Raspberry Pi 4 (64 bit) DUAL_HAT Hotspot

This is my attempt to create a lightweight hotspot for DMR, D-STAR and C4FM on a Raspberry Pi 4 using the 64 bit operating system "Raspberry Pi OS".

Preparation

> ⚠ Do not use a cheap SD-card for your hotspot. We do not mount the filesystem read-only like on Pi-Star.

1. Download the operating software image from raspberrypi.com.
2. Boot the Raspberry Pi and set it up to your needs.
    - Localisation
    - Keyboard layout
    - Install needed packages
    - Install your favourite shell (bash, zsh, fish, csh ...)
3. Disable the serial console (and we will also disable Bluetooth)
    - Open `/boot/cmdline.txt` and remove the text `console=serial0,115200`
    - Open `/boot/config.txt` and add to the end

    ```
    dtoverlay=disable-bt
    enable_uart=1
    ```

    You may also run `raspi-config` and disable the console from there. Choose *3 Interface Options* » *I6 Serial Port* » *No (login shell)* » *Yes (serial hardware)* » *OK*

    Disable the services

    ```
    $ sudo systemctl disable serial-getty@ttyAMA0.service
    $ sudo systemctl disable bluetooth.service
    ```

    Reboot the Raspberry Pi after those changes!

    ```
    $ sudo reboot
    ```

4. Install the needed packages to compile the software later

```
$ sudo apt update -y
$ sudo apt full-upgrade
$ sudo apt install git build-essential zsh nmap lsof vim cmake gcc-
```

Because **wiringPi** is currently not available from official sources, we have to install it manually.

```
$ git clone git@github.com:WiringPi/WiringPi.git
$ cd wiringPi
$ ./build
```

This should suffice, make sure to read INSTALL and follow its instructions.

Check *wiringPi* with `gpio -v` or `gpio readall`:

```
$ gpio -v
gpio version: 2.70
Copyright (c) 2012-2018 Gordon Henderson
This is free software with ABSOLUTELY NO WARRANTY.
For details type: gpio -warranty

Raspberry Pi Details:
Type: Pi 4B, Revision: 01, Memory: 4096MB, Maker: Sony
  * Device tree is enabled.
  *--> Raspberry Pi 4 Model B Rev 1.1
  * This Raspberry Pi supports user-level GPIO access.
```

5. Create a non-privileged user called *mmdvm* and add the user to the group *dialout*

```
$ sudo useradd mmdvm -g mmdvm -s /sbin/nologin
$ sudo usermod mmdvm -G dialout
```

I had some problems with wiringPi, so I finally added the user *mmdvm* to the groups *gpio* and *kmem*. (To let the user *mmdvm* access `/dev/mem` and `/dev/gpiomem`.)

```
$ sudo usermod -G gpio,kmem -a mmdvm
```

Firmware

I hope we did not forgot: we use a **DUAL_HAT** extension on our Raspberry Pi and we have to compile the correct firmware for this. The actual version of the firmware is at

1.6.0:

MMDVM_HS_Dual_Hat-v1.6.0 20210919 12.2880MHz dual ADF7021 FW by CA6JAU GitID #d4cb546

But let us move to the fun (work) part.

Get the code

Clone the git repository. I prefer this in a separate folder like `~/git`.

```
$ git clone --recursive git@github.com:juribeparada/MMDVM_HS.git
```

Configuration

Enter the directory and copy an appropriate default configuration for your board.

```
$ cp configs/MMDVM_HS_Dual_Hat-12mhz.h Config.h
```

Edit the file to your needs. The example below is what I use, be aware of the version of the installed TCXO of **12**.2880 MHz!

```
// file: "Config.h"
#if !defined(CONFIG_H)
#define  CONFIG_H

#define MMDVM_HS_DUAL_HAT_REV10
#define ENABLE_ADF7021
#define DUPLEX
#define ADF7021_12_2880
#define AD7021_GAIN_HIGH
#define STM32_USART1_HOST
#define I2C_ADDR 0x22
#define ENABLE_SCAN_MODE
#define SEND_RSSI_DATA
#define SERIAL_REPEATER
#define SERIAL_REPEATER_BAUD 115200
#define QUIET_MODE_LEDS
#define DISCREET_SRV_LED_INVERTED
#define ENABLE_UDID

#endif
```

Those settings are also interesting to play with in the file `ADF7021.h`:

```
// file: "ADF7021.h"
// Disable TX Raised Cosine filter for 4FSK modulation in ADF7021:
```

```
#define ADF7021_DISABLE_RC_4FSK

// Enable AFC support for DMR, YSF, P25, and M17 (experimental):
// (AFC is already enabled by default in D-Star)
// #define ADF7021_ENABLE_4FSK_AFC

// Configure AFC with positive initial frequency offset:
// #define ADF7021_AFC_POS
```

I only disabled the Raised Cosine filter for 4FSK modulation because this works best for me. Enabling AFC did only cause lots of synchronisation errors on DMR. Also more firmware read errors occured with 4FSK AFC enabled.

