“Prop-4m”
Ricetrasmettitore VHF 70 Mhz
Rel. 2.0

Paolo Pinto

IZ0ROO

IZ0ROO@fastwebnet.it

Roma, 10 settembre 2009

Indice

Introduzione 3
Specifiche tecniche 4
Parallax Propeller® 6
  Specifiche tecniche 8
  Piedinatura 8
  Organizzazione della memoria 9
Diagramma a blocchi del ricetrasmettitore 10
Scheda di controllo 11
  Assegnazione Linee dati della CPU 11
  Schema elettrico scheda di controllo 12
  LCD datasheet 13
  Elenco dei componenti scheda di controllo 14
  Circuito stampato scheda di controllo 15
Ricevitore 16
  Comandi 17
  Schema elettrico ricevitore 18
  Elenco dei componenti ricevitore 19
  Circuito stampato ricevitore 20
Trasmettitore 20
  Schema elettrico trasmettitore 22
  Circuito stampato trasmettitore 23
  Commutazioni segnali ed alimentazione 23
Calcolo dei fattori di divisione del PLL 24
Firmware 25
  Funzioni 25
  Comandi 25
  Descrizione 25
  Codice sorgente Spin 26
Foto 33
Bibliografia 43
  Documentazione Propeller 43
  Componenti elettronici 43
  Produttore circuiti stampati 43
  Taratura 43
  Software per PCB 44

Introduzione

Quando venne concessa ai Radioamatori dal Ministero delle Comunicazioni la possibilità di effettuare test di trasmissione e di ricezione sui 70 Mhz decisi di realizzare un ricetrasmettitore su tale banda.

Purtroppo la costruzione della radio ha preso almeno due anni di tempo e quando è stata terminata il Ministero delle Comunicazioni aveva revocato la concessione all'uso dei 4 metri. Se in futuro verrà nuovamente concessa l'autorizzazione, allora questo ricetrasmettitore potrà essere usato nel frattempo può solo considerarsi una esperienza di laboratorio.

Il ricetrasmettitore che ho realizzato è un apparato sintetizzato in FM, pilotato da microprocessore che monta solo componenti discreti non SMD. Io preferisco i componenti tradizionali ai componenti SMD poiché è sicuramente più facile il montaggio e quando si lavora su dei prototipi risulta più semplice la saldatura ed è possibile utilizzare gli zoccoli per gli integrati.

Questa radio è stata progettata adottando una tecnica che gli informatici chiamano "Bottom up" dal basso verso l'alto. Ho iniziato dai singoli componenti – scheda della CPU, scheda del ricevitore, scheda del trasmettitore – e sono approdato ad una radio completa. Progettando una scheda alla volta è possibile concentrare la propria attenzione sulle singole parti, farle funzionare mettendole a punto con la strumentazione e infine assemblarle fra loro.

Per prima cosa ho progettato e realizzato la scheda del controller, inizialmente lavorando su schede millefori per poi approdare ad un vero circuito stampato doppia faccia. La scheda controller è stata progettata anche con l'obiettivo di essere riutilizzata in altri progetti. Il controller presenta queste interfacce: tastiera, display LCD, linee dati generiche di input/output. A bordo della scheda ho previsto anche un clock per memorizzare la data e l'ora ed un convertitore analogico/digitale (ADC) per la conversione di tensioni in valori binari. Per la programmazione è presente un integrato max3232 che ne consente il collegamento direttamente alla porta seriale del PC. Come processore ho deciso di impiegare, al posto degli ottimi microcontrollori PIC generalmente utilizzati in questo tipo di progetti, un micro controller molto innovativo: il "Propeller" della Parallax.
Il propeller è un microprocessore con 8 processori RISC a 32 bit che lavorano in parallelo per una potenza di calcolo di 160 MIPS.

http://www.parallax.com/dl/docs/article/Propeller_Firmware_magazine_ITA.pdf

Qualcuno potrebbe obiettare che questa CPU sia troppo potente per una radio di questo tipo, e sicuramente questo è vero ma per testare un componente elettronico o un software è necessario provarlo, non è sufficiente studiare sulla carta le sue caratteristiche. Questa radio può essere considerata come un "Test bed" per sperimentare l'utilizzo del Propeller e la sua interazione con dei componenti esterni.

