18/12/2011 **Marco Ducco IK1PXM Torino**

Generatore triplo CW/RTTY/PSK31

Il generatore fornisce tre segnali contemporanei nello stesso canale audio. Per me è stato gratificante progettarlo, ma non sono poi riuscito a trovare per esso un utilizzo permanente.
Ho progettato il generatore, inizialmente, per provare in modo ripetibile dei decodificatori; successivamente, volevo collegarlo a un TX FM o SSB per i 2m, per costituire un beacon sperimentale che potesse permettere il confronto diretto didattico fra i modi, ma ho poi desistito per non aumentare le emissioni automatiche già presenti in banda. Ultimamente, lo adopero saltuariamente negli incontri HAM come generatore di suoni acchiappacuriosi.

Hardware

Il generatore è nato a pezzi: prima il generatore PSK31 e il generatore doppio CW/RTTY separati, poi uniti utilizzando piedini diversi dello stesso microcontrollore (uC). Infine, accorpamento dei programmi in uno unico. Lo schema elettrico della parte analogica del generatore doppio CW/FSK, cui fanno riferimento le sigle citate nel testo, è riportato nella fig.1. Lo schema elettrico della parte digitale, utilizzante una scheda Arduino Mini 04 dell'omonimo progetto open source (www.arduino.cc) è riportato nella fig.2.
Anche il generatore doppio è nato ed è stato provato a pezzi: prima ho realizzato il generatore audio la cui uscita viene modulata da un elemento di un interruttore quadruplo CD4066 (IC4a) comandato da una uscita del uC, su cui avevo caricato un programma che genera continuamente una codifica morse.
Il generatore è costituito da un circuito con due amplificatori operazionali (IC1a e IC1b) che genera un segnale triangolare nel campo 900 – 3500 Hz; rispetto a un oscillatore sinusoidale, il circuito presenta il vantaggio di poter variare la frequenza agendo su un solo potenziometro (P1) mantenendo l'ampiezza rigorosamente costante. Ho adoperato il classico doppio amplificatore operazionale LM358N, funzionante anche alimentato solo dal +5V del uC.
Il decodificatore morse da provare possiede di solito in ingresso un filtro passa banda che lo rende insensibile alle armoniche presenti nell'onda triangolare.
Per valutare le prestazioni di un decodificatore servono due segnali: il segnale che si vuole ricevere e quello interferente di disturbo che si vuole ignorare; di conseguenza ho inserito un secondo generatore (IC2). L' ho dimensionato per un campo di frequenze più esteso verso il basso di quello del primo. Le uscite dei due generatori confluiscono, tramite il potenziometro P4 di bilanciamento dei due segnali, nell'amplificatore sommatore IC3b che funge anche da filtro passa basso del primo ordine. Anche se non indispensabile, ho utilizzato il mezzo 358 libero per un filtro passa basso del secondo ordine (IC3a) per attenuare i segnali triangolari a partire dalla terza armonica.
Scegliendo il valore opportuno di 2,2 V per la tensione di riferimento dei segnali, li si rende simmetrici e si annulla la seconda armonica.
Ho poi aggiunto un amplificatore audio (IC5 LM386N) per ascoltare il segnale anche in altoparlante; detto amplificatore assorbe fino a 50 mA, funziona meglio se alimentato da 9 fino a 12 V, deve venire alimentato direttamente dalla tensione che alimenta tutto il generatore, la scheda Arduino non è in grado di fornire sul +5V la corrente assorbita.
Rimanevano ancora due interruttori liberi di IC4: ho usato IC4c per generare una variazione di

frequenza del primo generatore (modulazione FSK) regolabile tramite P2, e IC4d per dimezzare il guadagno dell'amplificatore sommatore simulando una evanescenza del segnale.