You may want to leave that file intact if you just want your hotspot work without fiddling around with the source code.

Fine tuning and experimenting

If you want a hotspot that works: move over to the compilation process. If you want better response times on DMR keep on reading.

You may want to consider making the following changes on those two files.

```
// file: "DMRDMOTX.cpp"
//m_txDelay(240U),      // 200ms
m_txDelay(120U),      // 200ms
```

```
// file: "DMRDMOTX.cpp"
void CDMRDMOTX::setTXDelay(uint8_t delay)
{
    //m_txDelay = 600U + uint16_t(delay) * 12U;        // 500ms + tx de
    m_txDelay = 60U + uint16_t(delay) * 12U;
}
```

```
// file: "DMRTX.cpp"
// const uint32_t STARTUP_COUNT = 20U;
const uint32_t STARTUP_COUNT = 2U;
```

I'll leave you with this, if you have no idea what you are doing here: you should probably leave those files just as they are...

Those values were determined by Volker (DL2SDG) and I found them on one of the links below. Those values may not suit your configuratoin or board, so take them as a starting point and try the best values for you and your board.

Compile and upload to the board

If no errors were made in `Config.h` this should be as easy as the steps above.

```
$ make clean
$ make -j4
$ sudo make mmdvm_hs_dual_hat
```

Finish this with a reboot of the Raspberry Pi.

```
$ sudo reboot
```

Note, that we used `-j4` as an argument of `make` because the Raspberry Pi 4 has 4 virtual cores and we use them to speed up the compilation process.

Finally

And, as many operators tend to ignore manuals a lot, here are a few links that you ~~can~~ should read.

- https://github.com/juribeparada/MMDVM_HS/blob/master/BUILD.md
- https://github.com/mathisschmieder/MMDVM_HS_Hat/blob/master/README.md
- http://kb.amft-it.de/doku.php?id=kb-afu:mmdvm-hospot
- http://kb.amft-it.de/doku.php?id=kb-afu:mmdvm-hs-config

MMDVMHost

This is the program that connects our Gateways with the hardware board. This could be referred as the modem probably.

Get the code

We run this command also in our separated `git` folder. Just to keep all those git repositories in one place.

```
$ git clone git@github.com:g4klx/MMDVMHost.git
```

Compile the code

Enter the newly created directory and compile it.

```
$ make -j4 -f Makefile.Pi
```

Install the executables into `/usr/local/bin` by executing

```
$ sudo make install
```

Configuration

Copy the default configuration file to `/etc`:

```
$ sudo cp MMDVM.ini /etc/
```