Una volta realizzata la CPU è venuto il turno del ricevitore. Su un sito di componentistica elettronica (http://www.alltronics.com) ho trovato una scheda surplus che conteneva un Tuner Mitsumi 407-A26. Questo Tuner possiede un front-end a mosfet molto sensibile, un mixer con uscita del segnale a 10.7 Mhz e un VCO. Il Tuner è in grado di sintonizzarsi in un range di frequenze da 60-150 Mhz. Sul sito è presente anche lo schema elettrico e la documentazione. Dopo avere dissaldato il Tuner l'ho collegato ad un PLL Fujitsu

utilizzato in alcuni kit di Nuova Elettronica e ad un MC3361 che è un ricevitore FM a banda stretta.

Il trasmettitore è sicuramente la parte meno complessa della radio. E' formato da un oscillatore basato su un VCO di Nuova Elettronica LX.1235/3 e sullo stesso PLL MB1502 utilizzato nel ricevitore. Sulla basetta del TX sono presenti anche i rele per le commutazioni dell'alimentazione. L'uscita del VCO è collegata direttamente ad un finale ibrido da 25W che semplifica molto la costruzione del trasmettitore. In pratica non ho dovuto avvolgere alcuna bobina.

La realizzazione di questa radio ha richiesto molto tempo ma grande è stata la soddisfazione nel vederla funzionare. Io credo che ogni radioamatore dovrebbe cimentarsi nella costruzione e nella sperimentazione. Spesso è difficile trovare il tempo, lo spazio e i soldi per questo tipo di attività ma vedere funzionare un oggetto realizzato con le proprie mani è sicuramente affascinante.

Spero che questo progetto possa essere utile ad altri radioamatori o appassionati di elettronica suggerendo loro possibili tecniche e idee per altri progetti. A volte, non è facile reperire i componenti utilizzati in un progetto e quindi risulta difficile costruirlo ma dallo studio della documentazione possono nascere nuovi spunti di progettazione per altre realizzazioni.

Specifiche tecniche

Queste sono le caratteristiche della radio:

Ricevitore:
Frequenza ricezione: 70-71 Mhz (impostabile via software)
Modulazione: FM stretta;
Sensibilità: 0.5 uV;
Canalizzazione: 25 Khz;
Selettività: elevata (non ancora quantificata);
Conversione: doppia – 1° conv. A 10.7 Mhz, 2° conv. 455 Khz;
VCO a PLL.

Trasmettitore:
Frequenza trasmissione: 70-71 Mhz (impostabile via software)
Modulazione: FM stretta;
Potenza: Finale ibrido da 25 W
VCO a PLL.

Parte digitale:
Propeller P8X32A-D40;
Encoder di tastiera;
Display LCD 2 righe;
DAC/ ADC;
Programmabile via RS-232 senza dispositivi esterni;

Controlli effettuati dalla CPU:

- impostazione frequenze RX e TX;
- Muting ON/OFF

- ricerca e scansione frequenze;
- impostazioni frequenze da tastiera;
- visualizzazione ora;
- potenza trasmissione;
- onde stazionarie;
- voltaggio di alimentazione.

Parallax Propeller®

Il nucleo operativo della radio è il Propeller di cui ho parlato nell'introduzione. Il Propeller è un micro controller della Parallax caratterizzato da 8 cores (Processori) a 32 bit integrati su singolo chip. La struttura di base del componente è mostrata in figura 1.

