raspi-config* pour accéder à cette fonction.



```
pi@raspberrypi: ~  
pi@raspberrypi:~ $ dmesg | grep blocks  
[ 5.540859] sd 0:0:0:0: [sda] 1096 512-byte logical blocks: (561 kB/548 KiB)  
[ 13.783643] EXT4-fs (mmcblk0p2): resizing filesystem from 425984 to 1901568 b  
locks  
[ 26.833053] EXT4-fs (mmcblk0p2): resized to 1048576 blocks  
pi@raspberrypi:~ $
```

Test de connexion Internet

Comme il va falloir installer des fichiers et mises à jour assez rapidement, on s'assurera que la Raspberry Pi peut bien accéder à Internet en exécutant par exemple la commande suivante : *ping 8.8.8.8*



```
pi@raspberrypi: ~  
pi@raspberrypi:~ $ ping 8.8.8.8  
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.  
64 bytes from 8.8.8.8: icmp_seq=1 ttl=113 time=36.4 ms  
64 bytes from 8.8.8.8: icmp_seq=2 ttl=113 time=34.4 ms  
64 bytes from 8.8.8.8: icmp_seq=3 ttl=113 time=37.4 ms  
^C  
--- 8.8.8.8 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 5ms  
rtt min/avg/max/mdev = 34.365/36.078/37.443/1.280 ms  
pi@raspberrypi:~ $
```

Appuyer sur la touche **Ctrl** du clavier, puis en la maintenant appuyée, appuyer sur la touche **C**. Cela permet de mettre fin à une tâche en cours d'exécution.

Cette action sera notée **Ctrl+C** dans la suite de ce document. Il en va de même pour toutes les autres combinaisons de touches, comme **Ctrl+Z** par exemple.

Navigation dans les répertoires

La commande permettant de changer de répertoire est à quelques détails près identique à celle utilisée sous DOS.

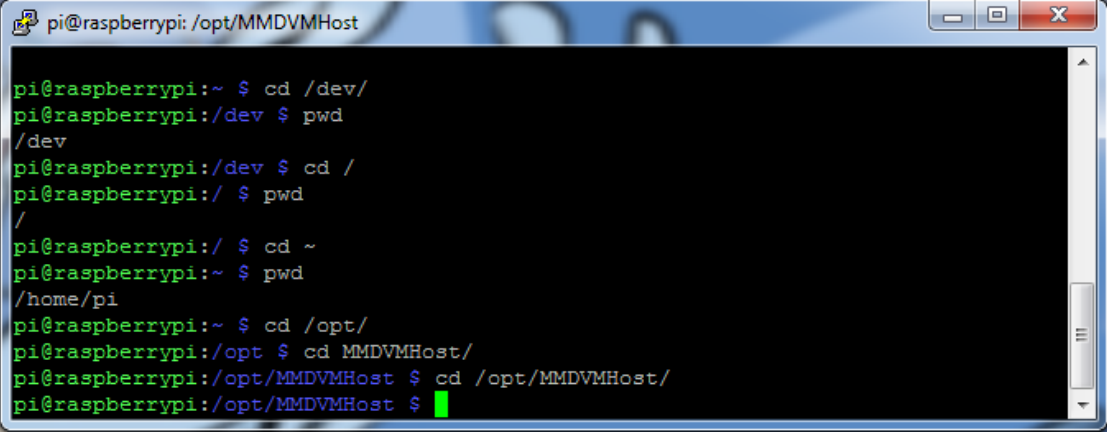
Dans l'exemple ci-dessous, je souhaite entrer dans le répertoire contenant les chemins d'accès aux périphériques de la carte Raspberry Pi :

A screenshot of a terminal window titled 'pi@raspberrypi: /dev'. The window has a blue title bar with standard Linux window controls. The terminal background is black with green text. The first line shows the prompt 'pi@raspberrypi:~ \$' followed by the command 'cd /dev/'. The second line shows the prompt 'pi@raspberrypi:/dev \$' followed by a green cursor block.

```
pi@raspberrypi:~ $ cd /dev/  
pi@raspberrypi:/dev $
```

Il est préférable de garder pour habitude d'utiliser des chemins absolus, c'est à dire qu'on considère toujours qu'on part de la racine de l'arborescence des répertoires. La racine est le niveau le plus bas de l'arborescence du « disque », comme [c:\](#) sous Windows. L'objectif étant de ne pas se soucier de l'emplacement courant avant le changement de répertoire. Vous pourrez utiliser des chemins relatifs quand vous vous serez familiarisé avec l'environnement Linux. Un chemin relatif permet de n'indiquer que le nom d'un sous-répertoire présent dans le répertoire courant.

En cas de doute concernant le chemin courant, on peut le vérifier à l'aide de la commande *pwd*.



```
pi@raspberrypi: /opt/MMDVMHost
pi@raspberrypi:~ $ cd /dev/
pi@raspberrypi:/dev $ pwd
/dev
pi@raspberrypi:/dev $ cd /
pi@raspberrypi:/ $ pwd
/
pi@raspberrypi:/ $ cd ~
pi@raspberrypi:~ $ pwd
/home/pi
pi@raspberrypi:~ $ cd /opt/
pi@raspberrypi:/opt $ cd MMDVMHost/
pi@raspberrypi:/opt/MMDVMHost $ cd /opt/MMDVMHost/
pi@raspberrypi:/opt/MMDVMHost $
```

Dans l'exemple ci-dessus, je vérifie que je suis bien allé dans le répertoire **/dev** avec la commande *pwd*.

Je me déplace ensuite dans la racine avec *cd /*.

Je me déplace ensuite dans le répertoire de l'utilisateur **pi** (le nom d'utilisateur par défaut sur le système d'exploitation Raspbian) avec la commande *cd ~*. On voit bien que cette commande nous emmène dans **/home/pi** grâce à la commande *pwd*.

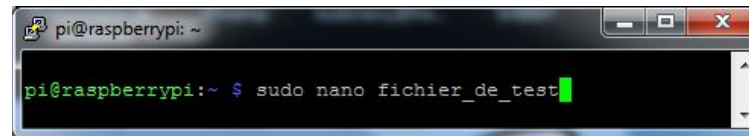
Enfin, je me déplace dans le répertoire **/opt/** en utilisant son chemin absolu avec la commande *cd /opt/*, puis dans **MMDVMHost** en utilisant son chemin relatif avec la commande *cd MMDVMHost/*. On voit enfin que la commande définissant le chemin absolu complet nous emmène au même endroit : **cd /opt/MMDVMHost/**.

Édition de fichiers depuis la console

Il sera nécessaire d'éditer des fichiers de configuration des modules logiciels du MMDVM, ou de créer des fichiers pour l'exécution automatique de ces modules par Linux. La façon la plus simple et la plus sûre d'effectuer ces actions consiste à utiliser un éditeur de texte depuis la console Linux. Ainsi, on ne perd pas le format spécifique des fichiers, notamment les caractères invisibles de sauts de ligne qui sont différents de ceux utilisés par Windows, ce qui évite d'être confronté à des messages d'erreur qui ne sont pas forcément explicites pour les personnes découvrant Linux. L'éditeur dont l'utilisation est la plus intuitive, et qui est présent nativement sur le système d'exploitation de la Raspberry Pi, s'appelle **nano**. L'utilisation de cet éditeur nécessite des droits « super utilisateur », car il permet de

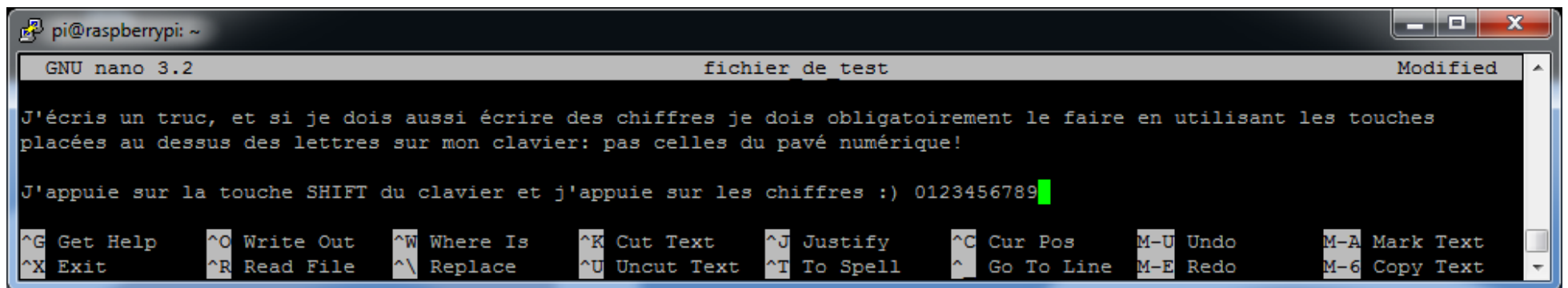
modifier n'importe quel fichier, avec tous les risques de corruption de ceux-ci qui peuvent en découler. Ainsi, on exécutera ce logiciel en le précédant de la commande **sudo**, comme pour d'autres actions tout aussi sensibles décrites plus loin dans ce document.

Exemple d'édition d'un fichier nommé « fichier_de_test » :



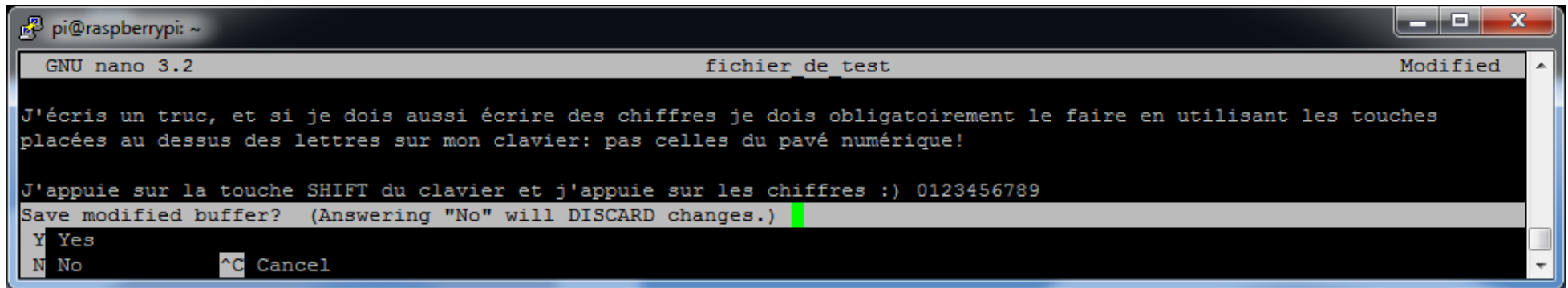
```
pi@raspberrypi: ~  
pi@raspberrypi:~ $ sudo nano fichier_de_test
```

Si ce fichier n'existe pas, il sera créé automatiquement au moment où nous demanderons à **nano** de sauvegarder les modifications.



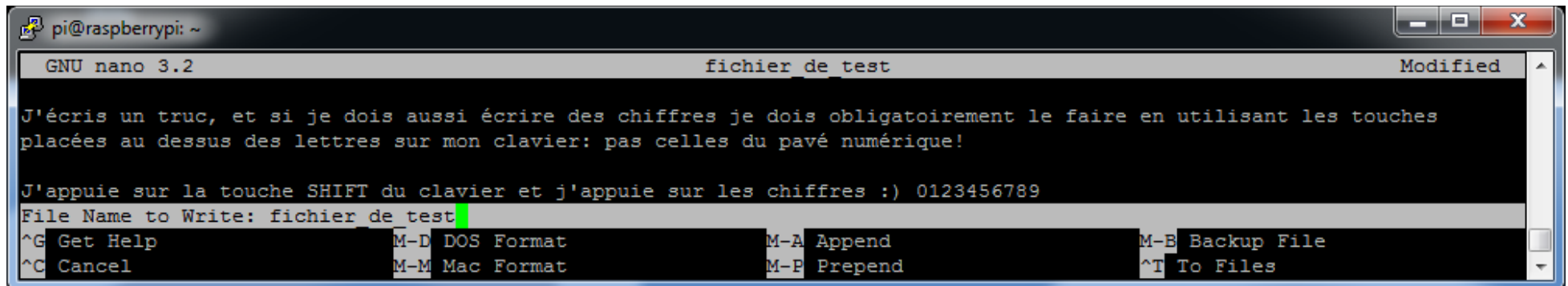
```
GNU nano 3.2      fichier_de_test      Modified  
J'écris un truc, et si je dois aussi écrire des chiffres je dois obligatoirement le faire en utilisant les touches  
placées au dessus des lettres sur mon clavier: pas celles du pavé numérique!  
  
J'appuie sur la touche SHIFT du clavier et j'appuie sur les chiffres :) 0123456789  
  
^G Get Help  ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos    M-U Undo      M-A Mark Text  
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell   ^ Go To Line  M-E Redo      M-6 Copy Text
```

La sauvegarde est proposée au moment où on quitte le **nano**, avec la combinaison de touches **Ctrl+X**:



The screenshot shows the GNU nano 3.2 editor interface. The title bar indicates the file is 'fichier_de_test' and it has been 'Modified'. The main text area contains two lines of French text. Below the text, a prompt asks 'Save modified buffer? (Answering "No" will DISCARD changes.)'. At the bottom, there are three options: 'Y Yes', 'N No', and '^C Cancel'. The 'Y' key is highlighted with a green cursor.

Il suffit enfin d'appuyer sur la touche **Y** pour confirmer la sauvegarde, **N** pour quitter sans sauvegarder, ou **Ctrl+C** pour revenir à l'édition du fichier sans effectuer de sauvegarde.



The screenshot shows the GNU nano 3.2 editor interface. The title bar indicates the file is 'fichier_de_test' and it has been 'Modified'. The main text area contains the same two lines of French text as the previous screenshot. Below the text, a prompt asks 'File Name to Write: fichier_de_test'. At the bottom, there is a table of keyboard shortcuts:

^G Get Help	M-D DOS Format	M-A Append	M-B Backup File
^C Cancel	M-M Mac Format	M-P Prepend	^T To Files

Ici le nom de fichier est rappelé après avoir appuyé sur **Y**. Si on souhaite garder ce nom de fichier, il suffit d'appuyer sur la touche **Entrée**, sinon on peut le modifier avant d'appuyer sur la touche **Entrée**. On peut aussi annuler cette action à tout moment en appuyant sur **Ctrl+C**.

Vous connaissez maintenant les bases de l'utilisation de la console Linux, nous allons pouvoir passer aux choses sérieuses !

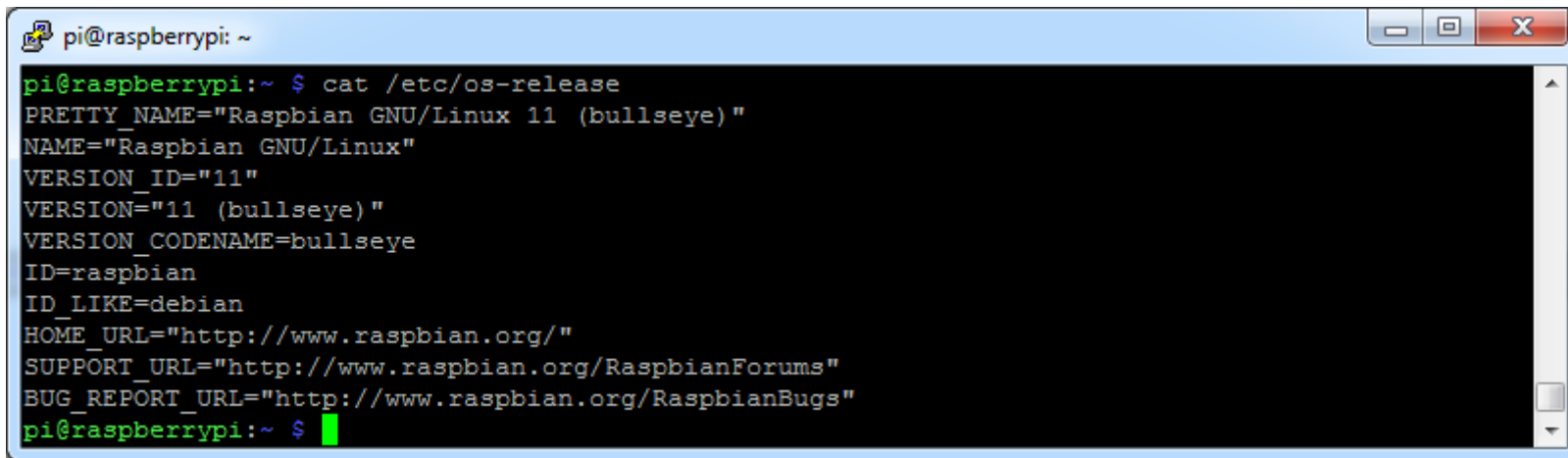
Conditions initiales

Les étapes décrites dans ce document sont basées sur le système d'exploitation **Raspbian 11 Bullseye** « light », qui ne dispose pas d'interface graphique. L'objectif étant de ne pas consommer de ressources matérielles inutilement sur la carte Raspberry Pi.

Le texte écrit ci-dessous en italique est correspond aux commandes à taper au clavier sur la console. Il faut ensuite les valider en appuyant sur la touche **Entrée**.

Version du système d'exploitation

cat /etc/os-release

A screenshot of a terminal window titled 'pi@raspberrypi: ~'. The terminal shows the command 'cat /etc/os-release' being executed, followed by its output. The output lists various system identifiers for Raspbian 11 Bullseye, including the pretty name, version ID, version, codename, ID, and support URLs. The prompt 'pi@raspberrypi:~ \$' is visible at the bottom of the terminal.

```
pi@raspberrypi:~ $ cat /etc/os-release
PRETTY_NAME="Raspbian GNU/Linux 11 (bullseye)"
NAME="Raspbian GNU/Linux"
VERSION_ID="11"
VERSION="11 (bullseye)"
VERSION_CODENAME=bullseye
ID=raspbian
ID_LIKE=debian
HOME_URL="http://www.raspbian.org/"
SUPPORT_URL="http://www.raspbian.org/RaspbianForums"
BUG_REPORT_URL="http://www.raspbian.org/RaspbianBugs"
pi@raspberrypi:~ $
```

Pré-requis

Mise à jour du système d'exploitation

```
sudo apt-get update  
sudo apt-get upgrade -y
```

Installation de GIT

```
sudo apt install git -y
```

Installation de l'outil de compilation

```
sudo apt-get install gcc-arm-none-eabi -y
```

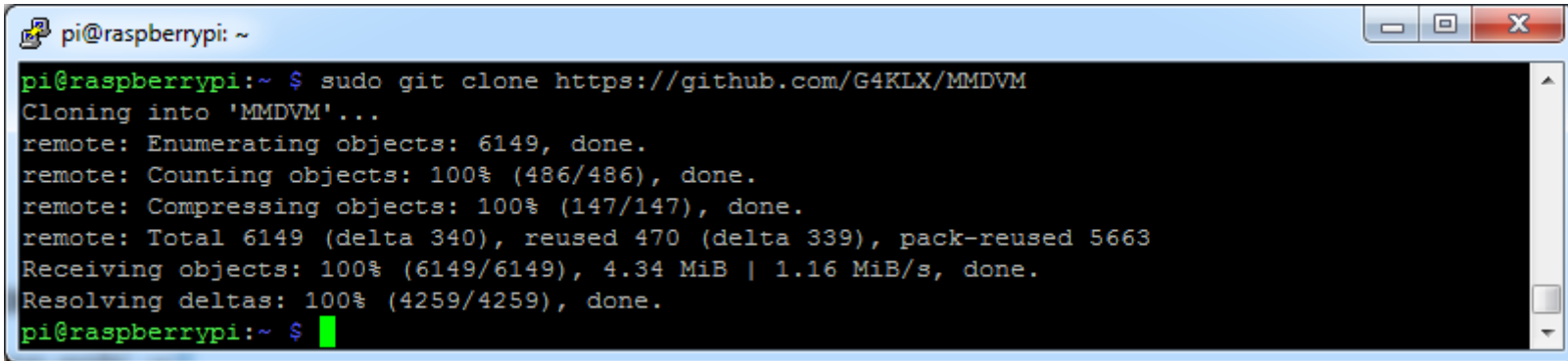
Programmation de la carte interface NUCLEO STM32F466 avec le firmware MMDVM

L'ancienne méthode utilisée est déplacée en **Annexe 1** à la fin de ce document, à des fins d'information uniquement, car elle n'est plus applicable aux versions récentes du logiciel MMDVM développé par G4KLX. Il doit être possible d'adapter cette ancienne méthode pour continuer à compiler depuis Windows, mais la nouvelle méthode utilisant la Raspberry Pi est nettement plus simple finalement.

Chargement des fichiers et compilation depuis la carte Raspberry Pi

La commande à exécuter pour charger les fichiers source est la suivante :

```
sudo git clone https://github.com/G4KLX/MMDVM
```



```
pi@raspberrypi: ~  
pi@raspberrypi:~ $ sudo git clone https://github.com/G4KLX/MMDVM  
Cloning into 'MMDVM'...  
remote: Enumerating objects: 6149, done.  
remote: Counting objects: 100% (486/486), done.  
remote: Compressing objects: 100% (147/147), done.  
remote: Total 6149 (delta 340), reused 470 (delta 339), pack-reused 5663  
Receiving objects: 100% (6149/6149), 4.34 MiB | 1.16 MiB/s, done.  
Resolving deltas: 100% (4259/4259), done.  
pi@raspberrypi:~ $
```

Ouvrir le répertoire MMDVM fraîchement créé, avec la commande suivante :

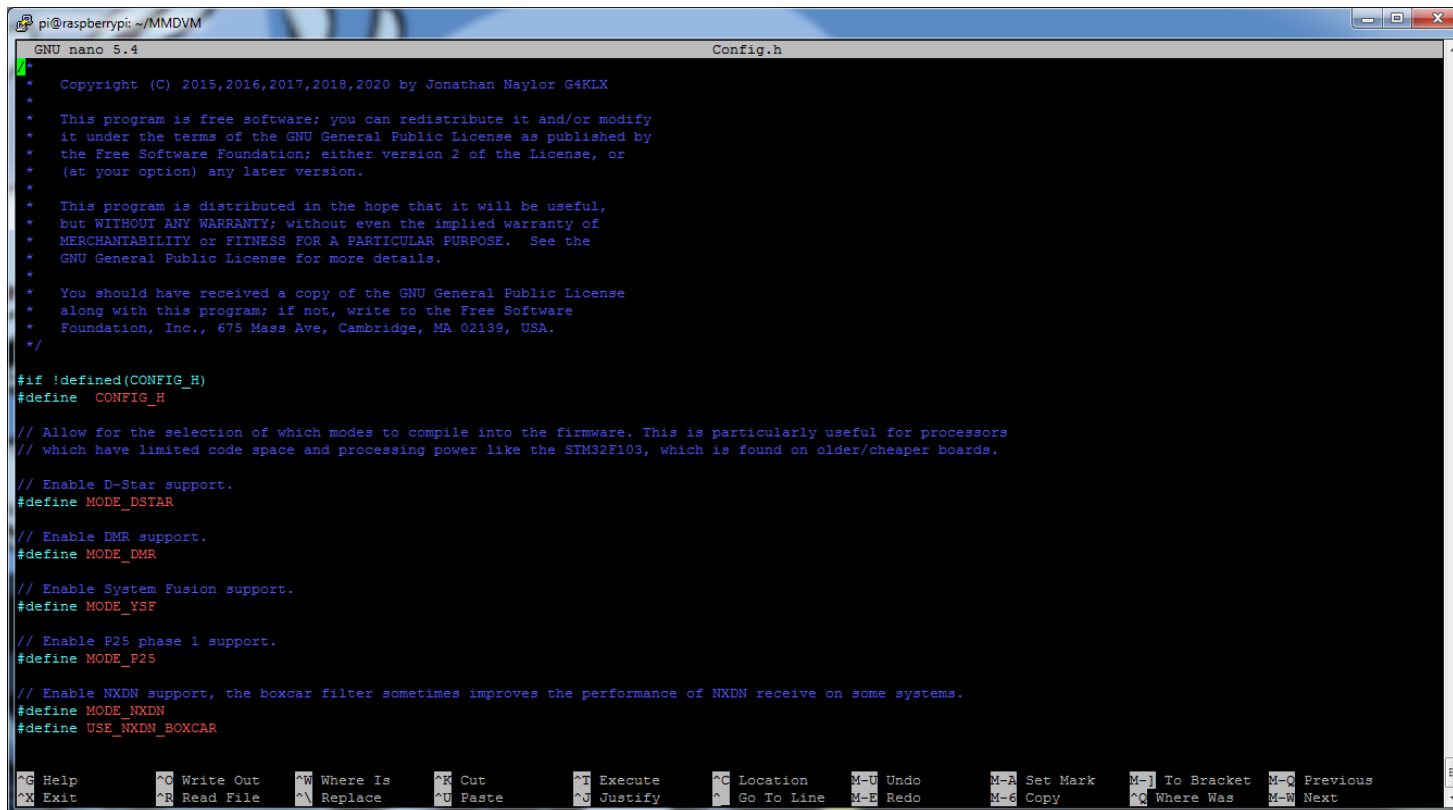
```
cd /home/pi/MMDVM
```

Charger les bibliothèques qui permettront de compiler le logiciel embarqué de la carte **Nucleo STM32F446**, avec cette commande :

`sudo git clone https://github.com/juribeparada/STM32F4XX\_Lib.git`

Éditer ensuite le fichier de configuration avec la commande suivante (attention, c'est un C majuscule) :

`sudo nano Config.h`



```
pi@raspberrypi: ~/MMDVM
GNU nano 5.4 Config.h
/*
 * Copyright (C) 2015,2016,2017,2018,2020 by Jonathan Naylor G4KLX
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
 */

#ifndef CONFIG_H
#define CONFIG_H

// Allow for the selection of which modes to compile into the firmware. This is particularly useful for processors
// which have limited code space and processing power like the STM32F103, which is found on older/cheaper boards.

// Enable D-Star support.
#define MODE_DSTAR

// Enable DMR support.
#define MODE_DMR

// Enable System Fusion support.
#define MODE_YSF

// Enable P25 phase 1 support.
#define MODE_P25

// Enable NXDN support, the boxcar filter sometimes improves the performance of NXDN receive on some systems.
#define MODE_NXDN
#define USE_NXDN_BOXCAR

^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo     M-A Set Mark  M-] To Bracket M-Q Previous
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^_ Go To Line  M-E Redo     M-G Copy      ^Q Where Was  M-W Next
```

On définit ici les paramètres de fonctionnement de la carte interface du MMDVM. Ils dépendront donc de votre besoin.

Pour information, voici le paramétrage que j'utilise pour la carte interface de F5UII :

/*

```
/*
 * Copyright (C) 2015,2016,2017,2018,2020 by Jonathan Naylor G4KLX
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
 */

#if !defined(CONFIG_H)
#define CONFIG_H

// Allow for the selection of which modes to compile into the firmware. This is particularly useful for processors
// which have limited code space and processing power.

// Enable D-Star support.
// #define MODE_DSTAR

// Enable DMR support.
#define MODE_DMR

// Enable System Fusion support.
#define MODE_YSF

// Enable P25 phase 1 support.
// #define MODE_P25

// Enable NXDN support, the boxcar filter sometimes improves the performance of NXDN receive on some systems.
// #define MODE_NXDN
```

```
//#define USE_NXDN_BOXCAR

// Enable M17 support.
//#define MODE_M17

// Enable POCSAG support.
#define MODE_POCSAG

// Enable FM support.
#define MODE_FM

// Enable AX.25 support, this is only enabled if FM is also enabled.
#define MODE_AX25

// Allow for the use of high quality external clock oscillators
// The number is the frequency of the oscillator in Hertz.
//
// The frequency of the TCX0 must be an integer multiple of 48000.
// Frequencies such as 12.0 Mhz (48000 * 250) and 14.4 Mhz (48000 * 300) are suitable.
// Frequencies such as 10.0 Mhz (48000 * 208.333) or 20 Mhz (48000 * 416.666) are not suitable.
//
// For 12 MHz
#define EXTERNAL_OSC 12000000
// For 12.288 MHz
// #define EXTERNAL_OSC 12288000
// For 14.4 MHz
// #define EXTERNAL_OSC 14400000
// For 19.2 MHz
// #define EXTERNAL_OSC 19200000

// Select a baud rate for host communication. The faster speeds are needed for external FM to work.
#define SERIAL_SPEED 115200 // Suitable for most older boards (Arduino Due, etc). External FM will NOT work with this!
// #define SERIAL_SPEED 230400 // Only works on newer boards like fast M4, M7, Teensy 3.x. External FM might work with
// this
// #define SERIAL_SPEED 460800 // Only works on newer boards like fast M4, M7, Teensy 3.x. External FM should work with
// this
// #define SERIAL_SPEED 500000 // Used with newer boards and Armbian on AllWinner SOC's (H2, H3) that do not support
460800
```

```
// Use pins to output the current mode via LEDs
#define MODE_LEDS

// For the original Arduino Due pin layout
// #define ARDUINO_DUE_PAPA

// For the ZUM V1.0 and V1.0.1 boards pin layout
// #define ARDUINO_DUE_ZUM_V10

// For the SP8NTH board
// #define ARDUINO_DUE_NTH

// For ST Nucleo-64 STM32F446RE board
#define STM32F4_NUCLEO_MORPHO_HEADER
// #define STM32F4_NUCLEO_ARDUINO_HEADER

// Use separate mode pins to switch external channel/filters/bandwidth for example
// #define MODE_PINS

// For the VK6MST Pi3 Shield communicating over i2c. i2c address & speed defined in i2cTeensy.cpp
// #define VK6MST_TEENSY_PI3_SHIELD_I2C

// Pass RSSI information to the host
// #define SEND_RSSI_DATA

// Use the modem as a serial repeater for Nextion displays
#define SERIAL_REPEATER

// Set the baud rate of the modem serial repeater for Nextion displays
#define SERIAL_REPEATER_BAUD_RATE 9600

// Use the modem as an I2C repeater for OLED displays
// #define I2C_REPEATER

// To reduce CPU load, you can remove the DC blocker by commenting out the next line
#define USE_DCBLOCKER
```

```
// Constant Service LED once repeater is running
// Do not use if employing an external hardware watchdog
// #define CONSTANT_SRV_LED

// Use the YSF and P25 LEDs for NXDN
// #define USE_ALTERNATE_NXDN_LEDS

// Use the D-Star and P25 LEDs for M17
#define USE_ALTERNATE_M17_LEDS

// Use the D-Star and DMR LEDs for POCSAG
#define USE_ALTERNATE_POCSAG_LEDS

// Use the D-Star and YSF LEDs for FM
#define USE_ALTERNATE_FM_LEDS

#endif
```

Si vous avez bien tout suivi, vous avez vu que j'ai commenté **MODE_M17**, et laissé **USE_ALTERNATE_M17_LEDS** dé-commenté. Le fait de laisser **USE_ALTERNATE_M17_LEDS** dé-commenté même si ce n'est pas utilisé permet d'éviter un échec lors de la compilation (test effectué le 14/04/2024).