and edit it to your needs. Most of the options are self-explanatory and I will only publish some excerpts of my `/etc/MMDVM.ini` file.

```
# file: "/etc/MMDVM.ini"
[Log]
# Logging levels, 0=No logging
DisplayLevel=2
FileLevel=2
FilePath=/var/log/mmdvm
FileRoot=MMDVM
FileRotate=0

[Modem]
Hardware=mmdvmhshat
# Valid values are "null", "uart", "udp", and (on Linux) "i2c"
Protocol=uart
# The port and speed used for a UART connection
# UARTPort=\\.\COM4
# UARTPort=/dev/ttyACM0
UARTPort=/dev/ttyAMA0
UARTSpeed=115200
# The port and address for an I2C connection
I2CPort=/dev/i2c
I2CAddress=0x22
# IP parameters for UDP connection
ModemAddress=192.168.2.100
ModemPort=3334
LocalAddress=192.168.2.1
LocalPort=3335
TXInvert=1
RXInvert=0
PTTInvert=0
TXDelay=100
RXOffset=-50
TXOffset=300
DMRDelay=0
RXLevel=50
TXLevel=50
RXDCOffset=0
```

```
TXDCOffset=0
RFLevel=100
# CWIdTXLevel=50
# D-StarTXLevel=50
# DMRTXLevel=50
# YSFTXLevel=50
# P25TXLevel=50
# NXDNTXLevel=50
# M17TXLevel=50
# POCSAGTXLevel=50
# FMTXLevel=50
# AX25TXLevel=50
RSSIMappingFile=/usr/local/etc/RSSI.dat
UseCOSAsLockout=0
Trace=0
Debug=0

[D-Star]
Enable=1
Module=B
SelfOnly=0
AckReply=1
AckTime=750
AckMessage=0
ErrorReply=1
RemoteGateway=0
# ModeHang=10
WhiteList=

[DMR]
Enable=1
# Set your hotspot ID here (use your own DMR-ID and append two digits t
Id=XXXXXXXXX
Beacons=0
BeaconInterval=60
BeaconDuration=3
ColorCode=1
SelfOnly=0
EmbeddedLCOnly=0
DumpTAData=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

[System Fusion]
Enable=1
LowDeviation=0
```

```
SelfOnly=0
TXHang=4
RemoteGateway=0
# ModeHang=10

[D-Star Network]
Enable=1
LocalAddress=127.0.0.1
LocalPort=20011
GatewayAddress=127.0.0.1
GatewayPort=20010
# ModeHang=3
Debug=0

[DMR Network]
Enable=1
# Type may be either 'Direct' or 'Gateway'. When Direct you must provic
# address as well as the Password, and for DMR+, Options also.
#Type=Direct
Type=Gateway
LocalAddress=127.0.0.1
LocalPort=62032
RemoteAddress=127.0.0.1
RemotePort=62031
#RemoteAddress=89.185.97.34
#RemotePort=55555
Password=passw0rd
# Password=P@ssw0rd1234
#Jitter=360
Jitter=0
Slot1=1
Slot2=1
# You would need the Options line if you want to connect MMDVMHost dire
# to a DMR server (without Gateway)
# Options="StartRef=4000;RelinkTime=15;UserLink=1;"
# ModeHang=3
Debug=0

[System Fusion Network]
Enable=1
LocalAddress=127.0.0.1
LocalPort=3200
GatewayAddress=127.0.0.1
GatewayPort=4200
# ModeHang=3
Debug=0
```

These are the most important parts of the configuration – make sure that you double check your file and insert your callsign and DMRID number correctly.

Within `` `[DMR]` `` you see the line starting with `` `Id=` ``. This is where you place the final Id for the DMR network, use your DMRID and append two digits to it, so you can run multiple hotspots on the same network/server (with different Ids, but still valid as they all start with your legit DMRID).

Create log file directory

```
$ sudo mkdir /var/log/mmdvm
$ sudo chown mmdvm:mmdvm /var/log/mmdvm
```

Check for errors

To check for errors, run `` `MMDVMHost` `` directly before setting up a system service. Make sure, that you have set `` `Daemon=0` `` in your `` `MMDVM.ini` `` file for this test. Start `` `MMDVMHost` `` as user *mmdvm*

```
$ sudo -u mmdvm MMDVMHost /etc/MMDVM.ini
```

Set `` `Daemon=1` `` back in your `` `MMDVM.ini` `` if you haven't already.

Setup the system service

Copy the unit file for Systemd to the system.

```
$ sudo cp linux/systemd/mmdvmhost.service /etc/systemd/system/
```

No modification should be needed, but for reference this is the content of the file:

```
# file: "/etc/systemd/system/mmdvmhost.service"
[Unit]
Description=MMDVMHost Radio Service
After=syslog.target network.target

[Service]
User=mmdvm
Type=forking
ExecStart=/usr/local/bin/MMDVMHost
Restart=on-abnormal

[Install]
WantedBy=multi-user.target
```

Start and enable this service in one go with

```
$ sudo systemctl daemon-reload
$ sudo systemctl enable --now mmdvmhost.service
```

DMRGateway

We want to connect our hotspot to the austrian IPSC2-OE-DMO server aswell as the austrian brandmeister master server (BM2321).