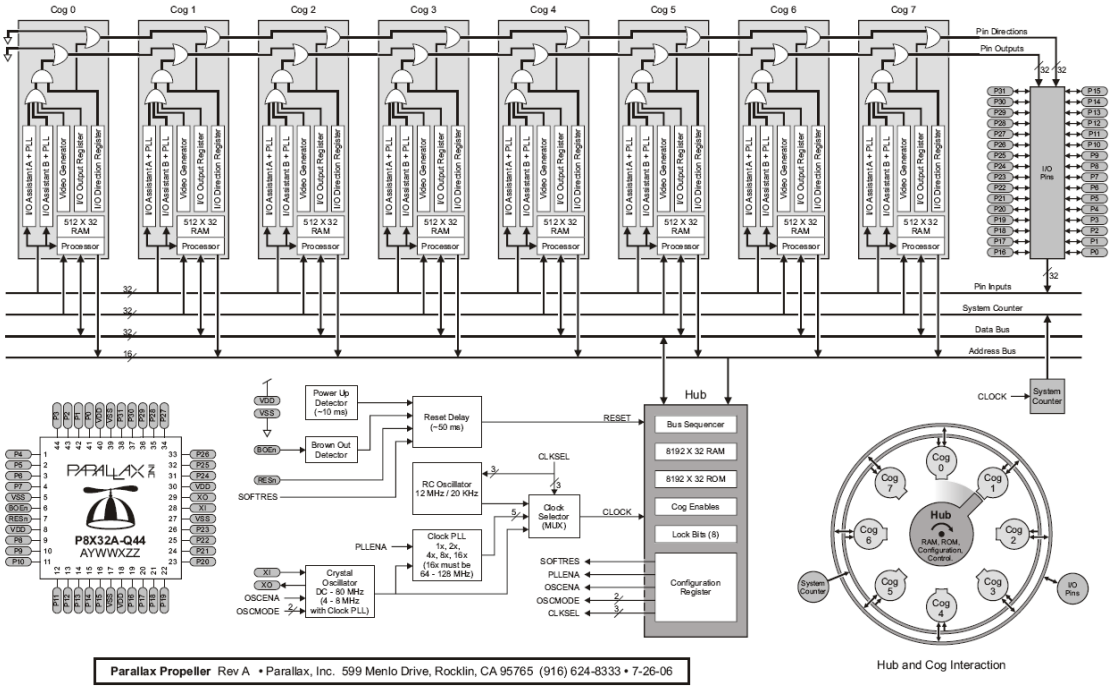


Figura 1

Ogni core è un microprocessore completo che può funzionare indipendentemente dagli altri, grazie ad una propria memoria RAM che viene inizializzata via software. Il Propeller può essere programmato in assembler o tramite un linguaggio compilato detto Spin vicino al Basic come sintassi, che permette di implementare algoritmi in modo relativamente semplice. Tra i costrutti del linguaggio Spin sono presenti alcune funzioni utilizzate per "istruire" i diversi Cog con il codice da eseguire.

Il Propeller dispone di risorse condivise tra i Cog mutualmente esclusive: memorie RAM e ROM. L'accesso a queste risorse è gestito da un meccanismo chiamato Hub: una logica round-robin consente a ciascun Cog di usare a turno le risorse. I pin di I/O sono organizzati in un'unica porta da 32 bit, sono condivisi tra i Cog e la loro direzione (input o output) dipende da come viene definita in ciascun Cog. Ai pin possono accedere contemporaneamente tutti i cog, una logica di arbitraggio evita possibili conflitti.

Alle linee dati si possono interfacciare sono dispositivi funzionanti a 3.3V. Per collegare periferiche funzionanti a 5V è necessario predisporre dei circuiti shift-level.

Per programmare il controller non servono programmatori. Il controller possiede una porta seriale RS-232 per collegarlo direttamente alla porta seriale del PC, (o alla porta USB utilizzando un semplice convertitore). La programmazione avviene mediante l'ambiente di sviluppo (IDE) della Parallax. I programmi vengono compilati ed inviati subito nella memoria del Propeller per l'esecuzione.

Il propeller ha un 'bootloader' integrato nella memoria ROM, quindi alla prima accensione, questo controlla se il PC deve inviare un programma da eseguire. In caso positivo, la comunicazione viene stabilita e il programma viene ricevuto. All'interno del propeller, i programmi vengono eseguiti sempre dalla memoria RAM principale, questo significa che in assenza di alimentazione il software viene perduto. Per ovviare a questo problema, il chip è stato progettato per ospitare una memoria seriale I2C EEPROM esterna. All'accensione del propeller, se il software è presente nella memoria EEPROM, viene copiato in RAM ed eseguito.

Ciascun Cog ha a disposizione un generatore video in grado di inviare informazioni ad un monitor VGA o un televisore con ingresso video-composito.

Per scaricare il software sviluppato direttamente nella EEPROM, basta farlo dal compilatore con i tasti F11 o CTRL-F11 (Load EEPROM) invece dei tasti F10 o CTRL+F10 (Load RAM) per la normale esecuzione in RAM.

Il propeller P8X32A è disponibile in tre differenti packeage (figura 3): DIP a 40 pin, LQFP a 44 pin e QFN. La possibilità di avere la versione DIP a 40 pin farà sicuramente contenti tutti i sviluppatori che desiderano iniziare a lavorare con questo dispositivo, in quanto spesso le versioni più piccole sono a volte un ostacolo da superare sotto il punto di vista delle saldature. Le altre versioni invece combinano il vantaggio della straordinaria potenzialità che offre questo dispositivo con le ridottissime dimensioni.