Une fois le fichier édité, quitter **nano** en sauvegardant les modifications apportées, avec **Ctrl+X** suivi de **Y** puis d'**Entrée**.

Compiler ensuite le fichier binaire pour la carte **Nucleo STM32F446** en exécutant la commande suivante :

sudo make nucleo

Les étapes de compilation défilent à l'écran, jusqu'à la création du fichier binaire qui sera injecté dans la carte **Nucleo STM32F446**.


```
pi@raspberrypi: ~/MMDVM
xx.c -o obj_f4/startup_stm32f4xx.o
Compiled STM32F4XX_Lib/Device/startup/startup_stm32f4xx.c!

arm-none-eabi-g++ obj_f4/AX25Demodulator.o obj_f4/AX25Frame.o obj_f4/AX25RX.o obj_f4/AX25Twist.o obj_f4/AX25TX.o obj_f4/CalDMR.o obj_f4/CalDStarRX.o obj_f4/CalDStar
f4/CalPOCSAG.o obj_f4/CalRSSI.o obj_f4/CWidTX.o obj_f4/DMRDMORX.o obj_f4/DMRDMOTX.o obj_f4/DMRIdleRX.o obj_f4/DMRRX.o obj_f4/DMRSlotRX.o obj_f4/DMRSlotType.o obj_f4
o obj_f4/FMCTCSSRX.o obj_f4/FMCTCSSTX.o obj_f4/FMDownSampler.o obj_f4/FMKeyer.o obj_f4/FMNoiseSquelch.o obj_f4/FMTimeout.o obj_f4/FMTimer.o obj_f4/FMUpSampler.o obj
sy.o obj_f4/M17RX.o obj_f4/M17TX.o obj_f4/MMDVM.o obj_f4/NXDNRX.o obj_f4/NXDNTX.o obj_f4/P25RX.o obj_f4/P25TX.o obj_f4/POCSAGTX.o obj_f4/SerialArduino.o obj_f4/Seri
o obj_f4/STMUART.o obj_f4/Utils.o obj_f4/YSFRX.o obj_f4/YSFTX.o obj_f4/misc.o obj_f4/stm32f4xx_adc.o obj_f4/stm32f4xx_dac.o obj_f4/stm32f4xx_gpio.o obj_f4/stm32f4xx
2f4xx.o obj_f4/startup_stm32f4xx.o -Os --specs=nano.specs -Wl,-Map=bin/mmdvm.map -T stm32f4xx_link.ld -mcpu=cortex-m4 -mthumb -mlittle-endian -mfpv4-sp-d16 -mfl
Lib/GCC/libarm_cortexM4lf_math.a -o bin/mmdvm_f4.elf
Linking complete!

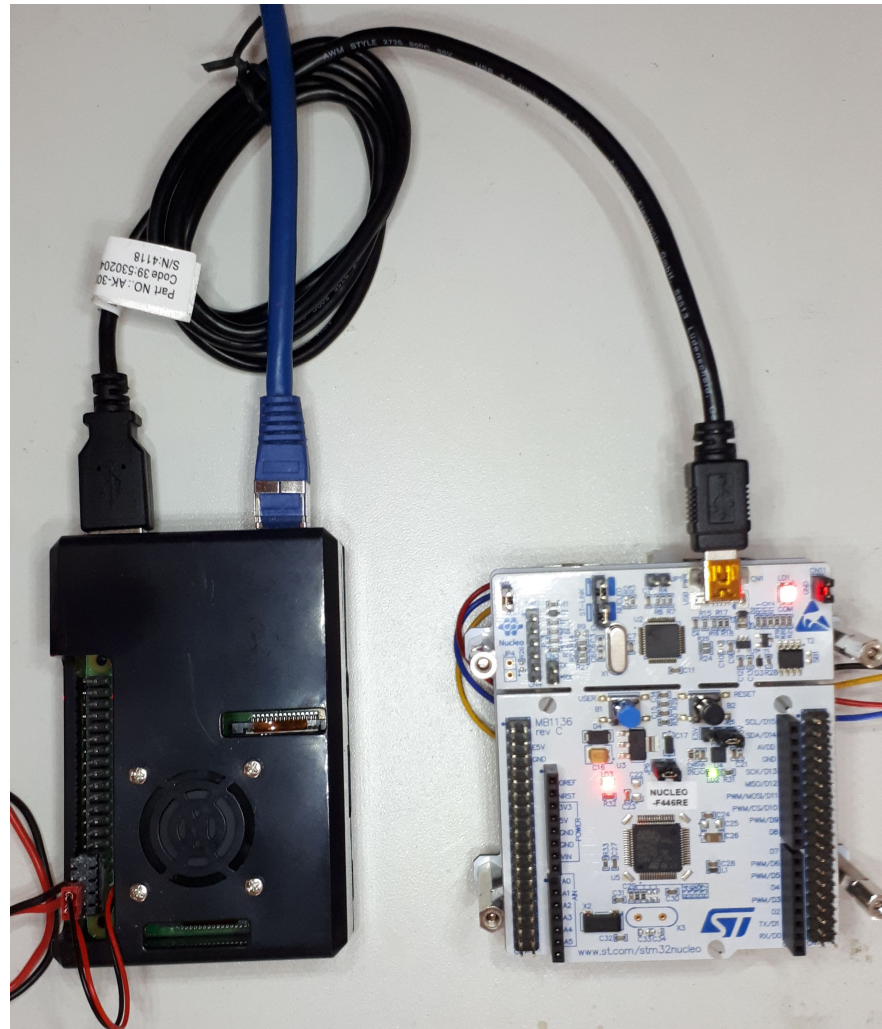
arm-none-eabi-size bin/mmdvm_f4.elf
   text    data     bss     dec     hex filename
 299408    1656    71388   372452   5aee4 bin/mmdvm_f4.elf
arm-none-eabi-objcopy -O ihex bin/mmdvm_f4.elf bin/mmdvm_f4.hex
Objcopy from ELF to IHEX complete!

arm-none-eabi-objcopy -O binary bin/mmdvm_f4.elf bin/mmdvm_f4.bin
Objcopy from ELF to BINARY complete!

pi@raspberrypi:~/MMDVM $
```

Il ne reste plus qu'à charger un de ces fichiers dans la carte **Nucleo STM32F446**.

La carte **Nucleo STM32F446** peut maintenant être connectée sur un des ports USB de la carte **Raspberry Pi** :

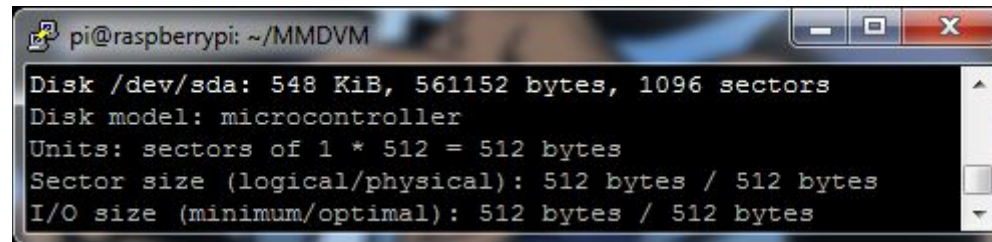


Cette opération nécessite de créer un lien logiciel vers l'interface de disque USB donnant accès à une partie de la mémoire de cette carte, afin d'y déposer un des fichiers qui viennent d'être générés. Le logiciel déjà présent dans cette carte se chargera de transférer ce fichier au bon endroit dans le microcontrôleur. Ce n'est pas plus compliqué que ça !

Vérifier tout d'abord que le lien USB est fonctionnel, en utilisant cette commande :

sudo fdisk -l

Parmi les informations affichées, vous devez voir celles-ci :



```
pi@raspberrypi: ~/MMDVM
Disk /dev/sda: 548 KiB, 561152 bytes, 1096 sectors
Disk model: microcontroller
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

Notez bien le chemin d'accès au disque, car il va nous servir juste après à créer le point de montage. Ici, il est écrit **/dev/sda**. Le chemin n'est pas forcément celui visible ci-dessus, donc soyez bien attentifs.

Créer le point de montage avec la commande suivante :

sudo mkdir /mnt/usb

Monter le disque USB :

sudo mount /dev/sda /mnt/usb

Copier le fichier portant l'extension **.bin** généré à l'issue de la compilation dans ce disque USB :

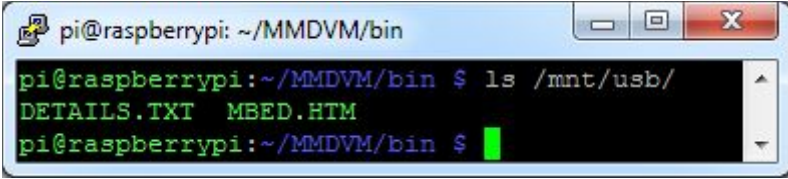
sudo cp /home/pi/MMDVM/bin/mmdvm_f4.bin /mnt/usb/

Patienter après cette dernière commande. Le logiciel embarqué de la carte **Nucleo STM32F446** ne traite pas forcément instantanément ce nouveau fichier.

Lorsque le fichier est pris en compte, la LED la plus proche du connecteur USB de la carte **Nucleo STM32F446** se met à clignoter en rouge et vert en alternance le temps du chargement du fichier. Cette LED reste allumée ensuite en vert fixe pendant une trentaine de secondes, puis la carte redémarre automatiquement une fois le chargement du fichier terminé. Il ne faut surtout pas couper l'alimentation électrique de cette carte avant qu'elle redémarre de son propre gré.

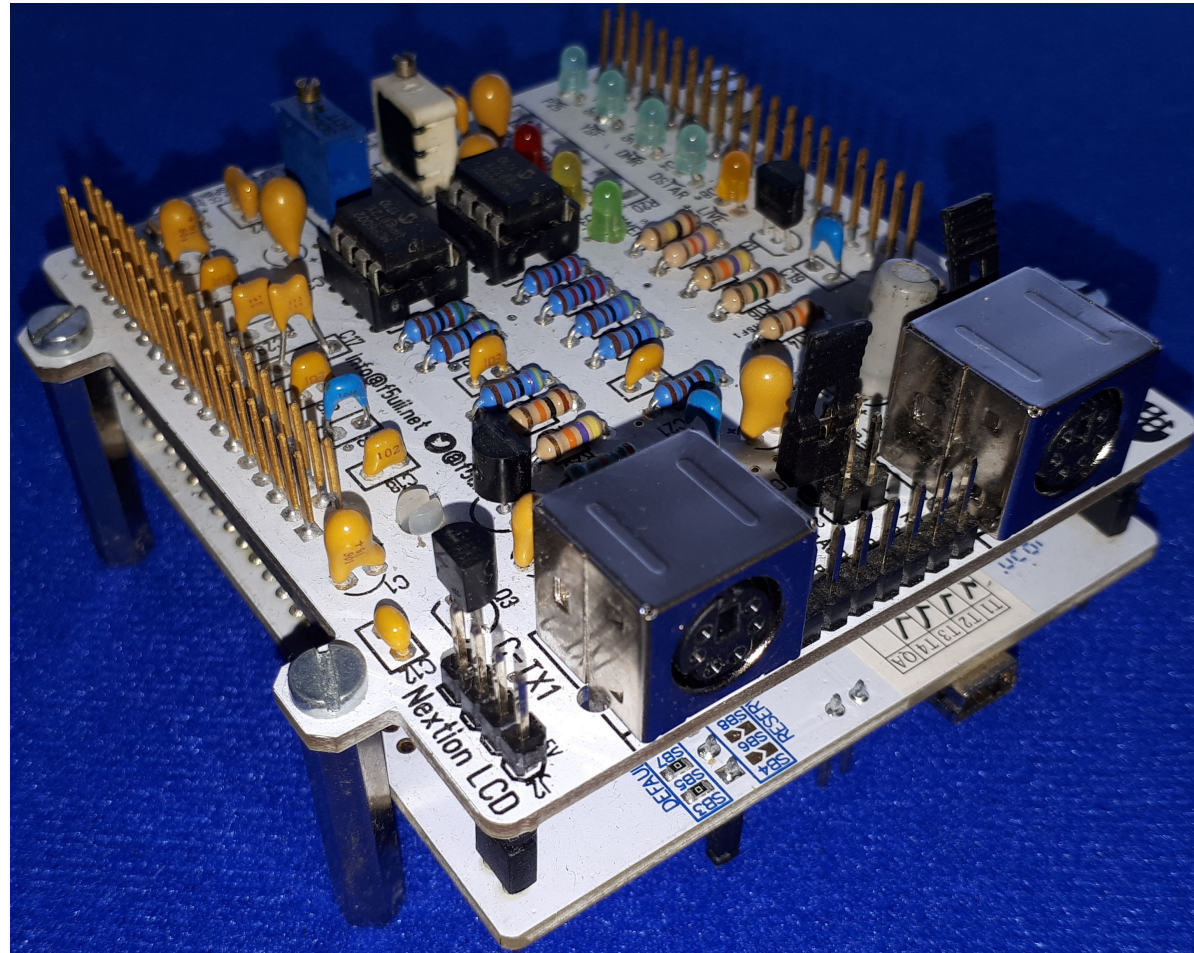
On peut d'ailleurs constater que le fichier précédemment chargé sur le disque USB de la carte **Nucleo STM32F446** a été supprimé automatiquement après avoir été chargé dans le microcontrôleur, en vérifiant le contenu du répertoire où il avait été déposé :

`ls /mnt/usb/`



```
pi@raspberrypi: ~/MMDVM/bin
pi@raspberrypi:~/MMDVM/bin $ ls /mnt/usb/
DETAILS.TXT  MBED.HTM
pi@raspberrypi:~/MMDVM/bin $
```

C'est fini pour cette partie du MMDVM. Vous pouvez enficher la carte d'interface radio sur la carte **Nucleo STM32F446**:



Chargement et installation des logiciels sur la carte Raspberry Pi

Pré-requis

Installation du module de communication par le port série pour Python

```
sudo apt-get install python3-serial
```

Installation de la [librairie libsamplerate obligatoire](#) pour le fonctionnement du mode relais FM

```
sudo apt install libsamplerate0-dev
```

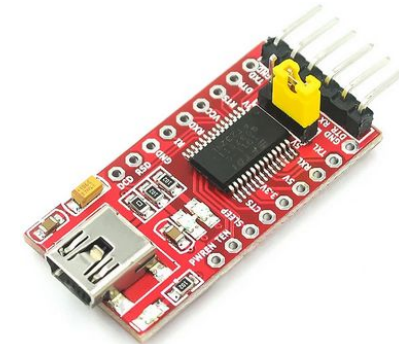
Installation de MMDVMHost

```
cd /opt  
sudo git clone https://github.com/G4KLX/MMDVMHost  
cd /opt/MMDVMHost/  
sudo make
```

Chargement des fichiers d'affichage dans l'écran Nextion

Même si vous n'effectuez qu'une mise à jour d'un MMDVM déjà en service, cette action peut rester nécessaire car le dialogue avec l'écran est susceptible d'évoluer lui aussi.

Les procédures décrites habituellement utilisent le port RS232 déjà présent sur la carte Raspberry Pi, mais cette action est devenue compliquée depuis que cette voie de communication est utilisée par défaut par l'interface WiFi / Bluetooth. Je préfère utiliser une carte interface USB vers RS232 qui est reconnue automatiquement par le système d'exploitation de la carte Raspberry Pi, et qui évite d'éditer un fichier de configuration sensible sur la carte **Raspberry Pi**.



On trouve très facilement ce type de carte sur les sites de vente en ligne, avec les mots clé **USB FT232 TTL**, pour moins de 2€. Cette carte est donc connectée par son port USB sur la carte Raspberry Pi, et on relie les contacts de l'afficheur sur les broches mâles.

Le brochage est le suivant :

Fil noir = GND (masse)

Fil rouge = +5V (alimentation électrique de l'écran)

Fil jaune = RX de l'écran, à relier sur le TX du module USB RS232

Fil bleu = TX de l'écran, à relier sur le RX du module USB RS232

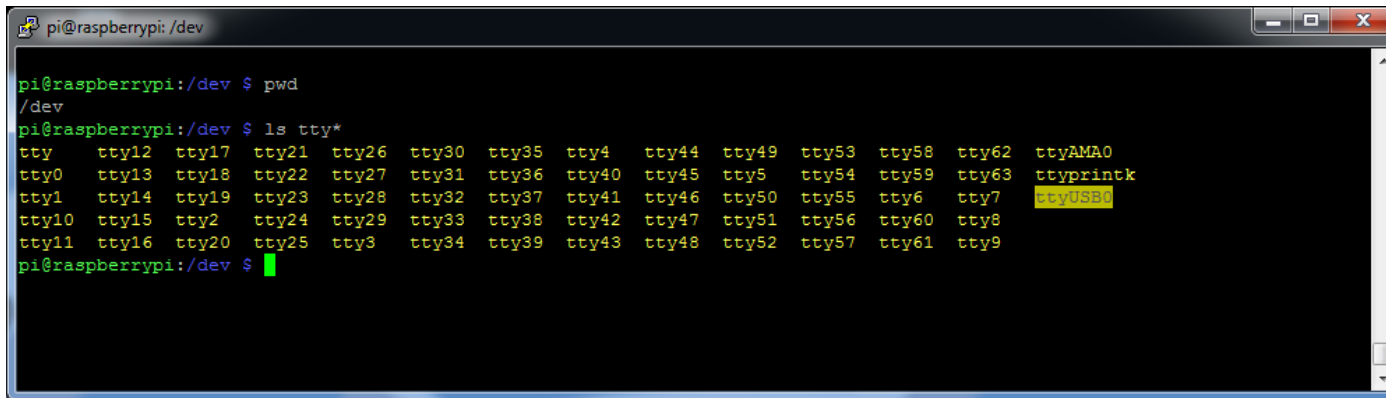


On prendra soin de sélectionner au préalable une tension de fonctionnement de 5V sur la carte interface USB RS232, si celle-ci n'est pas câblée nativement pour être connectée à des signaux UART à la norme TTL.

Côté Raspberry Pi, on doit voir le nom **tttyUSB0** dans la liste des périphériques :

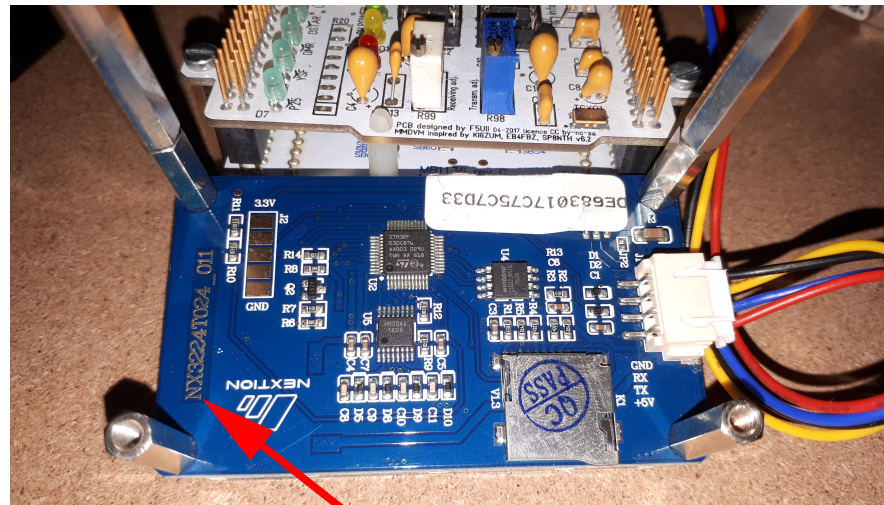
`cd /dev`

`ls ttty*`

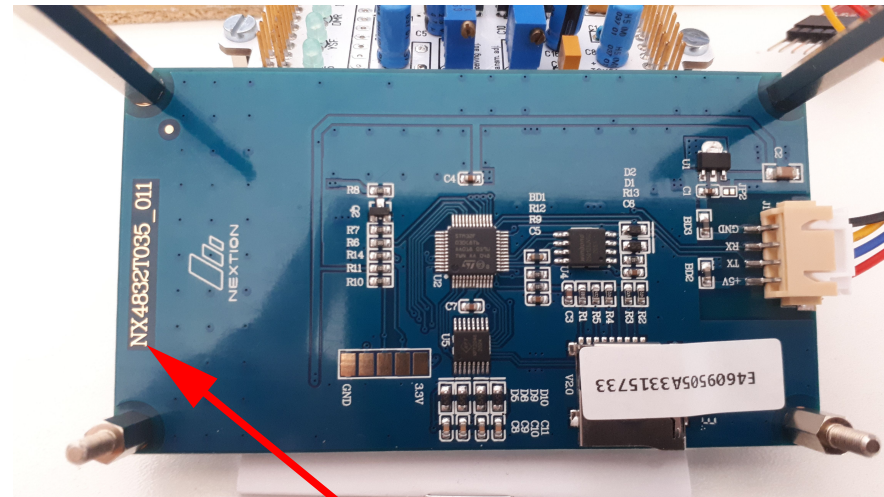


```
pi@raspberrypi:/dev $ pwd
/dev
pi@raspberrypi:/dev $ ls ttty*
ttty  ttty12  ttty17  ttty21  ttty26  ttty30  ttty35  ttty4  ttty44  ttty49  ttty53  ttty58  ttty62  tttyAMA0
ttty0  ttty13  ttty18  ttty22  ttty27  ttty31  ttty36  ttty40  ttty45  ttty5  ttty54  ttty59  ttty63  tttyprintk
ttty1  ttty14  ttty19  ttty23  ttty28  ttty32  ttty37  ttty41  ttty46  ttty50  ttty55  ttty6  ttty7  tttyUSB0
ttty10 ttty15  ttty2  ttty24  ttty29  ttty33  ttty38  ttty42  ttty47  ttty51  ttty56  ttty60  ttty8
ttty11 ttty16  ttty20  ttty25  ttty3  ttty34  ttty39  ttty43  ttty48  ttty52  ttty57  ttty61  ttty9
pi@raspberrypi:/dev $
```

Avant de charger le fichier prêt à l'emploi dans l'écran, il faut noter la référence de celui-ci, indiquée sur sa carte électronique :



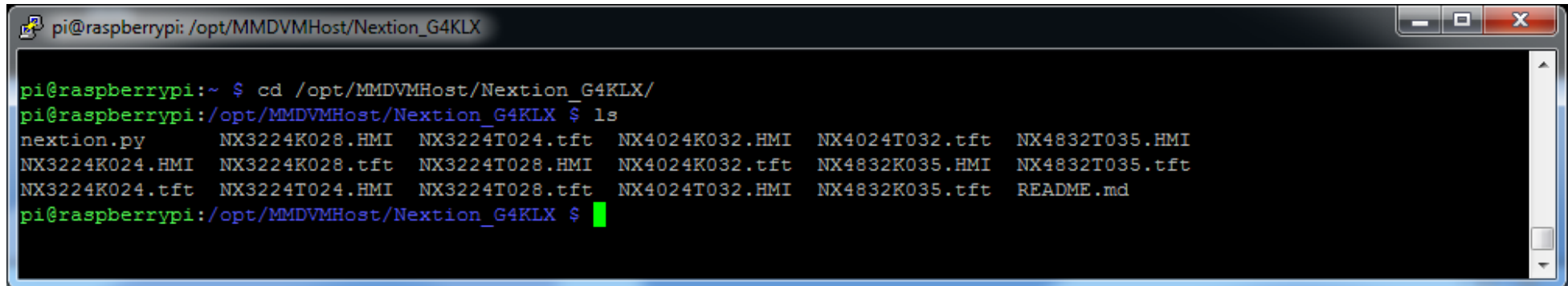
NX3224T024



NX4832T035

On peut lister toutes les références compatibles en se rendant dans le répertoire contenant les fichiers :

`cd /opt/MMDVMHost/Nextion_G4KLX/`



```
pi@raspberrypi: /opt/MMDVMHost/Nextion_G4KLX
pi@raspberrypi:~ $ cd /opt/MMDVMHost/Nextion_G4KLX/
pi@raspberrypi:/opt/MMDVMHost/Nextion_G4KLX $ ls
nextion.py      NX3224K028.HMI  NX3224T024.tft  NX4024K032.HMI  NX4024T032.tft  NX4832T035.HMI
NX3224K024.HMI  NX3224K028.tft  NX3224T028.HMI  NX4024K032.tft  NX4832K035.HMI  NX4832T035.tft
NX3224K024.tft  NX3224T024.HMI  NX3224T028.tft  NX4024T032.HMI  NX4832K035.tft  README.md
pi@raspberrypi:/opt/MMDVMHost/Nextion_G4KLX $
```

On choisit dans cette liste le fichier portant la référence de l'écran.

Le programme qui est fourni pour reprogrammer l'écran n'est compatible qu'avec la version 2 de Python. Si vous utilisez la version 3 de Python, il faut télécharger une version adaptée du programme, avec cette commande :

`wget http://dl.shibby.fr/Radio/MMDVM/tftwrite.py`

Le fichier tftwrite.py a été extrait de ce projet <https://pypi.org/project/GPS-clock/>

En prenant comme exemple l'écran portant la référence NX4832T035, le chargement s'effectue avec la commande suivante :

- Depuis **Python 2** :
 - `sudo python /opt/MMDVMHost/Nextion_G4KLX/nextion.py /opt/MMDVMHost/Nextion_G4KLX/NX4832T035.tft /dev/ttyUSB0`
- Depuis **Python 3** :
 - `sudo python /opt/MMDVMHost/Nextion_G4KLX/tftwrite.py /opt/MMDVMHost/Nextion_G4KLX/NX4832T035.tft /dev/ttyUSB0`

On peut observer la progression du chargement sur la console Linux et sur l'écran **Nextion** :



L'écran redémarre sur le logo MMDVM à la fin du chargement.

On peut alors déconnecter l'écran de sur le convertisseur USB RS232, et le relier à la carte interface radio sur le connecteur dédié:

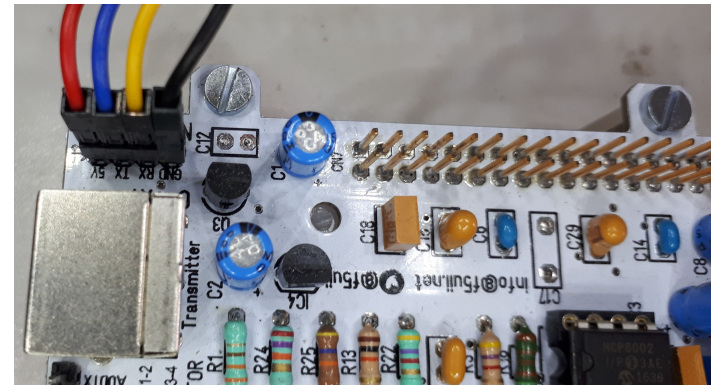
Sur la carte interface de F5UII, le brochage est le suivant :

Fil noir = GND

Fil rouge = 5V

Fil jaune = RX

Fil bleu = TX



Installation de MMDVMCal

```
cd /opt
```

```
sudo git clone https://github.com/G4KLX/MMDVMCal
```

```
cd /opt/MMDVMCal/
```

```
sudo make
```

Installation de DMRGateway (seulement si l'accès à un serveur d'interconnexion de relais phonie est souhaité) :

```
cd /opt
```

```
sudo git clone https://github.com/G4KLX/DMRGateway
```

```
cd /opt/DMRGateway/
```

```
sudo make
```

Installation de DAPNETGateway (seulement si l'accès à un serveur d'interconnexion de relais POCSAG est souhaité) :

```
cd /opt
sudo git clone https://github.com/G4KLX/DAPNETGateway
cd /opt/DAPNETGateway/
sudo make
```

Installation de MMDVM-Dashboard

```
sudo apt-get install lighttpd -y
sudo groupadd www-data
sudo usermod -G www-data -a pi
sudo chown -R www-data:www-data /var/www/html
sudo chmod -R 775 /var/www/html
sudo apt-get install php7.4-common php7.4-cgi php -y
sudo lighty-enable-mod fastcgi
sudo lighty-enable-mod fastcgi-php
sudo service lighttpd start
sudo service lighttpd force-reload
cd ~
sudo git clone https://github.com/dg9vh/MMDVMHost-Dashboard.git
sudo cp -R /home/pi/MMDVMHost-Dashboard/* /var/www/html/
```

À présent vous devez pouvoir vous connecter à l'interface web du tableau de bord MMDVM sur un explorateur internet, en tapant son IP dans la barre d'adresse :

MMDVM-Dashboard by DG9VH for Hotspot:
DMR-Id:



MMDVMHost by G4KLX Version:
Firmware:

You forgot to remove setup.php in root-directory of your dashboard or you forgot to configure it! Please delete the file or configure your Dashboard by calling [setup.php!](#)

Currently TXing

Time (UTC)	Mode	Callsign	Target	Source	TX-Time
------------	------	----------	--------	--------	---------

System Info

CPU-Temperature	CPU-Frequency	System-Load	CPU-Usage	Uptime	Idle
34.324 °C	1400 MHz	0.77 / 0.29 / 0.18	46.1 %	12 mins, 33 s	11 mins, 37 s

Disk Use

File System	Mount Point	Use	Free	Used	Total
/dev/root	/	21%	5.38 GB	1.41 GB	7.11 GB
/dev/mmcblk0p1	/boot	19%	204.41 MB	47.64 MB	252.05 MB

Repeater Info

Current Mode

idle

Location	TX-Freq.	RX-Freq.
	. MHz	. MHz

Enabled Modes

DMR	DMR Network	D-Star	D-Star Network	System Fusion	System Fusion Network	P25	P25 Network	NXDN	NXDN Network	POCSAG	POCSAG Network
-----	-------------	--------	----------------	---------------	-----------------------	-----	-------------	------	--------------	--------	----------------

Last Heard List of today's 0 callsigns.

Show entries

Search:

Time (UTC)	Mode	Callsign	Target	Source	Dur (s)	Loss	BER
No data available in table							

Showing 0 to 0 of 0 entries

Previous Next

Today's local transmissions

Show entries

Search:

Time (UTC)	Mode	Callsign	Target	Source	Dur (s)	Loss	BER	RSSI (avg)
No data available in table								

Showing 0 to 0 of 0 entries

Previous Next

Paramétrage du tableau de bord MMDVM-Dashboard

AJOUTER ICI LA PROCÉDURE

Test rapide de dialogue entre la Raspberry Pi, l'interface MMDVM et l'écran Nextion

Tous les logiciels sont installés, et les cartes sont reliées entre elles. Nous allons donc pouvoir faire un premier test de bon fonctionnement de l'ensemble des cartes constituant le MMDVM, c'est toujours rassurant de voir que le temps passé à exécuter toutes les actions précédentes n'a pas été perdu.

Dans un premier temps, seuls les paramètres définissant les interfaces de communication entre chaque carte sont nécessaires.

Éditer le fichier MMDVM.ini avec la commande suivante :