Get the code

```
$ git clone git@github.com:g4klx/DMRGateway.git
```

Configuration

```
# file: "/etc/DMRGateway.ini"
[General]
Timeout=10
# RFTimeout=10
# NetTimeout=7
RptAddress=127.0.0.1
RptPort=62032
LocalAddress=127.0.0.1
LocalPort=62031
RuleTrace=0
Daemon=1
Debug=0

[Log]
# Logging levels, 0=No logging
DisplayLevel=2
FileLevel=2
FilePath=/var/log/mmdvm
FileRoot=DMRGateway
FileRotate=0

[Voice]
Enabled=1
Language=en_GB
Directory=/home/pi/git/DMRGateway/Audio

[Info]
Latitude=0.0
Longitude=0.0
Height=0
Location=Location, GRID
Description=Multi-Mode Hotspot
URL=https://oe7drt.com

[XLX Network]
Enabled=0
File=XLXHosts.txt
Port=62030
```

```
Password=passw0rd
ReloadTime=60
# Local=3351
Slot=1
TG=6
Base=64000
Startup=950
Relink=10
Debug=0
#Allow user linking control using Private Calls
UserControl=1
#Override default module for startup reflector
#Module=A

# BrandMeister
[DMR Network 1]
Enabled=1
Name=BM
Address=94.199.173.125
Port=62031
#Local=3352
# Local cluster
#TGRewrite=1,9,1,9,1
# Echo on RF slot 1 TG9990 to network slot 1 9990
TypeRewrite=1,9990,1,9990
SrcRewrite=1,9990,1,9990,1
# Dynamic rewriting of slot 2 TGs 90-999999 to TG9 to emulate reflector
#TGDynRewrite=2,90,4000,5000,9,999910,9990
# For normal repeater operation disable TGDyn coment out the above line
# After that remove the coments below
PassAllTG=1
# PassAllTG=2
# Pass all of the other private traffic on slot 1 and slot 2
PassAllPC=1
#PassAllPC=2
Password=passw0rd
Location=0
Debug=0

# DMR+
[DMR Network 2]
Enabled=1
Name=DMR+
Address=89.185.97.34
Port=55555
# Local=3352
Password=PASSWORD
Options="StartRef=4000;RelinkTime=15;UserLink=1;TS1_1=110;"
TGRewrite=2,6,1,6,2
TGRewrite=2,110,1,110,1
TGRewrite=2,120,1,120,1
TGRewrite=2,130,1,130,1
```

```
TGRewrite=2,232,1,232,1
TGRewrite=2,8180,1,8180,10
TGRewrite=2,8191,1,8191,9
TGRewrite=2,9990,2,9990,1
TGRewrite=2,9,2,9,1
#PCRewrite=2,4000,2,4000,1001
PassAllPC=2
Location=0
Debug=0

[...]

[GPSD]
Enable=0
Address=127.0.0.1
Port=2947

[APRS]
Enable=0
Address=127.0.0.1
Port=8673
Description=APRS Description
Suffix=3

[Dynamic TG Control]
Enabled=1
Port=3769

[Remote Control]
Enable=0
Address=127.0.0.1
Port=7643
```

Test your setup

Change the configuration from above to `` `Daemon=0` `` and start DMRGateway in a terminal with

```
$ sudo -u mmdvm DMRGateway /etc/DMRGateway.ini
```

Check for errors and move on when everything is fine. You should hear the announcement coming from the ISPC2 server. In our case (Ref 4000) we should hear something like “Not connected” or “Repeater not connected”.

Create a system service

I copied the unit file from mmdvmhost.service and made my changes. If you don’t want to do that by yourself, use this one to start with.

```
# file: "/etc/systemd/system/dmrgateway.service"
[Unit]
Description=DMRGateway Radio Service
After=syslog.target network.target mmdvmhost.service

[Service]
User=mmdvm
Type=forking
ExecStart=/usr/local/bin/DMRGateway
Restart=on-abnormal

[Install]
WantedBy=multi-user.target
```

Change the configuration back `Daemon=1` and enable the service/unit.

```
$ sudo systemctl enable --now dmrgateway.service
```

YSFGateway (C4FM, YSF, YCS)

What we define: connect our hotspot to the `AT-C4FM-Austria` reflector via YSF.

Get the code

Use a quite actual but modified version of YSFClients

This is a fork of the original repository by *Jonathan, G4KLX*.

The only reason for using this version is the proper display in the servers dashboard. You will see your location aswell as the used frequency and the *YSFGateway* type in the *YCS232 dashboard*.

This version might not be as actual as the original repository, but it is ready-to-use with the *changes made by Kurt, OE1KBC* back in 2021.

