MMDVM RF7021SE STM32F103C8T6

Este documento pretende describir y ayudar a configurar las diferentes formas de trabajar del software MMDVM con la versión de Andy CA6JAU (https://www.youtube.com/channel/UCM3Q9w1icLkhq1sTxRScRbg)

Página de referencia SOFTWARE

https://github.com/juribeparada/MMDVM_HS

Página de referencia HARDWARE DB4PLE

https://github.com/db4ple/MMDVM_HS_PCB_Single

Atención especial al fichero:

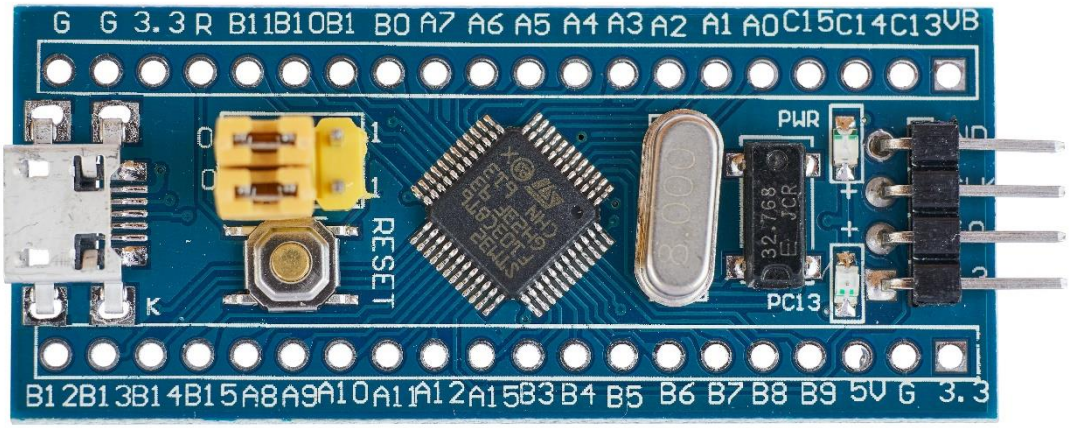
- **BUILD.TXT**

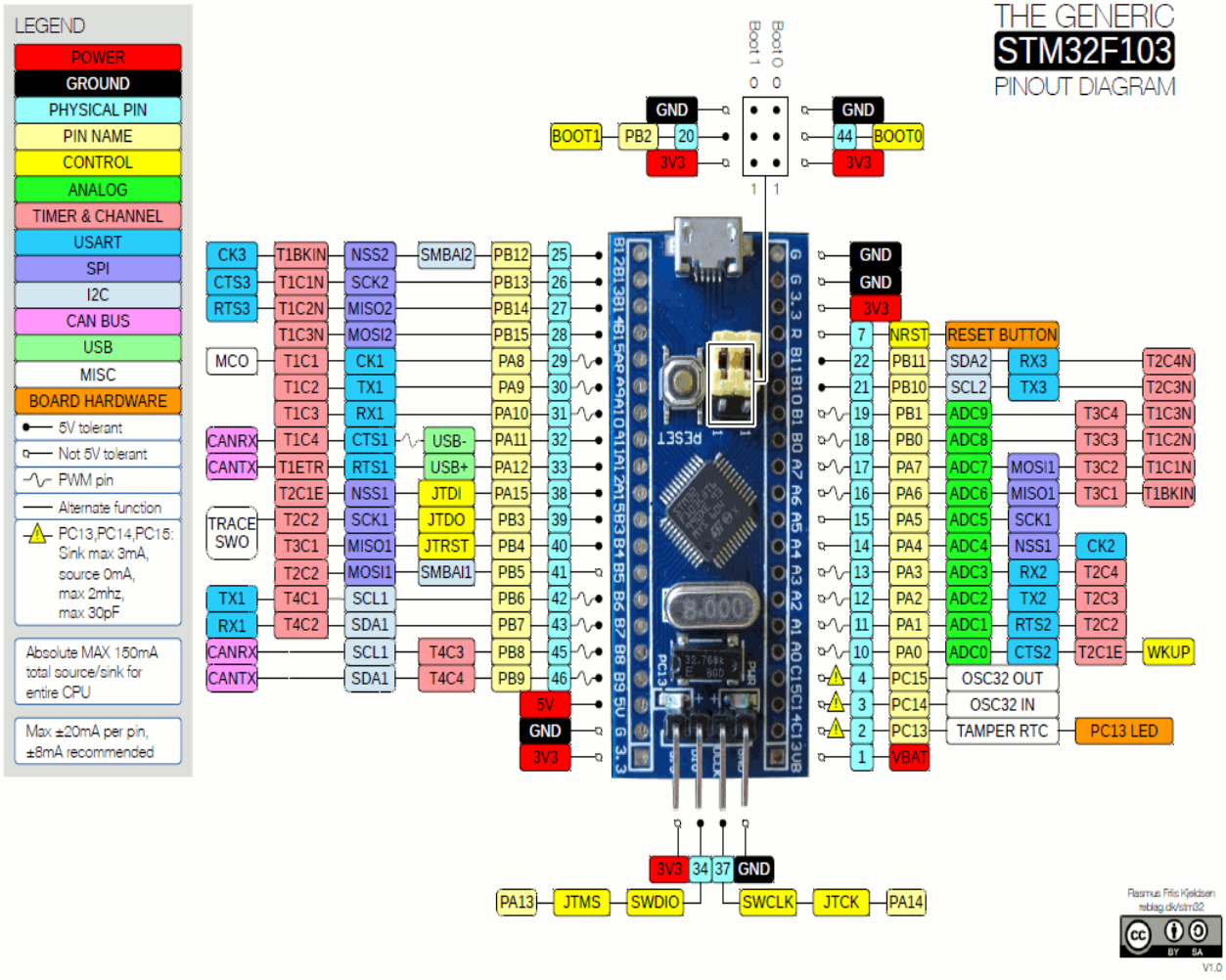
En ese documento se detalla toda la información sobre compilación, pines, etc.