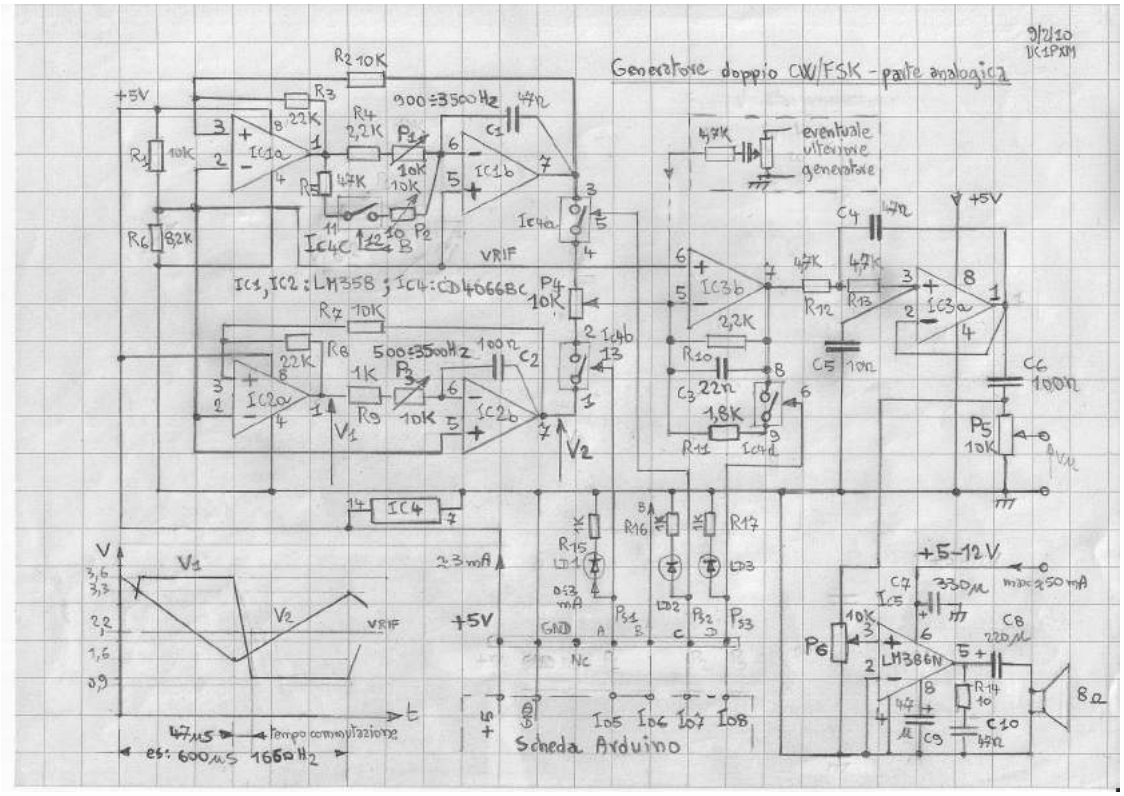


Fig. 1 Generatore CW/FSK parte analogica

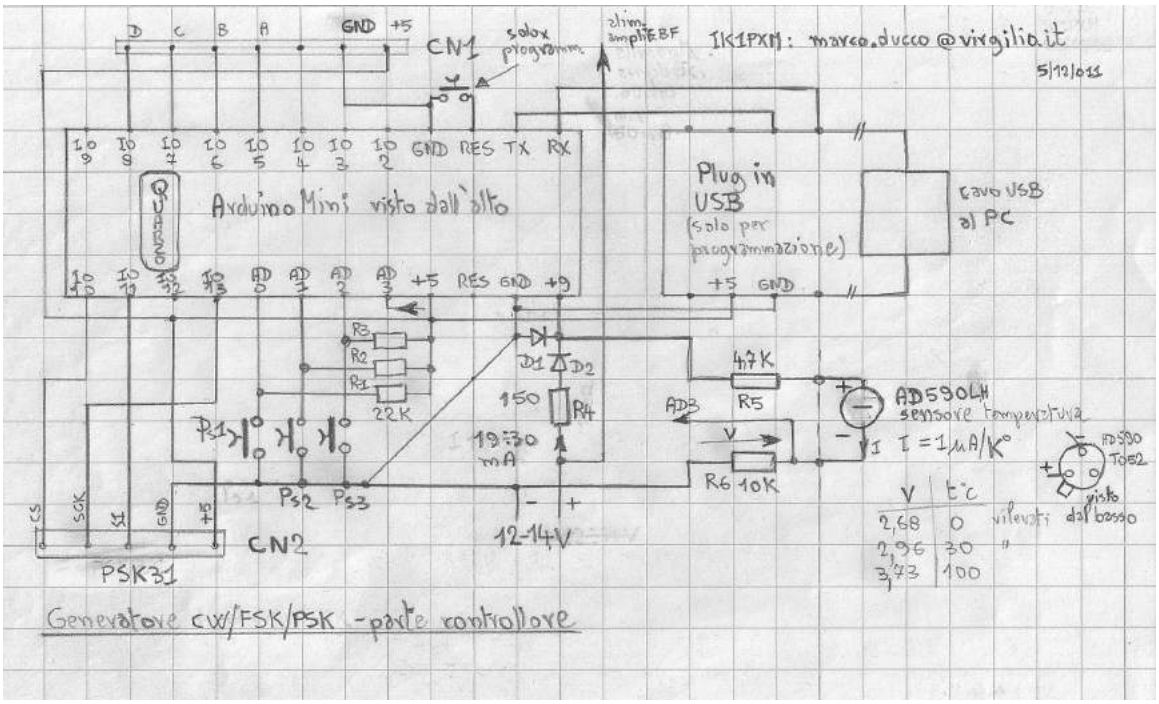


Fig.2 Generatore CW/FSK/PSK parte controllore

Il uC inizialmente pilotava un display LCD a 2x16 caratteri sul quale compariva lo stato dei

comandi, semplificando, l'ho sostituito con i LED messi in parallelo alle uscite di comando degli interruttori.

Lo schema del generatore PSK31 è riportato nella fig.3. E' costituito da un oscillatore sinusoidale audio connesso tramite un potenziometro digitale agli ingressi di un amplificatore differenziale. Comandando opportunamente la partizione del potenziometro si realizza un guadagno variabile con continuità da -1 a 1 che costituisce l'inversione di fase graduale della modulazione PSK31.
Il potenziometro è comandato ogni 2 millisecondi dal uC tramite il bus SPI (Serial Peripheral Interface). Per una descrizione completa si rimanda alla bibliografia.

