MMDVM FOR DUMMIES

MMDVM è attualmente un sistema in via di sviluppo, l'autore è Jonathan Naylor G4KLX già noto per il software D-star repeater e Ircddbgateway. E' un sistema multipiattaforma, supporta DMR, D-STAR e Fusion anche se la parte più interessante è quella DMR.
Principalmente MMDVM serve per la gestione di un ripetitore auto costruito anche se esiste la possibilità di utilizzarlo per un sistema hotspot in simplex perdendo però una delle caratteristiche basilari del DMR, la comunicazione contemporanea su due slot.
Per chi è interessato ad una soluzione hot-spot di accesso casalingo alla rete DMR consiglio di guardare altri sistemi più semplici e nati per quello scopo come ad esempio il progetto DV4MINI o il progetto DVMEGA, ciò non toglie che comunque MMDVM utilizzato in DMO mode e quindi in simplex con una sola radio funzioni comunque molto bene, ovviamente con tutte le limitazioni del caso.
A proposito di DVMEGA, è possibile utilizzare la parte host di MMDVM (il software MMDVMHost) collegato ad una scheda DVMEGA, anziché la classica Arduino due, realizzando di fatto un HotSpot DMR ma qui tratteremo solo la parte relativa alla messa in servizio di un vero ripetitore auto costruito.

Il sistema MMDVM di gestione di un ripetitore è diviso in due sezioni:

- La parte modem sviluppata su piattaforma Arduino 2 più una interfaccia tra Arduino e la parte RF composta da un filtro attivo passa basso.
- La parte Host di interfacciamento con la rete che gira su piattaforma Linux o windows. (Ottima soluzione l'economica Raspberry)

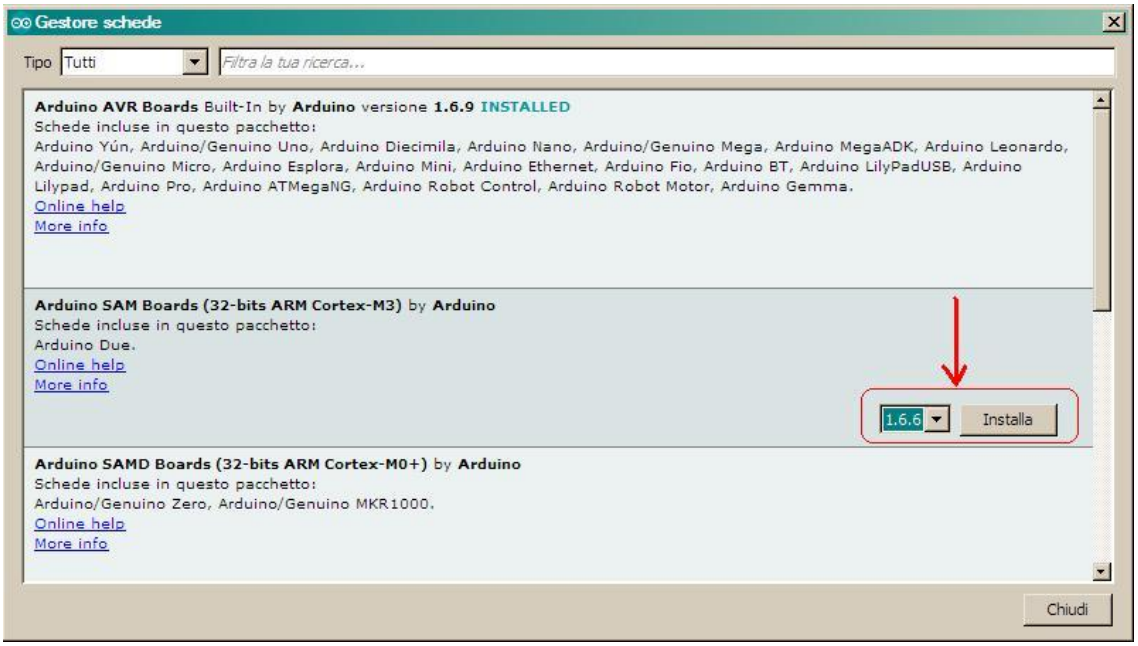
Il sito di riferimento è l'omonimo gruppo Yahoo praticamente unica fonte di informazioni sul sistema: https://groups.yahoo.com/neo/groups/mmdvm/info
Sul gruppo yahoo nella sezione file si trovano delle versioni di software per la scheda Arduino e per la parte host su computer già compilate, il mio consiglio è di partire con queste che teoricamente sono testate e funzionanti.
In alternativa chi invece vuole provare il sistema man mano che viene sviluppato può scaricare i sorgenti da GitHub al seguente indirizzo web:
https://github.com/g4klx
ovviamente essendo sorgenti anche la parte host va compilata, con Visual Studio su piattaforma windows o con C++ su piattaforma linux.
Per questo consiglio di partire con l'ultima versione presente sul gruppo Yahoo, oltre ad essere testata trovate anche l'eseguibile per windows e linux e vi complicate meno la vita nelle fasi iniziali.