I've included his changes in my fork and created a branch called *YCS232*.

```
$ git clone git@github.com:oe7drt/YSFClients.git
$ cd YSFClients
$ git fetch --all
$ git checkout YCS232
```

You can also add the original sources as an additional remote:

```
$ git remote add upstream git@github.com:g4klx/YSFClients.git
$ git fetch --all
```

Original, most actual codebase

```
$ git clone git@github.com:g4klx/YSFClients.git
```

Either choice of source, compile it

Enter the created directory *YSFClients* and compile the code. Install them into `` `/usr/local/bin` ``

```
$ cd YSFClients
$ make -j4
$ sudo make install
```

Configuration

Copy the default configuration file into `` `/etc` ``

```
$ sudo cp YSFGateway/YSFGateway.ini /etc/
```

and edit the file. Here are some excerpts of mine:

```
# file: "/etc/YSFGateway.ini"
[General]
Callsign=OE7DRT
Suffix=RPT
# Suffix=ND
Id=XXXXXXX
RptAddress=127.0.0.1
RptPort=3200
LocalAddress=127.0.0.1
LocalPort=4200
WiresXMakeUpper=1
WiresXCommandPassthrough=0
Debug=0
Daemon=1

[Info]
RXFrequency=430600000
TXFrequency=439075000
Power=1
Latitude=0.0
Longitude=0.0
Height=0
Name=Location, GRID
Description=Multi-Mode Hotspot

[Log]
# Logging levels, 0=No logging
```

```
DisplayLevel=2
FileLevel=2
FilePath=/var/log/mmdvm
FileRoot=YSFGateway
FileRotate=1

[APRS]
Enable=0
Address=127.0.0.1
Port=8673
Description=APRS Description
Suffix=Y

[Network]
# Startup=FCS00120
Startup=AT-C4FM-Austria
# book DG-ID for Reflector
Options=10,20,22,24,28,32,62,35
InactivityTimeout=0
Revert=0
Debug=0

[YSF Network]
Enable=1
Port=42000
Hosts=/usr/local/etc/YSFHosts.txt
ReloadTime=60
ParrotAddress=127.0.0.1
ParrotPort=42012
YSF2DMRAddress=127.0.0.1
YSF2DMRPort=42013
YSF2NXDNAddress=127.0.0.1
YSF2NXDNPort=42014
YSF2P25Address=127.0.0.1
YSF2P25Port=42015

[FCS Network]
Enable=0
Rooms=./FCSRooms.txt
Port=42001
```

This is **not the full** file, but pretty much of it. Adopt to your needs but check and double-check that file like the other configuration files.

Test your setup

Change `` `Daemon=0` `` in the configuration file for the test and run

```
$ sudo -u mmdvm YSFGateway /etc/YSFGateway.ini
```

You should see yourself in the dashboard and also the screen should print something like `Linked to AT-C4FM-AUSTRIA` and `Link successful to MMDVM`.

Stop it with `CTRL + C` and move on to create a system service (unit file) for Systemd.

Change `Daemon=1` back in your config if not already done.

Setup a system service

Create the unit file `/etc/systemd/system/ysfgateway.service`

```
# file: "/etc/systemd/system/ysfgateway.service"
[Unit]
Description=YSFGateway Service
After=syslog.target network.target

[Service]
User=mmdvm
Type=forking
ExecStart=/usr/local/bin/YSFGateway
Restart=on-abnormal

[Install]
WantedBy=multi-user.target
```

Start and enable the service.

```
$ sudo systemctl daemon-reload
$ sudo systemctl enable --now ysfgateway.service
```

ircDDBGateway (D-STAR)

Okay, this was the hardest part for me because I haven't found much information about what programs and configuration files are really needed for D-STAR to work properly – I'm not even sure now if it is working fine or if something is still missing.

So the information in this section may be inaccurate or even wrong. Consider that, but if you have a correct answer I would be happy to hear about it.

I think that *dstarrepeater* was only used for the first hardware and it's work is now done from MMDVMHost. Again, correct me if I'm wrong.

Get the code

In our `~/git` directory (where else?)

```
$ git clone git@github.com:g4klx/ircDDBGateway.git
```

Compile the code and install the programs

Open `Makefile` and change the parameters on top of the file to your needs. I like to use `/var/log/mmdvm`, `/usr/local/bin` and `/usr/local/etc` for example...

```
$ cd ircDDBGateway
$ make -j4
$ sudo make install
```

Configuration

Sadly, I haven't found any configuration file as an example, so I referred to the configuration files on anothor hotspot that was running Pi-Star.

These are the files that I use at the moment, changes may be done if needed.