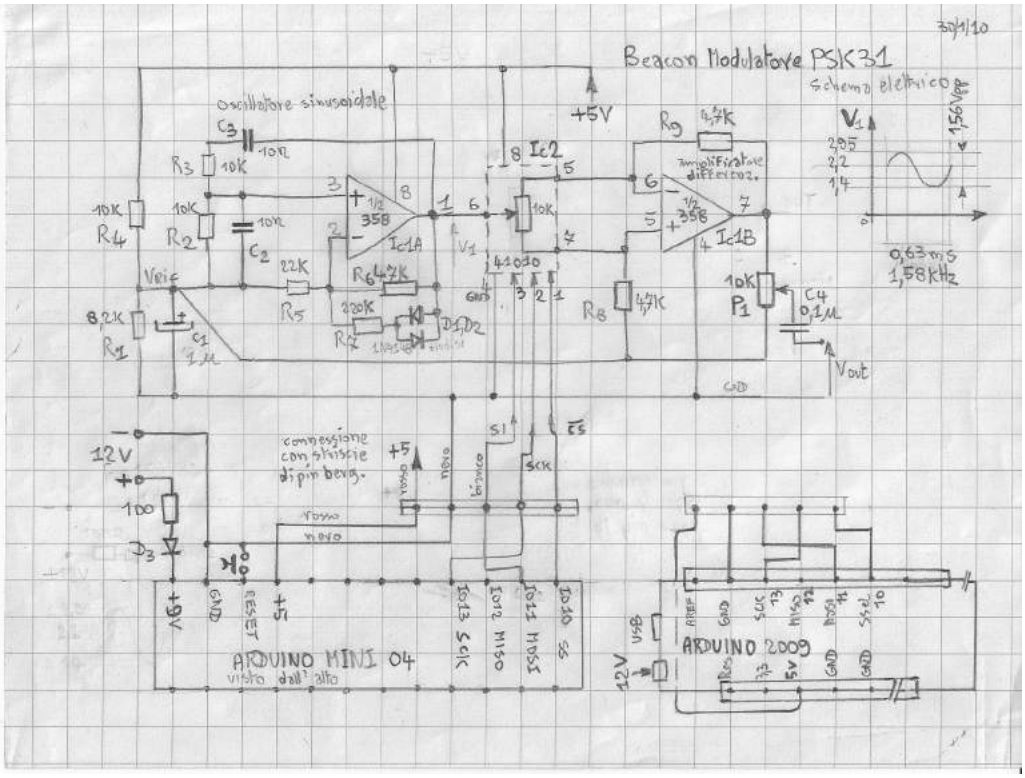


Fig.3 Generatore PSK31

Ho utilizzato un ingresso analogico libero del uC per connettere un sensore di temperatura, il valore viene trasmesso come esempio di telemetria nel pacchetto PSK; altrimenti, potrei venire criticato per non aver adoperato per il generatore un più semplice registratore/ riproduttore audio.

Elenco componenti
parte analogica:
Tutti i resistori sono ¼ W 5%: R1, R2, R7 10 k Ω; R3,R8 22 k Ω; R4, R10 2,2 k Ω; R5 47 k Ω; R6 8,2 k Ω; R9,R15,R16,R17 1 k Ω; R11 1,8 k Ω; R12,R13 4,4 k Ω; R14 10 Ω; R18 4,7 k Ω.
C1,C4,C10 47 nF; C2,C6 100 nF; C3,C5 10 nF; C7 330 uF 16 V; C8 220 uF 16 V.
P1, P2, P3, P4, P5, P6 potenziometri trimmer 10kΩ.
IC1, IC2, IC3 LM358N; IC4 CD4066BC; IC5 LM386N; LD1, LD2, LD3 led ; altoparlante 8 Ω.

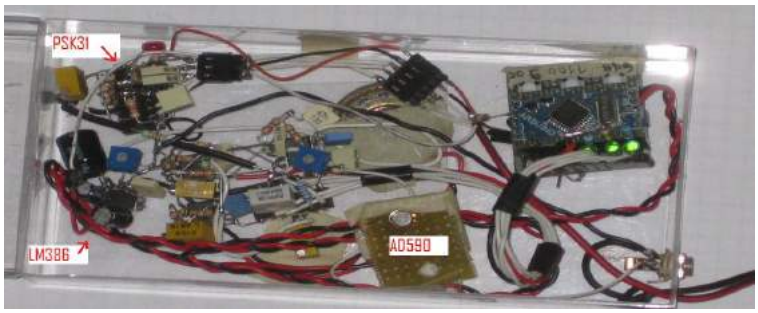
parte controllore:
R1, R2, R3 22 kΩ; R4 150 Ω; R5 4,7kΩ; R6 10kΩ; D1, D2 diodi 1N4001 o simili
Scheda Arduino mini (oppure Arduino Uno o 2009).
Plug-in USB-TTL (eventuale) mod. A0015 www.smartprj.com

Regolazione potenziometri:
P1 per 1275 Hz mark o portante generatore CW 1
P2 per 1445 Hz space (shift 170 Hz)
P3 per 900 Hz portante generatore CW 2
P4 in modo che (visto con l' analizzatore spettro del programma TCube di Alberto I2PHD) le ampiezze dei segnali dei due generatori siano uguali.
P5 per regolare l'ampiezza del segnale in uscita (all'ingresso line del PC o al microfono TX SSB)
P6 per il volume desiderato in altoparlante