```
sudo nano /opt/MMDVMHost/MMDVM.ini
```

Chaque fonction a ses paramètres regroupés dans un même bloc, identifié par le texte compris entre les crochets [et].

Afin de vous aider à trouver facilement les paramètres à modifier, vous trouverez ci-dessous dans la colonne de gauche le nom du bloc concerné par la modification, et à sa droite le paramètre à modifier, avec les modifications apportées surlignées en vert :

[General]	Display=Nextion
[Modem]	UARTPort=/dev/ttyACM0
[Modem]	UARTSpeed=115200
[Nextion] Ici on va enlever le signe # au début de la ligne suivante :	Port=modem
[Nextion] Ici on va ajouter le signe # au début de la ligne suivante :	#Port=/dev/ttyAMA0
[Nextion]	ScreenLayout=0

Quitter enfin l'édition en sauvegardant les modifications apportées : **Ctrl+X** suivi de **Y** puis d'**Entrée**.

Une fois les paramètres de fonctionnement définis, on peut tester rapidement le bon fonctionnement du MMDVM en exécutant le programme principal avec la commande suivante :

```
sudo /opt/MMDVMHost/MMDVMHost /opt/MMDVMHost/MMDVM.ini
```

Si l'exécution se termine par ce message, vérifier que le MMDVM est bien connecté à la Raspberry Pi :

```
E: 2022-06-04 19:25:03.626 Cannot open device - /dev/ttyACM0
```

Si la liaison entre la Raspberry Pi et l'interface MMDVM fonctionne, on doit voir ce type de message :

```
I: 2022-06-04 19:28:29.291 This software is for use on amateur radio networks only,  
I: 2022-06-04 19:28:29.292 it is to be used for educational purposes only. Its use on  
I: 2022-06-04 19:28:29.292 commercial networks is strictly prohibited.  
I: 2022-06-04 19:28:29.292 Copyright(C) 2015-2021 by Jonathan Naylor, G4KLX and others  
M: 2022-06-04 19:28:29.292 MMDVMHost-20220523 is starting  
M: 2022-06-04 19:28:29.292 Built 16:40:46 Jun 4 2022 (GitID #fe195c4)
```

— Les informations indiquées ensuite n'ont aucun intérêt pour ce premier test, je les ai donc supprimées pour une meilleure lisibilité —

```
M: 2022-06-04 19:28:32.344 MMDVMHost-20220523 is running
```

Cette dernière ligne nous confirme que le MMDVM est en cours d'exécution.

La communication avec l'écran Nextion est elle aussi validée car celui-ci affiche des informations liées au MMDVM à présent :



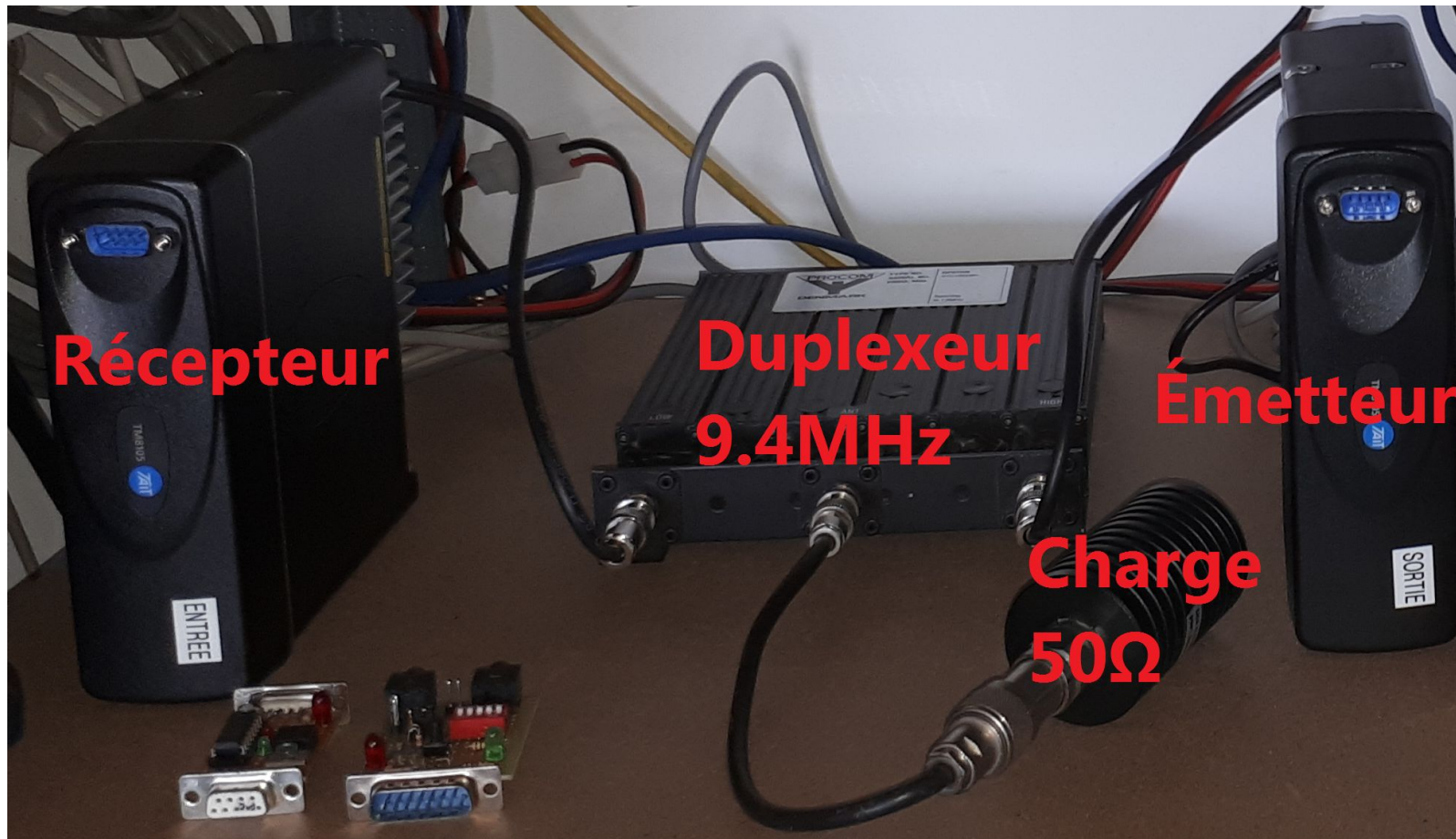
On peut quitter l'application avec **Ctrl+Z**.

Le message suivant apparaît alors à l'écran:

```
^Z
[2]+ Stopped sudo /opt/MMDVMHost/MMDVMHost /opt/MMDVMHost/MMDVM.ini
pi@raspberrypi:/opt $
```

On voit la commande qui nous a permis de quitter le logiciel « ^Z », le nom du logiciel qui était en cours d'exécution, et le retour à l'invite de commande sur la dernière ligne.

Configuration de test du relais MMDVM

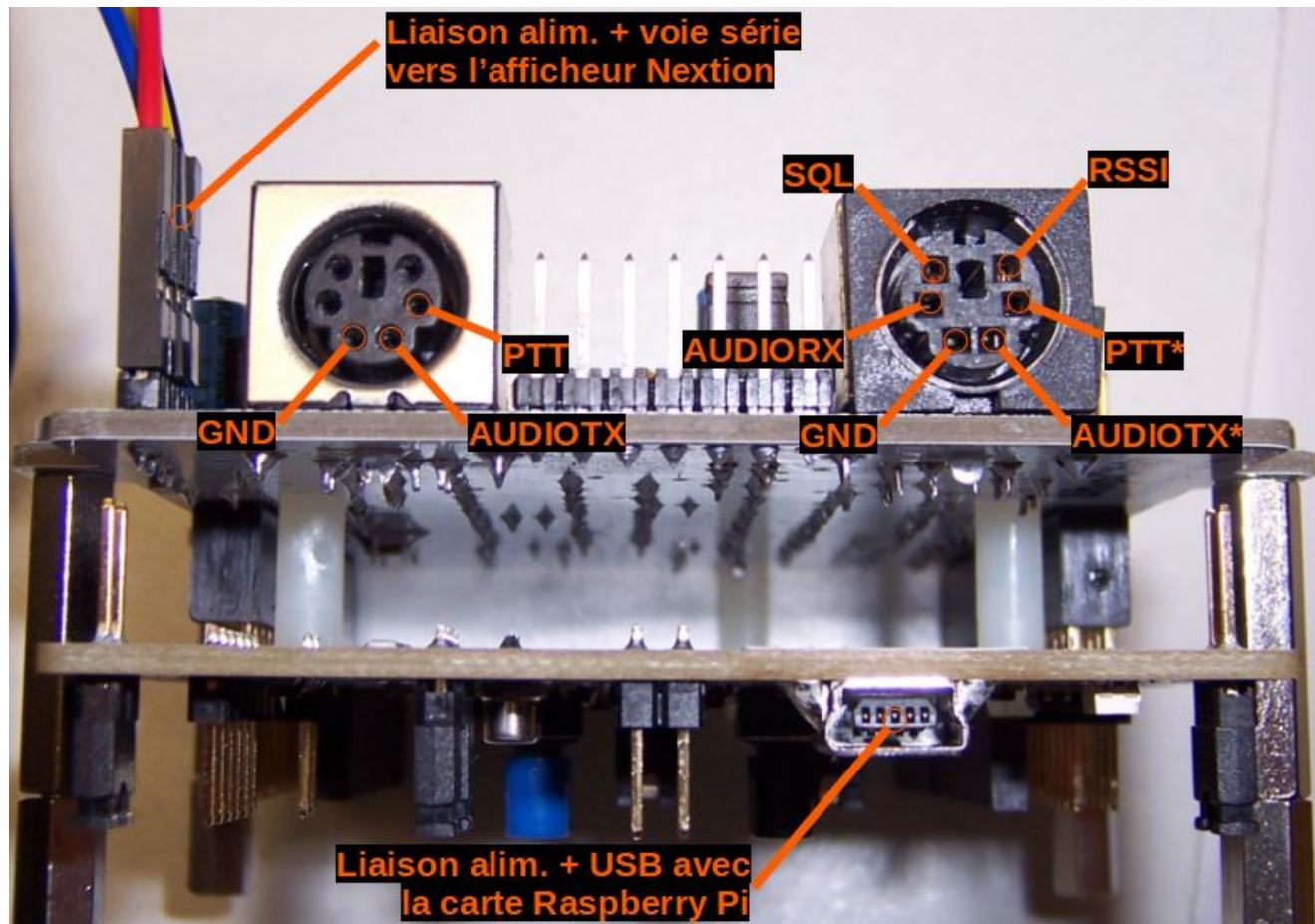


Chaque poste est connecté à l'interface MMDVM par l'intermédiaire de sa prise « accessoires » en face arrière, jusqu'à la prise mini-DIN 6 broches de l'interface avec la carte **Nucleo STM32F446**. Les entrées/sorties audio des postes sont récupérées **avant filtrage en interne**.



Il faut noter ici que la connexion sur l'interface MMDVM des équipements d'émission/réception constituant le relais est **obligatoire** pour les étapes de calibration.

Brochage des connecteurs Mini-Din 6 points et liaisons externes



* Seulement valable les broches 1 et 2 de JP1 sont reliées ensemble, et si les broches 3 et 4 sont reliées ensemble.

Calibration du MMDVM avec MMDVMCal

L'étape de calibration permet de régler les niveaux de réception et d'émission sur la carte interface analogique, qui est chargée d'adapter les signaux issus des équipements radio avant de les appliquer à l'entrée et à la sortie de la carte processeur STM32.

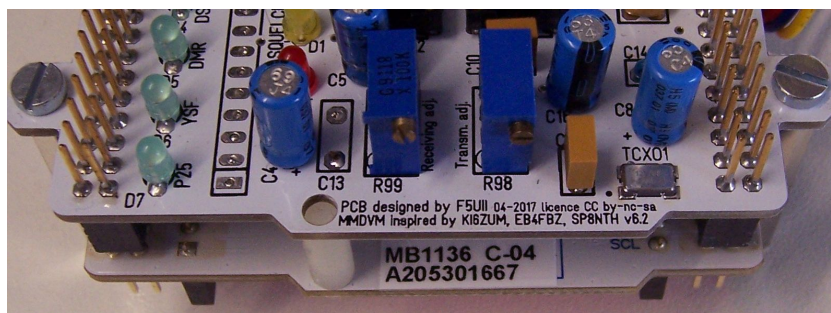
Cette étape est cruciale et doit être réalisée avec le plus grand soin, car elle assure le bon fonctionnement du MMDVM en réception et en émission. Des signaux audio trop forts ou trop faibles impliqueraient un taux d'erreur important, dégradant significativement la qualité des transmissions, allant jusqu'à rendre impossible le décodage des émissions, aussi bien du côté du relais que de celui des équipements radio des opérateurs.

Cette opération est grandement simplifiée grâce aux informations fournies par MMDVMCal d'une part, et à celles obtenues avec une simple clé SDR.

Pour que MMDVMCal puisse être exécuté, il faut que le MMDVM ne soit pas déjà sollicité par un autre logiciel, comme MMDVMHost par exemple. Il faut donc quitter si besoin MMDVMHost avant d'exécuter MMDVMCal, avec la commande suivante :

```
sudo systemctl stop mmdvmhost.service
```

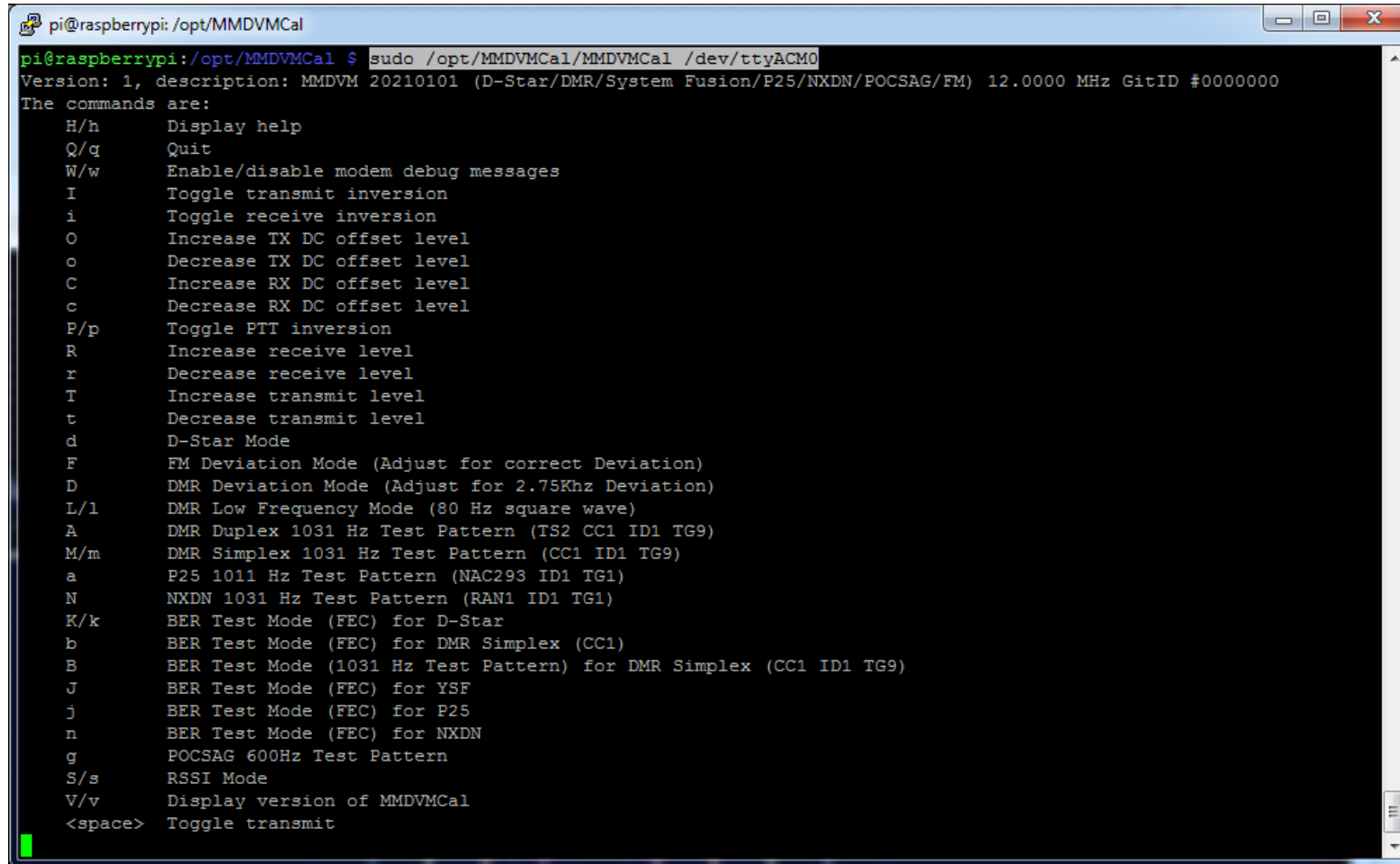
On ajustera le niveau audio du signal reçu à l'aide de la résistance variable multi-tours R99 (à gauche), et celui du signal émis par l'intermédiaire de la résistance variable multi-tours R98 (à droite) :



Commande permettant d'exécuter MMDVMCal :

```
sudo /opt/MMDVMCal/MMDVMCal /dev/ttyACM0
```


Exemple de calibration pour un émetteur DAPNET



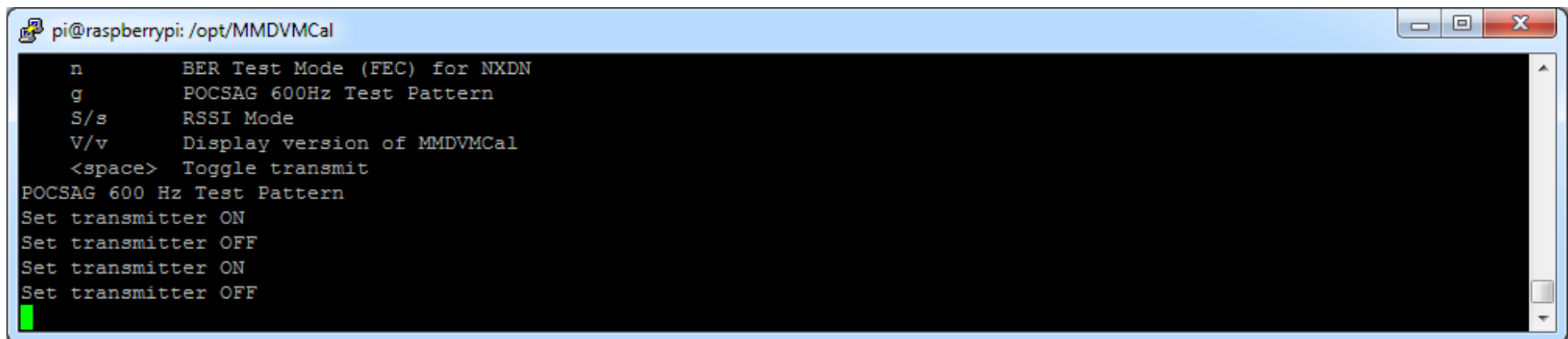
```
pi@raspberrypi: /opt/MMDVMCal
pi@raspberrypi:/opt/MMDVMCal $ sudo /opt/MMDVMCal/MMDVMCal /dev/ttyACM0
Version: 1, description: MMDVM 20210101 (D-Star/DMR/System Fusion/P25/NXDN/POCSAG/FM) 12.0000 MHz GitID #0000000
The commands are:
H/h      Display help
Q/q      Quit
W/w      Enable/disable modem debug messages
I        Toggle transmit inversion
i        Toggle receive inversion
O        Increase TX DC offset level
o        Decrease TX DC offset level
C        Increase RX DC offset level
c        Decrease RX DC offset level
P/p      Toggle PTT inversion
R        Increase receive level
r        Decrease receive level
T        Increase transmit level
t        Decrease transmit level
d        D-Star Mode
F        FM Deviation Mode (Adjust for correct Deviation)
D        DMR Deviation Mode (Adjust for 2.75Khz Deviation)
L/l      DMR Low Frequency Mode (80 Hz square wave)
A        DMR Duplex 1031 Hz Test Pattern (TS2 CC1 ID1 TG9)
M/m      DMR Simplex 1031 Hz Test Pattern (CC1 ID1 TG9)
a        P25 1011 Hz Test Pattern (NAC293 ID1 TG1)
N        NXDN 1031 Hz Test Pattern (RAN1 ID1 TG1)
K/k      BER Test Mode (FEC) for D-Star
b        BER Test Mode (FEC) for DMR Simplex (CC1)
B        BER Test Mode (1031 Hz Test Pattern) for DMR Simplex (CC1 ID1 TG9)
J        BER Test Mode (FEC) for YSF
j        BER Test Mode (FEC) for P25
n        BER Test Mode (FEC) for NXDN
g        POCSAG 600Hz Test Pattern
S/s      RSSI Mode
V/v      Display version of MMDVMCal
<space> Toggle transmit
```

Notez bien que la casse (majuscule/minuscule) est à respecter car certaines lettres ont deux fonctions différentes !

Le logiciel démarre en indiquant toutes les commandes disponibles. On pourra par la suite lui demander de les afficher à nouveau en appuyant simplement sur la touche **h** du clavier.

Dans cet exemple, nous souhaitons ajuster l'excursion de fréquence d'un émetteur DAPNET, utilisant le protocole radio **POCSAG**

Il faut donc appuyer sur la touche **g** du clavier pour entrer dans le mode de test **POCSAG**, puis ensuite piloter le signal **PTT** de l'émetteur en appuyant sur la **barre d'espace** du clavier. Chaque appui sur celle-ci alternera le mode de fonctionnement de l'émetteur en émission ou en réception :