/etc/ircddbgateway

```
# file: "/etc/ircddbgateway"
gatewayType=1
# gatewayType=0 is the default, but Pi-Star uses 1 here
# 0 repeater, 1 hotspot, 2 dongle, 3 starnet
gatewayCallsign=N0SIGN
gatewayAddress=0.0.0.0
icomAddress=172.16.0.20
icomPort=20000
hbAddress=127.0.0.1
hbPort=20010
latitude=0.0
longitude=0.0
description1=Location, GRID
description2=Location State
url=https://qrz.com/db/N0SIGN
repeaterCall1=N0SIGN
repeaterBand1=B
# repeaterType1=0 homebrew, 1 icom, 2 dummy
repeaterType1=0
repeaterAddress1=127.0.0.1
repeaterPort1=20011
reflector1=DCS009 A
atStartup1=1
reconnect1=0
frequency1=439.07500
offset1=-8.4750
rangeKms1=1.000
latitude1=0.0
longitude1=0.0
agl1=3.000
```

```
description1_1=Location, GRID
description1_2=Location State
url1=
band1_1=0
band1_2=0
band1_3=0
repeaterCall2=
repeaterBand2=
repeaterType2=0
repeaterAddress2=127.0.0.1
repeaterPort2=20012
reflector2=
atStartup2=0
reconnect2=0
frequency2=0.00000
offset2=0.0000
rangeKms2=0.000
latitude2=0.000000
longitude2=0.000000
agl2=0.000
description2_1=
description2_2=
url2=
band2_1=0
band2_2=0
band2_3=0
repeaterCall3=
repeaterBand3=
repeaterType3=0
repeaterAddress3=127.0.0.1
repeaterPort3=20013
reflector3=
atStartup3=0
reconnect3=0
frequency3=0.00000
offset3=0.0000
rangeKms3=0.000
latitude3=0.000000
longitude3=0.000000
agl3=0.000
description3_1=
description3_2=
url3=
band3_1=0
band3_2=0
band3_3=0
repeaterCall4=
repeaterBand4=
repeaterType4=0
repeaterAddress4=127.0.0.1
repeaterPort4=20014
reflector4=
atStartup4=0
reconnect4=0
frequency4=0.00000
offset4=0.0000
rangeKms4=0.000
latitude4=0.000000
longitude4=0.000000
agl4=0.000
description4_1=
description4_2=
url4=
band4_1=0
band4_2=0
band4_3=0
ircddbEnabled=1
ircddbHostname=ircv4.openquad.net
ircddbUsername=N0SIGN
ircddbPassword=
ircddbEnabled2=0
ircddbHostname2=rr.openquad.net
ircddbUsername2=
ircddbPassword2=
ircddbEnabled3=0
ircddbHostname3=
ircddbUsername3=
ircddbPassword3=
ircddbEnabled4=0
ircddbHostname4=
ircddbUsername4=
ircddbPassword4=
aprsEnabled=1
aprsHostname=euro.aprs2.net
aprsPassword=00000
aprsPort=14580
dextraEnabled=1
dextraMaxDongles=5
dplusEnabled=1
dplusMaxDongles=5
dplusLogin=N0SIGN
dcsEnabled=1
ccsEnabled=1
ccsHost=CCS704
xlxEnabled=0
xlxOverrideLocal=0
xlxHostsFileUrl=http://xlxapi.rlx.lu/api.php?do=GetXLXDMRMaster
starNetBand1=A
starNetCallsign1=
starNetLogoff1=
starNetInfo1=
starNetPermanent1=
starNetUserTimeout1=300
starNetGroupTimeout1=300
```

```
starNetCallsignSwitch1=0
starNetTXMsgSwitch1=1
starNetReflector1=
starNetBand2=A
starNetCallsign2=
starNetLogoff2=
starNetInfo2=
starNetPermanent2=
starNetUserTimeout2=300
starNetGroupTimeout2=300
starNetCallsignSwitch2=0
starNetTXMsgSwitch2=1
starNetReflector2=
starNetBand3=A
starNetCallsign3=
starNetLogoff3=
starNetInfo3=
starNetPermanent3=
starNetUserTimeout3=300
starNetGroupTimeout3=300
starNetCallsignSwitch3=0
starNetTXMsgSwitch3=1
starNetReflector3=
starNetBand4=A
starNetCallsign4=
starNetLogoff4=
starNetInfo4=
starNetPermanent4=
starNetUserTimeout4=300
starNetGroupTimeout4=300
starNetCallsignSwitch4=0
starNetTXMsgSwitch4=1
starNetReflector4=
starNetBand5=A
starNetCallsign5=
starNetLogoff5=
starNetInfo5=
starNetPermanent5=
starNetUserTimeout5=300
starNetGroupTimeout5=300
starNetCallsignSwitch5=0
starNetTXMsgSwitch5=1
starNetReflector5=
remoteEnabled=1
remotePassword=raspberry
#remotePort=54321
remotePort=10022
language=0
infoEnabled=1
echoEnabled=1
logEnabled=1
dratsEnabled=0
```

```
dtmfEnabled=1
mobileGPSEnabled=0
mobileGPSAddress=127.0.0.1
mobileGPSPort=7834
windowX=-1
windowY=-1
```