La programación se puede realizar de diferentes formas, para poder posteriormente trabajar VIA SERIAL USART o USB HOST – AMA0 / ACMx / COMx, en función del sistema serial elegido y el sistema operativo.

DIAGRAMAS

En el citado gáfico se muestra las conexiones del STM.





La sección de RF se basa en el circuito: ADF7021

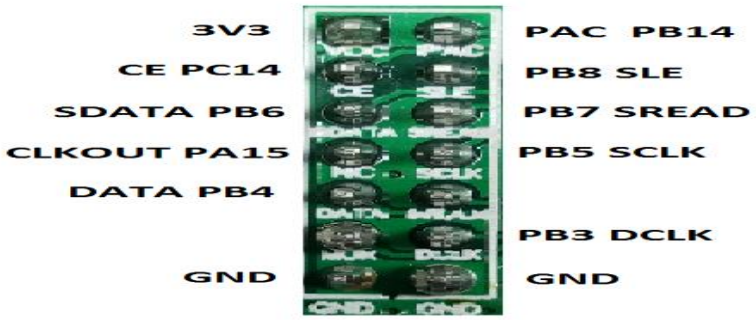
http://www.analog.com/media/en/technical-documentation/data-sheets/ADF7021.pdf

Como hemos indicado anteriormente, es necesario sustituir el TXCO por un valor de los contemplados en el fichero Config.h

- Frequency range 80Mhz - 650Mhz band.
- Transfer rate support 0.05kbps-32.8 kbps.
- Programmable output power from -16dBm to +13 dBm, stepping 0.3dBm.
- Power consumption : Transmit mode (10dBm) 29.2mA; receive mode 26.4mA.
- 2.2-3.6V supply voltage.

SEÑALES UTILIZADAS

La comunicación entre el STM y el RR7021SE son las siguientes (módulo principal):



Y para la versión DUPLEX es la siguiente:

Las líneas TxRxDATA (PB4/PA4) y TxRxCLK (PB3/PA5) es conveniente añadir en serie una Resistencia de 10 Ohm.

Otras señales son:

SERIAL

TXD – PA9 – SERIAL HOST / RXD – PA10 – SERIAL HOST

NEXTION LCD

DISP_TXD – PA2 NEXTION LCD / DISP_RXD – PA3 NEXTION LCD

LEDS

COS_LED – PB15 / PTT_LED – PB14 / P25_LED – PB0 / YSF_LED – PB1

DMR_LED – PB13 / DSTAR_LED – PB12

STATUS

PIN_LED – PC13 STATUS / PIN_DEB – PB9 DEBUGGING

MATERIAL NECESARIO

RF7021SE ADF7021

https://es.aliexpress.com/item/Rf7021se-adf7021-wireless-module-wireless-module/1331907442.html?spm=2114.13010608.0.0.RZlWoz

STM32F103C8T6 STM32

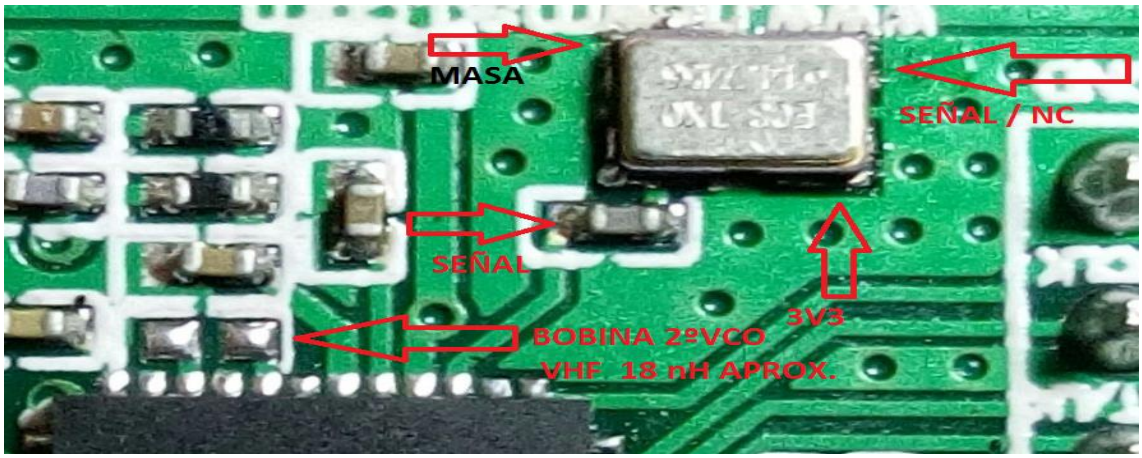
https://es.aliexpress.com/item/1pcs-STM32F103C8T6-ARM-STM32-Minimum-System-Development-Board-Module-For-arduino/32583160323.html?spm=2114.13010608.0.0.RZlWoz