Preparazione scheda Arduino due

Vediamo a questo punto come rendere tutto operativo, partiamo con la preparazione della scheda Arduino due.
Una piccola premessa, di scheda Arduino oltre alla originale esistono anche vari cloni Cinesi, personalmente vi consiglio l'originale anche se comunque pare che anche le Cinesi funzionino discretamente.

Primo passaggio scaricate ed installate l'interfaccia di programmazione di Arduino (IDE), la versione da scaricare è la 1.6.7, una volta installata vanno scaricate le librerie per Arduino due non presenti di default, per farlo andate su:
Strumenti -> Scheda -> Gestore schede

Nella finestra che si apre cliccare su “Arduino Sam boards (32-bits ARM Cortex-M3) “ sulla destra si apre un menu a discesa da cui selezionare la versione.
Selezionate la V. 1.6.6 e cliccate su “INSTALLA” (Vedere immagine seguente)



Terminata l’installazione tornate su:
Strumenti -> Scheda
potrete adesso selezionare dal menu a discesa la scheda “**Arduino Due (Programming Port**)” che andrà a sostituire la Arduino Uno presente di default.
A questo punto la parte di configurazione dell’IDE di Arduino si può dire conclusa.

Prima di scaricare i sorgenti e programmare la scheda c’è da fare però ancora un passaggio, per poter compilare mmdvm sono necessarie le librerie CMSIS DSP che in arduino non sono caricate di default, per abilitarle va editato il file “*platform.txt* ” che si trova normalmente nel seguente percorso:
C:\Users\UtenteWindows\AppData\Local\Arduino15\packages\arduino\hardware\sam\1.6.6
Attenzione perché il percorso varia in base alla versione dell’ide, questo è riferito alla 1.6.7

al posto di “UtenteWindows” dovete mettere l’utente con cui entrate nel sistema.
Dovete cercare la seguente sezione:

Combine gc-sections, archives, and objects
la riga da modificare è quella immediatamente successiva:
recipe.c.combine.pattern="{compiler.path}{compiler.c.elf.cmd}" -mcpu={build.mcu} -mthumb {compiler.c.elf.flags} "-T{build.variant.path}/{build.ldscript}" "-Wl,-Map,{build.path}/{build.project_name}.map" {compiler.c.elf.extra_flags} -o "{build.path}/{build.project_name}.elf" "-L{build.path}" -Wl,--cref -Wl,--check-sections -Wl,--gc-sections -Wl,--entry=Reset_Handler -Wl,--unresolved-symbols=report-all -Wl,--warn-common -Wl,--warn-section-align -Wl,--start-group "{build.path}/core/syscalls_sam3.c.o" {object_files} "{build.variant.path}/{build.variant_system_lib}" "{build.path}/{archive_file}" -Wl,--end-group -lm –gcc