I wasn't aware what numbers need to be set on *gatewayType* or *repeaterType*, but a look into the source code made my decision easier. I'm not sure if that is correctly interpreted as I never finished learning C++ but I learned the basics back in school…

And so I found some information in `Common/Defs.h`:

```
enum HW_TYPE {
    HW_HOMEBREW,
    HW_ICOM,
    HW_DUMMY
};
```

```
enum GATEWAY_TYPE {
    GT_REPEATER,
    GT_HOTSPOT,
    GT_DONGLE,
    GT_STARNET
};
```

View those two online on github: HW_TYPE, GATEWAY_TYPE

/etc/timeserver

timeserverd is used to produce time announcements. The interval of these can be set to

- every 15 minutes: `interval=0`
- every 30 minutes: `interval=1`
- every hour: `interval=2`

Adopt those changes in `/etc/timeserver`.

```
# file: "/etc/timeserver"
callsign=N0SIGN
sendA=0
sendB=1
sendC=0
sendD=0
```

```
sendE=0
address=127.0.0.1
language=3
format=1
interval=2
windowX=0
windowY=0
```

First run

```
$ sudo -u mmdvm ircddbgatewayd
```

Abort with `` `CTRL + C` `` when done.

Create the system services

You could copy over the unit files from `` `./debian` ``, but the paths in those files need adjustement, so we create them directly with `` `sudo vim /etc/systemd/system/...` ``

```
# file: "/etc/systemd/system/ircddbgateway.service"
[Unit]
Description=D-STAR Gateway Daemon
After=network.target

[Service]
User=mmdvm
ExecStart=/usr/local/bin/ircddbgatewayd
Restart=on-abort

[Install]
WantedBy=multi-user.target
```

```
# file: "/etc/systemd/system/timeserver.service"
[Unit]
Description=Timeserver (ircDDBGateway) Daemon
After=syslog.target network.target ircddbgateway.service

[Service]
User=mmdvm
Type=forking
ExecStart=/usr/local/bin/timeserverd -daemon
#ExecStop=/usr/local/sbin/timeserver.service stop
#ExecReload=/usr/local/sbin/timeserver.service restart
Restart=on-abort

[Install]
```

```
WantedBy=multi-user.target
```

Enable the services

```
$ sudo systemctl daemon-reload
$ sudo systemctl enable --now ircddbgateway.service
$ sudo systemctl enable --now timeserver.service
```

The hotspot works

This is the end of the procedure if you want a working hotspot. If you want visualisation on a dashboard continue reading: we will install nginx as our webserver and host the dashboard made by Kim, DG9VH.

Dashboard

Preparation

There are a few things to prepare, before we can finally visualize the electro-magnetic stream in the air.

Install some needed packages

```
$ sudo apt install python3-websockets python3-pip
$ sudo pip3 install ansi2html
$ sudo apt install python3-gpiozero python3-psutil python3-serial
```

Refer to the installation instructions on the repository for more information and the full instructions.

Allow the *logtailer* program access to *MMDVMHost* with `` `sudo visudo` `` and add this line to the user section of the sudoers file:

```
www-data ALL=(ALL) NOPASSWD: /usr/local/bin/MMDVMHost
```

Webserver or built-in solution?

We could use a built-in solution to host our dashboard using Python. This is a fancy way to host a dashboard but I've never used such solutions myself, so I'll stick to a real webserver in this article.

Install nginx with php support with

```
$ sudo apt install nginx php-fpm
```

Configure and limit webserver

I've already added some limiting directives into the default configuration. We should not need this with the websockets based dashboard, but I add this anyway.