Es necesario sustituir el TCXO por uno de los siguientes por un valor de 14.7456 o 12.2888 Mhz

N.º Ref. Mouser: 520-TXO-3225-14.74T / Osciladores TCXO 14.7456MHz 3.3V HCMOS 2.5ppm

N.º Ref. Mouser: 520-TXO-3225-12.288T / Osciladores TCXO 12.288MHz 3.3V HCMOS 2.5ppm

Tamaño 3.2 mm x 2.5 mm

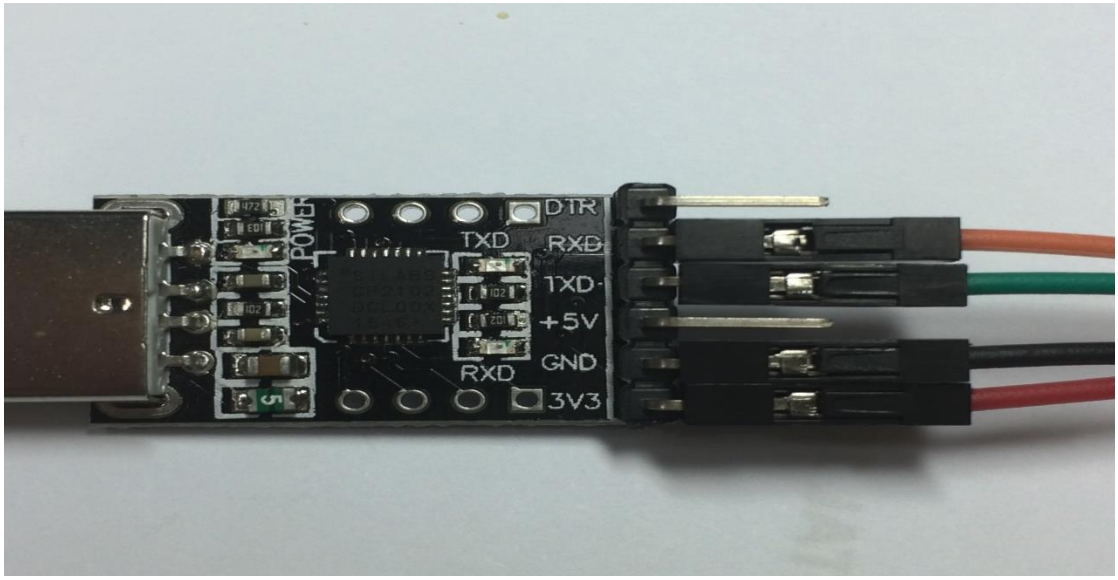


Marca de montaje, punto o chaflanado es el terminal NC (no conectado).

Es necesario un convertidor USB a UART TTL se puede localizar en ebay de forma barata.

Para la primera programación no es necesario poner el conector USB HOST al STM, ya que la placa se alimentara desde el convertidor USB.

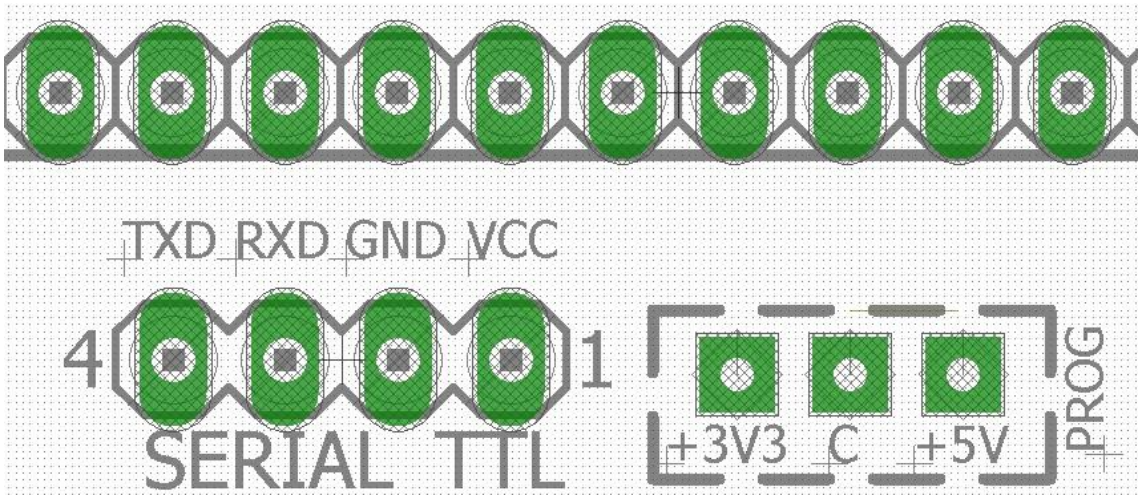
Hemos de puentear el conector PROG en función de la tensión de salida que facilita el USB TTL ADPATER, en este caso la placa es capaz de dar tanto 5 V y 3V3.



PROGRAMACION VIA SERIAL USART

La conexión se realizar a los pines A9 – TXD y A10 – RXD, estos pines han de ser conectados al USB TTL ADAPTER a las señales adecuadas.