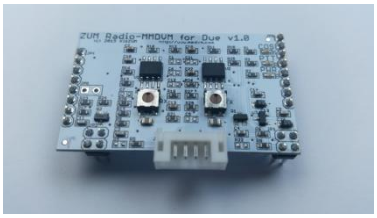
e va modificata come segue:
recipe.c.combine.pattern="{compiler.path}{compiler.c.elf.cmd}" -mcpu={build.mcu} -mthumb {compiler.c.elf.flags} "-T{build.variant.path}/{build.ldscript}" "-Wl,-Map,{build.path}/{build.project_name}.map" {compiler.c.elf.extra_flags} -o "{build.path}/{build.project_name}.elf" "-L{build.path}" -Wl,--cref -Wl,--check-sections -Wl,--gc-sections -Wl,--entry=Reset_Handler -Wl,--unresolved-symbols=report-all -Wl,--warn-common -Wl,--warn-section-align -Wl,--start-group "{build.path}/core/syscalls_sam3.c.o" {object_files} "{build.variant.path}/{build.variant_system_lib}" **"{build.system.path}/CMSIS/CMSIS/Lib/ARM/arm_cortexM3l_math.lib"** "{build.path}/{archive_file}" -Wl,--end-group -lm -gcc

La modifica consiste nell’aggiunta verso la fine della riga del seguente testo che sopra ho evidenziato:
"{build.system.path}/CMSIS/CMSIS/Lib/ARM/arm_cortexM3l_math.lib"

MMDVM FOR DUMMIES

Siccome è sufficiente dimenticare una virgola e vi ritrovate che non funziona nulla vi consiglio di aprire il file: *BUILD.txt*
che trovate nei sorgenti MMDVM per Arduino dove è ben illustrata la procedura che ho descritto sopra e fate un copia/incolla di tutta la riga dal BUILD.txt al platform.txt sostituendo la riga originale.
Naturalmente tutto il resto presente nel file platform.txt va lasciato invariato.
C'è ancora una cosa da verificare, la piedinatura della scheda di interfaccia con la parte RF che avete inserito sopra alla Arduino, ad oggi esistono tre varianti:

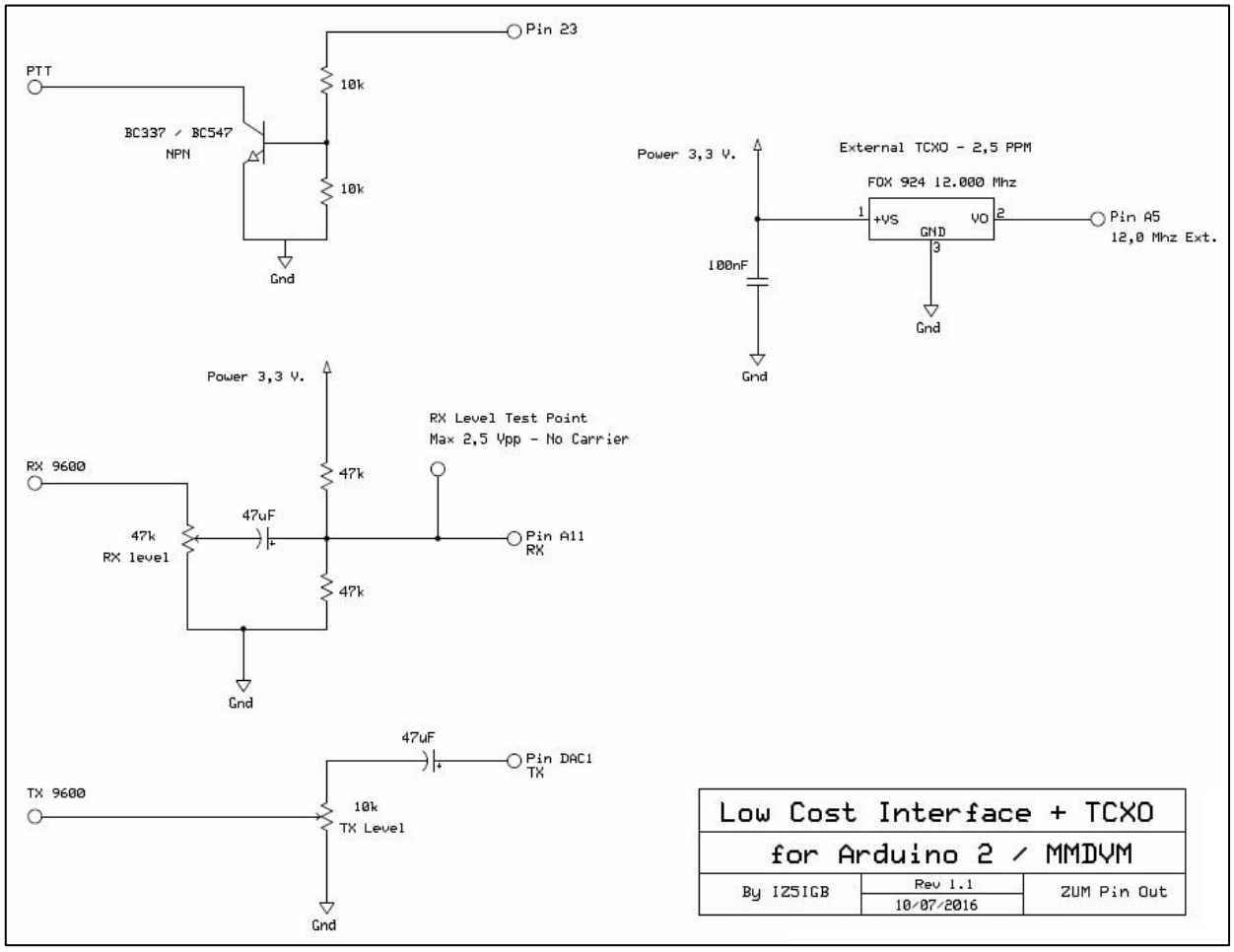
ARDUINO DUE PAPA



ARDUINO DUE ZUM

ARDUINO DUE NTH

Se l'avete acquistata dalle foto già sapete quale è, se invece l'avete autocostruita avrete copiato da uno degli schemi delle tre varianti per cui anche in quel caso potete risalire alla famiglia originale per la piedinatura, dico questo perché a seconda della scheda dovrete, all'interno dell'ide di Arduino, aprire MMDVM visualizzare il file *Config.h* e de commentare quella di vostro interesse commentando tutte le altre col simbolo "//". Sempre nello stesso file va settato l'utilizzo di un eventuale TXCO esterno sempre de comentando l'apposita riga.
Per chi vuole costruirsi tutto da solo cito la cosidetta "Interfaccia Low Cost" ideata da Graziano IZ5IGB che pur non avendo i filtri funziona molto bene, lo schema è il seguente:



La piedinatura della “low cost” corrisponde alla “Arduino Due Zum”.

A questo punto siete pronti per programmare la Arduino due, mi raccomando di verificare che la porta COM configurata nell’ide di Arduino sia effettivamente quella dove è presente la scheda (verificare tramite il pannello di controllo) e di collegare la Arduino alla usb del PC tramite la mini usb “Programming Port” (quella vicina al connettore di alimentazione) che è la configurazione di default.

Se la programmazione è andata a buon fine nell’ide non devono comparire errori e il Led “L” deve iniziare a lampeggiare al ritmo di circa un secondo.