```
pi@raspberrypi: /opt/MMDVMCal
n      BER Test Mode (FEC) for NXDN
g      POCSAG 600Hz Test Pattern
S/s    RSSI Mode
V/v    Display version of MMDVMCal
<space> Toggle transmit
POCSAG 600 Hz Test Pattern
Set transmitter ON
Set transmitter OFF
Set transmitter ON
Set transmitter OFF
█
```

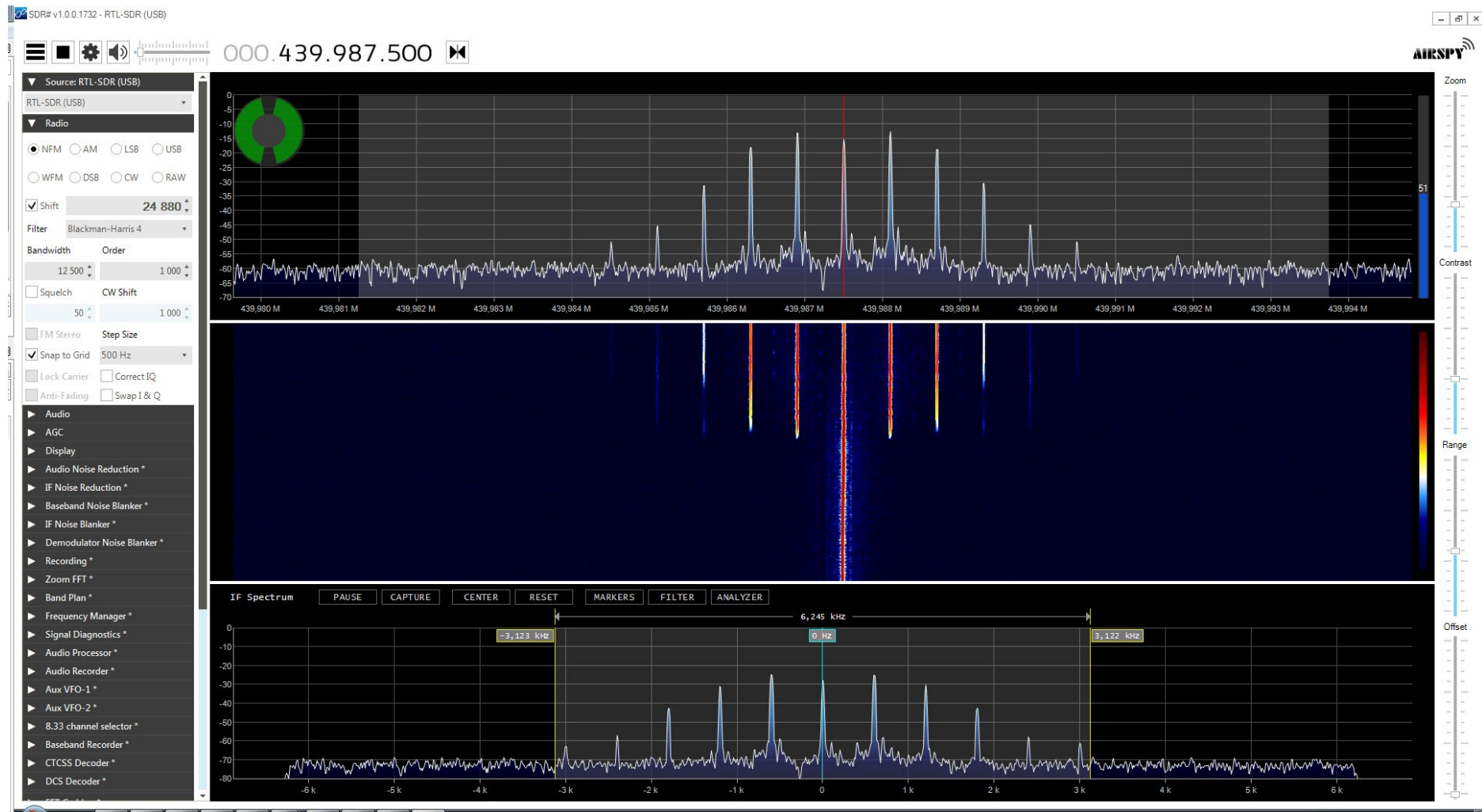
On doit entendre la tonalité de 600Hz sur un récepteur calé sur la même fréquence de l'émetteur. Si on n'entend qu'une porteuse, il faut vérifier que le signal nommé « AUDIOTX » sur la carte interface est bien relié sur l'entrée audio du poste émetteur, et le cas échéant ajuster R98 pour entendre le signal audio émis par le MMDVM.

Une fois cette vérification effectuée, on peut procéder au réglage de l'excursion de fréquence en s'aidant d'un récepteur SDR (même un modèle premier prix) et d'un logiciel de décodage quelconque.

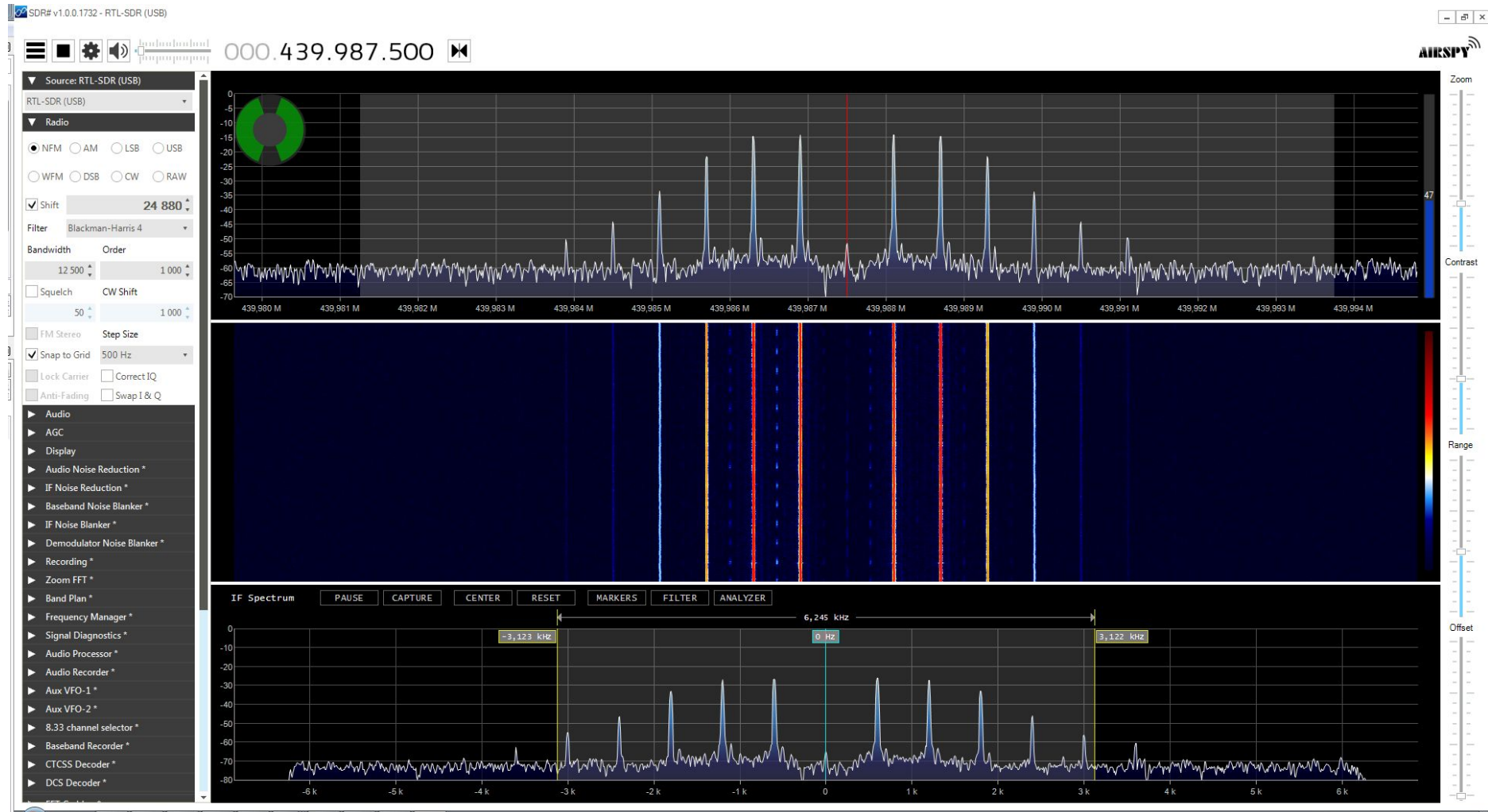
Dans l'exemple suivant, j'utilise une de ces clés « DVB-T+FM+DAB » montées dans un boîtier plastique bleu, et le logiciel **SDR#**.

Réglage audio au plus bas, on ne voit qu'une porteuse. On notera le réglage « Bandwidth » à 12500 Hz :

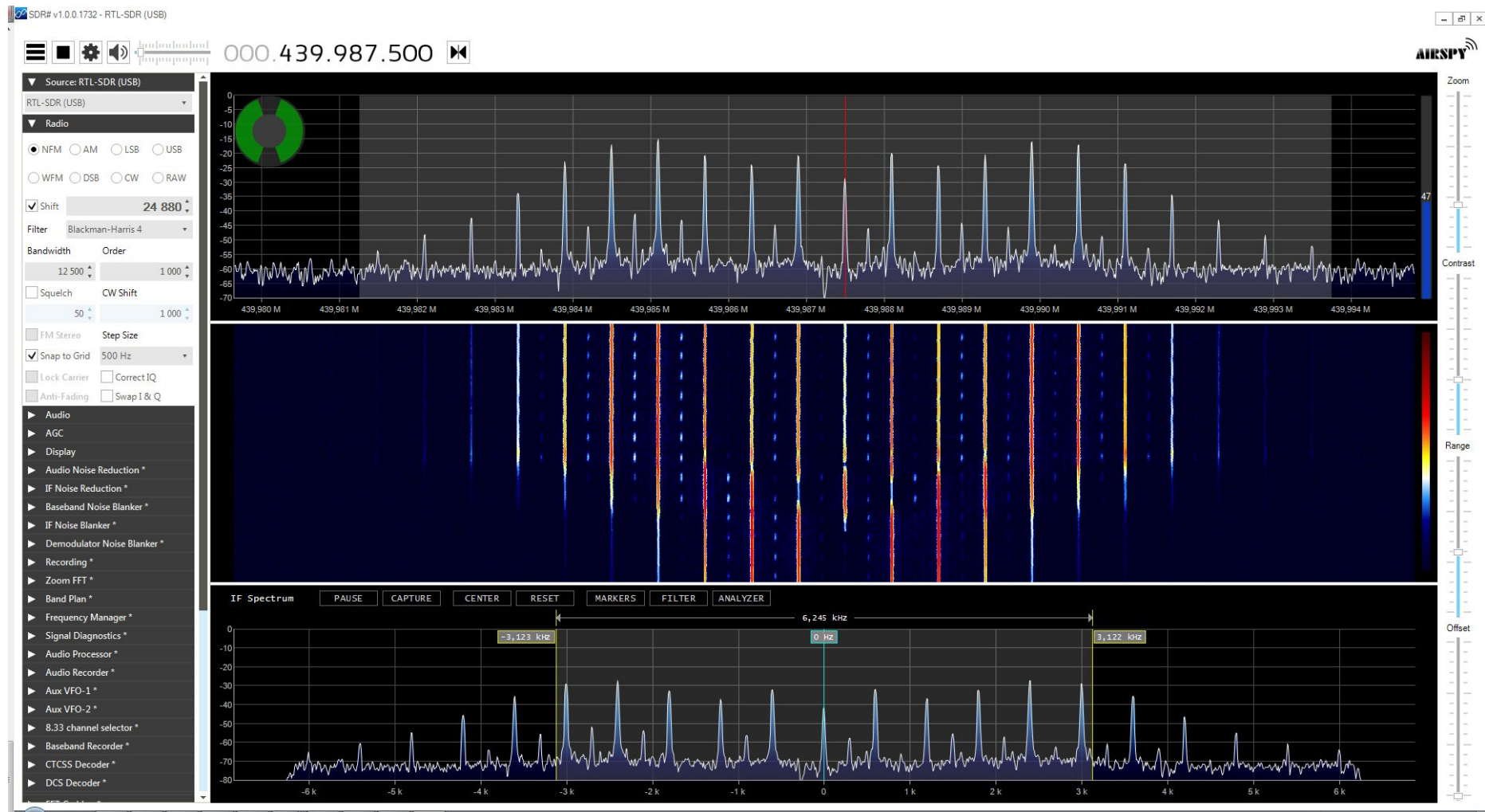
J'augmente petit à petit le niveau audio de la carte interface. Les raies du signal modulé apparaissent et celle de la porteuse est toujours bien visible au centre. Elle est placée pile sur la ligne verticale rouge en haut de l'écran, et sur la ligne verticale bleu clair en bas :



L'objectif est maintenant de continuer à augmenter le niveau audio jusqu'à ce que la raie centrale disparaisse. Ce réglage est très précis, et on s'aidera du spectre audio en bas de l'écran pour s'assurer qu'on n'a pas dépassé le point de réglage :



Si on dépasse le point de réglage, on constate que l'émission dépasse de la bande passante allouée, ce qu'il faut à tout prix éviter car cela rend l'émission inexploitable par les opérateurs qui utiliseront votre relais MMDVM :



Cas d'un relais Multi Mode

Lorsqu'on utilise le MMDVM en tant qu'émetteur DAPNET, il est très fortement déconseillé de l'utiliser en même temps dans d'autres modes. La raison est simple : les messages texte émis sur le DAPNET ne sont pas mémorisés en vue d'être émis lorsque c'est possible. Une émission DAPNET a donc de très fortes chances d'être perdue si le relais MMDVM est en cours d'utilisation dans un autre mode au même moment.

Une autre raison imposant l'incompatibilité du DAPNET avec d'autres modes sur le même relais est l'utilisation d'une fréquence d'émission unique pour les réseau DAPNET, dont les différents relais se partagent l'utilisation selon une fenêtre temporelle définie. Un même message est donc émis avec un léger décalage dans le temps entre des relais proches, afin d'assurer l'émission d'un seul relais à la fois et ainsi la bonne réception du signal. Ce décalage de transmission du message ne doit pas être confondu avec une mise en mémoire des messages. Il s'agit juste d'une distribution de ceux-ci étalée dans le temps par le serveur d'interconnexion au réseau.

Parlons à présent des autres modes gérés par le MMDVM. Même s'ils ne peuvent pas être utilisés simultanément lors d'une discussion, puisque le MMDVM n'effectue pas de conversion d'un mode vers un autre, il est tout à fait possible d'effectuer une conversation dans un mode, puis une autre dans un autre mode. Il faudra donc s'assurer que chacun des modes de transmission sera correctement retransmis par le MMDVM.

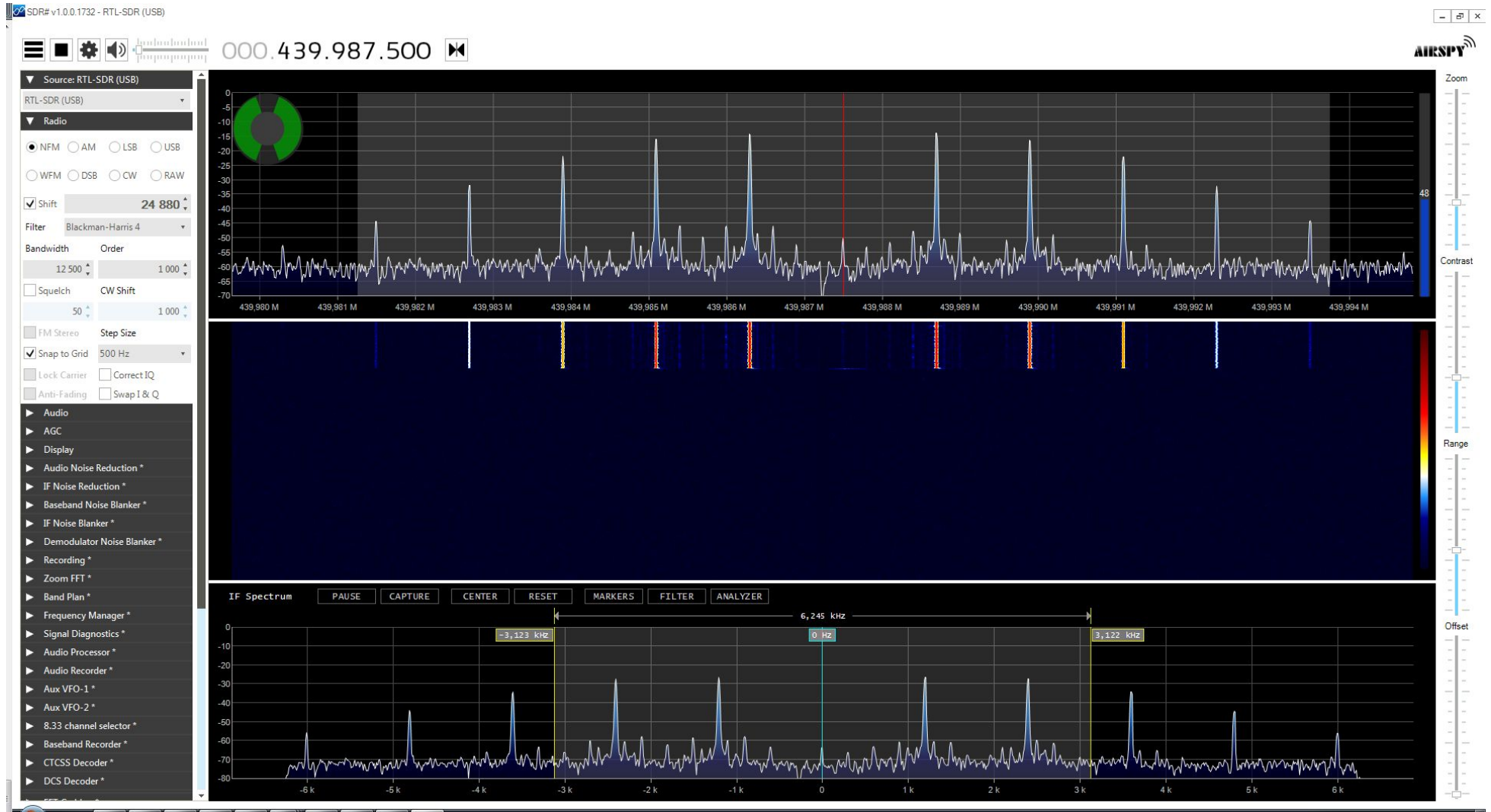
Ceci est défini d'une façon semblable au réglage d'excursion décrit précédemment pour le DAPNET, mais il faut aussi s'assurer que le réglage est valable pour tous les modes de transmission qui seront garés par le MMDVM. On utilisera alors une approche légèrement différente, qui consiste à définir un niveau d'émission fixe avec R98, et ensuite définir un niveau audio spécifique pour chaque mode et permettant d'aboutir au réglage d'excursion idéal pour chacun d'entre eux.

Réglage de l'excursion en émission

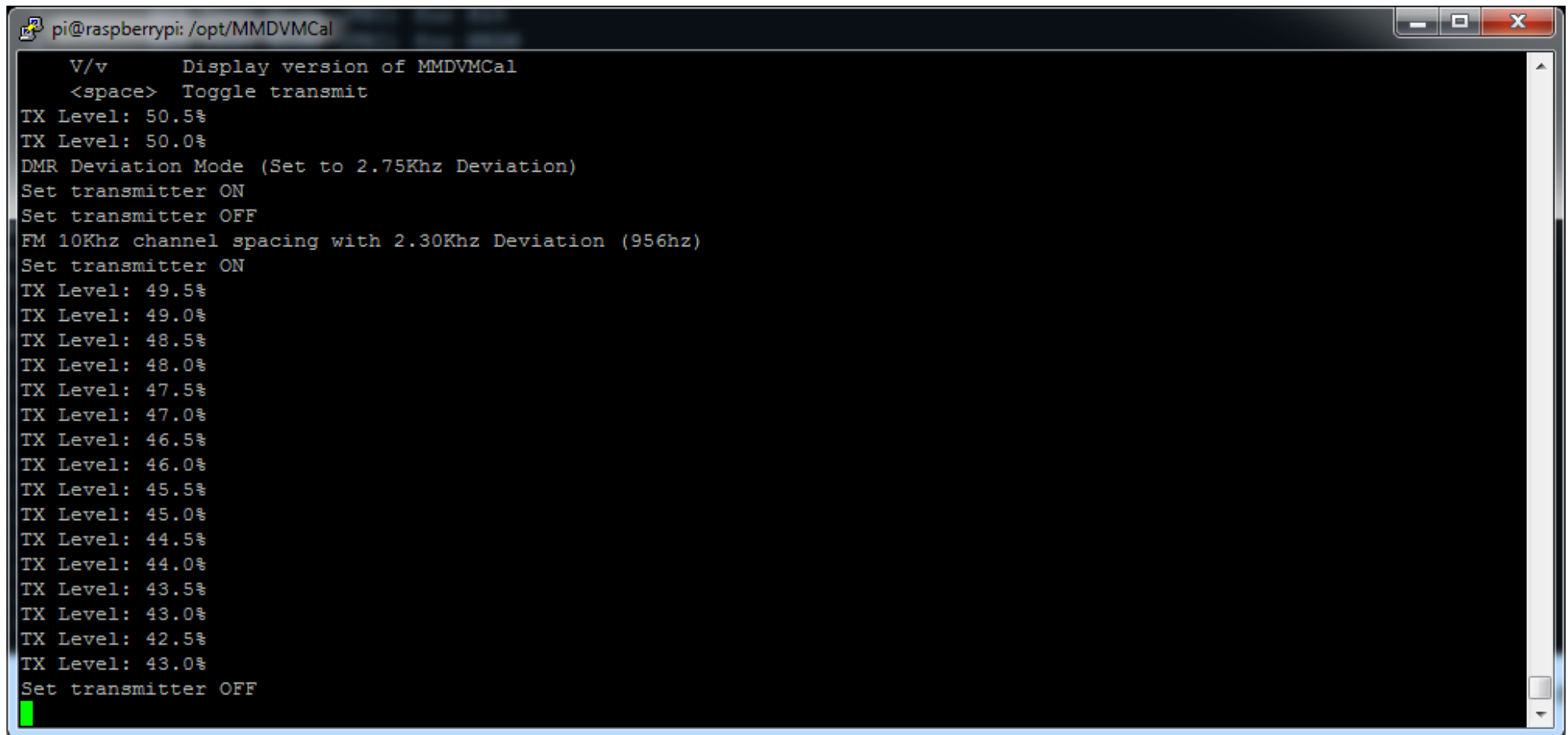
Pour cet exemple, je commence par le réglage de l'excursion en mode DMR, en laissant le niveau audio au réglage logiciel médian (50%). Le mode de test DMR est sélectionné en appuyant sur la touche **D** du clavier. Un appui rapide sur **T** puis **t** permet de vérifier que le niveau audio défini de façon logicielle est bien à 50 %. On active ensuite l'émission avec la **barre d'espace** et on règle le niveau de signal audio transmis avec R98, jusqu'à la disparition de la raie centrale sur les signaux affichés par le logiciel SDR.

```
pi@raspberrypi: /opt/MMDVMCal
pi@raspberrypi:/opt/MMDVMCal $
pi@raspberrypi:/opt/MMDVMCal $ sudo /opt/MMDVMCal/MMDVMCal /dev/ttyACM0
Version: 1, description: MMDVM 20210101 (D-Star/DMR/System Fusion/P25/NXDN/POCSAG/FM) 12.0000 MHz GitID #0000000
The commands are:
H/h      Display help
Q/q      Quit
W/w      Enable/disable modem debug messages
I        Toggle transmit inversion
i        Toggle receive inversion
O        Increase TX DC offset level
o        Decrease TX DC offset level
C        Increase RX DC offset level
c        Decrease RX DC offset level
P/p      Toggle PTT inversion
R        Increase receive level
r        Decrease receive level
T        Increase transmit level
t        Decrease transmit level
d        D-Star Mode
F        FM Deviation Mode (Adjust for correct Deviation)
D        DMR Deviation Mode (Adjust for 2.75Khz Deviation)
L/l      DMR Low Frequency Mode (80 Hz square wave)
A        DMR Duplex 1031 Hz Test Pattern (TS2 CC1 ID1 TG9)
M/m      DMR Simplex 1031 Hz Test Pattern (CC1 ID1 TG9)
a        P25 1011 Hz Test Pattern (NAC293 ID1 TG1)
N        NXDN 1031 Hz Test Pattern (RAN1 ID1 TG1)
K/k      BER Test Mode (FEC) for D-Star
b        BER Test Mode (FEC) for DMR Simplex (CC1)
B        BER Test Mode (1031 Hz Test Pattern) for DMR Simplex (CC1 ID1 TG9)
J        BER Test Mode (FEC) for YSF
j        BER Test Mode (FEC) for P25
n        BER Test Mode (FEC) for NXDN
g        POCSAG 600Hz Test Pattern
S/s      RSSI Mode
V/v      Display version of MMDVMCal
<space> Toggle transmit
TX Level: 50.5%
TX Level: 50.0%
DMR Deviation Mode (Set to 2.75Khz Deviation)
Set transmitter ON
Set transmitter OFF
█
```


J'obtiens le signal suivant en réglant R89 :



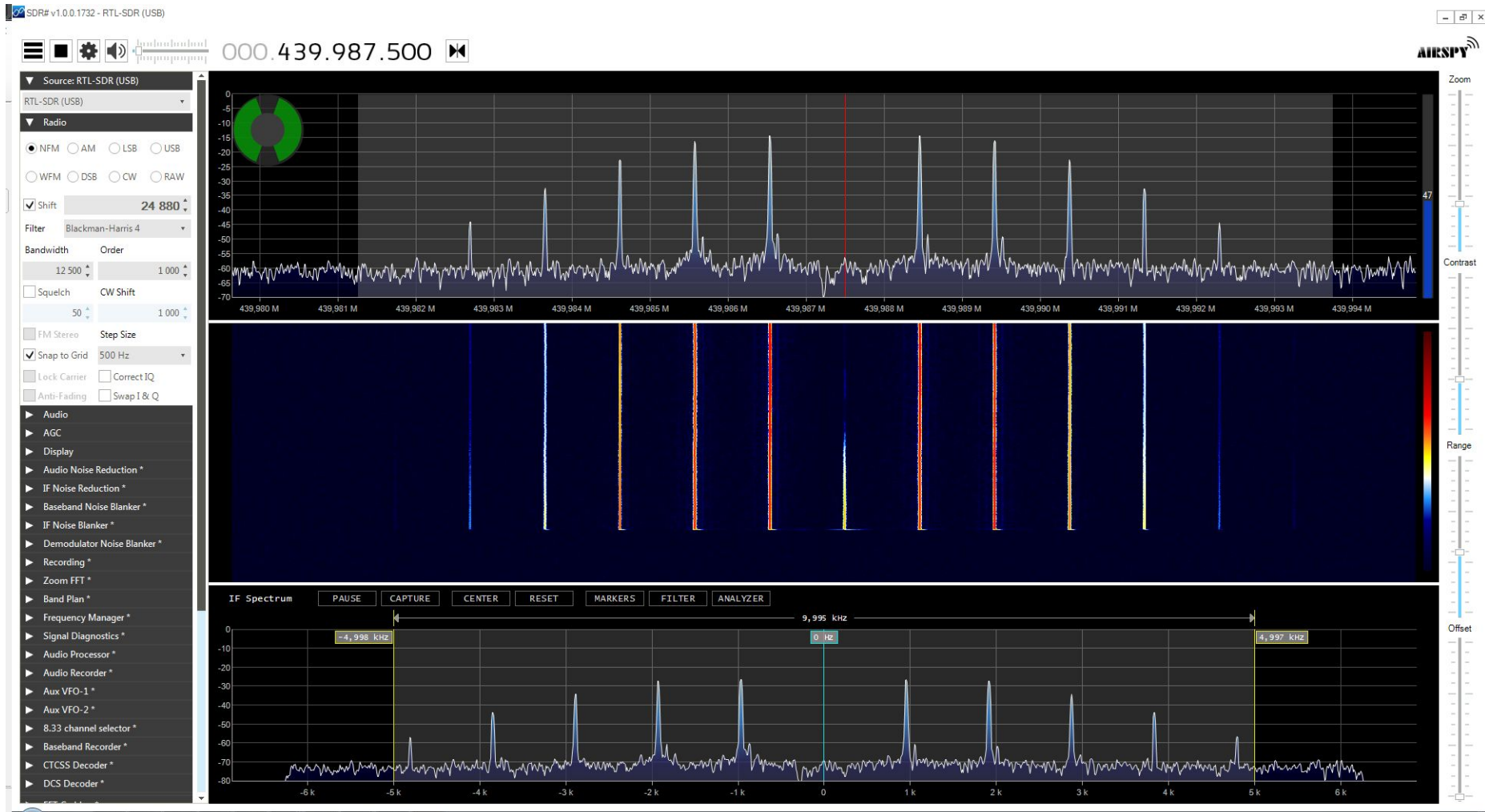
Maintenant, je passe dans le mode FM par exemple en appuyant sur **F**, je force l'émission de la tonalité de test avec la **barre d'espace** et je règle à présent le niveau audio d'émission de façon logicielle, avec les touches **T** pour augmenter ou **t** pour diminuer :



```
pi@raspberrypi: /opt/MMDVMCal
V/v      Display version of MMDVMCal
<space>  Toggle transmit
TX Level: 50.5%
TX Level: 50.0%
DMR Deviation Mode (Set to 2.75Khz Deviation)
Set transmitter ON
Set transmitter OFF
FM 10Khz channel spacing with 2.30Khz Deviation (956hz)
Set transmitter ON
TX Level: 49.5%
TX Level: 49.0%
TX Level: 48.5%
TX Level: 48.0%
TX Level: 47.5%
TX Level: 47.0%
TX Level: 46.5%
TX Level: 46.0%
TX Level: 45.5%
TX Level: 45.0%
TX Level: 44.5%
TX Level: 44.0%
TX Level: 43.5%
TX Level: 43.0%
TX Level: 42.5%
TX Level: 43.0%
Set transmitter OFF
```

Une fois le réglage effectué, j'appuie à nouveau sur la **barre d'espace** pour cesser l'émission.

Le résultat obtenu est le suivant :



Il n'y a pas d'autre mode de calibration en émission, donc on note les niveaux audio logiciels, et on ne touche plus jamais à R98, sauf pendant les campagnes d'entretien et de maintenance du relais :

DMR = 50 %

FM = 43 %

On reportera ces valeurs dans le fichier MMDVM.ini plus tard dans ce document.

Réglage du taux d'erreur binaire en réception

Comme son nom l'indique, il s'agit d'une valeur indiquant un pourcentage d'erreur de décodage des données reçues. L'acronyme couramment utilisé est **BER** : **Bit Error Rate**. L'objectif est donc d'obtenir ici une valeur stable et nulle.

En plus du niveau audio du signal reçu s'ajoutent des contraintes de stabilité de la base de temps fournie par l'oscillateur câblé sur la carte interface analogique. Comme indiqué dans le fichier **Config.h** modifié lors de la compilation du logiciel du MMDVM STM32F466 effectuée précédemment dans ce document, j'utilise un oscillateur fournissant au STM32 une horloge de 12MHz. Ce signal lui sert à gérer les fenêtres temporelles des signaux numériques, dont celle des « Time Slots » en DMR. Pour rappel, un Time Slot est une portion de temps pendant laquelle une émission va avoir lieu, ce qui permet de diffuser plusieurs émissions simultanément sur une même fréquence. La moindre dérive de la base de temps créée par l'horloge 12MHz rendrait le MMDVM incapable de savoir quand traiter le Time Slot 1, et le Time Slot 2 en DMR.

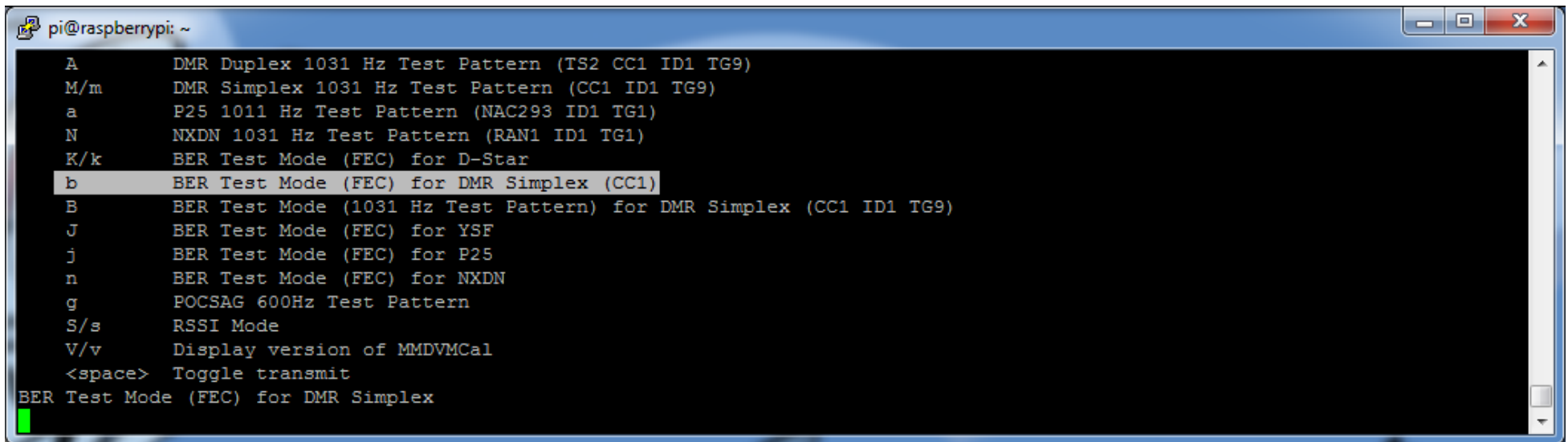
C'est pour cette raison qu'il faut impérativement utiliser une source d'horloge la plus stable possible, ce qui exclut l'utilisation d'un simple oscillateur. L'utilisation d'un TCXO (**T**emperature **C**ontrolled **X**tal **O**scillator / Oscilleteur à quartz compensé en température) assurera ainsi l'absence d'erreur causée par une dérive de fréquence.

Cette contrainte technique ne concerne que les relais utilisés en half-duplex. En DMR simplex (émission et réception sur la même fréquence), il n'y a pas de notion de Time Slot. L'utilisation d'un TCXO ne sera pas nécessaire dans ce cas précis.

Si votre interface est utilisée en half-duplex et n'est pas équipée d'un TCXO, attendez-vous donc à rencontrer des problèmes de stabilité de transmission en DMR.

Pour effectuer le réglage en réception, il faut avant tout paramétrer le poste DMR en simplex. Pour ce faire, il suffit de paramétrer un canal avec la même fréquence de réception et d'émission, identique à celle paramétrée sur le récepteur de votre relais MMDVM. Ainsi, le poste DMR utilisé pour les tests n'utilisera pas de Time Slots, et MMDVMCal pourra décoder correctement le signal émis.

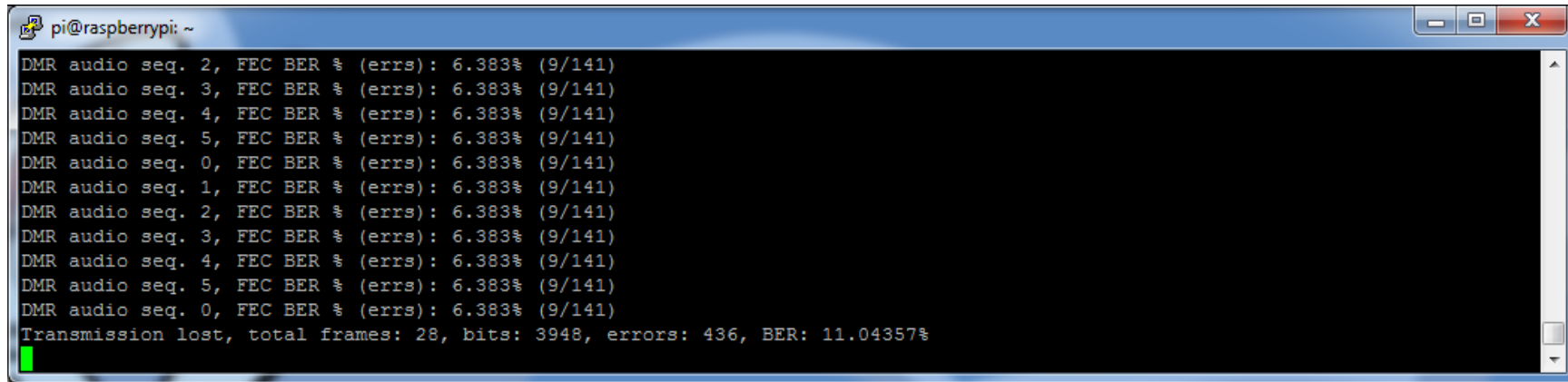
On entre dans ce mode de test en appuyant sur la touche **b** :



