Instalación, configuración y programación del STM32 MMDVM_HS para Modo USB (Modem ACM0)

Expandir el sistema de archivos para ocupar toda la microSD:

```
rpi-rw
sudo pistar-expand
sudo reboot
```

Después de reiniciar activaremos el modo de escritura:

```
rpi-rw
```

Actualizaremos el sistema:

```
sudo apt-get update
sudo apt-get install gcc-arm-none-eabi gdb-arm-none-eabi libstdc++-arm-none-eabi-newlib libnewlib-arm-none-eabi
```

Instalaremos los paquetes necesarios:

```
cd ~
cd /home/pi
git clone https://github.com/juribeparada/MMDVM_HS
cd MMDVM_HS/
git clone https://github.com/juribeparada/STM32F10X_Lib
```

Editamos el Config.h

```
sudo nano Config.h
```

Nos Saldrá algo así como esto:

Las lineas con # en primer lugar, son las que están activas.

```
#if !defined(CONFIG_H)
#define  CONFIG_H

// Select one board (STM32F103 based boards)

// 1) ZUMspot RPi or ZUMspot USB:

// #define ZUMSPOT_ADF7021

// 2) Libre Kit board or any homebrew hotspot with modified RF7021SE and Blue Pill STM32F103:

#define LIBRE_KIT_ADF7021

// 3) MMDVM_HS_Hat revisions 1.1, 1.2 and 1.4 (DB9MAT & DF2ET)

// #define MMDVM_HS_HAT_REV12

// 4) MMDVM_HS_Dual_Hat revisions 1.0 (DB9MAT & DF2ET & DO7EN)

// #define MMDVM_HS_DUAL_HAT_REV10

// 5) Nano hotSPOT (BI7JTA)

// #define NANO_HOTSPOT

// 6) Nano DV revisions 1.0 (BG4TGO & BG5HHP)

// #define NANO_DV_REV10

// Enable ADF7021 support:
```

```
#define ENABLE_ADF7021

// Enable full duplex support with dual ADF7021 (valid for homebrew
hotspots only):
// #define DUPLEX

// TCXO of the ADF7021
// For 14.7456 MHz:
#define ADF7021_14_7456
// For 12.2880 MHz:
// #define ADF7021_12_2880

// Host communication selection:
// #define STM32_USART1_HOST
#define STM32_USB_HOST

// Enable mode detection:
#define ENABLE_SCAN_MODE

// Send RSSI value:
// #define SEND_RSSI_DATA

// Enable Nextion LCD serial port repeater on USART2 (ZUMspot Libre Kit
and ZUMspot RPi):
#define SERIAL_REPEATER

// Enable Nextion LCD serial port repeater on USART1 (Do not use with
STM32_USART1_HOST enabled):
// #define SERIAL_REPEATER_USART1

// Enable P25 Wide modulation:
// #define ENABLE_P25_WIDE

// Disable mode LEDs blink during scan mode:
// #define QUIET_MODE_LEDS

// Constant Service LED once repeater is running
// #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

// Enable modem debug messages

// #define ENABLE_DEBUG

#endif
```

Modificaremos las siguientes líneas:

En este caso ya tenemos activado por defecto el modo USB, pero por si acaso explico el funcionamiento de activar y desactivar una línea.

Activaremos modo STM32_USB_HOST (quitando las // que tiene delante) y desactivaremos el modo STM32_USART1_HOST (poniendo // delante de #define STM32_USART1_HOST)

```
// Host communication selection:
// #define STM32_USART1_HOST
#define STM32_USB_HOST
```

Con estos cambios sería suficiente. Ahora si queremos activaremos la pantalla Nextion y el modo RSSI (si no está activado) en las siguientes líneas:

```
// Enable Nextion LCD serial port repeater on USART1 (Do not use with STM32_USART1_HOST enabled):

#define SERIAL_REPEATER_USART1

// Send RSSI value:
#define SEND_RSSI_DATA
```

Una vez modificado todo guardamos el documento pulsado Ctrl+c
Ahora solo nos falta compilar con el comando sudo make bl.

Si todo ha ido bien y no nos ha dado ningún error al compilar procederemos a cargar el firmware al STM.

Conectaremos un extremo de los pines a la placa donde pone Serial TTL (TX,RX,GND Y VCC), en el otro extremos lo conectaremos al conversor TTL-USB de la siguiente forma:

```
  TXD --- RXD
  RXD --- TXD
  GND --- GND
  +5  --- VCC (Hay opción a 3v3)
```

Después conectaremos el conversor a cualquier usb de la Raspberry/Orange pi.

Detendremos los procesos mmdvmhost para liberar el stm:
Abrimos un terminal el remoto o el local y copiamos los siguientes comandos:

```
sudo pistar-watchdog.service stop
sudo systemctl stop mmdvmhost.timer
sudo systemctl stop mmdvmhost.service
```

Cargar el firmware:

Abrimos un terminal sea en remoto o en local, nos situamos en el carpeta MMDVM_HS, con el comando, cd /home/pi/MMDVM_HS/ cambiaremos de posición el jumper del STM más alejado del RESET, le pondremos de posición 0 a posición 1, seguidamente copiaremos el comando sudo make serial-bl devser=/dev/ttyUSB0 en el terminal (sin pulsar ENTER AUN),
(Nota: si tenemos ya conectado algún que otro usb sería ttyUSB1).
Pulsaremos el botón RESET y lo más rápido posible pulsamos ENTER, entonces empezará a programar el STM.
Cuando termine y si todo ha ido bien te dirá que está al 100%, cambiaremos el jumper movido a su posición original y ya tendremos programado el STM.

NOTA: Es posible que de error al terminar de programar el stm, si pasa eso repetiremos el proceso por si en algún momento no lo ha hecho bien.

Si queremos actualizar el firmware no es necesario hacer todo este proceso de nuevo. Nos colocamos en la carpeta cd /home/pi/MMDVM_HS/, lanzamos el comando sudo git pull, si hay actualizaciones nos lo dirá. Procedemos a limpiar la carpeta con la comando sudo make clean, editamos el fichero Config.h por si necesitamos cambiar algo, sudo nano Config.h. Seguidamente compilaremos con el comando sudo make bl, si todo ha ido bien procederemos a cargar el firmware como hemos citado arriba.

EA4GAX - Sergio.