Preparazione software Host su PC

SISTEMI WINDOWS

Il software che gira sul PC si chiama *MMDVMHost.exe*, lo trovate già compilato nella sezione file del gruppo yahoo ed è un unico file eseguibile quindi non necessita di nessuna installazione, su sistemi Windows (testato su Win7, Win8 e Win10) va avviato in una finestra DOS passandogli come parametro il nome del file di inizializzazione che di default è *MMDVM.ini*, i due files andranno messi entrambi in una cartella facilmente accessibile all’apertura della finestra DOS e il tutto va avviato col seguente comando al prompt del DOS:
MMDVMHost MMDVM.ini
(per i dettagli sull’impostazione del file MMDVM.ini vedere più avanti nella documentazione)
Sul computer dovrà essere stato preventivamente installato il pacchetto “Visual C++ Redistributable for Visual Studio 2015” prelevandolo dal sito Microsoft.
Alla data di oggi 04-11-2016 sul gruppo Yahoo è presente una vecchia release di settembre 2016, pur non essendo aggiornata va benissimo per chi vuole provare DMR e D-STAR, per chi vuole invece utilizzare le versioni più aggiornate attualmente non c’è altro da fare che scaricare i sorgenti e compilarli.
I sorgenti vanno scaricati da GitHub al seguente link:
https://github.com/g4klx/MMDVMHost
Per la compilazione va installato Visual Studio 2015 che può essere prelevato dal sito Microsoft o anche dall’area file del gruppo Yahoo.

SISTEMI LINUX E RASPBERRY

Per Linux non esistono per ora file compilati o pacchetti di installazione per cui si dovranno scaricare i sorgenti e compilarli. Non è comunque una procedura complicata a patto di avere una minima conoscenza sull’utilizzo di Linux da riga di comando.

Visto che dobbiamo scaricare i sorgenti tanto vale scaricare gli ultimi aggiornati presenti su GitHub, attenzione a non mischiare le versioni, se compilate l’ultima versione di MMDVMHost usatelo con l’ultima versione di MMDVM sulla scheda Arduino, versioni differenti possono funzionare come no per cui meglio non rischiare. Il tutorial che segue è riferito alla raspberry, ma è applicabile a qualsiasi sistema linux.
Creiamo una cartella a piacere dove lavorare, chiamiamola ad esempio lavoro:
sudo mkdir lavoro
spostiamoci al suo interno:
cd lavoro
e scarichiamo i sorgenti:
sudo git clone https://github.com/g4klx/MMDVMHost
con questo comando abbiamo “clonato” la cartella dei sorgenti dal repository comprese tutte le eventuali cartelle necessarie per la compilazione. A questo punto spostiamoci all’interno della cartella dei sorgenti:
cd MMDVMHost
e avviamo la compilazione col comando:
sudo make

Se tutto va a buon fine senza errori all'interno della cartella troveremo l'eseguibile compilato che si chiamerà *MMDVMHost* (senza estensione… siamo in linux!).
In via teorica si potrebbe avviare anche li dove sta io però consiglio di spostare i files necessari in un'altra cartella lasciando quella dei sorgenti pronta per compilazioni successive. I files da copiare sono tre:
MMDVMHost (L'eseguibile)
MMDVM.ini (Il file di inizializzazione)
DMRIds.dat (Il file di conversione ID DMR – CALL)

Prima di avviare il sistema va configurato il file di inizializzazione MMDVM.ini questo ovviamente sia per sistemi Windows che Linux, apritelo con un editor ed inserite i vostri dati, mi raccomando dati veritieri, ID DMR mettete il vostro ID non 123456, non facciamoci riprendere dai soliti puristi del DMR che però in questo caso hanno tutte le ragioni del mondo! E già che siamo in argomento, non fate prove e tarature sul TG 222 o altri TG Regionali trafficati, finché il sistema non è a posto e ben funzionante cerchiamo di non creare disturbi sulla rete.
Il file .ini è abbastanza semplice da configurare, è sufficiente completarlo con i propri dati, nella sezione:
[Modem]
inserite la porta com corretta rispettando la sintassi di esempio presente, per i sistemi Windows es.:
Port=\\.\COM35
per i sistemi Linux è sempre uguale, de commentate:
Port=/dev/ttyACM0

Nella sezione [DMR Network] mettete:
[DMR Network]
Enable=1
Address=95.110.161.52
Port=62031
# Local=3350
Password=passw0rd
Slot1=1
Slot2=1
Debug=1
Attenzione! In passw0rd non c'è una O maiuscola ma zero.
L'indirizzo IP è quello del BrandMeister Italiano.
Completato l'editing del file con tutti i propri dati salviamo e siamo pronti per avviare il software.