```
pi@raspberrypi: ~
A      DMR Duplex 1031 Hz Test Pattern (TS2 CC1 ID1 TG9)
M/m    DMR Simplex 1031 Hz Test Pattern (CC1 ID1 TG9)
a      P25 1011 Hz Test Pattern (NAC293 ID1 TG1)
N      NXDN 1031 Hz Test Pattern (RAN1 ID1 TG1)
K/k    BER Test Mode (FEC) for D-Star
b      BER Test Mode (FEC) for DMR Simplex (CC1)
B      BER Test Mode (1031 Hz Test Pattern) for DMR Simplex (CC1 ID1 TG9)
J      BER Test Mode (FEC) for YSF
j      BER Test Mode (FEC) for P25
n      BER Test Mode (FEC) for NXDN
g      POCSAG 600Hz Test Pattern
S/s    RSSI Mode
V/v    Display version of MMDVMCal
<space> Toggle transmit
BER Test Mode (FEC) for DMR Simplex
```

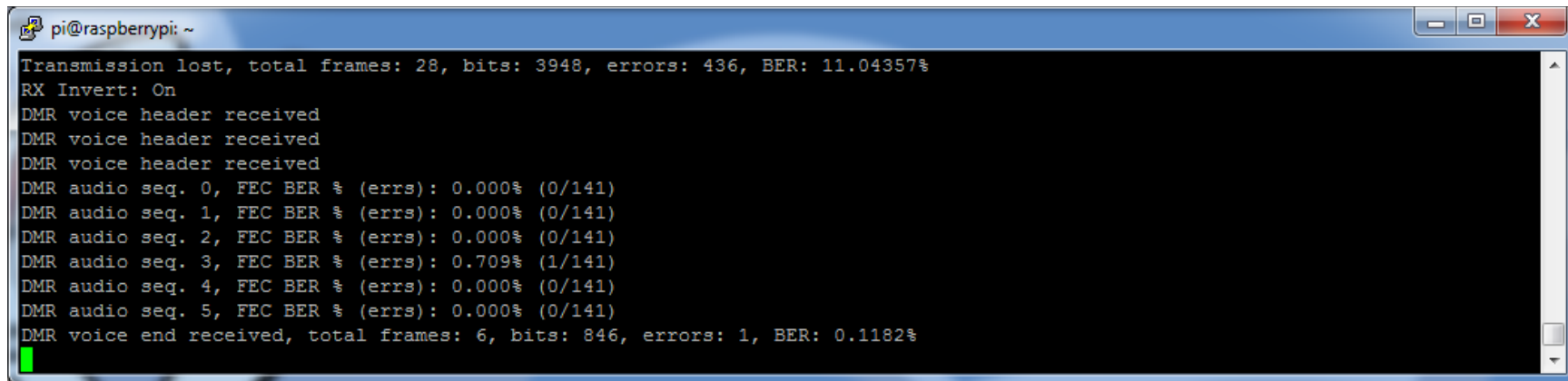
Il faut ensuite appuyer sur le bouton PTT du poste DMR et vérifier la valeur de BER indiquée à la fin de chaque ligne qui s'affiche à l'écran. Durant mes essais, j'ai constaté que l'émission de mon poste DMR (TYT MD380) avec son antenne d'origine et à faible puissance a tout de même tendance à perturber les signaux analogiques et/ou les amplis de l'interface du MMDVM. Pensez à éloigner autant que possible votre poste du MMDVM lors des essais.

Si vous ne voyez les indications de BER s'afficher uniquement lorsque vous relâchez le PTT, cela signifie que vous devez inverser le mode de réception du MMDVM :



```
pi@raspberrypi: ~  
DMR audio seq. 2, FEC BER % (errs): 6.383% (9/141)  
DMR audio seq. 3, FEC BER % (errs): 6.383% (9/141)  
DMR audio seq. 4, FEC BER % (errs): 6.383% (9/141)  
DMR audio seq. 5, FEC BER % (errs): 6.383% (9/141)  
DMR audio seq. 0, FEC BER % (errs): 6.383% (9/141)  
DMR audio seq. 1, FEC BER % (errs): 6.383% (9/141)  
DMR audio seq. 2, FEC BER % (errs): 6.383% (9/141)  
DMR audio seq. 3, FEC BER % (errs): 6.383% (9/141)  
DMR audio seq. 4, FEC BER % (errs): 6.383% (9/141)  
DMR audio seq. 5, FEC BER % (errs): 6.383% (9/141)  
DMR audio seq. 0, FEC BER % (errs): 6.383% (9/141)  
Transmission lost, total frames: 28, bits: 3948, errors: 436, BER: 11.04357%
```

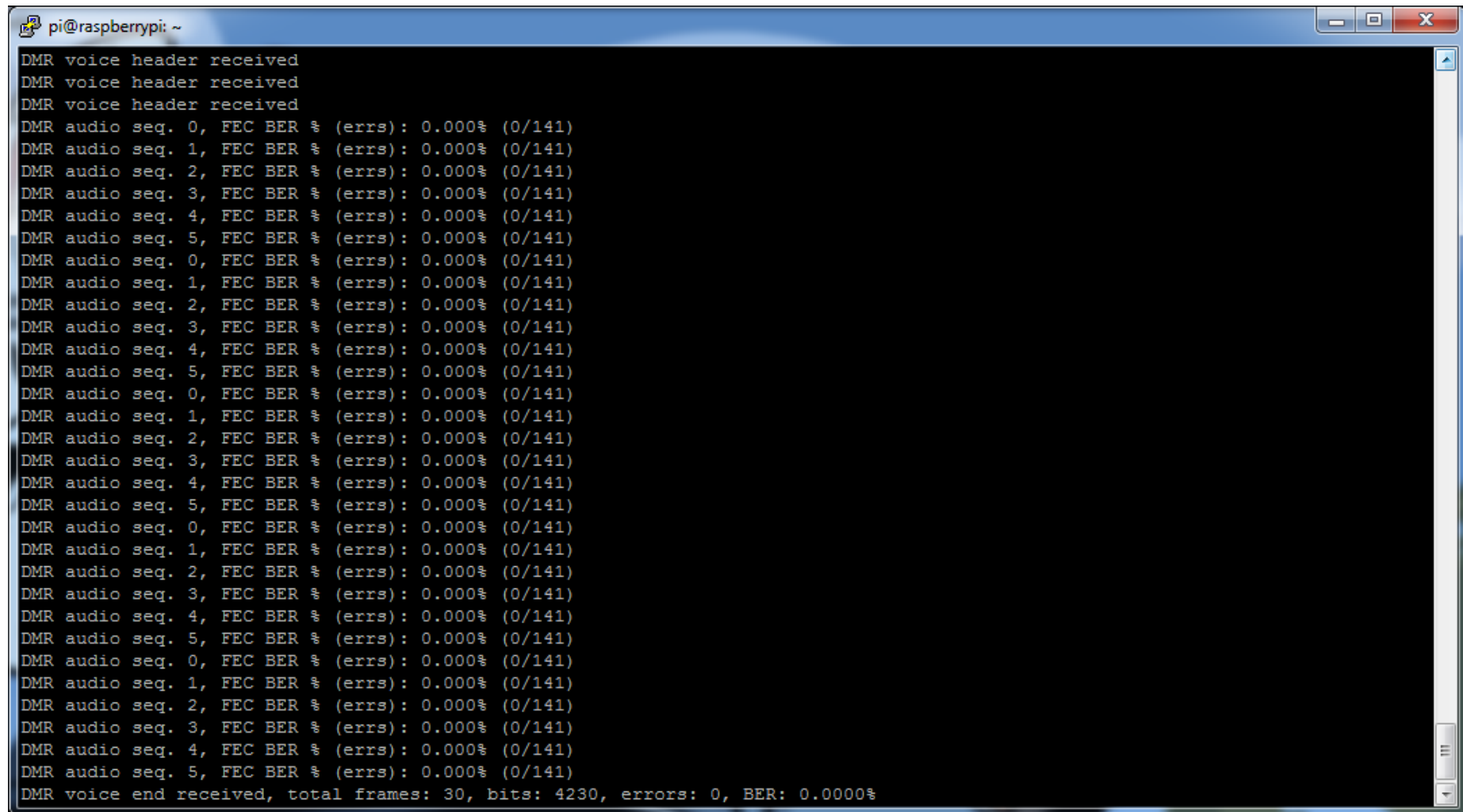
Cas typique de test dans le mauvais mode de réception ci-dessus. Et même test après avoir appuyé sur la touche **i** ci-dessous :



```
pi@raspberrypi: ~  
Transmission lost, total frames: 28, bits: 3948, errors: 436, BER: 11.04357%  
RX Invert: On  
DMR voice header received  
DMR voice header received  
DMR voice header received  
DMR audio seq. 0, FEC BER % (errs): 0.000% (0/141)  
DMR audio seq. 1, FEC BER % (errs): 0.000% (0/141)  
DMR audio seq. 2, FEC BER % (errs): 0.000% (0/141)  
DMR audio seq. 3, FEC BER % (errs): 0.709% (1/141)  
DMR audio seq. 4, FEC BER % (errs): 0.000% (0/141)  
DMR audio seq. 5, FEC BER % (errs): 0.000% (0/141)  
DMR voice end received, total frames: 6, bits: 846, errors: 1, BER: 0.1182%
```

L'indication « **DMR voice header received** » prouve qu'on intercepte bien les données dès l'appui sur le PTT du poste DMR. On reportera ce paramétrage de réception inversés dans le fichier de configuration du MMDVM un peu plus loin dans ce document.

Ensuite, tout en restant en émission, on tourne **R99** jusqu'à obtenir un BER de 0.000% :



```
pi@raspberrypi: ~  
DMR voice header received  
DMR voice header received  
DMR voice header received  
DMR audio seq. 0, FEC BER % (errs): 0.000% (0/141)  
DMR audio seq. 1, FEC BER % (errs): 0.000% (0/141)  
DMR audio seq. 2, FEC BER % (errs): 0.000% (0/141)  
DMR audio seq. 3, FEC BER % (errs): 0.000% (0/141)  
DMR audio seq. 4, FEC BER % (errs): 0.000% (0/141)  
DMR audio seq. 5, FEC BER % (errs): 0.000% (0/141)  
DMR audio seq. 0, FEC BER % (errs): 0.000% (0/141)  
DMR audio seq. 1, FEC BER % (errs): 0.000% (0/141)  
DMR audio seq. 2, FEC BER % (errs): 0.000% (0/141)  
DMR audio seq. 3, FEC BER % (errs): 0.000% (0/141)  
DMR audio seq. 4, FEC BER % (errs): 0.000% (0/141)  
DMR audio seq. 5, FEC BER % (errs): 0.000% (0/141)  
DMR audio seq. 0, FEC BER % (errs): 0.000% (0/141)  
DMR audio seq. 1, FEC BER % (errs): 0.000% (0/141)  
DMR audio seq. 2, FEC BER % (errs): 0.000% (0/141)  
DMR audio seq. 3, FEC BER % (errs): 0.000% (0/141)  
DMR audio seq. 4, FEC BER % (errs): 0.000% (0/141)  
DMR audio seq. 5, FEC BER % (errs): 0.000% (0/141)  
DMR audio seq. 0, FEC BER % (errs): 0.000% (0/141)  
DMR audio seq. 1, FEC BER % (errs): 0.000% (0/141)  
DMR audio seq. 2, FEC BER % (errs): 0.000% (0/141)  
DMR audio seq. 3, FEC BER % (errs): 0.000% (0/141)  
DMR audio seq. 4, FEC BER % (errs): 0.000% (0/141)  
DMR audio seq. 5, FEC BER % (errs): 0.000% (0/141)  
DMR audio seq. 0, FEC BER % (errs): 0.000% (0/141)  
DMR audio seq. 1, FEC BER % (errs): 0.000% (0/141)  
DMR audio seq. 2, FEC BER % (errs): 0.000% (0/141)  
DMR audio seq. 3, FEC BER % (errs): 0.000% (0/141)  
DMR audio seq. 4, FEC BER % (errs): 0.000% (0/141)  
DMR audio seq. 5, FEC BER % (errs): 0.000% (0/141)  
DMR voice end received, total frames: 30, bits: 4230, errors: 0, BER: 0.0000%
```

Paramétrage de MMDVMHost

Le principe du MMDVM, c'est de pouvoir servir de relais radio pour différents modes de transmission, numériques ou analogique. Cela n'implique pas pour autant que tous ces modes seront activés par défaut. Il faut donc définir entre autre, dans le fichier de configuration du logiciel principal, les modes qui seront gérés par le relais MMDVM. À cela s'ajoutent les paramètres relatifs à la balise d'identification du relais en code morse, les niveaux audio en émission pour chaque mode, le paramétrage des différents périphériques de la carte MMDVM et la connexion éventuelle aux passerelles d'interconnexion des relais par l'intermédiaire de serveurs Internet.

L'accès en édition au fichier de configuration est effectué avec la commande suivante :
sudo nano /opt/MMDVMHost/MMDVM.ini

Ci-dessous une liste non exhaustive des paramètres, avec quelques explications.

Paramètres généraux

[General] Callsign=Fxyyy Id=208aabbbb Timeout=180 Duplex=1 # ModeHang=10 RFModeHang=10 NetModeHang=3 Display=Nextion Daemon=0	Indicatif du relais, affiché sur l'écran relié au MMDVM Identifiant CCS7 du relais, affiché sur l'écran relié au MMDVM Fonctionnement en half-duplex (relais), 0 = simplex (point d'accès) Type d'afficheur Exécution de MMDVMHost en tant que démon (pas utilisé, exécuté en tant que service grâce au paramétrage détaillé plus loin dans ce document)
--	--

Informations sur le MMDVM

<pre>[Info] RXFrequency=439987500 TXFrequency=430587500 Power=5 # The following lines are only needed if a direct connection to a DMR master is being used Latitude=0.00 Longitude=0.00 Height=10 Location=JN12BL Description=Relais MMDVM URL=www.shibby.fr</pre>	Données complémentaires communiquées aux serveurs d'interconnexion Fréquence de réception Fréquence d'émission Puissance d'émission (P.A.R.) Coordonnée GPS : Latitude Coordonnée GPS : Longitude Hauteur de l'antenne par rapport au niveau de la mer Locator Description de l'installation radio Site Internet en lien avec le relais
--	--

Journaux d'événements

<pre>[Log] # Logging levels, 0=No logging DisplayLevel=1 FileLevel=1 FilePath=. FileRoot=MMDVM FileRotate=1</pre>	
---	--

Balise d'identification en code morse

[CW Id] Enable=1 Time=10 Callsign=fxyyy locator 73	1 = activ��, 0 = d��sactiv�� D��lai en minutes entre chaque ��mission de la balise Message �� ��mettre
---	--

Fichiers de correspondance entre les identifiants (ex. : CCS7) et les indicatifs

[DMR Id Lookup] File=DMRIds.dat Time=24 [NXDN Id Lookup] File=NXDN.csv Time=24	
---	--

Param  trage du modem MMDVM

[Modem] Port=/dev/ttyACM0 # Port=/dev/ttyAMA0 # Port=\\.COM4 Protocol=uart # Address=0x22 TXInvert=1 RXInvert=1 PTTInvert=0 TXDelay=100	Interface de communication Raspberry Pi �� MMDVM Protocole de communication 1 = Inversion de l'��mission 1 = Inversion de la r��ception 1 = Inversion de la commande PTT, 0 = pas d'inversion
--	---

<code>RXOffset=0 TXOffset=0 DMRDelay=83 RXLevel=50 TXLevel=50 RXDCOffset=0 TXDCOffset=0 RFLevel=100 CWIdTXLevel=50 # D-StarTXLevel=50 DMRTXLevel=50 # YSFTXLevel=50 # P25TXLevel=50 # NXDNTXLevel=50 POCSAGTXLevel=18.5 FMTXLevel=43 RSSIMappingFile=RSSI.dat UseCOSAsLockout=0 Trace=0 Debug=0</code>	Niveau audio spécifique à l'émission de la balise CW Niveau audio spécifique à l'émission DMR Niveau audio spécifique à l'émission DAPNET Niveau audio spécifique à l'émission analogique
--	---

Paramétrage DMR « local »

<code>[DMR] Enable=1 Beacons=0 BeaconInterval=60 BeaconDuration=3 ColorCode=1 SelfOnly=0</code>	1 = Relayage des émissions DMR activé, 0 = désactivé
---	---

EmbeddedLCOnly=0 DumpTADData=1 # Prefixes=234,235 # Slot1TGWhiteList= # Slot2TGWhiteList= CallHang=3 TXHang=4 # ModeHang=10 # OVCM Values, 0=off, 1=rx_on, 2=tx_on, 3=both_on, 4=force off # OVCM=0	
---	--

Paramétrage général DAPNET

[POCSAG] Enable=1 Frequency=439987500	1 = Activé Fréquence d'émission
---	------------------------------------

Paramétrage général FM (relais analogique)

[FM] Enable=0 Callsign=fxyyy locator 73 CallsignSpeed=15 CallsignFrequency=820 CallsignTime=10 CallsignHoldoff=0 CallsignHighLevel=50 CallsignLowLevel=20	1 = activé, 0 = désactivé Indicatif émis par la balise CW Nombre de mots par minute Fréquence de la tonalité CW Délai en minutes entre chaque émission de la balise Niveau audio de la balise en l'absence d'émission FM Niveau audio si une émission phonie est en cours
---	---

<code>CallsignAtStart=1</code> <code>CallsignAtEnd=1</code> <code>CallsignAtLatch=0</code> <code>RFAck=K</code> <code>ExtAck=N</code> <code>AckSpeed=20</code> <code>AckFrequency=1750</code> <code>AckMinTime=4</code> <code>AckDelay=1000</code> <code>AckLevel=50</code> <code># Timeout=180</code> <code>TimeoutLevel=80</code> <code>CTCSSFrequency=88.4</code> <code>CTCSSThreshold=30</code> <code># CTCSSHighThreshold=30</code> <code># CTCSSLowThreshold=20</code> <code>CTCSSLevel=20</code> <code>KerchunkTime=0</code> <code>HangTime=7</code> <code>AccessMode=1</code> <code>COSInvert=0</code> <code>RFAudioBoost=1</code> <code>MaxDevLevel=90</code> <code>ExtAudioBoost=1</code>	<code>Émission de la balise au déclenchement du relais</code> <code>Émission de la balise à la fermeture du relais</code> <code>Indication de fin de prise de parole « K »</code> <code>Nombre de mots par minutes du « K »</code> <code>Fréquence de la tonalité du « K »</code> <code>Durée min d'une émission en secondes suivie d'un « K »</code> <code>Délai en millisecondes entre la fin d'émission et le K</code> <code>Niveau audio du « K »</code> <code>Fréquence de la tonalité subaudible « CTCSS »</code> <code>Seuil de détection du CTCSS</code> <code>Seuil haut d'hystérésis de détection du CTCSS</code> <code>Seuil bas d'hystérésis de détection du CTCSS</code> <code>Délai avant ouverture du relais</code>
--	--

Paramétrage d'accès à la passerelle DMR (DMRGateway)

<code>[DMR Network]</code> <code>Enable=0</code> <code># Type may be either 'Direct' or 'Gateway'. When</code>	<code>1 = activé, 0 = désactivé</code>
--	--

```
Direct you must provide the Master's
# address as well as the Password, and for DMR+,
Options also.
Type=Gateway
Address=127.0.0.1
Port=62031
Local=62032
Password=P@ssw0rd
Jitter=360
Slot1=1
Slot2=1
# Options=
# ModeHang=3
Debug=0
```

Laisser tel quel
Adresse du DMRGateway : 127.0.0.1 = local
Port distant, laisser tel quel
Port local, laisser tel quel
Mot de passe d'accès à la passerelle locale

Paramétrage d'accès à la passerelle DAPNET (DAPNETGateway)

```
[POCSAG Network]
Enable=1
LocalAddress=127.0.0.1
LocalPort=3800
GatewayAddress=127.0.0.1
GatewayPort=4800
# ModeHang=3
Debug=1
```

Paramétrage de l'afficheur Nextion

```
[Nextion]
```

Port=modem #Port=/dev/ttyAMA0 Brightness=30 DisplayClock=1 UTC=0 #Screen Layout: 0=G4KLX 2=ON7LDS ScreenLayout=0 IdleBrightness=20	Port sur lequel est relié l'afficheur, modem = connecteur dédié sur l'interface MMDVM Luminosité quand le MMDVM est actif (TX en cours) Affichage de l'horloge : 1 = activé, 0 = désactivé Affichage de l'heure UTC : 1 = oui, 0 = non Mode d'affichage, selon la source du fichier chargé dans l'écran Nextion avec Nextion.py Luminosité de l'écran lorsque le relais est inactif
---	---

Il faut noter que la journalisation des événements servant au « debug » crée des fichiers pouvant devenir volumineux, selon le niveau de détail des événements enregistrés. La carte mémoire peut être saturée par ces fichiers, ce qui causera de nombreux dysfonctionnements du système Linux lui-même, et rendra le MMDVM inopérant. Lorsque la présence de ces fichiers n'est plus nécessaire, soit une fois que le MMDVM est stable, pensez à supprimer les fichiers se terminant par l'extension .log présents dans les répertoires d'installation des différents logiciels constituant le MMDVM, et à désactiver la création de ces fichiers en remplaçant le paramétrage Debug=1 par Debug=0 partout où il apparaît dans les fichiers de configuration mentionnés dans ce document.

Exécution automatique des logiciels au démarrage de la Raspberry Pi

Ici nous allons nous intéresser au démarrage automatisé des services liés au MMDVM, afin que cet équipement puisse fonctionner de façon autonome tant que son alimentation électrique est présente, ainsi qu'après une perte temporaire de sa source d'alimentation électrique.

Certains logiciels ne vous seront pas forcément nécessaires, selon le type de relais que vous souhaitez mettre en place.

MMDVMHost constitue la base du MMDVM, ce qui rend son exécution obligatoire.

DMRGateway permet le lien entre votre relais MMDVM DMR et un serveur spécifique permettant l'interconnexion avec d'autres relais par un lien Internet.

DAPNETGateway réalise aussi un lien entre votre relais MMDVM DAPNET et un serveur spécifique permettant la diffusion des messages transitant par celui-ci. Ceci permet par exemple de diffuser une info vers des destinataires quel que soit le relais dont ils dépendent.

Pré-requis

Installation du multiplexeur de terminaux « screen »

```
sudo apt-get install screen -y
```


MMDVMHost

sudo nano /lib/systemd/system/mmdvmhost.service

Copier/coller le texte affiché en gras ci-dessous dans le fichier mmdvmhost.service en cours de création :

[Unit]

Description=MMDVM Host Service

After=syslog.target network.target

[Service]

User=root

WorkingDirectory=/opt/MMDVMHost

ExecStart=/usr/bin/screen -S MMDVMHost -D -m /opt/MMDVMHost/MMDVMHost /opt/MMDVMHost/MMDVM.ini

ExecStop=/usr/bin/screen -S MMDVMHost -X quit

Restart=always

RestartSec=10

[Install]

WantedBy=multi-user.target

Quitter l'éditeur de texte en sauvegardant le fichier avec **Ctrl+X**, puis valider l'enregistrement avec la touche **Y** et **Entrée**.

sudo nano /lib/systemd/system/mmdvmhost.timer

Copier/coller le texte affiché en gras ci-dessous dans le fichier mmdvmhost.timer en cours de création :

[Timer]

OnStartupSec=60

[Install]

WantedBy=multi-user.target

Quitter l'éditeur de texte en sauvegardant le fichier avec **Ctrl+X**, puis valider l'enregistrement avec la touche **Y** et **Entrée**.

```
sudo chmod 755 /lib/systemd/system/mmdvmhost.service
sudo chmod 755 /lib/systemd/system/mmdvmhost.timer
sudo ln -s /lib/systemd/system/mmdvmhost.service /etc/systemd/system/mmdvmhost.service
sudo ln -s /lib/systemd/system/mmdvmhost.timer /etc/systemd/system/mmdvmhost.timer
```

Activer l'exécution automatique du timer à chaque mise sous tension de la Raspberry Pi :

```
sudo systemctl enable mmdvmhost.timer
```

Redémarrer le démon pour prendre en compte les nouveaux paramètres :

```
sudo systemctl daemon-reload
```

Fonctions de gestion du service (à mémoriser, il n'est pas utile de les exécuter maintenant) :

```
sudo systemctl start mmdvmhost.service
sudo systemctl restart mmdvmhost.service
sudo systemctl stop mmdvmhost.service
```

Information sur le statut du service (à mémoriser, il n'est pas utile de l'exécuter maintenant) :

```
sudo systemctl status mmdvmhost.service
```

DMRGateway

sudo nano /lib/systemd/system/dmrgateway.service

Copier/coller le texte affiché en gras ci-dessous dans le fichier dmrgateway.service en cours de création :

[Unit]

Description=DAPNET Gateway Service

After=syslog.target network.target

[Service]

User=root

WorkingDirectory=/opt/DMRGateway

ExecStart=/usr/bin/screen -S DMRGateway -D -m /opt/DMRGateway/DMRGateway /opt/DMRGateway/DMRGateway.ini

ExecStop=/usr/bin/screen -S DMRGateway -X quit

Restart=always

RestartSec=10

[Install]

WantedBy=multi-user.target

Quitter l'éditeur de texte en sauvegardant le fichier avec **Ctrl+X**, puis valider l'enregistrement avec la touche **Y** et **Entrée**.

sudo nano /lib/systemd/system/dmrgateway.timer

Copier/coller le texte affiché en gras ci-dessous dans le fichier dmrgateway.timer en cours de création :

[Timer]

OnStartupSec=60

[Install]

WantedBy=multi-user.target

Quitter l'éditeur de texte en sauvegardant le fichier avec **Ctrl+X**, puis valider l'enregistrement avec la touche **Y** et **Entrée**.

```
sudo chmod 755 /lib/systemd/system/dmrgateway.service
sudo chmod 755 /lib/systemd/system/dmrgateway.timer
sudo ln -s /lib/systemd/system/dmrgateway.service /etc/systemd/system/dmrgateway.service
sudo ln -s /lib/systemd/system/dmrgateway.timer /etc/systemd/system/dmrgateway.timer
```

Activer l'exécution automatique du timer à chaque mise sous tension de la Raspberry Pi :

```
sudo systemctl enable dmrgateway.timer
```

Redémarrer le démon pour prendre en compte les nouveaux paramètres :

```
sudo systemctl daemon-reload
```

Fonctions de gestion du service (à mémoriser, il n'est pas utile de les exécuter maintenant) :

```
sudo systemctl start dmrgateway.service
sudo systemctl restart dmrgateway.service
sudo systemctl stop dmrgateway.service
```

Information sur le statut du service (à mémoriser, il n'est pas utile de l'exécuter maintenant) :

```
sudo systemctl status dmrgateway.service
```

DAPNETGateway

sudo nano /lib/systemd/system/dapnetgateway.service

Copier/coller le texte affiché en gras ci-dessous dans le fichier dapnetgateway.service en cours de création :

[Unit]

Description=DAPNET Gateway Service

After=syslog.target network.target

[Service]

User=root

WorkingDirectory=/opt/DAPNETGateway

**ExecStart=/usr/bin/screen -S DAPNETGateway -D -m /opt/DAPNETGateway/DAPNETGateway
/opt/DAPNETGateway/DAPNETGateway.ini**

ExecStop=/usr/bin/screen -S DAPNETGateway -X quit

Restart=always

RestartSec=10

[Install]

WantedBy=multi-user.target

Quitter l'éditeur de texte en sauvegardant le fichier avec **Ctrl+X**, puis valider l'enregistrement avec la touche **Y** et **Entrée**.

sudo nano /lib/systemd/system/dapnetgateway.timer

Copier/coller le texte affiché en gras ci-dessous dans le fichier dapnetgateway.timer en cours de création :

[Timer]

OnStartupSec=60

[Install]

WantedBy=multi-user.target

Quitter l'éditeur de texte en sauvegardant le fichier avec **Ctrl+X**, puis valider l'enregistrement avec la touche **Y** et **Entrée**.