We also added `` `index.php` `` to the `` `index` `` directive, but not in front of the filenames → we want to disable the dashboard with an empty `` `index.html` `` file that we create in the document root of our webserver (`` `sudo touch /var/www/html/index.html` ``).

```
# file: "/etc/nginx/sites-enabled/default"
limit_req_zone $binary_remote_addr zone=mylimit:10m rate=2r/s;

# Default server configuration
#
server {
  listen 80 default_server;
  listen [::]:80 default_server;
  limit_req zone=mylimit burst=5 nodelay;
  # limit_req_status 444;

  # SSL configuration
  #
  # listen 443 ssl default_server;
  # listen [::]:443 ssl default_server;
  #
  # [...]
  #
  # include snippets/snakeoil.conf;

  root /var/www/html;

  # Add index.php to the list if you are using PHP
  index index.html index.php index.htm index.nginx-debian.html;

  server_name _;

  location / {
    # First attempt to serve request as file, then
    # as directory, then fall back to displaying a 404.
    try_files $uri $uri/ =404;
  }

  # pass PHP scripts to FastCGI server
  #
  location ~ \.php$ {
    include snippets/fastcgi-php.conf;

    # With php-fpm (or other unix sockets):
    fastcgi_pass unix:/run/php/php7.4-fpm.sock;
    # With php-cgi (or other tcp sockets):
    #fastcgi_pass 127.0.0.1:9000;
  }

  # deny access to .htaccess files, if Apache's document root
  # concurs with nginx's one
  #
  location ~ /\.ht {
    deny all;
  }
}


# Virtual Host configuration for example.com
#
# [...]
```

Get the code

```
$ sudo mkdir /opt/MMDVMDash
$ sudo chown pi /opt/MMDVMDash
$ git clone --recursive git@github.com:dg9vh/MMDVMHost-Websocketboard.g
```

We saved the dashboard repository now in `` `/opt/MMDVMDash` ``.

Configuration

Open `` `logtailer.ini` ``, this should look something like this (shrinked together)

```
# file: "/opt/MMDVMDash/logtailer.ini"
[DEFAULT]
Host=0.0.0.0
Port=5678
Ssl=False
SslCert=/path/to/cert
SslKey=/path/to/keyfile
MaxLines=500
Filerotate=True
[MMDVMHost]
Logdir=/var/log/mmdvm/
Prefix=MMDVM
DMR_ID_Lookup=1
DMR_ID_LookupFile=/usr/local/etc/DMRIds.dat
DMR_ID_Reload_Time=1440
MMDVM_ini=/etc/MMDVM.ini
MMDVM_bin=/usr/local/bin/MMDVMHost
[DAPNETGateway]
Logdir=/var/log/mmdvm/
Prefix=DAPNETGateway
[ServiceMonitoring]
BinaryName1=MMDVMHost
```

```
BinaryName2=ircddbgatewayd
BinaryName3=YSFGateway
BinaryName4=timeserverd
```

Next, open `/opt/MMDVMDash/html/js/config.js` and modify it to your needs

```
// file: "/opt/MMDVMDash/html/js/config.js"
var config_struc_ver = 20210501.1;
var qrz = 1;
var debug = 0;
var warnlevel = 200;
var emergencylevel = 500;
var currtx = 1;
var lastheard = 2;
var localheard = 1;
var allheard = 1;
var qso = 1;
var dapnet = 0;
var sysinfo = 1;
var services = 1;
var about = 0;
var useClientTimezone = 1;
var showBMTGLink = 0;
var qrz_blacklist = [
"N0CALL",
]
var dashboard_blacklist = [
"MY0CALL",
]
var useDarkTheme = 0;
var customHeadlineText = ``;
var customText = ``;
```

Create and enable a system service

```
$ sudo cp systemd/logtailer.service /etc/systemd/system/
$ sudo systemctl daemon-reload
$ sudo systemctl enable --now logtailer.service
```

Enjoy the multi-mode hotspot

Finally, a few last words should be written. I've spent quite some time with my research about the tools needed for a hotspot. Most of us use Pi-Star and go with it. And that's fine, because Pi-Star is an awesome jack of all trades, but in this article I tried to explain my experiences that I learned when I created my lightweight hotspot that only runs programs that I really need.

You gain more control over the system tasks on your Raspberry Pi, but you loose some comfort functions. I always liked slim and lightweight systems since I learned about the linux world back in my school time. I went with Gentoo and fluxbox for some years and also used FreeBSD in combination with fluxbox on my older Lenovo T60 laptop.

So that’s why I try to make my systems small and compact and I’m very happy with the result. I personally thought that I’d stuck on the D-STAR configuration because I’ve found so less information about that mode – specially it’s configuration. But now it’s working fine and I can say: it wasn’t that hard (or abstract) as I thought before.