Per avviare il sistema il comando è il seguente:
sudo ./MMDVMHost ./MMDVM.ini

Avremo come output sullo schermo tutti i messaggi del sistema compresi quelli di debug che in questa fase sono molti.
Naturalmente se usciamo dalla console il programma si arresterà, quando tutto sarà funzionante andrà avviato come servizio ma questo tutorial è "For Dummies!" il resto in una futura guida "Avanzata" ;-))

Aggiornamento del 04-11-2016, sul sito ik1whn.com è presente il documento "MMDVM Advanced" dove l'argomento della partenza all'avvio viene esaurientemente spiegato.

Qualche accenno sulla parte RF

Una delle parti critiche del sistema sono le radio utilizzate, in via teorica tutte le radio con porta dati a 9600 o comunque quelle già testate col sistema D-Star dovrebbero funzionare anche con MMDVM. In pratica non è proprio così, il DMR è molto più critico e non tutte le radio sono idonee. Al link seguente trovate un elenco senz'altro incompleto delle radio testate:
https://bm.pd0zry.nl/index.php/Homebrew_Repeaters
La taratura del livello del segnale dal ricevitore alla scheda Arduino e dalla scheda Arduino al trasmettitore è la parte più critica e per alcuni molto difficoltosa.
Per tarare i livelli ci sono due trimmer, uno per linea, inoltre ci sono anche i valori RXLevel e TXLevel nella sezione MODEM del file MMDVM.ini che spostano il punto di intervento dei trimmer se vanno a posizionarsi tutti da una parte.
Per effettuare una prima taratura senza strumentazione si può procedere come segue:
Portare entrambi i trimmer circa a metà corsa.
Scaricare da GitHub o dalla sezione file del gruppo yahoo il software MMDVMCal.
MMDVMCal permette di mandare in trasmissione in modalità D-Star il nostro ripetitore e di variare in tempo reale da tastiera vari parametri, per un elenco completo delle funzioni lanciarlo e premere il tasto "h". Il programma va lanciato in una finestra dos o da terminale in Linux passandogli come parametro la porta seriale dove si trova la scheda Arduino, per sistemi Linux il comando di avvio sarà pertanto:
./MMDVMCal /dev/ttyACM0

Con MMDVMCal vedremo di trovare il giusto valore di PTTInvert , TXInvert e TXLevel da inserire nel file MMDVM.ini, in pratica la catena di trasmissione del nostro sistema.
Lanciato il programma e premendo la barra spaziatrice si manda in tx il sistema in modalità D-Star.
Con questa funzione se il valore di PTT non è corretto il trasmettitore va in tx ma arduino non trasmette nulla per cui con la radio D-Star non decodificate nulla, stessa cosa se il valore di inversione TX è errato, la radio D-Star non decodifica. Avrete quindi da giocare variando questi due parametri finché non si trova la combinazione corretta.
Sintonizziamo una radio D-Star (meglio gli ultimi ID-51 o ID-5100 perché hanno i filtri più stretti e quindi permettono una taratura migliore) sulla frequenza di trasmissione del nostro sistema, la radio D-Star deve decodificare la stringa di testo "MMDVM / TEST" se ciò non avviene e il simbolo DV lampeggia alternandosi con la scritta FM provate a invertire il PTT con il tasto "P" premete nuovamente la barra spaziatrice e verificate se la radio decodifica, se così non fosse provate a invertire il valore TX col tasto "I" e/o variate il livello TX con i tasti "T" (Aumenta) e "t" (Diminuisce).
Fate vari tentativi con livelli diversi e con PTT normale o invertito e TX normale o invertito finché la radio d-star comincia a decodificare. A questo punto trovate il livello minimo e massimo (con i tasti "t" e "T") a cui avviene ancora la decodifica e poi calcolate il valore medio.
Esempio:
Valore minimo di decodifica 26
Valore massimo di decodifica 75
(75+26):2=50.5
Andremo per cui ad impostare in MMDVM.ini TXLevel=50