Modalità funzionamento
All'accensione, qualche secondo dopo l'inizializzazione del uC, partono contemporaneamente le tre emissioni automatiche. L'emissione in CW assume in sequenza due velocità: 7 WPM, 24 WPM.
Si può agire sul programma del uC tramite tre pulsanti, ciascuno dei quali premuto in successione determina le seguenti modalità di funzionamento:
Premendo il pulsante P1:
Una prima volta mette in OFF l'emissione in CW
Una seconda volta P1 mette in ON continuo il tono CW
Una terza volta P1 riabilita la manipolazione automatica CW a 900 Hz
Premendo il pulsante P2:
Una prima volta emette uno strano CW bitonale casuale, decodificabile solo con un decodificatore con una banda > 200 Hz.
Una seconda volta mette in ON continuo il tono SPACE alto
Una terza volta mette in ON continuo il tono MARK basso
Una quarta volta emette CW con il tono MARK, CW ritardato rispetto a quello del tono 900 Hz
Una quinta volta mette in OFF l'emissione
Una sesta volta riabilita la manipolazione automatica RTTY
Premendo il pulsante P3:
Una prima volta dimezza l'ampiezza dei segnali in uscita.
Una seconda volta ripristina l'ampiezza dei segnali in uscita.
Una terza volta riabilita il comando automatico dell'ampiezza dei segnali, impostato all'interno della stringa di testo della emissione CW.

Montaggio
Fino ad ora ho realizzato solo un prototipo e ho il vezzo di operare il meno possibile pur di raggiungere il risultato voluto: ho montato la parte analogica nel modo “dead bug”, senza inserire i componenti in una basetta, ma saldandoli direttamente sui piedini degli integrati capovolti come un insetto morto. Ho poi fissato il tutto in un contenitore con del nastro biadesivo. Nella fig.4 si vede una immagine dell'insieme. Invierò via mail gli schemi di montaggio a chi li chiede.

Fig.4 Insieme generatore



Software

Il programma funziona indifferentemente su una scheda Arduino 2009 o Arduino Uno o in modo più economico su una cablata con un chip ATMEGA168-20PU, un risuonatore ceramico e un regolatore di tensione.

Lo sviluppo del programma nel C++ dell'ambiente di sviluppo Arduino mi ha impegnato molto di più di quello dell'hardware, il SW è quasi impossibile da descrivere bene e sarebbe difficile da comprendere. Fornisco solo poche indicazioni:

Il programma è temporizzato ogni 2 ms, con contatori ogni 16 cicli e ogni 11 cicli.

Esegue sempre (ogni 2 ms) la sagomatura del PSK31

Ogni 16 cicli (32 ms) aggiorna un bit PSK31.

Ogni 11 cicli (22 ms) aggiorna un bit RTTY, ogni 7 bit (1 start, 5 dati, 1 stop) legge e converte nei bit Baudot un nuovo carattere da trasmettere.

Sempre ogni 11 cicli decrementa il contatore della durata di una unità morse (punto) lenta 7*22 ms (7 WPM) o veloce 2*22 ms (24WPM). Il comando lento o veloce viene fornito da un carattere minuscolo "l" o "v", mentre i caratteri effettivi sono scritti maiuscoli. Il canale analogico di temperatura viene acquisito in un ciclo in cui la CPU è libera da operazioni particolari.

Invierò il programma a chi lo chiede, e/o un chip già programmato dietro rimborso spese.