```
sudo chmod 755 /lib/systemd/system/dapnetgateway.service
sudo chmod 755 /lib/systemd/system/dapnetgateway.timer
sudo ln -s /lib/systemd/system/dapnetgateway.service /etc/systemd/system/dapnetgateway.service
sudo ln -s /lib/systemd/system/dapnetgateway.timer /etc/systemd/system/dapnetgateway.timer
```

Activer l'exécution automatique du timer à chaque mise sous tension de la Raspberry Pi :

```
sudo systemctl enable dapnetgateway.timer
```

Redémarrer le démon pour prendre en compte les nouveaux paramètres :

```
sudo systemctl daemon-reload
```

Fonctions de gestion du service (à mémoriser, il n'est pas utile de les exécuter maintenant) :

```
sudo systemctl start dapnetgateway.service
sudo systemctl restart dapnetgateway.service
sudo systemctl stop dapnetgateway.service
```

Information sur le statut du service (à mémoriser, il n'est pas utile de l'exécuter maintenant) :

```
sudo systemctl status dapnetgateway.service
```

Et maintenant...

Après avoir réalisé toutes les étapes décrites ci-dessus, votre MMDVM devrait être en mesure de retransmettre vos premiers essais radio. Quelques modifications et ajustements de paramétrage de MMDVMHost peuvent être encore nécessaires, mais vous êtes proche du but. Ce document est encore en cours d'édition. Vous pouvez tout de même poster vos questions éventuelles sur mon blog en cliquant sur le lien fourni au début de la première page. Ceci m'aidera à rendre ce document le plus complet possible.

Quelques pièges qui peuvent faire perdre du temps

Noms des interfaces USB

Par défaut, la carte **Nucleo STM32F446** se voit affecter `/dev/ttyACM0` pour l'interface de communication série avec la carte Raspberry Pi, et `/dev/sda` pour son espace de stockage. Si la carte **Nucleo STM32F446** est déconnectée puis reconnectée sans redémarrer la carte Raspberry Pi, ces chemins d'accès seront modifiés, ce qui ne permettra plus d'assurer un bon fonctionnement de l'ensemble. Faites donc bien attention à utiliser les chemins par défaut dans les fichiers de configuration. Tenez-en compte aussi lors de vos essais, cela vous fera gagner du temps:)

Débordement de mémoire tampon

Ce phénomène est facilement identifiable, lorsque le MMDVM est en fonctionnement : le message « Waiting » reste affiché sur l'écran **Nextion**.

Il est confirmé en ouvrant le fichier d'événements en cours de validité, dans **/opt/MMDVMHost** :



```
pi@raspberrypi: ~  
pi@raspberrypi:~ $ ls /opt/MMDVMHost/*.log  
/opt/MMDVMHost/MMDVM-2022-06-05.log  
pi@raspberrypi:~ $
```

Ici le fichier contenant les événements de la journée en cours est **MMDVM-2022-06-05.log**. On consultera son contenu avec la commande suivante :

```
cat /opt/MMDVMHost/MMDVM-2022-06-05.log
```

On doit voir un nombre très important de lignes indiquant **MMDVM RX buffer has overflowed**.

Annexe 1 – Ancienne méthode de compilation, sous Windows

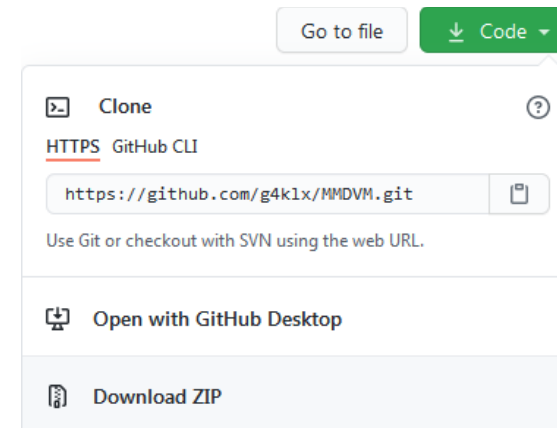
Les informations fournies ci-dessous sont conservées à titre purement informatif. Une nouvelle méthode, beaucoup plus simple, est fournie précédemment dans ce document.

Chargement du projet MMDVM et modification des paramètres de compilation

Aller sur la page Github suivante :

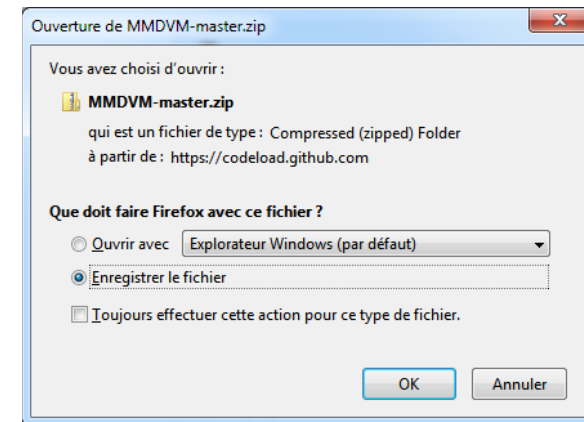
<https://github.com/g4klx/MMDVM>

Cliquer sur le bouton vert en haut à droite :

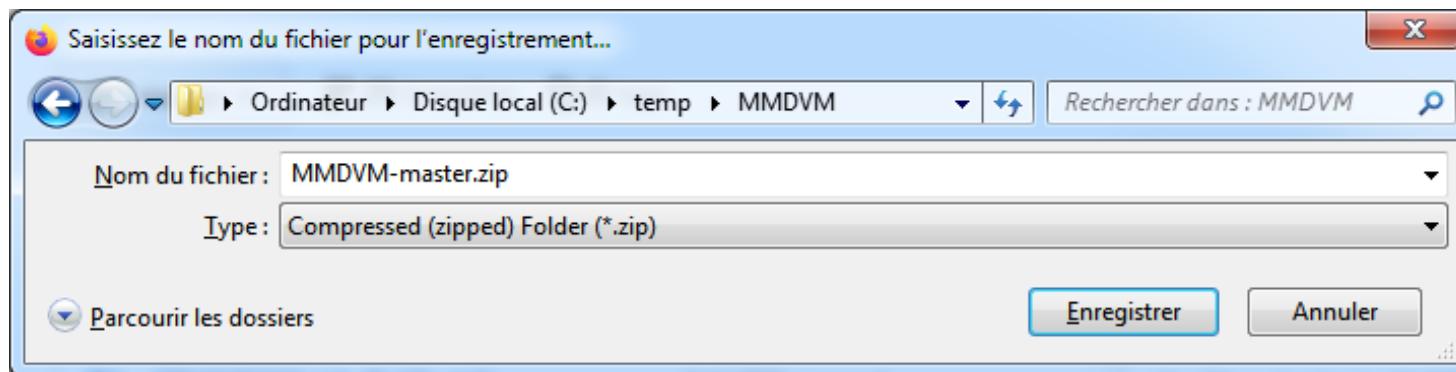


Cliquer sur le sous-menu de téléchargement des fichiers sous format d'archive ZIP : **Download ZIP**.

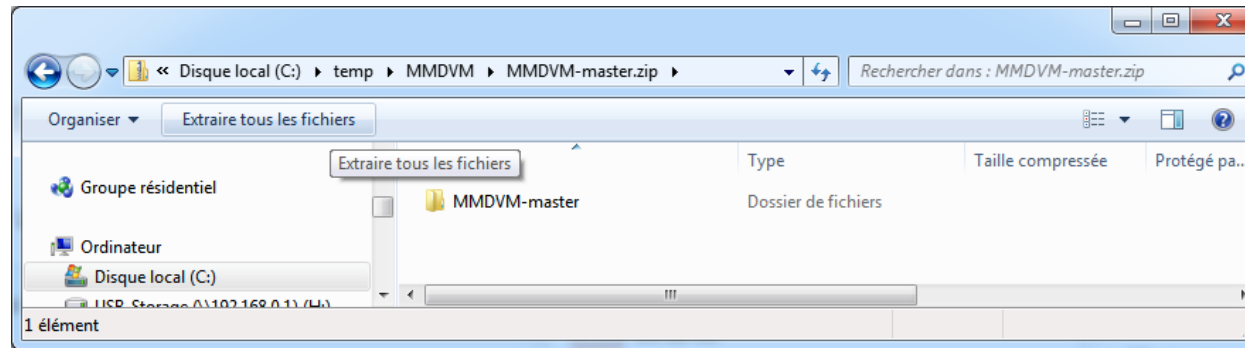
Enregistrer le fichier dans le répertoire souhaité :



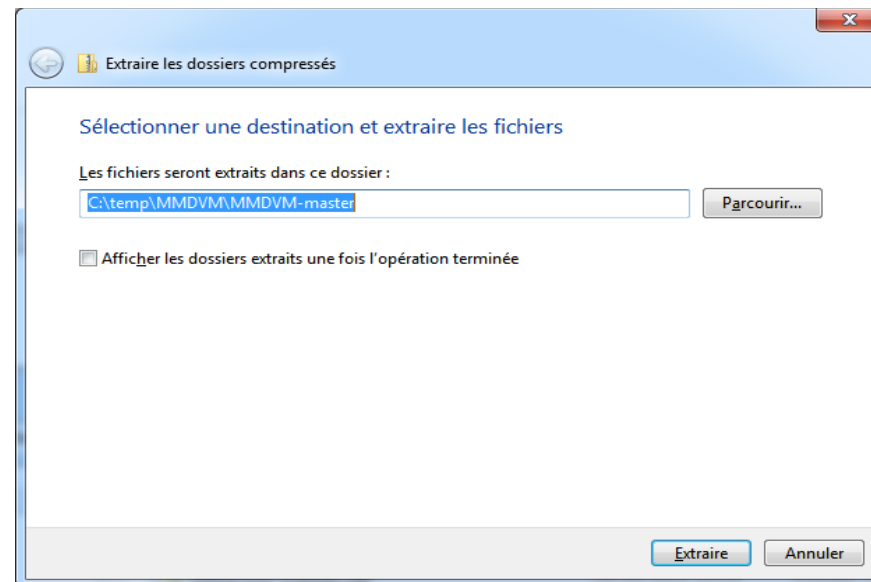
Ici je choisis de l'enregistrer dans **c:\temp\MMDVM** :



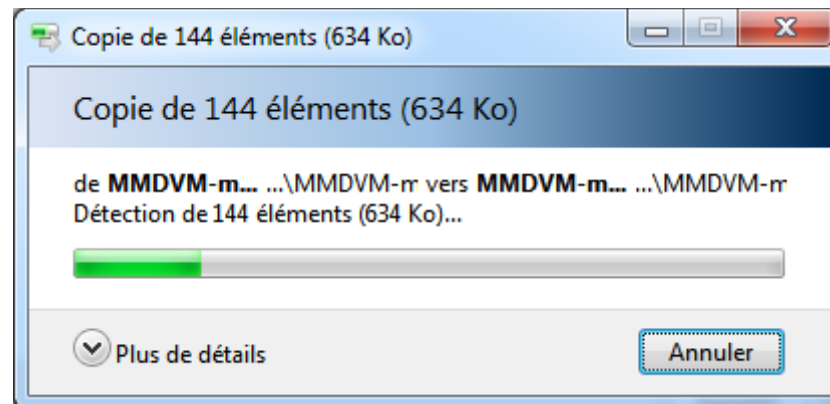
Depuis le répertoire où est enregistré le fichier, double-cliquer sur le fichier et cliquer sur le bouton **Extraire les fichiers** :



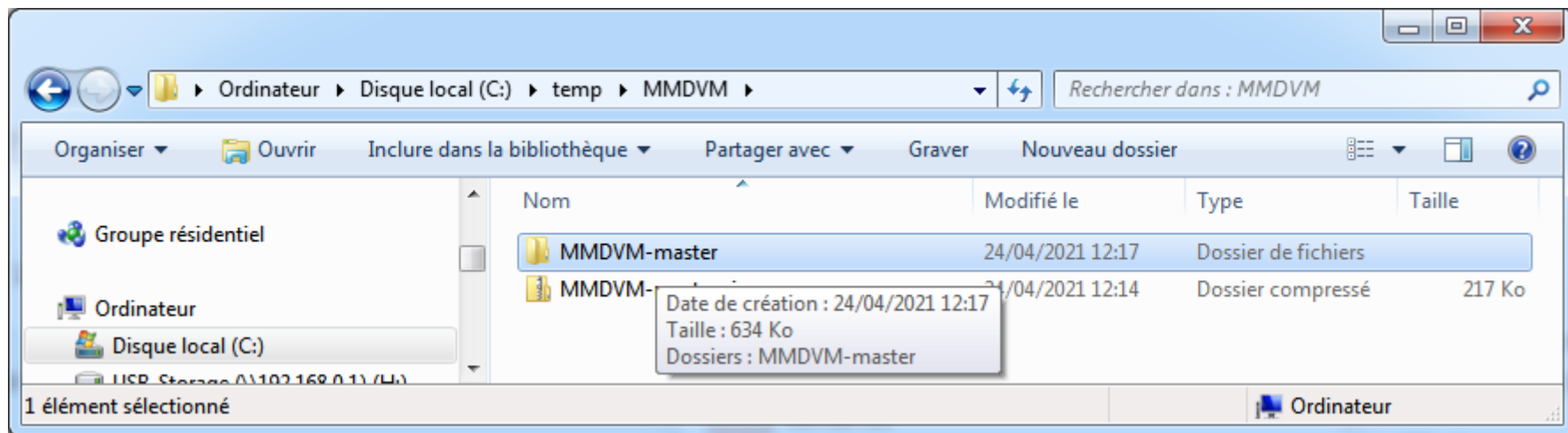
Définir le répertoire dans lequel les fichiers vont être extraits. Je conserve le chemin défini par défaut dans cet exemple :



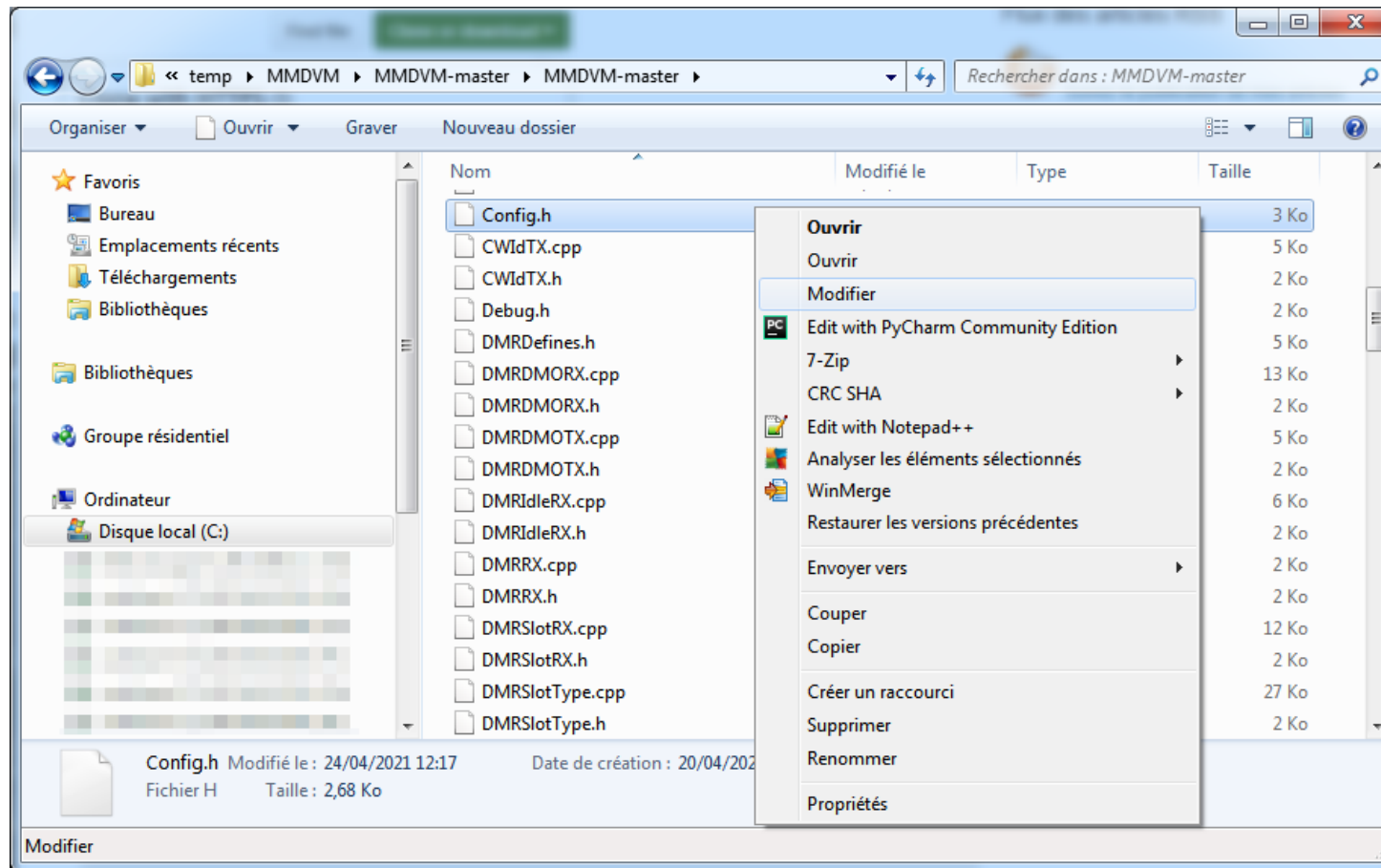
Cliquer sur Extraire. L'extraction est exécutée :



Ce nouveau répertoire apparaît à l'endroit défini à la fin de l'opération :



Accéder au contenu de ce nouveau répertoire pour atteindre le fichier **config.h**, effectuer un clic droit sur le nom du fichier et cliquer sur **Modifier** :



Si vous utilisez l'interface de F5UII sur une carte ST NUCLEO-64 STM32F446, vous devez définir les paramètres suivants :

```
/*  
 * Copyright (C) 2015,2016,2017,2018,2020 by Jonathan Naylor G4KLX  
 *  
 * This program is free software; you can redistribute it and/or modify  
 * it under the terms of the GNU General Public License as published by  
 * the Free Software Foundation; either version 2 of the License, or  
 * (at your option) any later version.  
 *  
 * This program is distributed in the hope that it will be useful,  
 * but WITHOUT ANY WARRANTY; without even the implied warranty of  
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the  
 * GNU General Public License for more details.  
 *  
 * You should have received a copy of the GNU General Public License  
 * along with this program; if not, write to the Free Software  
 * Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.  
 */  
  
#if !defined(CONFIG_H)  
#define CONFIG_H  
  
// Allow for the use of high quality external clock oscillators  
// The number is the frequency of the oscillator in Hertz.  
//  
// The frequency of the TCXO must be an integer multiple of 48000.  
// Frequencies such as 12.0 Mhz (48000 * 250) and 14.4 Mhz (48000 * 300) are suitable.  
// Frequencies such as 10.0 Mhz (48000 * 208.333) or 20 Mhz (48000 * 416.666) are not suitable.
```

```
//  
// For 12 MHz  
#define EXTERNAL_OSC 12000000  
// For 12.288 MHz  
// #define EXTERNAL_OSC 12288000  
// For 14.4 MHz  
// #define EXTERNAL_OSC 14400000  
// For 19.2 MHz  
// #define EXTERNAL_OSC 19200000  
  
// Use pins to output the current mode via LEDs  
#define MODE_LEDS  
  
// For the original Arduino Due pin layout  
// #define ARDUINO_DUE_PAPA  
  
#if defined(STM32F1)  
// For the SQ6POG board  
#define STM32F1_POG  
#else  
// For the ZUM V1.0 and V1.0.1 boards pin layout  
#define ARDUINO_DUE_ZUM_V10  
#endif  
  
// For the SP8NTH board  
// #define ARDUINO_DUE_NTH
```

```
// For ST Nucleo-64 STM32F446RE board  
#define STM32F4_NUCLEO_MORPHO_HEADER  
// #define STM32F4_NUCLEO_ARDUINO_HEADER  
  
// Use separate mode pins to switch external channel/filters/bandwidth for example  
#define MODE_PINS  
  
// For the VK6MST Pi3 Shield communicating over i2c. i2c address & speed defined in i2cTeensy.cpp  
// #define VK6MST_TEENSY_PI3_SHIELD_I2C  
  
// Pass RSSI information to the host  
#define SEND_RSSI_DATA  
  
// Use the modem as a serial repeater for Nextion displays  
#define SERIAL_REPEATER  
  
// To reduce CPU load, you can remove the DC blocker by commenting out the next line  
#define USE_DCBLOCKER  
  
// Constant Service LED once repeater is running  
// Do not use if employing an external hardware watchdog  
// #define CONSTANT_SRV_LED  
  
// Use the YSF and P25 LEDs for NXDN  
#define USE_ALTERNATE_NXDN_LEDS  
  
// Use the D-Star and DMR LEDs for POCSAG
```



```
#define USE_ALTERNATE_POCSAG_LEDS
```

```
// Use the D-Star and YSF LEDs for FM
```

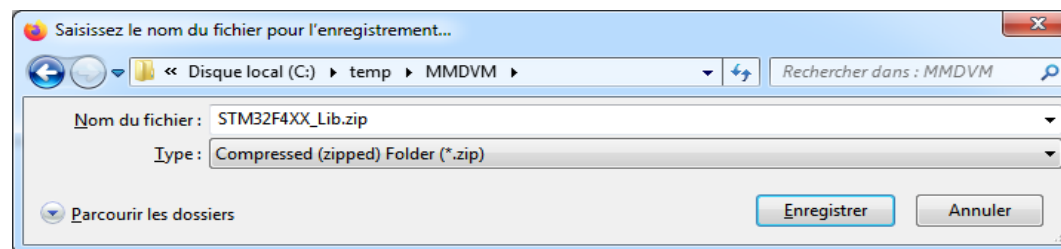
```
#define USE_ALTERNATE_FM_LEDS
```