Uscendo da MMDVMCal col tasto "Q" vi appariranno gli ultimi valori impostati da inserire in MMDVM.ini. Uscite da MMDVmCal e, seguendo i valori del nostro esempio, prima di lanciare MMDVMHost settate nel vostro MMDVM.ini TXLevel=50 e PTTInvert=0 se ricevete con ptt invert OFF oppure PTTInvert=1 se ricevete con ptt invert ON.
A questo punto la regolazione della parte TX dovrebbe essere accettabile anche per il DMR.
Per dare una regolazione grossolana sul lato RX partite con un valore di 95 in RXLevel nel file .ini, partiamo con un valore molto alto perché ciò ci consente di avere un valore di soglia ADC superiore, lanciate ancora MMDVMCal e trasmettete con una radio D-Star, regolate il trimmer RX per un valore di "diff" che sia compreso tra 400 e 600 che corrisponde ad un livello di circa 1Vpp per il D-Star e circa 2Vpp per il DMR.

Uscite da MMDVMCal e avviate MMDVMHost, mandate in trasmissione la vostra radio DMR (Radio DMR impostata sul TG9 in modo da non portare disturbi sulla rete!) dovreste essere in grado di transitare sul sistema, se così non fosse provate a invertire lo stato di RXIvert nel file MMDVM.ini e a variare la posizione del trimmer RX lasciando fermo il trimmer TX.
Una volta che riuscite a vedere il vostro traffico sul monitor si può procedere con la taratura del livello di ricezione mediante il trimmer RX, se il livello in ingresso alla scheda Arduino è troppo alto sullo schermo appare la scritta “MMDVM ADC levels have overflowed”.
Se mentre trasmettete la scritta non appare mediante il trimmer RX aumentate il livello fino a farla apparire quindi tornate indietro lentamente finché non viene più visualizzata o viene visualizzata molto raramente, a questo punto sarete sicuri di non mandare in overflow l’arduino ma nello stesso tempo di avere un discreto livello di segnale.
Siccome può essere difficoltoso visualizzare l’avvertimento in mezzo a tutto l’output sullo schermo per chi usa Linux consiglio di utilizzare il seguente comando per lanciare MMDVMHost in modo da filtrare solo ciò che ci interessa:
./MMDVMHost ./MMDVM.ini | *grep overflow*
Anche qui nell’eventualità che la posizione del trimmer RX fosse tutta da un lato variate il valore di RXLevel nel file MMDVM.ini in modo da spostare il punto di intervento ricordando che comunque, compatibilmente con la posizione del trimmer è meglio avere un valore il più alto possibile, ripetete quindi la taratura.

A questo punto avete terminato, sarete in grado di utilizzare il vostro sistema in modalità DMR e D-Star.
Ovviamente la miglior taratura sarà sempre quella strumentale ma per chi non dispone di strumentazione questo sistema permette di rendere il vostro MMDVM utilizzabile per le prime prove e posso confermare che nel mio caso non è molto distante da quella ideale.
Aggiornamento del 04-11-2016, sul sito ik1whn.com è presente il documento “MMDVM Advanced” dove è spiegato come effettuare la taratura della deviazione utilizzando una comune chiavetta SDR.