STM	USB ADAPTER
A9 TXD	RXD
A10 RXD	TXD
GND	GND
VCC	VCC



- Instalación de las herramientas de programación

cd ~

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

git clone https://git.code.sf.net/p/stm32flash/code stm32flash

cd stm32flash

make clean

make

sudo make install

- Instalación código fuente y librerías

cd ~

git clone https://github.com/juribeparada/MMDVM_HS

cd MMDVM_HS/

git clone https://github.com/juribeparada/STM32F10X_Lib

- Creación y programación del STM

make clean

make

sudo make zumspot-pi

El resultado de la instrucción es la siguiente:

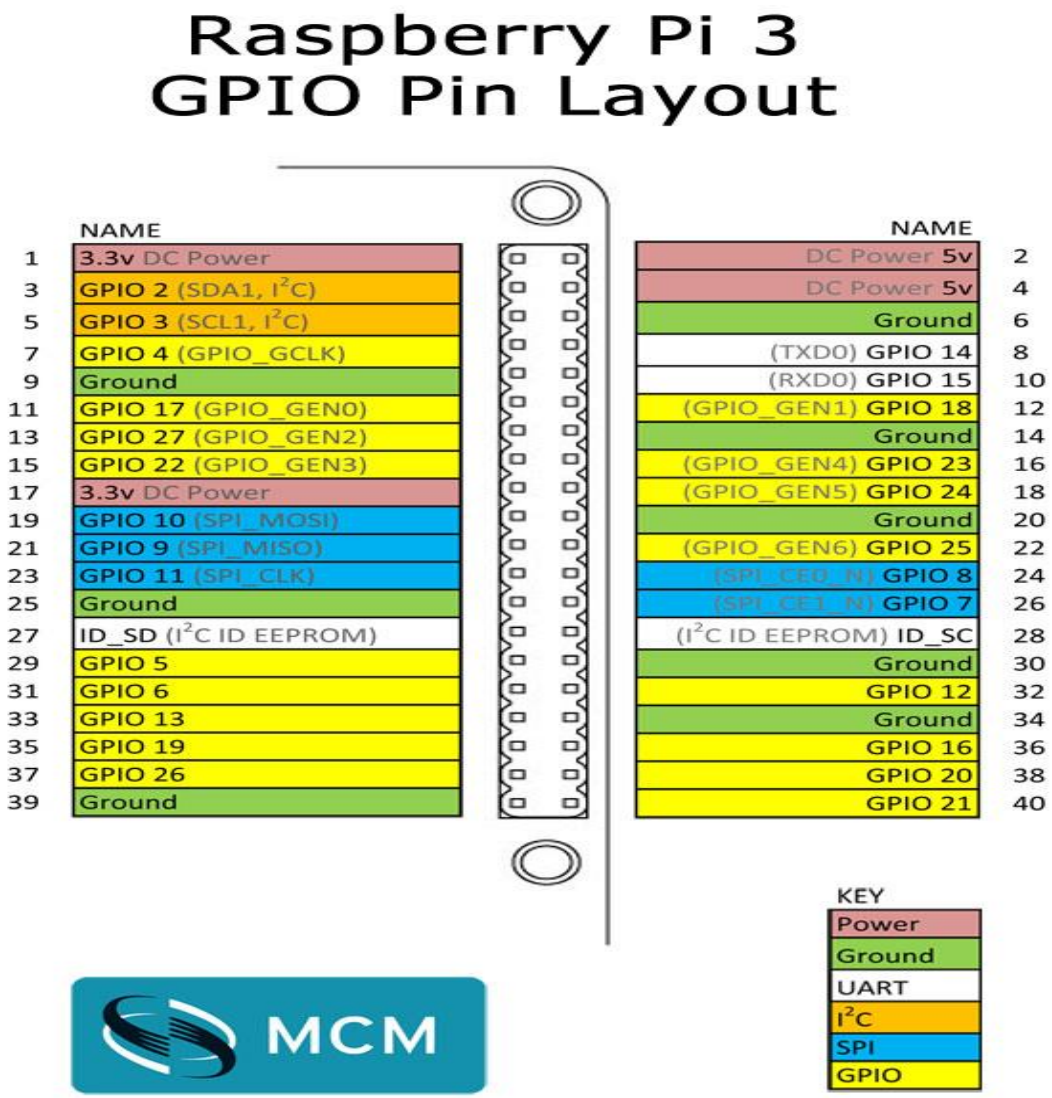
```
pi@raspberrypi:~/MMDVM_HS $ sudo make zumspot-pi
/usr/local/bin/stm32flash -v -w bin/outp.bin -g 0x0 -R -i 20,-21,21:-20,21 /dev/ttyAMA0
stm32flash 0.5

http://stm32flash.sourceforge.net/

Using Parser : Raw BINARY
Interface serial_posix: 57600 8E1

GPIO sequence start
 setting gpio 20 to 1... OK
 delay 100000 us
 setting gpio 21 to 0... OK
 delay 100000 us
 setting gpio 21 to 1... OK
GPIO sequence end

Version      : 0x22
Option 1     : 0x00
Option 2     : 0x00
Device ID    : 0x0410 (STM32F10xxx Medium-density)
- RAM        : Up to 20KiB  (512b reserved by bootloader)
- Flash      : Up to 128KiB (size first sector: 4x1024)
- Option RAM : 16b
- System RAM : 2KiB
Write to memory
Erasing memory
Wrote and verified address 0x0800b48c (100.00%) Done.

Starting execution at address 0x08000000... done.

GPIO sequence start
 setting gpio 20 to 0... OK
 delay 100000 us
 setting gpio 21 to 1... OK
GPIO sequence end


pi@raspberrypi:~/MMDVM_HS $
```

El funcionamiento es simple, cuando se ejecuta sudo make zumspot-pi, desde los pines del GPIO se manda la orden de LOAD y RESET del STM, por tanto la programación es automática, únicamente hay que quitar el puente tal como muestra el dibujo.