Riporto un programma ridotto della parte PSK31 che ritengo più interessante:

```
 ….....................................
//31/1/10 beacon PSK essenziale
long memtempo;
byte cnt = 0; byte cnt_t = 0; byte cntb = 0;
byte vetalfa16[17]={200,200,198,194,188,180,168,150,127,100, 71,47, 28,15,  6, 2, 0}; //valori al potenziometro di sagomatura a coseno
byte vetbitpsk[]={0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,1,1,1,0,0,1,1,0,1,0,0,1,0,1,1,0,0,1,1,1,1,0,0,1,0,0}; //corrisponde a: "ciao "
boolean invers = true;   int nel_vetbitpsk;

void setup() {  nel_vetbitpsk = sizeof(vetbitpsk)-1;   inipot(); }

void loop()
{ if ((millis() - memtempo) >= 2 ) //periodicità elaborazione bit 2 ms
  { memtempo = millis();
  if (invers == true) {if(cnt_t < 16){cnt_t++;}}  else {if(cnt_t > 0){cnt_t--;}};
  write_pot(vetalfa16[cnt_t]);
  cnt++; if (cnt>=16) //periodicità elaborazione bit 32 ms
    { cnt =0;
      cntb++; if (cntb > nel_vetbitpsk){cntb=0;};
      if (vetbitpsk[cntb]==0) {if (invers== true){invers=false;}else{invers=true;}} //se il bit è zero cambia fase
    }
  }
}

void inipot()  //inizializza SPI del potenziometro
{ //setup potenziometro; piedini usati:  SLAVESELECT ss pin 10, DATAOUT  MOSI pin 11, SPICLOCK sck  pin 13
  byte clr;
  pinMode(11, OUTPUT); pinMode(13,OUTPUT); pinMode(10,OUTPUT); digitalWrite(10,HIGH);
  SPCR = (1<<SPE)|(1<<MSTR); clr=SPSR; clr=SPDR; delay(10);
}

char spi_transfer(volatile char data)
{ SPDR = data; while (!(SPSR & (1<<SPIF))){ };  return SPDR; }
byte write_pot(int value) // per il pot digitale  MCP41010
{ digitalWrite(10,LOW); spi_transfer(B00010001); spi_transfer(value); //2 byte:comando, dato.
  digitalWrite(10,HIGH);}
….....................................................................
```

Prove svolte

Ho provato il generatore con i programmi multimodo MixW e MULTIPSK.

Ovviamente funzionano entrambi bene, ma con qualche diversità: nella decodifica CW, al passaggio da 7 a 24 WPM MULTIPSK impiega più tempo per sincronizzarsi.