Problemi noti

Come già detto non tutti gli RTX che funzionano in D-Star funzionano anche in DMR, questo potrebbe essere uno dei problemi se il vostro sistema non funziona bene e l’audio si sente male e/o distorto insieme ovviamente ad una taratura dei livelli non ottimale.
Altro problema conosciuto per il DMR è la poca precisione del quarzo della scheda Arduino.
la temporizzazione dei due Timeslot viene ricavata dal clock di Arduino che purtroppo non eccelle per la sua precisione, se fosse solo un problema di poca precisione sarebbe sufficiente prevedere una correzione via software cosa che è stata implementata ma il problema è che oltre ad essere poco preciso ha una deriva che con certe schede Arduino è abbastanza accentuata e che porta dopo un certo numero di secondi di trasmissione ad un degrado irreversibile dell’audio perché i due timeslot shiftano e tendono ad accavallarsi e a quel punto l’unica soluzione è lasciare il ptt ed eventualmente riprendere.
Il problema è più o meno grave a seconda della scheda Arduino, con alcune si presenta abbondantemente dopo 3 minuti quindi poco avvertibile in un normale qso, con altre però si presenta anche dopo solo 30 secondi.

La soluzione è adottare un clock esterno con un TXCO con una precisione di almeno 2,5ppm cosa prevista nelle ultime versioni del software per Arduino e che va abilitata (se presente il clock esterno) prima di programmare la scheda come detto all’inizio.

Altro problema che può inficiare la qualità dell’audio della nostra comunicazione è la qualità della rete che trasporta i dati e quindi della linea ADSL, non è tanto importante avere elevata larghezza di banda quanto avere una latenza minima. È pur vero che normalmente una ADSL veloce è anche di buona qualità e permette in condizioni di stress, quando la linea è utilizzata per altri servizi, di avere comunque sufficiente spazio per le nostre comunicazioni ma per trasportare efficacemente i nostri frame è sufficiente una 7Mbit di buona qualità e con una bassa latenza.

MMDVM FOR DUMMIES

Il protocollo di trasporto dello streaming DV è UDP che non garantisce la consegna ma è stato scelto per la sua tempestività nel veicolare i dati. A differenza di altre applicazioni internet il problema della gestione dello streaming è notevole, se si interrompe il flusso anche per poche frazioni di secondo in mancanza di correzioni software l'audio si interrompe e si sentono i classici disturbi sull'audio, inoltre se l'interruzione è un po' più lunga nel caso del D-Star e del System Fusion la portante cade.
Non è ovviamente percorribile la strada di inserire un buffer perché introdurremmo un ritardo inaccettabile per cui, oltre all'efficacia delle correzioni software per cercare di tamponare il problema di frame persi, è importante che il problema non si presenti adottando per i nostri sistemi connessioni di rete con bassa latenza.

Per testare la connessione potete fare una serie di PING all'indirizzo IP del server a cui il sistema DV si deve collegare, ad esempio al BrandMeister Italiano:
ping 95.110.161.52
Ovviamente più il valore è basso migliore è la connessione, valori ottimali se inferiori a 30ms, oltre i 55-60ms vi possono essere seri problemi soprattutto se la alta latenza è associata a jitter e perdita di pacchetti, oltre 90-100ms il DMR non funziona più. Attenzione, se il ping è su valori ottimali ma se ne perde anche uno solo ogni 60-70 secondi è anche peggio, situazione quest'ultima che accade spesso se si adottano tratte con connessioni wifi casalinghe molto trafficate che sono assolutamente da evitare.

Questo è tutto per ora, spero che questa piccola guida possa esservi d'aiuto nei primi approcci con mmdvm, un sistema che nonostante la sua giovane età sta riscuotendo molto interesse e che personalmente mi sta dando grosse soddisfazioni.

Un particolare ringraziamento a Graziano IZ5IGB che mi ha dato lo spunto per realizzare questa piccola guida e per le preziose informazioni tecniche che con grande ham spirit mi ha fornito.