```
#endif
```

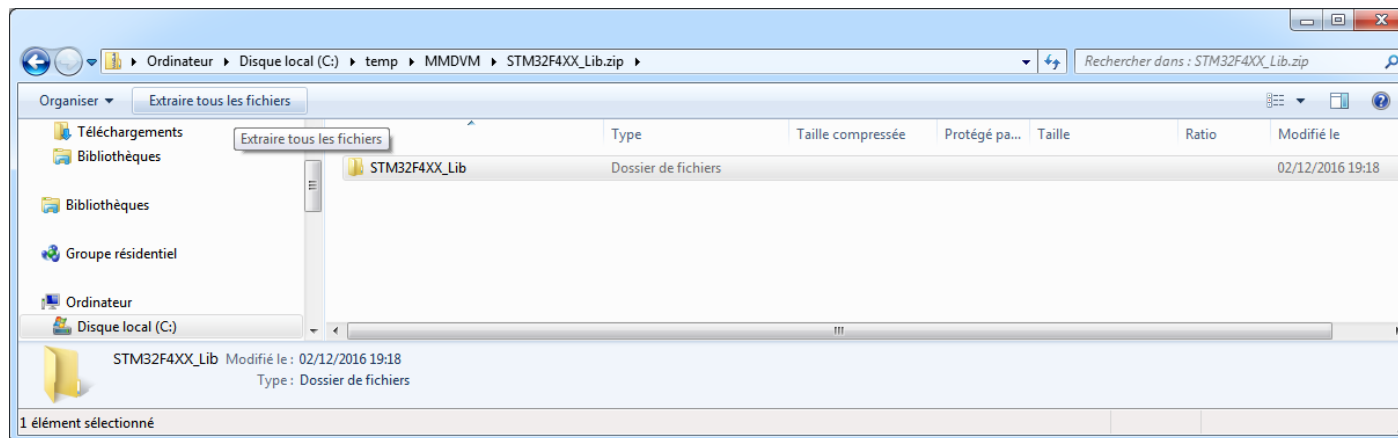
Enregistrez les modifications dans le fichier **Config.h** et quittez l'éditeur. **Librairie STM32F4XX**

Télécharger l'archive ZIP disponible à cette adresse : http://dl.shibby.fr/Radio/MMDVM/STM32F4XX_Lib.zip

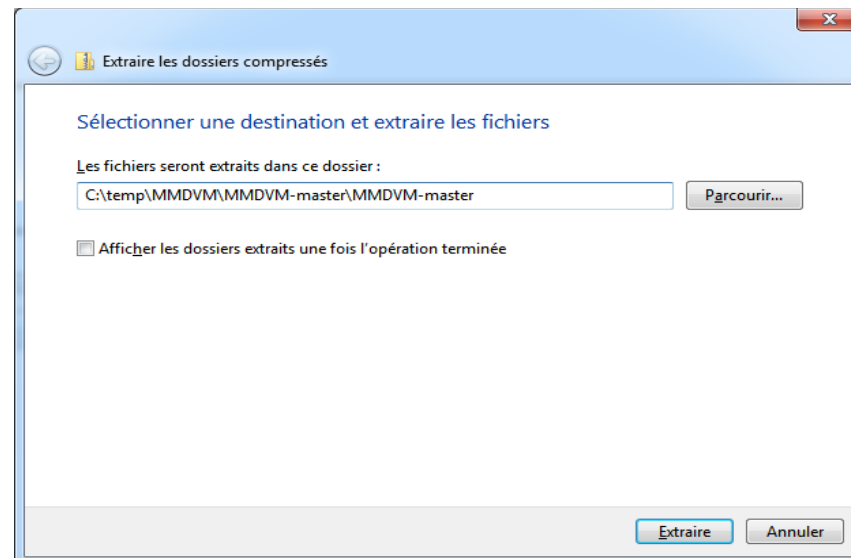
Ici je choisis de l'enregistrer dans **c:\temp\MMDVM** :



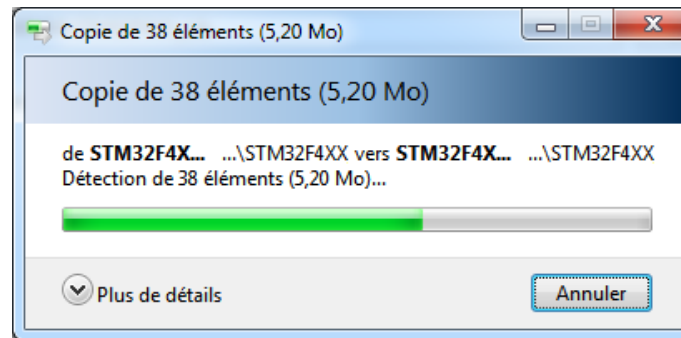
Depuis le répertoire où est enregistré le fichier, double-cliquer sur le fichier et cliquer sur le bouton **Extraire les fichiers** :



Définir le répertoire dans lequel les fichiers vont être extraits. Il faut que le répertoire de destination soit celui qui contient les fichiers MMDVM. Si vous suivez à la lettre cet exemple, il faut définir le chemin suivant : **C:\temp\MMDVM\MMDVM-master\MMDVM-master**



Cliquer sur Extraire. L'extraction est exécutée :

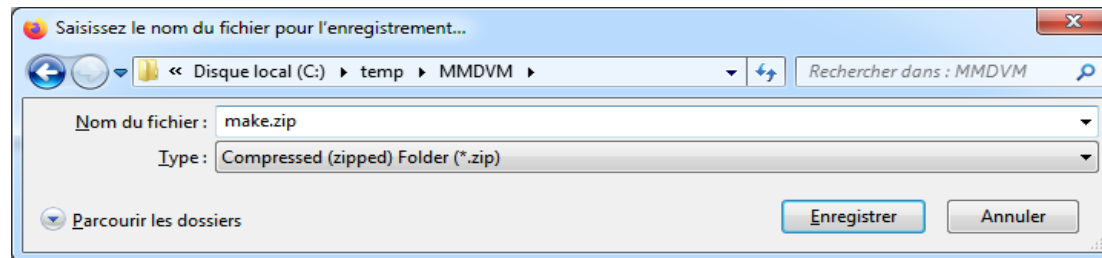


De retour dans le répertoire contenant les fichiers MMDVM, on constate que le répertoire a été créé et contient les fichiers contenus dans l'archive ZIP :

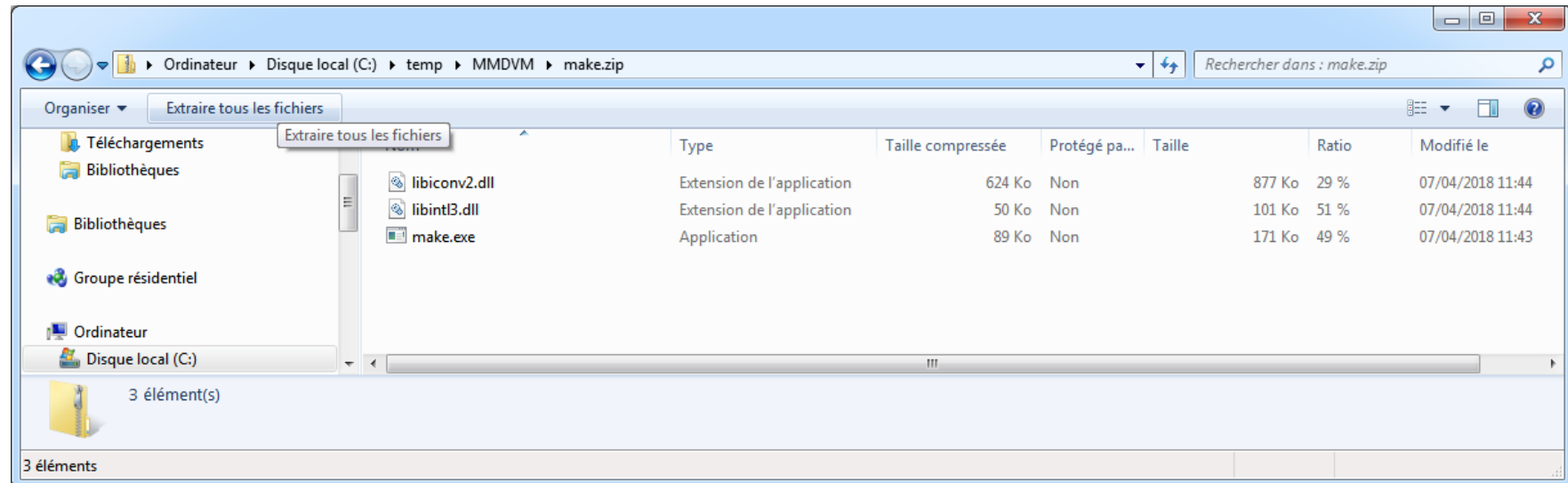
Compilateur gnuwin32

Télécharger l'archive ZIP disponible à cette adresse : <http://dl.shibby.fr/Radio/MMDVM/make.zip>

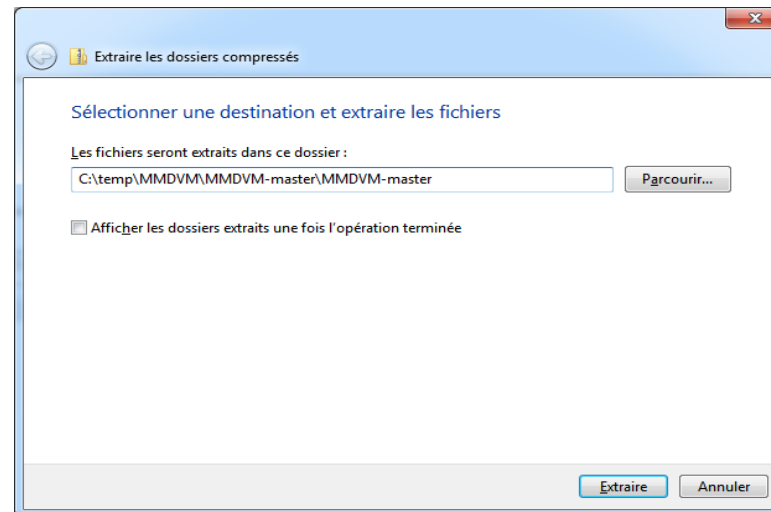
Ici je choisis de l'enregistrer dans **c:\temp\MMDVM** :



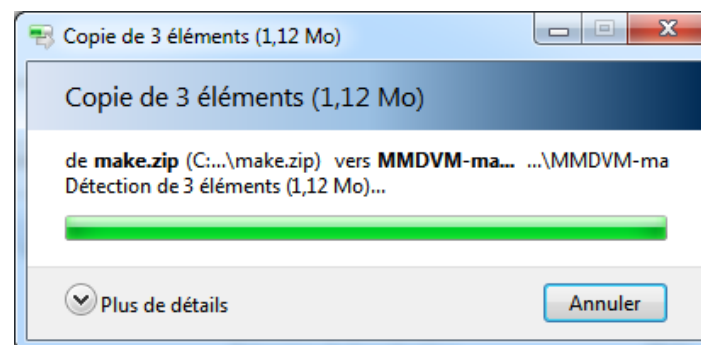
Depuis le répertoire où est enregistré le fichier, double-cliquer sur le fichier et cliquer sur le bouton **Extraire les fichiers** :



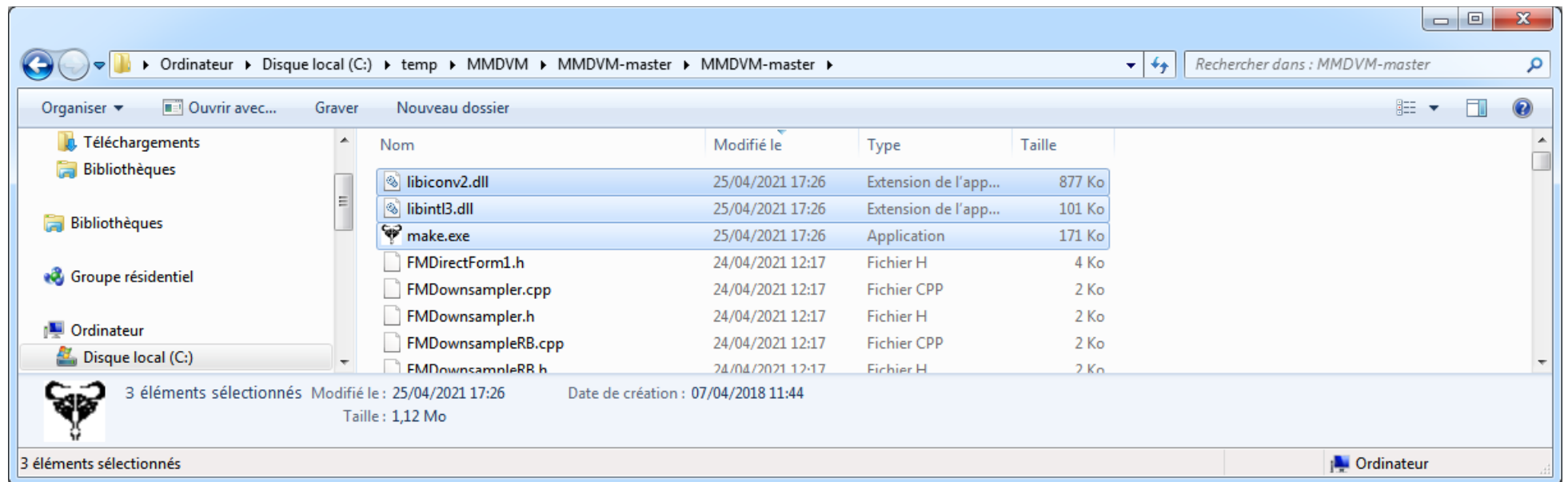
Définir le répertoire dans lequel les fichiers vont être extraits. Il faut que le répertoire de destination soit celui qui contient les fichiers MMDVM. Si vous suivez à la lettre cet exemple, il faut définir le chemin suivant : **C:\temp\MMDVM\MMDVM-master\MMDVM-master**



Cliquer sur Extraire. L'extraction est exécutée :



De retour dans le répertoire contenant les fichiers MMDVM, on constate que les fichiers contenus dans l'archive ZIP ont bien été copiés :

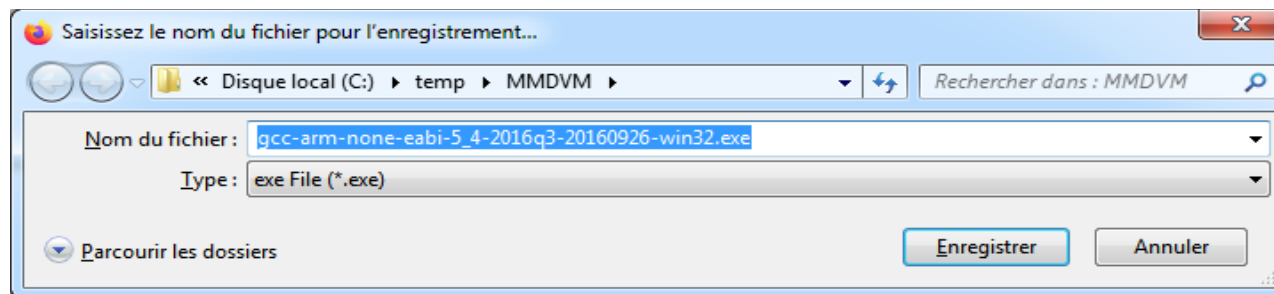


Télécharger l'outil de compilation pour processeur ARM

Cet outil permet de créer un fichier qui pourra être exécuté par le microcontrôleur présent sur la carte STM32F466.

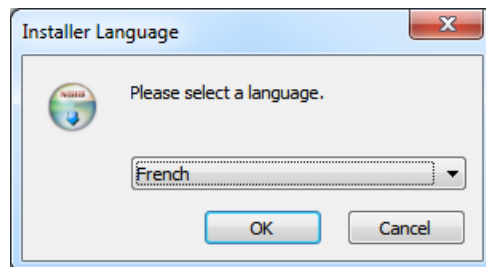
Télécharger l'archive ZIP disponible à cette adresse : http://dl.shibby.fr/Radio/MMDVM/gcc-arm-none-eabi-5_4-2016q3-20160926-win32.exe

Ici je choisis de l'enregistrer dans **c:\temp\MMDVM** :

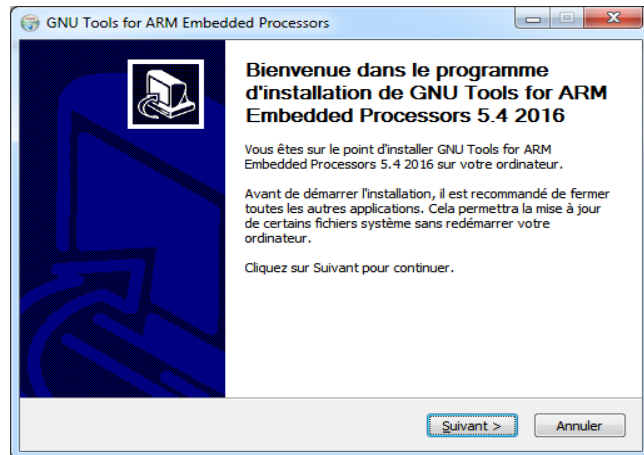


Depuis le répertoire où est enregistré le fichier, double-cliquer sur le fichier et suivre les étapes ci-dessous :

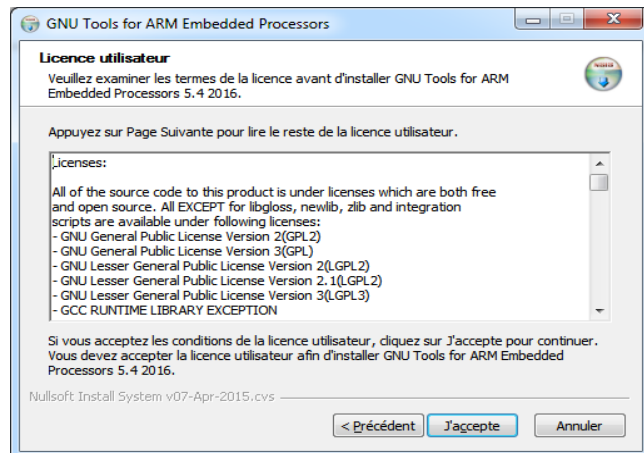
- Valider le langage utilisé par le programme d'installation en cliquant sur **OK** :



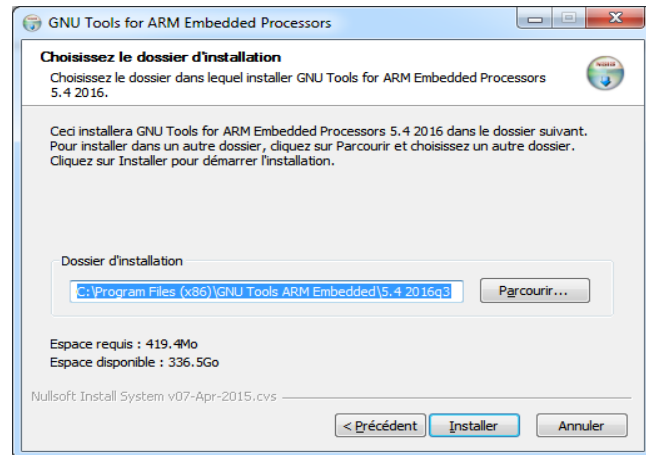
- Cliquer sur **Suivant** :



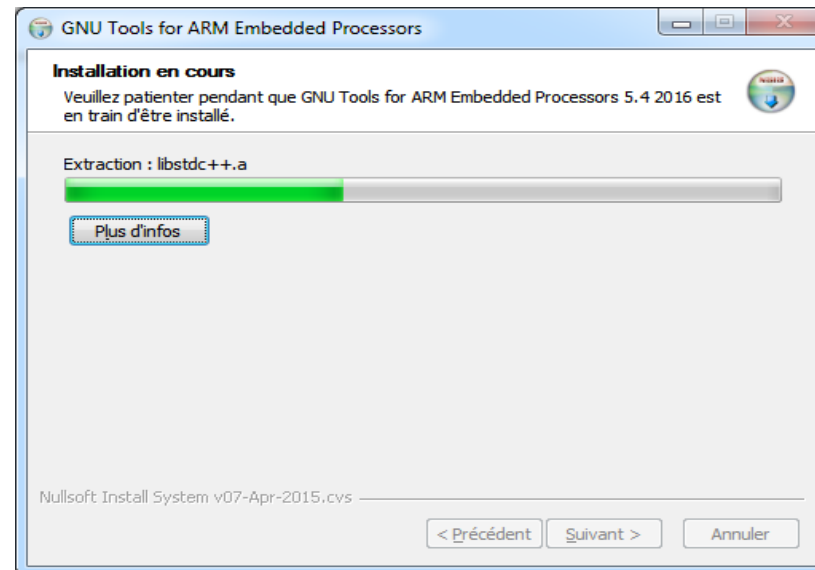
- Accepter la licence utilisateur en cliquant sur **J'accepte** :



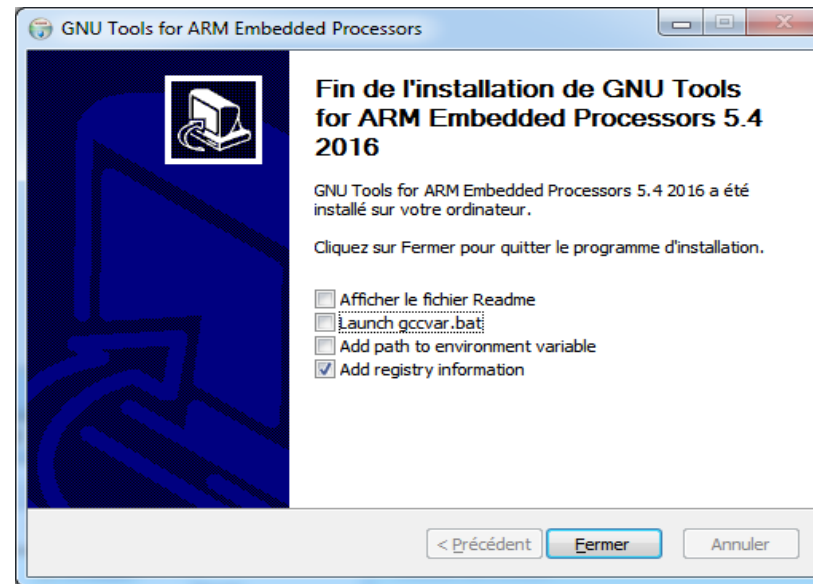
- Laisser le dossier d'installation par défaut et valider l'installation en cliquant sur **Installer** :



L'installation est exécutée. Attendre qu'elle se termine :



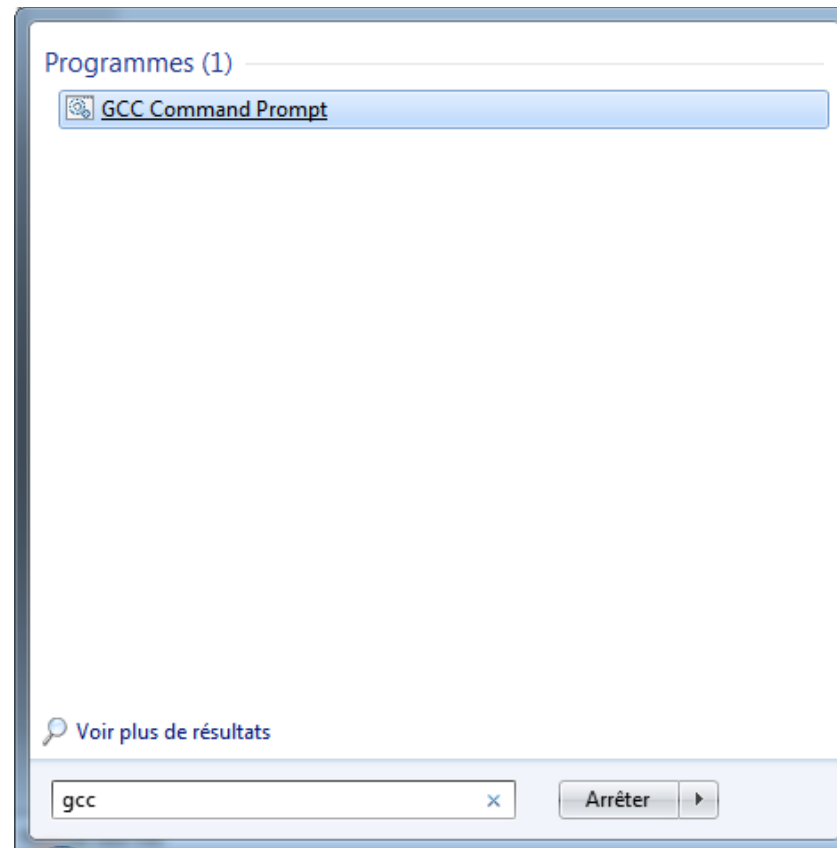
À la fin de l'installation, cocher la case indiquée ci-dessous et cliquer sur **Fermer** :



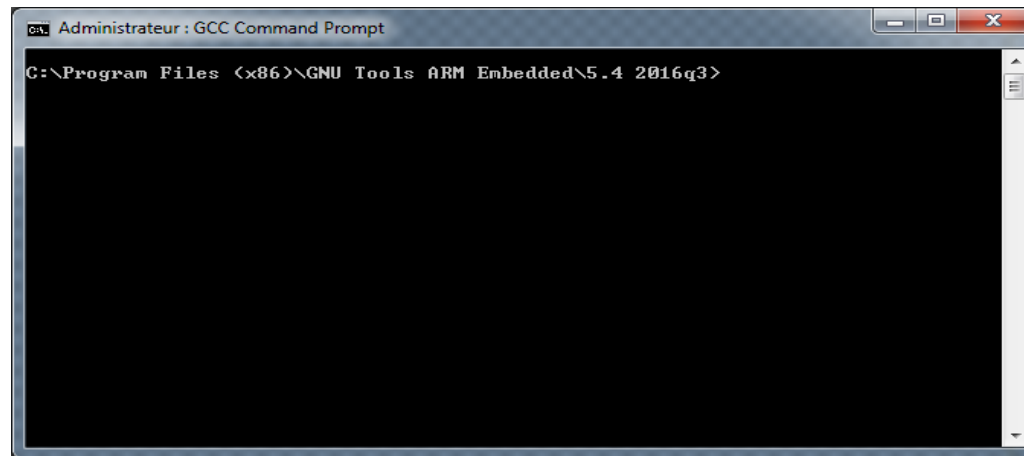
Compilation

On va enfin pouvoir compiler le logiciel embarqué destiné à la carte STM32F446 !

Dans la barre des tâches de Windows, cliquer sur l'icône à gauche (menu « démarrer »), et taper **GCC** :



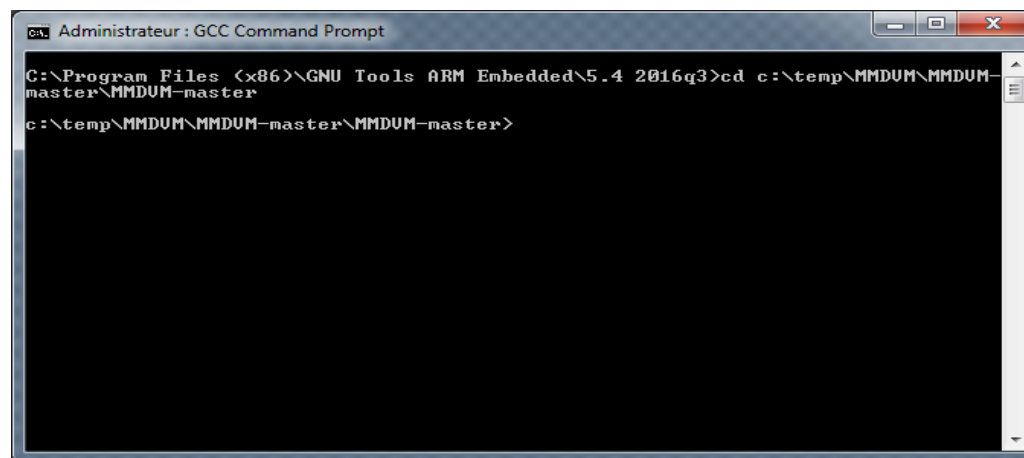
Cliquer sur **GCC Command Prompt**. La fenêtre suivante apparaît à l'écran :



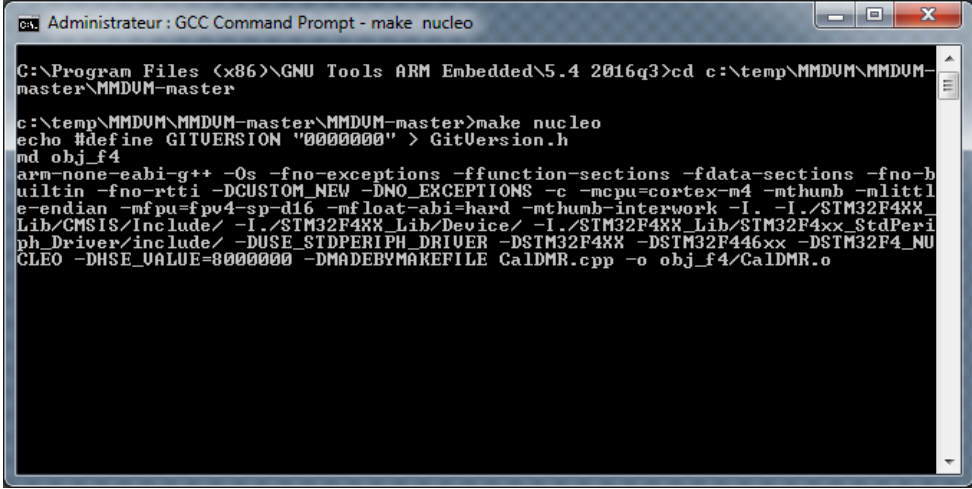
Aller dans le répertoire où se trouvent tous les fichiers MMDVM avec la commande suivante :

cd c:\temp\MMDVM\MMDVM-master\MMDVM-master

Valider avec la touche **Entrée**.



Taper **make nucleo** et valider avec la touche **Entrée**. La compilation des fichiers commence alors :

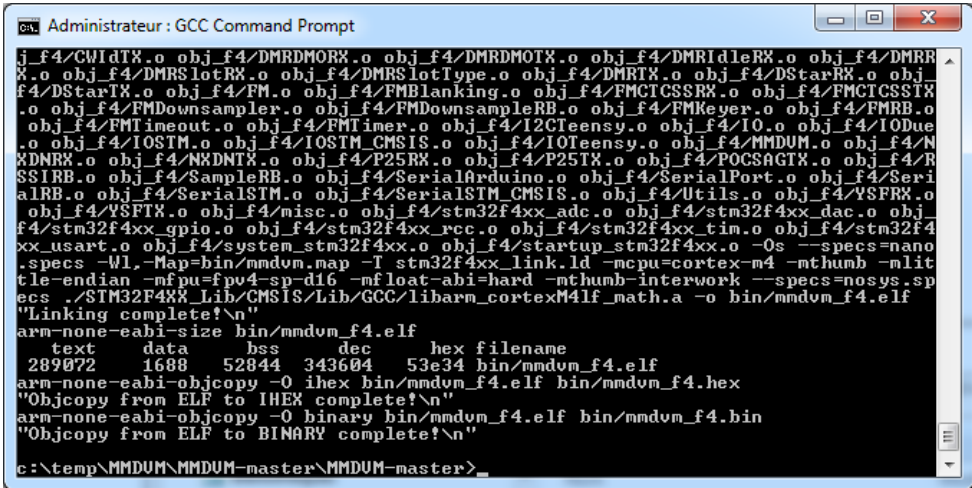


```
Administrateur : GCC Command Prompt - make nucleo

C:\Program Files (x86)\GNU Tools ARM Embedded\5.4 2016q3>cd c:\temp\MMDUM\MMDUM-master\MMDUM-master

c:\temp\MMDUM\MMDUM-master\MMDUM-master>make nucleo
echo #define GITVERSION "0000000" > GitVersion.h
md obj_f4
arm-none-eabi-g++ -Os -fno-exceptions -ffunction-sections -fdata-sections -fno-b
uiltin -fno-rtti -DCUSTOM_NEW -DNO_EXCEPTIONS -c -mcpu=cortex-m4 -mthumb -mlittl
e-endian -mfpv4-sp-d16 -mfloat-abi=hard -mthumb-interwork -I. -I./STM32F4XX_
Lib/CMSIS/Include/ -I./STM32F4XX_Lib/Device/ -I./STM32F4XX_Lib/STM32F4xx_StdPeri
ph_Driver/include/ -DUSE_STDPERIPH_DRIVER -DSTM32F4XX -DSTM32F446xx -DSTM32F4_NU
CLEO -DHSE_VALUE=8000000 -DMADEBYMAKEFILE CalDMR.cpp -o obj_f4/CalDMR.o
```

À la fin de cette opération, vous devez voir le message suivant s'afficher :