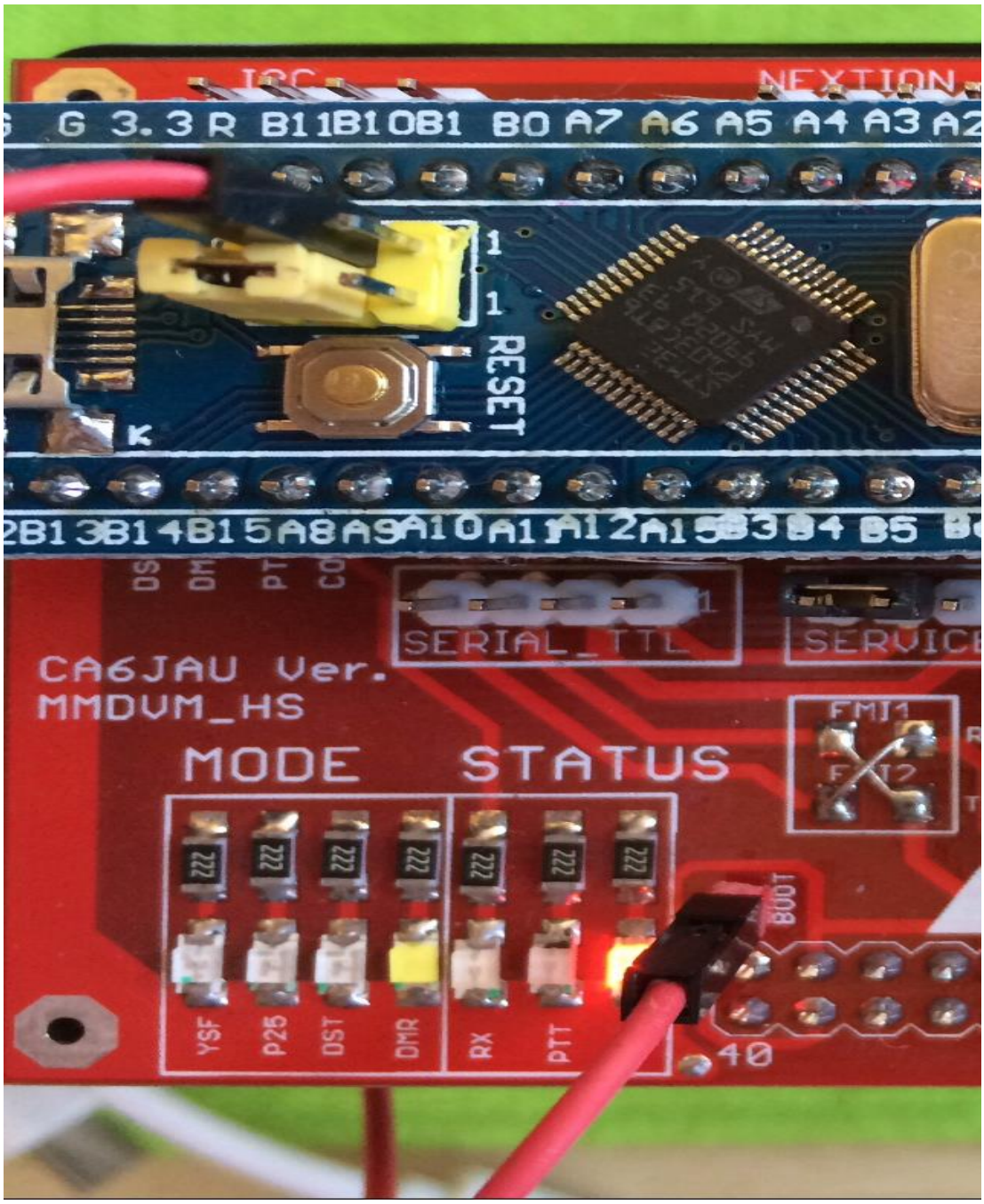
Se usan del GPIO los pines 38 y 40.

GPIO / PIN	STM
GPIO 20 - 38	BOOT
GPIO 21 - 40	RESET

El estado de las señales GPIO desde ventana terminal es **gpio readall**



Cable para conexión de BOOTLOADER, una vez programado, podemos quitar el puente y el JUMPER en su posición.