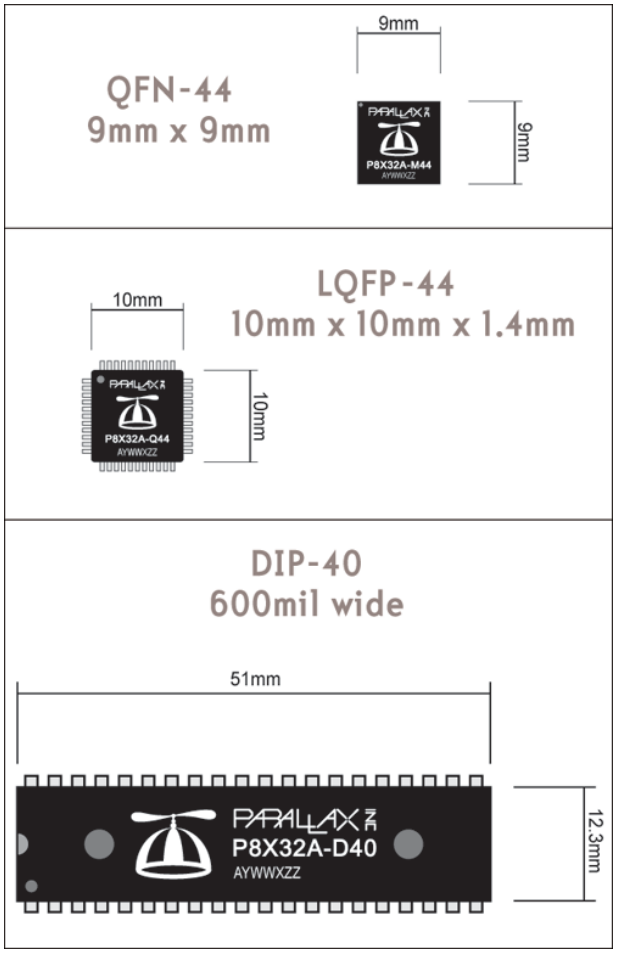


Figura 2

Specifiche tecniche

Modello	Packeage	I/O Pins	Alimentazione	Clock Esterno	Oscillatore interno RC	Velocità esecuzione	RAM/ROM Globale	COG RAM
P8X32A-D40	40-pin DIP	32 CMOS	3.3 Volt DC	DC ÷ 80 Mhz	12 Mhz o 20 Khz	0 ÷ 160 MIPS (20 MIPS/cog)	32K RAM 32K ROM	512x32 bits per COG
P8X32A-Q40	44-pin LQFP							
P8X32A-M40	44-pin QFN							