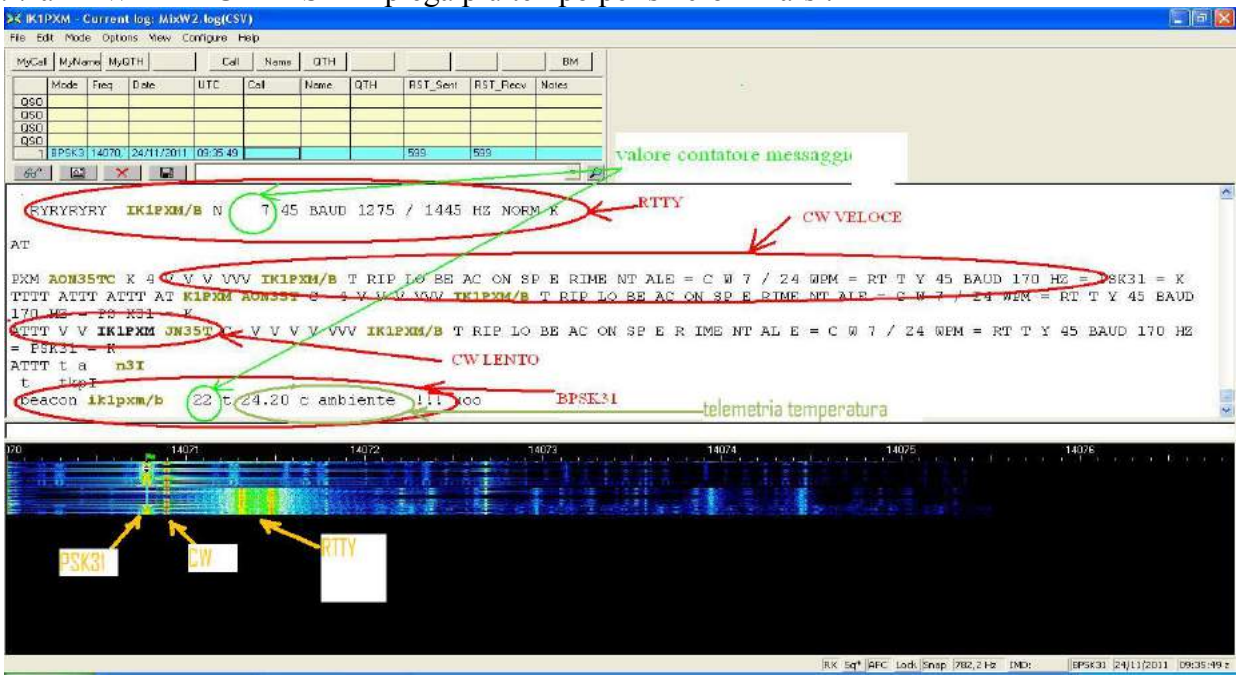


Fig.5 Schermata ricezione con MixW

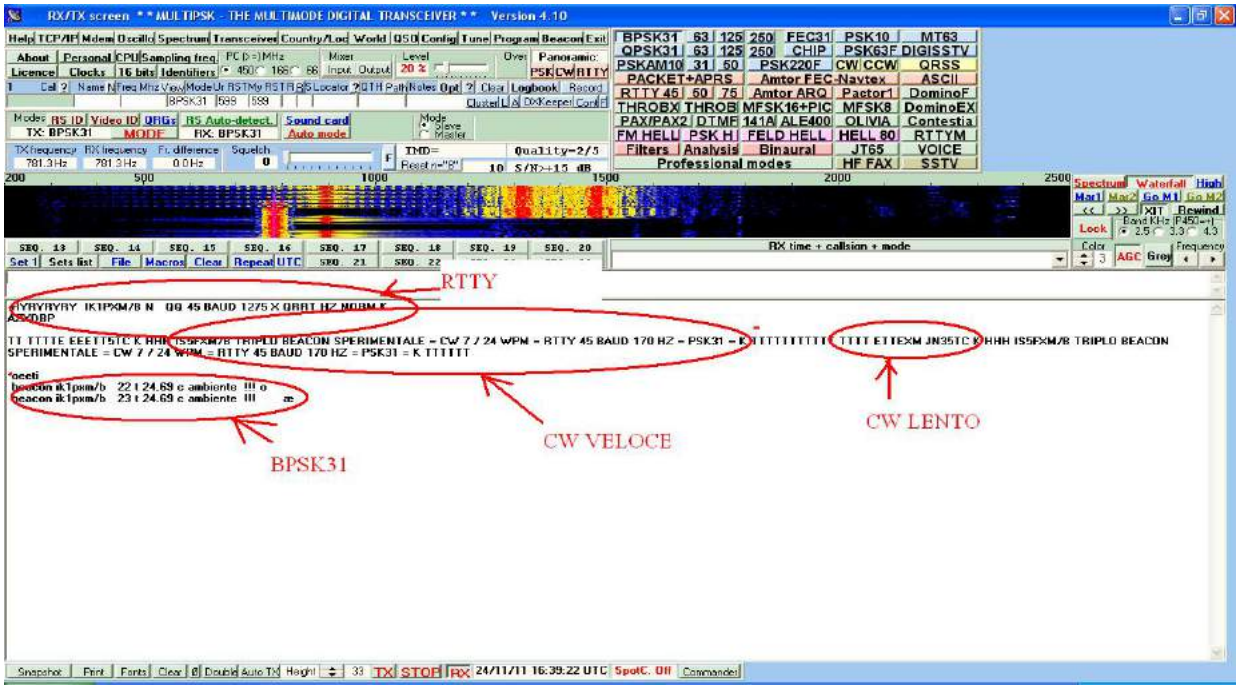


Fig.6 Schermata ricezione con MULTIPSK

Bibliografia:

Generatore PSK31 con Arduino di Marco Ducco - Radiokit maggio 2011