```
Administrateur : GCC Command Prompt

obj_f4/CWidTX.o obj_f4/DMRDMORX.o obj_f4/DMRDMOTX.o obj_f4/DMRidleRX.o obj_f4/DMRR
X.o obj_f4/DMRSlotRX.o obj_f4/DMRSlotType.o obj_f4/DMRTX.o obj_f4/DStarRX.o obj_
f4/DStarTX.o obj_f4/FM.o obj_f4/FMBlanking.o obj_f4/FMCTCSSRX.o obj_f4/FMCTCSSTX
.o obj_f4/FMDownsampler.o obj_f4/FMDownsampleRB.o obj_f4/FMKeyer.o obj_f4/FMRB.o
obj_f4/FMTimeout.o obj_f4/FMTimer.o obj_f4/I2CTeensy.o obj_f4/IO.o obj_f4/IODue
.o obj_f4/IOSTM.o obj_f4/IOSTM_CMSIS.o obj_f4/IOTeensy.o obj_f4/MMDUM.o obj_f4/N
XDNRX.o obj_f4/NXDNTX.o obj_f4/P25RX.o obj_f4/P25TX.o obj_f4/POCSAGTX.o obj_f4/R
SSIRB.o obj_f4/SampleRB.o obj_f4/SerialArduino.o obj_f4/SerialPort.o obj_f4/Seri
alRB.o obj_f4/SerialSTM.o obj_f4/SerialSTM_CMSIS.o obj_f4/Utils.o obj_f4/V5FRX.o
obj_f4/V5FTX.o obj_f4/misc.o obj_f4/stm32f4xx_adc.o obj_f4/stm32f4xx_dac.o obj_
f4/stm32f4xx_gpio.o obj_f4/stm32f4xx_rcc.o obj_f4/stm32f4xx_tim.o obj_f4/stm32f4
xx_usart.o obj_f4/system_stm32f4xx.o obj_f4/startup_stm32f4xx.o -Os -specs=nano
.specs -Wl,-Map=bin/mmdvm.map -I stm32f4xx_link.ld -mcpu=cortex-m4 -mthumb -mlit
tle-endian -mfpv4-sp-d16 -mfloat-abi=hard -mthumb-interwork -specs=nosys.sp
ecs ./STM32F4XX_Lib/CMSIS/Lib/GCC/libarm_cortexM4lf_math.a -o bin/mmdvm_f4.elf
"Linking complete!\n"
arm-none-eabi-size bin/mmdvm_f4.elf
text      data      bss      dec      hex filename
289072    1688     52844    343604    53e34 bin/mmdvm_f4.elf
arm-none-eabi-objcopy -O ihex bin/mmdvm_f4.elf bin/mmdvm_f4.hex
"Objcopy from ELF to IHEX complete!\n"
arm-none-eabi-objcopy -O binary bin/mmdvm_f4.elf bin/mmdvm_f4.bin
"Objcopy from ELF to BINARY complete!\n"
c:\temp\MMDUM\MMDUM-master\MMDUM-master>
```

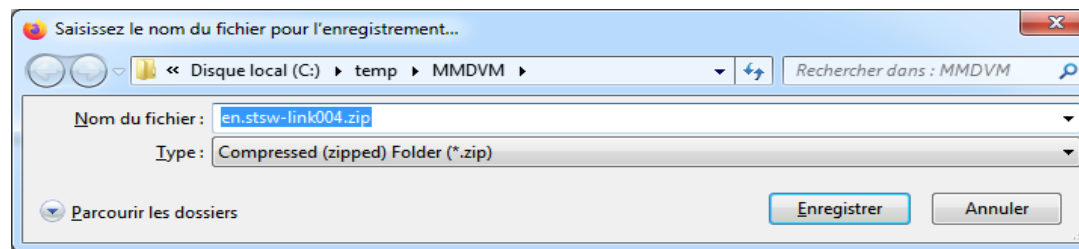
On voit que les fichiers compilés **mmdvm_f4.hex** et **mmdvm_f4.bin** ont été créés. Il s'agit du même fichier compilé, mais dans deux formats différents.

Programmation de la carte STM32F466

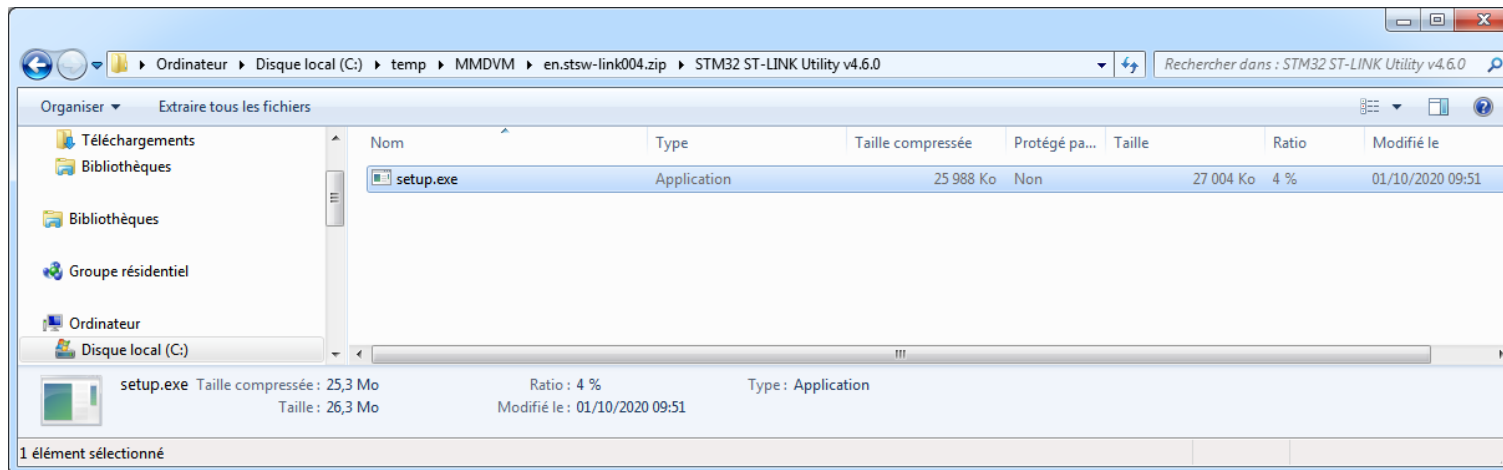
Installation du logiciel de programmation

Télécharger l'archive ZIP disponible à cette : <http://dl.shibby.fr/Radio/MMDVM/en.stsw-link004.zip>

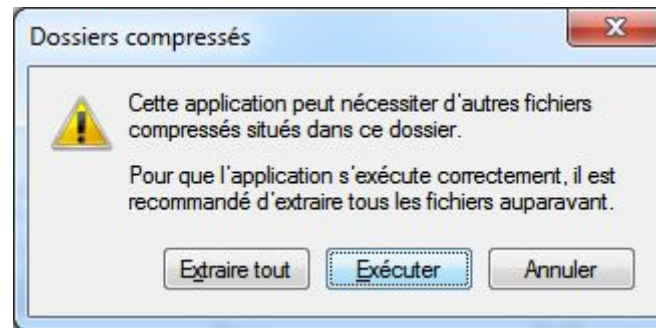
Ici je choisis de l'enregistrer dans **c:\temp\MMDVM** :



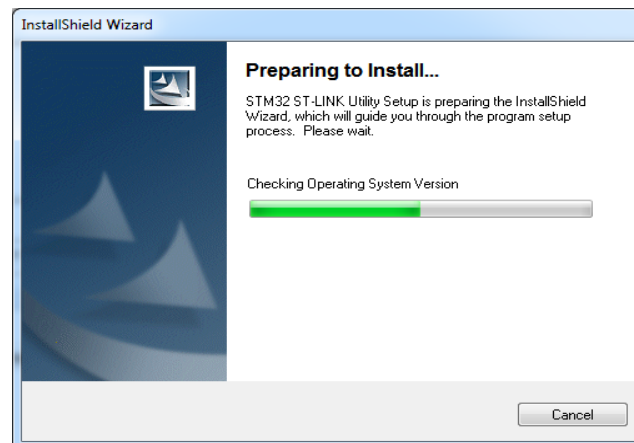
Depuis le répertoire où est enregistré le fichier, double-cliquer sur le fichier, puis sur le répertoire, puis sur le fichier **setup.exe**



Cliquer ensuite sur **Exécuter** :

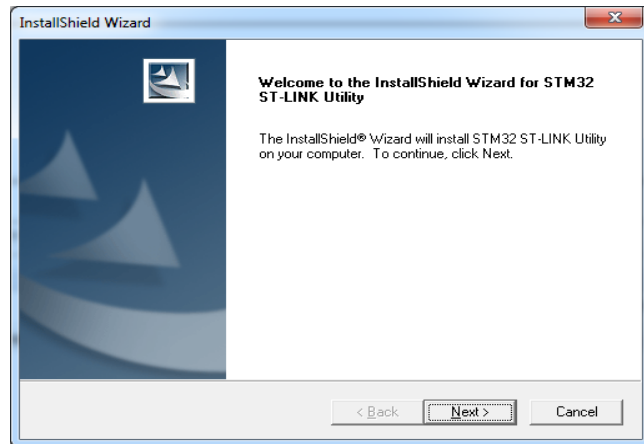


L'installation du logiciel s'exécute :



Suivre les étapes décrites ci-dessous :

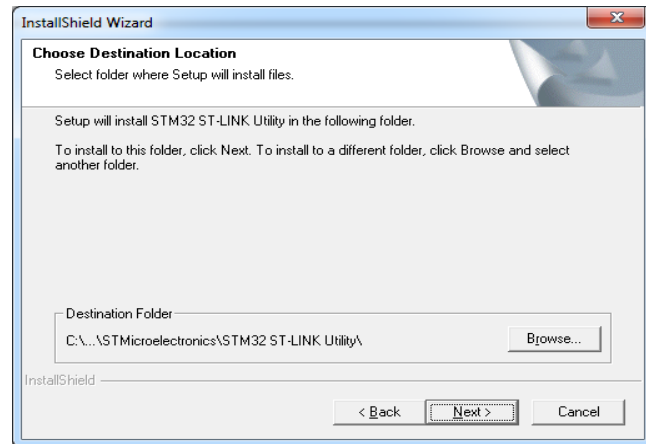
- Cliquer sur **Next** :



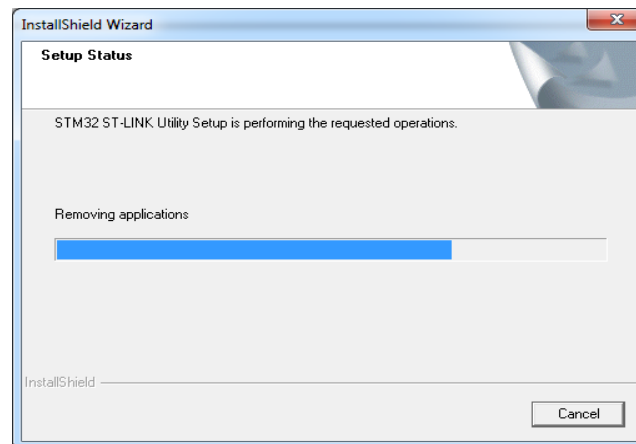
- Accepter la licence utilisateur en cliquant sur **Yes** :



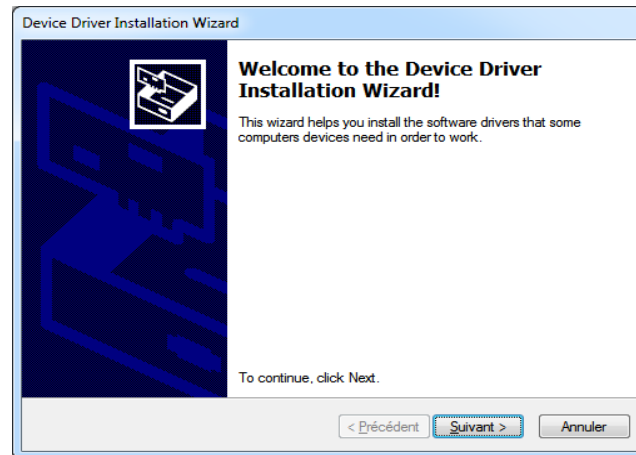
- Conserver le chemin d'installation par défaut et valider en cliquant sur **Next** :



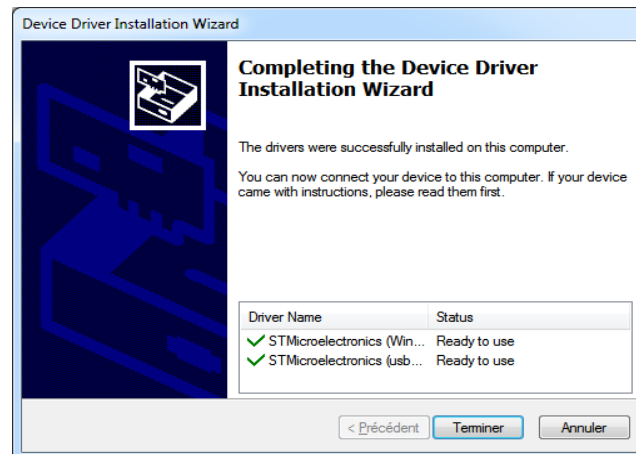
L'installation débute :



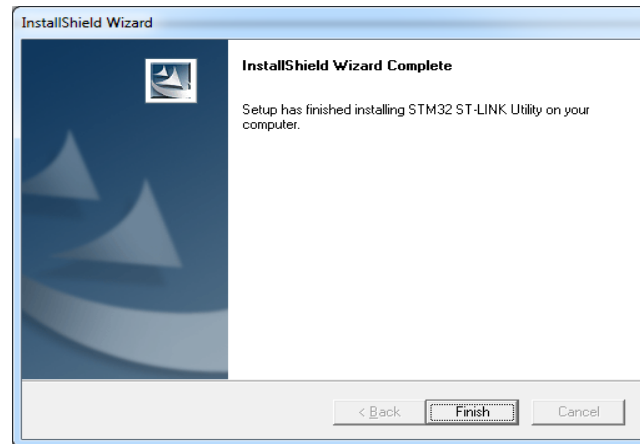
Valider l'installation des pilotes nécessaires pour communiquer avec la carte STM32F466 en cliquant sur **Suivant** :



Puis sur **Terminer** à la fin de l'installation :

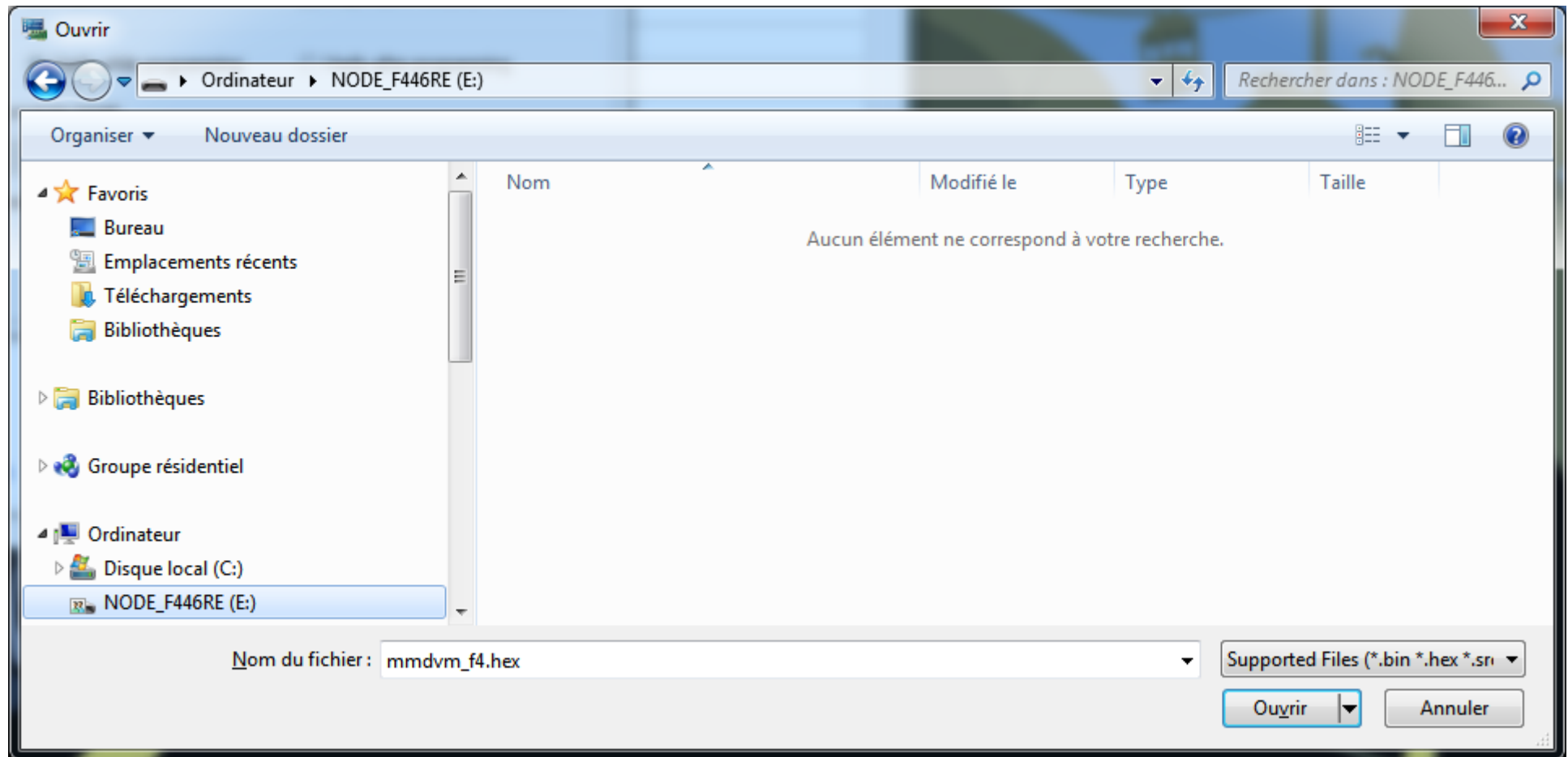


Quitter enfin le programme d'installation en cliquant sur **Finish** :



Chargement du logiciel MMDVM dans la carte STM32F466

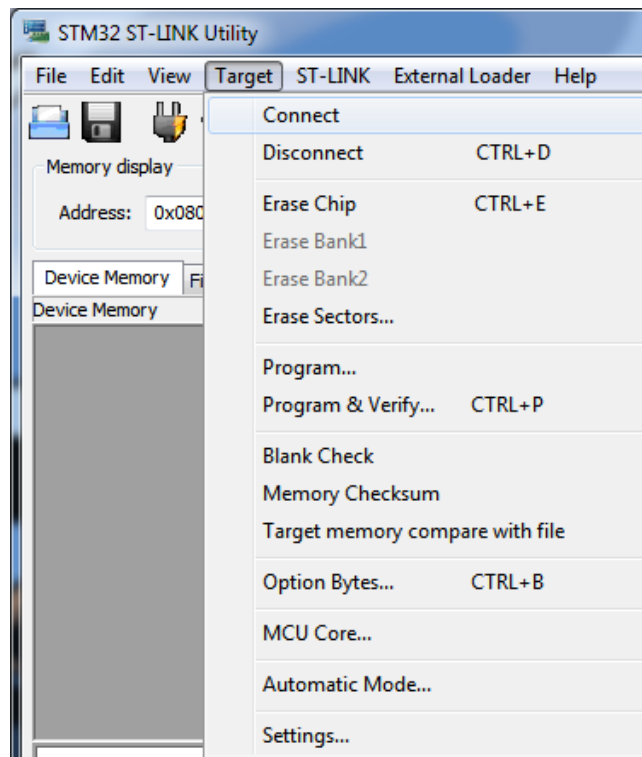
Connecter un cordon Mini USB entre le PC et la carte STM32F466. Si tout se passe bien, vous devez avoir 3 LED allumées sur la carte STM32 et vous devez voir un nouveau disque amovible nommé **NODE_F466RE** dans l'explorateur Windows :



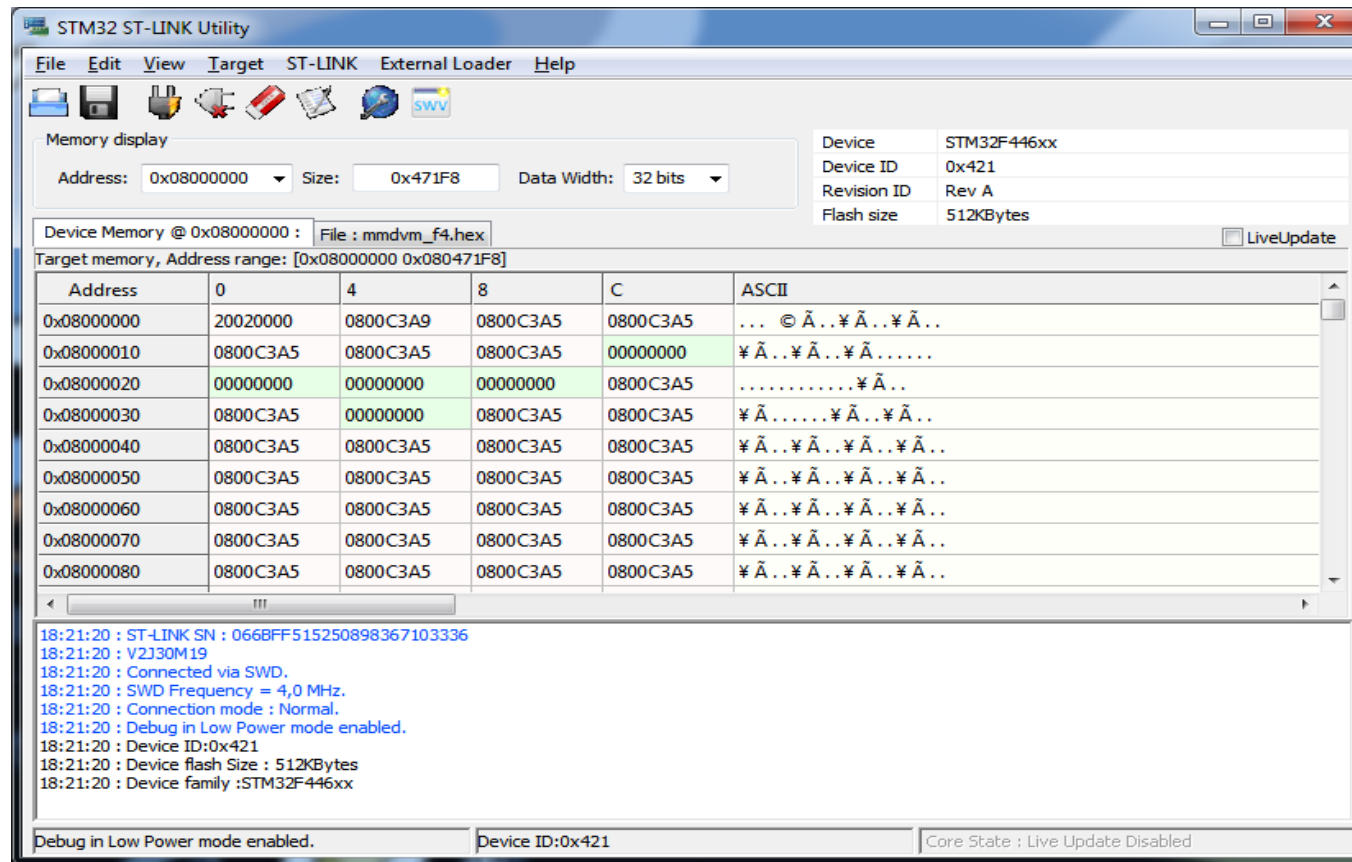
Afficher le bureau de Windows et double-cliquer sur l'icône **STM32 ST-LINK Utility** :



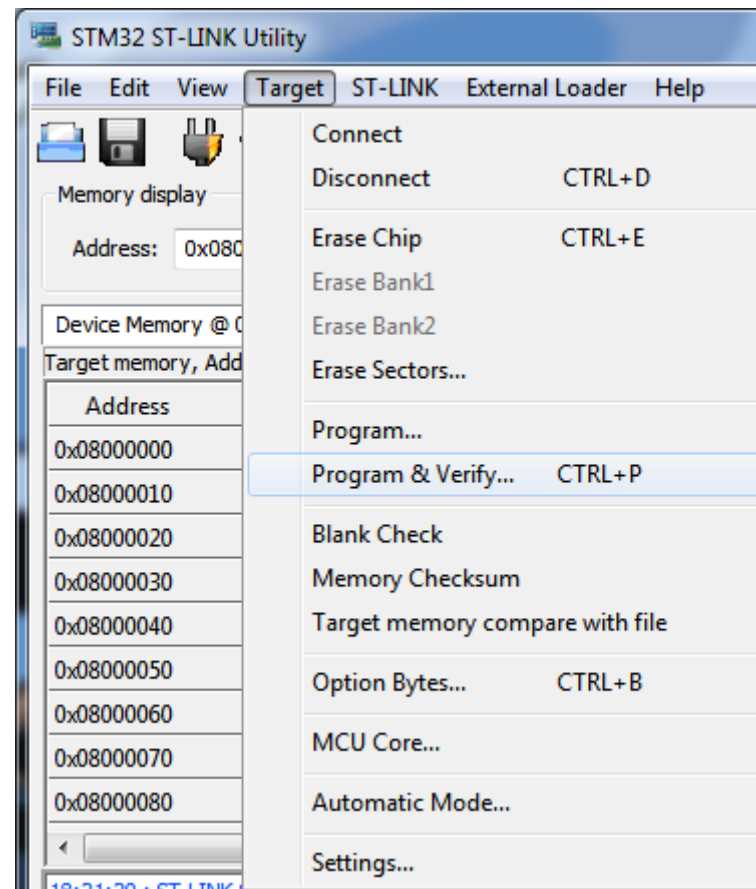
Dans la barre de menus affichée en haut du logiciel **ST-LINK Utility**, cliquer sur **Target** puis **Connect** :



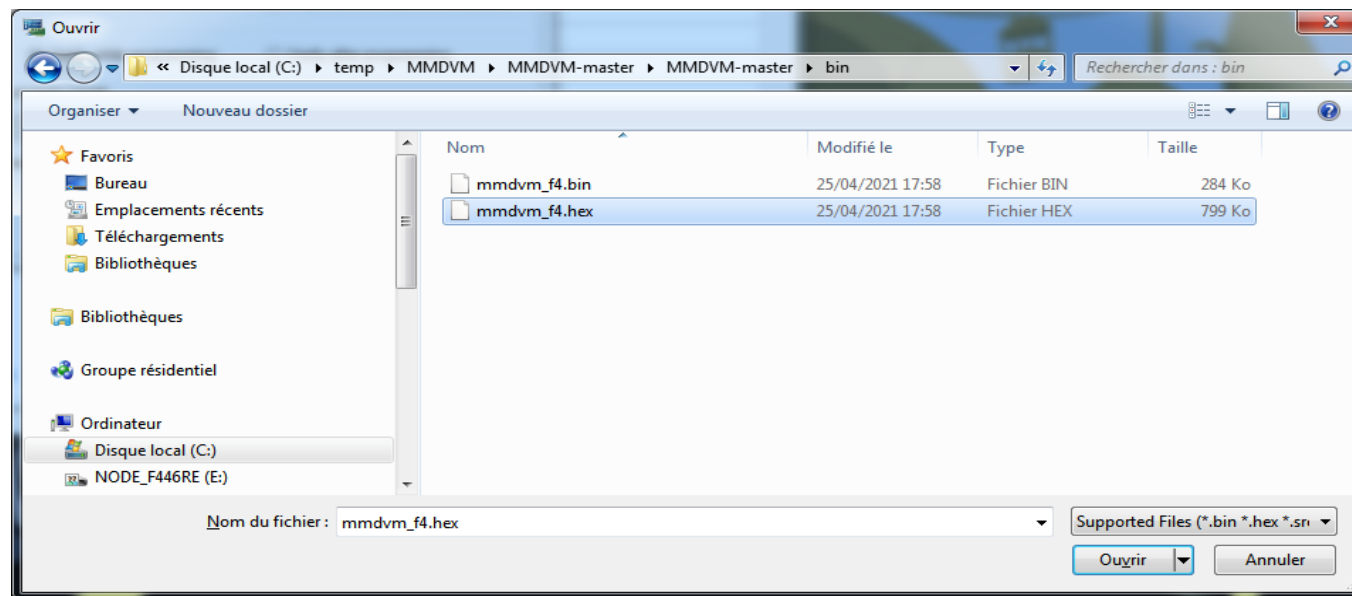
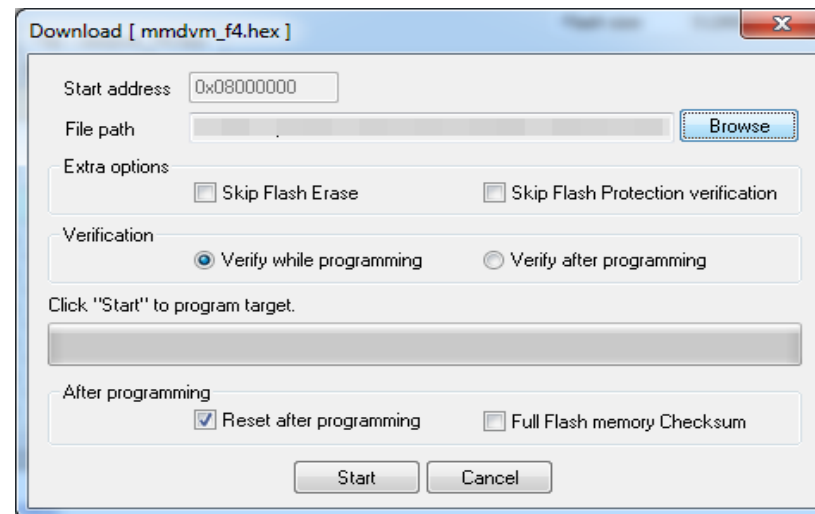
Un affichage de ce type doit être visible à l'écran. Cela signifie que la connexion a été exécutée avec succès :



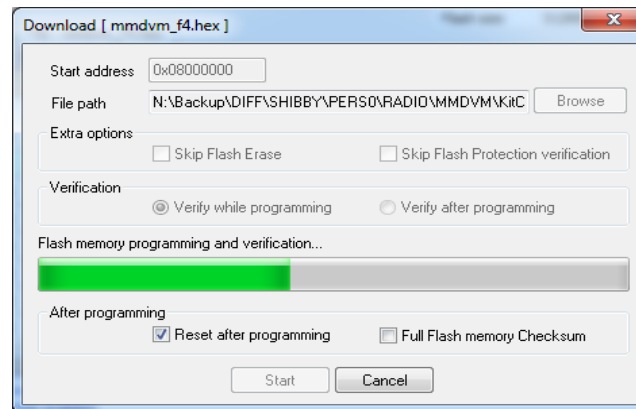
Cliquer à nouveau sur **Target** dans la barre de menu du haut, puis sur **Program & Verify** dans le menu déroulant :



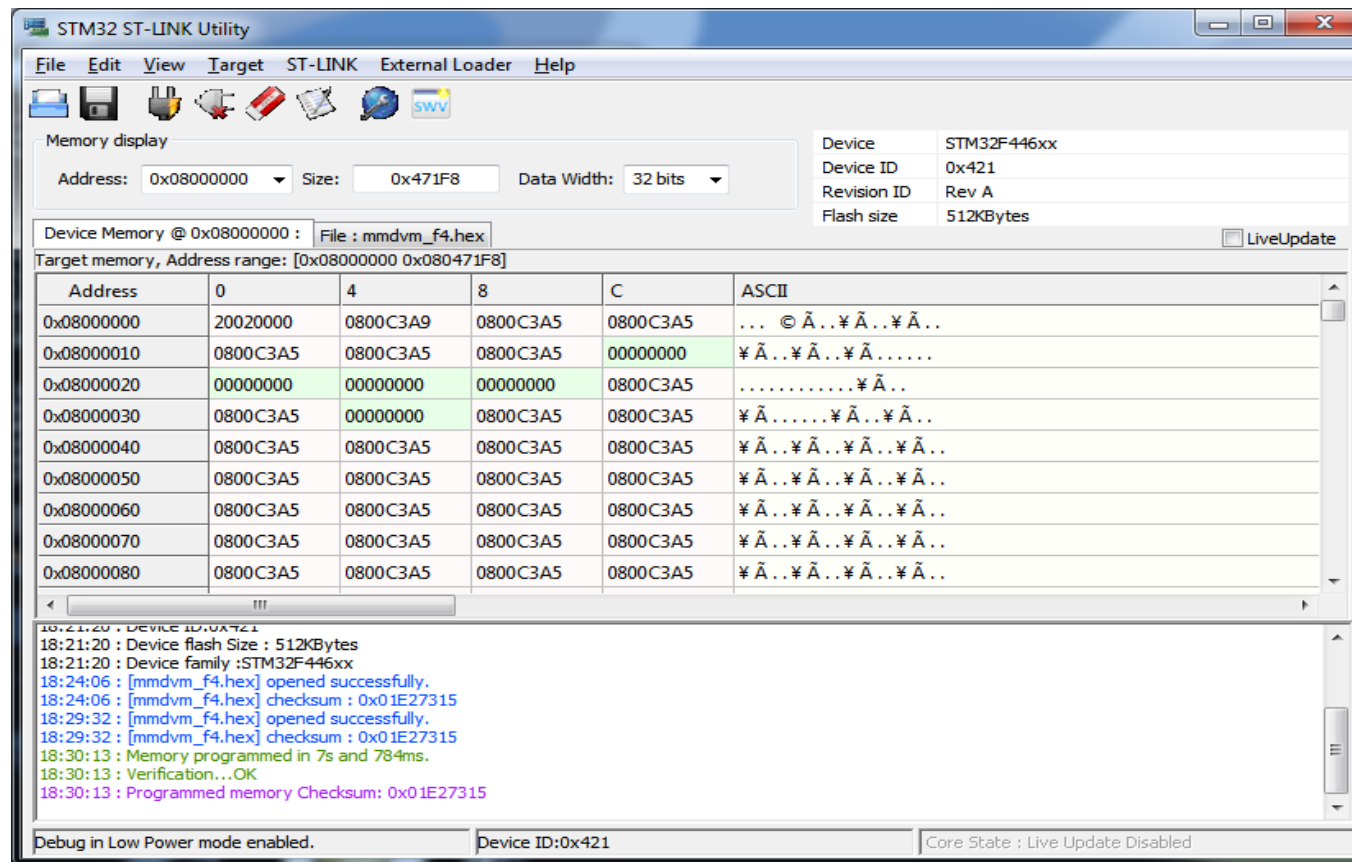
Une fenêtre s'affiche et vous invite à définir le chemin d'accès au fichier. Cliquer sur **Browse** et naviguer dans l'explorateur de fichiers jusqu'au fichier créé lors de la compilation :



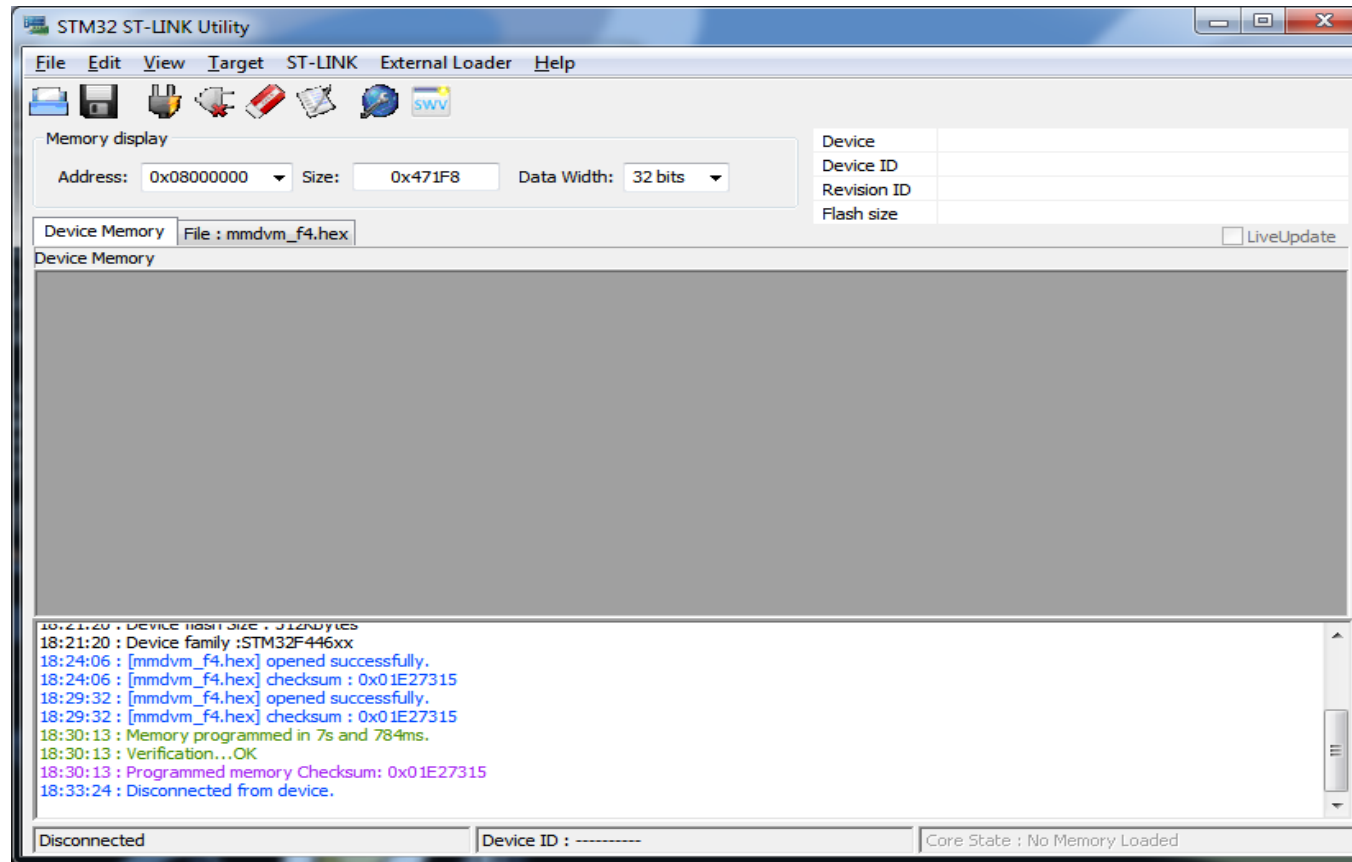
Valider en cliquant sur **Ouvrir**, puis lancer le chargement du fichier en cliquant sur **Start**.



À la fin de la programmation, le LED la plus proche du connecteur USB de la carte STM32 clignote rapidement en rouge et en vert. L'écran d'accueil du logiciel **ST-LINK Utility** doit afficher des lignes supplémentaires en vert et en mauve dans le journal des événements, en bas :



Cliquer sur **Target** dans la barre de menu du haut puis sur **Disconnect** dans le menu déroulant :



La carte STM32 peut être déconnectée définitivement du PC, enfin jusqu'à une future mise à jour de son logiciel, qui évolue périodiquement pour corriger des anomalies ou pour ajouter des fonctions. Il suffira de télécharger à nouveau l'archive ZIP mise à jour sur la page Github, d'extraire les fichiers en lieu et place de ceux utilisés précédemment, de compiler ces nouveaux fichiers et de les charger dans la carte STM32. Rien de compliqué maintenant que vous maîtrisez toutes ces étapes !