PROGRAMACION VIA HOST O USB

Necesitamos tener un convertior SERIAL para conectar al USART del STM, desde hay hacemos la carga del HEX (desde Windows podemos hacer ese proceso con una herramienda de STM muy fácil de usar). Abrimos ventana de terminal

cd ~

- Instalación de las herramientas de programación

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

- Instalación código fuente y librerías

git clone https://github.com/juribeparada/MMDVM_HS

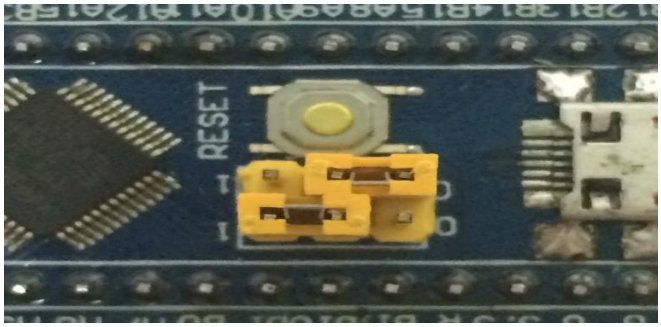
cd MMDVM_HS/

git clone https://github.com/juribeparada/STM32F10X_Lib

make clean

make bl

Suponemos que lo tenemos asignado al USB0 (ls /dev/ - podemos ver los dispositivos)



Mover el puente BOOT0 hacia la izquierda, lado contrario al conector USB y pulsar RESET(el LED deja de parpadear) , entramos en modo carga.

Cambiar BOOT0 de 0 a 1

Pulsar RESET

sudo make serial-bl devser=/dev/ttyUSB0

Cambiar BOOT0 de 1 a 0

El proceso de grabación hace primero la carga de BOOTLOADER y luego el fichero HEX.

Una vez cargado el BOOTLOADER, podemos actualizar, para futuras actualizaciones con la instrucción:

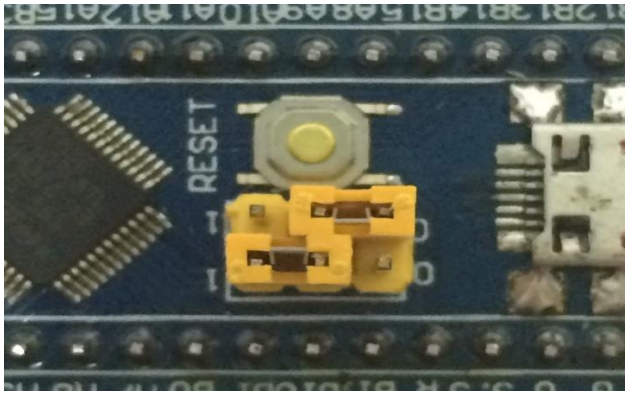
sudo make dfu devser=/dev/ttyACM0

NOTA: IGUAL PERO SIN BOOTLOADER

make clean

make

Suponemos que lo tenemos asignado al USB0



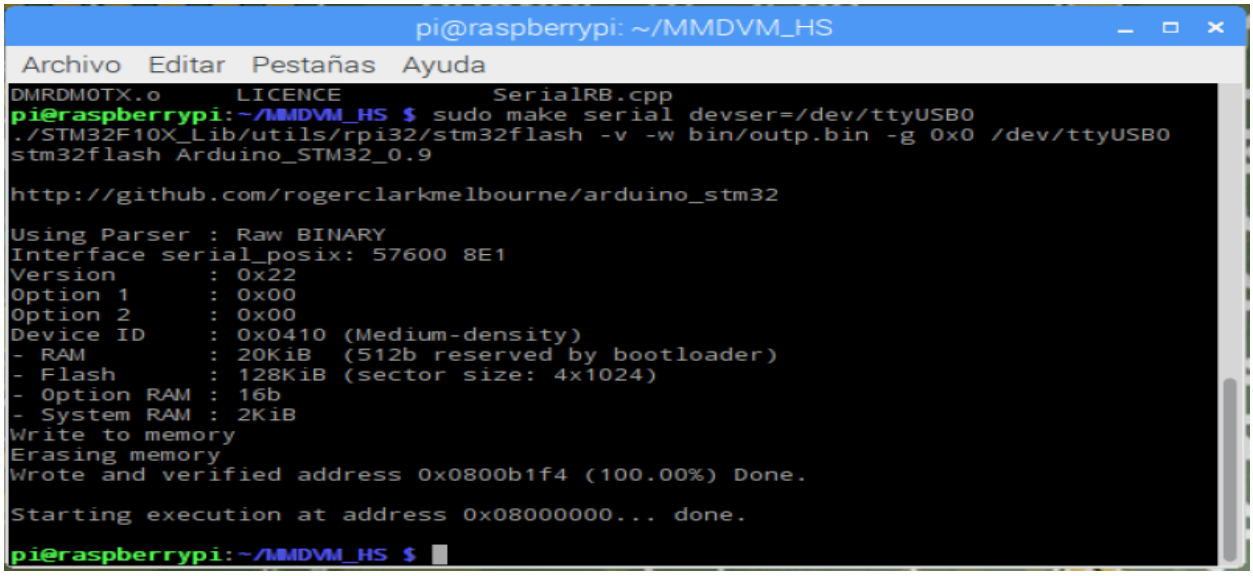
Mover el puente BOOT0 hacia la izquierda, lado contrario al conector USB y pulsar RESET, entramos en modo carga.

Cambiar BOOT0 de 0 a 1

Pulsar RESET

sudo make serial devser=/dev/ttyUSB0

Cambiar BOOT0 de 1 a 0



Edición de los parámetros de funcionamiento:

Desde la ventada de terminal:

cd ~

cd MMDVM_HS/

nano Config.h

#define ENABLE_ADF7021

#define BIDIR_DATA_PIN

#define ADF7021_14_7456 – Valor TCXO

#define STM32_USART1_HOST – Para usar mediante AMA0, PINES GPIO

#define STM32_USB_HOST – Para usar mediante el USB, ACM0

#define ENABLE_SCAN_MODE – Activa el rastreo de modos

#define SEND_RSSI_DATA - Permite el envío de la señal RRSI

#define SERIAL_REPEATER – Salida de datos para pantalla NEXTION

make clean – borra lo compilado anteriormente

make – genera fichero HEX

make bl – genera fichero HEX con BOOTLOADER

make zumspot-pi – serial conectado al USART1

make serial devser=/dev/ttyXXX – serial conectado a USB HOST

make serial-BL devser=/dev/ttyXXX – igual con carga del BOOTLOADER

make dfu devser=/dev/ttyACM0 – carga desde ACM0, requiere BOOTLOADER

VERSION DUPLEX MMDVM_HS

https://github.com/juribeparada/MMDVM_HS/tree/duplex

https://github.com/juribeparada/MMDVM_HS/blob/duplex/BUILD.txt

CONEXIONADO PINES DE LA PLACA

- **PANTALLA NEXTION**
  La pantalla NEXTION puede trabajar de forma directa conectada al STM, activando en el fichero Config.h la opción de:

#define SERIAL_REPEATER – Salida de datos para pantalla NEXTION

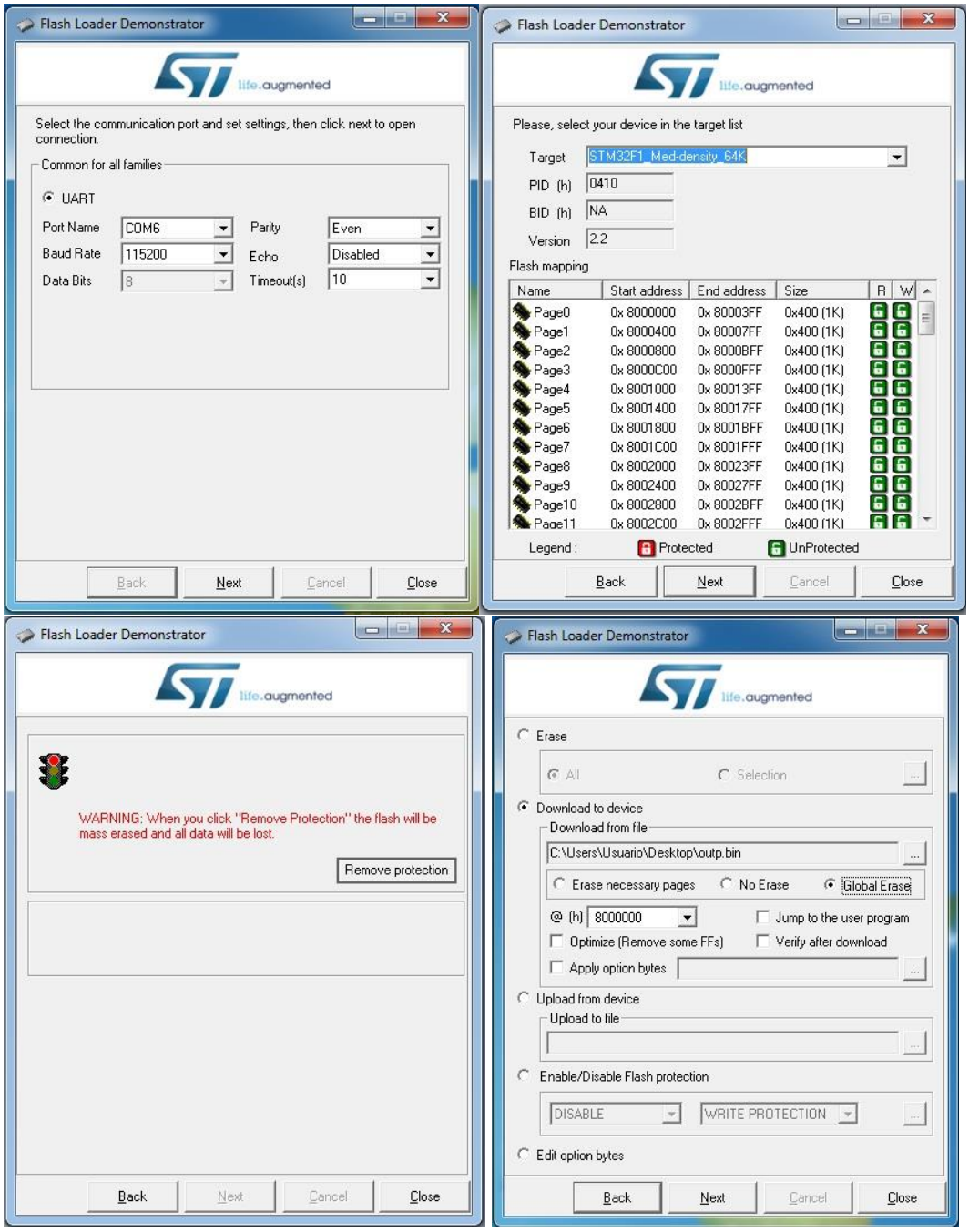
El conexionado es el siguiente:

PLACA	GND	5V	RXD	TXD
NEXTION	NEGRO	ROJO	TXD	RXD

- **CONECTOR I2C – SIN SERVICIO**
- **CONECTOR SERIAL – Conector para programación STM usando USB TTL ADAPTER.**
- **CONECTOR SERVICE PROGRAMACION – 3V3 o 5V**
- **CONECTOR POWER 5V – permite seleccionar desconectar la tensión de 5V del GPIO, para tareas de mantenimiento y poder alimentar de forma externa.**
- **CONECTOR I2C GPIO – CONECTOR PANTALLA GPIO -, para conectar pantalla I2C directamente al GPIO Raspberry.**

FLASH LOADER DEMONSTRATOR

Aplicación de Windows para recarga de BIN/HEX desde el SERIAL TTL ADAPTER

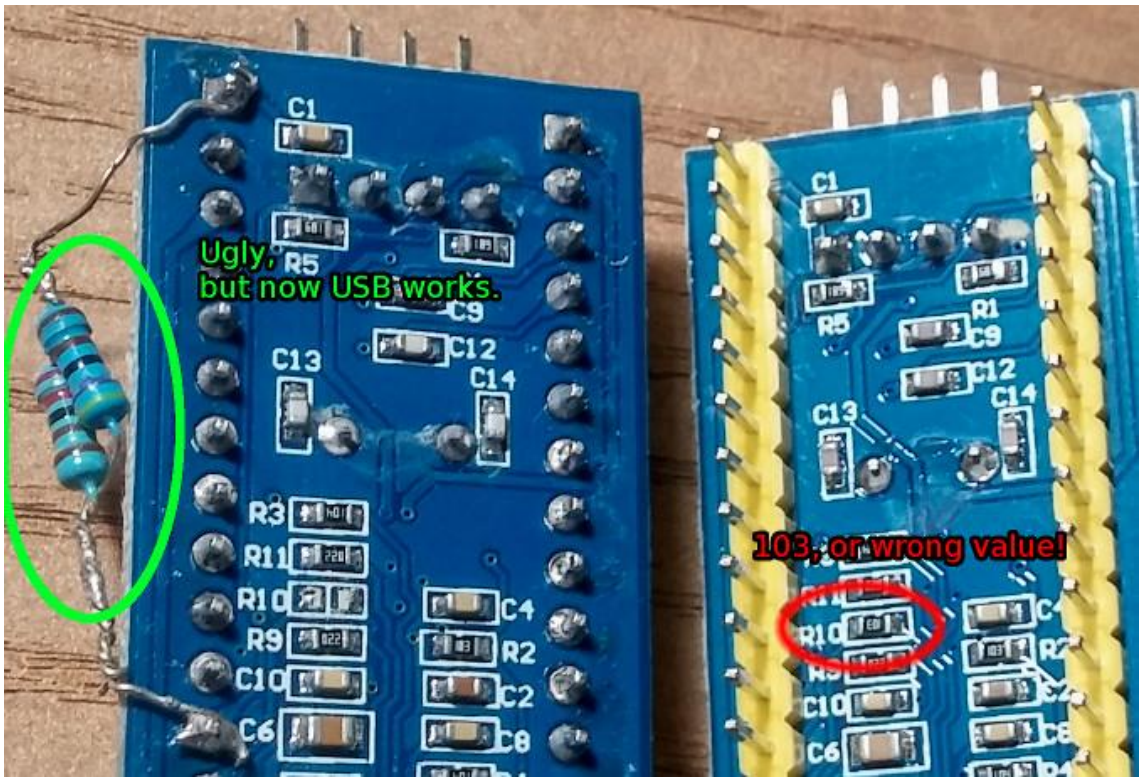


MODIFICACION SMT “PROBLEMAS USB EN WINDOWS”

En algunas STM existe un problema con la comunicación del USB HOST con el USB del ordenador.

http://wiki.stm32duino.com/index.php?title=Blue_Pill

Es debido a un valor no adecuado en la R10, el valor correcto debería ser 1K5.



Se soluciona añadiendo entre los pines 3V3 y A12 una resistencia de 1K8, siendo la resistencia equivalente de unos 1K5. Lógicamente, se puede cambiar la R10 por el valor correcto en formato SMD.

APLICACIÓN WINDOWS

La forma usual de trabajar con este tipo de dispositivos es desde LINUX. Manel, EA3EIZ (http://www.associacioader.com/index.php) ha realizado un programa "lanzadera" que permite trabajar desde WINDOWS.

En la web anterior teneis la aplicación y el manual de instalación/uso.

El programa hace las llamadas al mmdvm, ircDDBGateway y YSF, que han sido compilados para usar en WINDOWS.

Instrucciones para la configuración del programa FTM-400D

1º Instalar el programa FTM-400D-TOTAL.exe

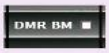
2º Instalamos el ircDDBGateway desde el directorio
C:/Archivos de progra/EA3EIZ/Menu_Radio/ircDDBGateway_32.exe
ó en C:/Archivos de progra (x86)/EA3EIZ/Menu_Radio/ircDDBGateway_32.exe

3º Abrimos el programa desde el icono que nos ha creado en el escritorio

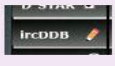
4º Hacemos la configuraciones desde el botón

5º Configurar DMR+
click en check DMR+ y elegimos el servidor

6º Comprobamos los datos que han ingresados
y hacemos alguna modificaciones

7º Configurar DMR BM
click en check DMR BM 

8º Comprobamos los datos que han ingresados
y hacemos alguna modificaciones

9º Configurar YSF
click en

10º Configurar ircDDB
click en