Piedinatura

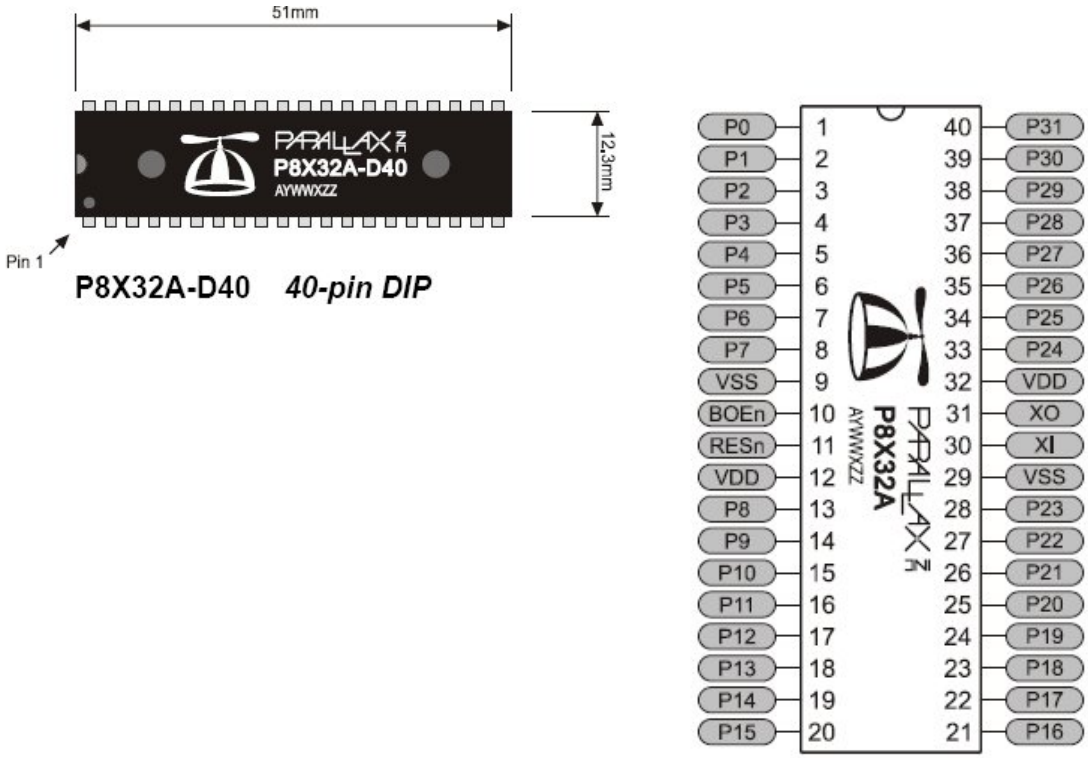


Figura 3

Organizzazione della memoria

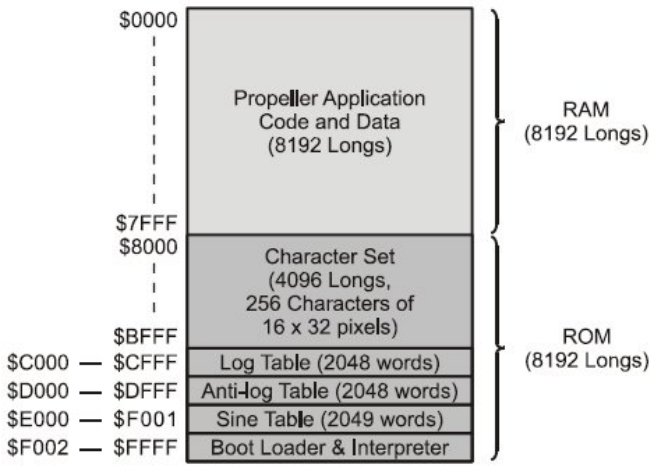
Il propeller ha a disposizione un totale di **64K di memoria**, divisa tra 32K di RAM e 32K di ROM.

I primi 32K di RAM sono riservati ai programmi e ai dati che desideriamo conservare negli applicativi, i secondi 32K di RAM sono divisi ulteriormente in cinque gruppi avente diverse funzionalità.

Font di caratteri: Il primo gruppo contiene tutti i caratteri che fanno parte della font Parallax, questi potrebbero essere utilizzati ad esempio per un ouput a video (VGA o video-composito), oppure per disegnare uno schema elettrico. Infatti la speciale font Parallax, include caratteri personalizzati che rappresentano parti di comuni schemi elettrici (es. condensatori, resistenze, induttanze, ecc..). Questi caratteri visualizzati in sequenza possono comporre uno schema elettrico.

Tabella per Funzioni Matematiche: Tre tabelle sono state riservate per speciali funzioni matematiche, Log, Anti-Log (per facilitare i calcoli con esponente) e una tabella per il calcolo del seno. Il manuale di programmazione spiega in dettaglio come utilizzare queste tabelle.

Interprete e Bootloader: Contiene l'interprete per i programmi scritti in SPIN (questo viene caricato nella RAM di ciascun Cog quando attivo) e un piccolo Bootloader che ha la funzione di trasferire i programmi compilati dal PC nella memoria del propeller.



RAM:

$0000÷$7FFF: Programmi applicativi e dati

ROM:

$8000÷$BFFF: Font di caratteri (256 cartteri composti da 16x32 pixel)

$C000÷CFFF: Tavola Log

$D000÷DFFF: Tavola Anti-Log

$E000÷F001: Tavola del seno

$F002÷$FFFF: Interprete e Bootloader

Figura 4

Diagramma a blocchi del ricetrasmettitore

Il ricetrasmettitore è composto da quattro schede:

- Parte di controllo digitale;
- Scheda ricevitore (RX);
- Scheda trasmettitore (TX) + commutazioni alimentazione;
- Amplificatore di Potenza da 25 W.

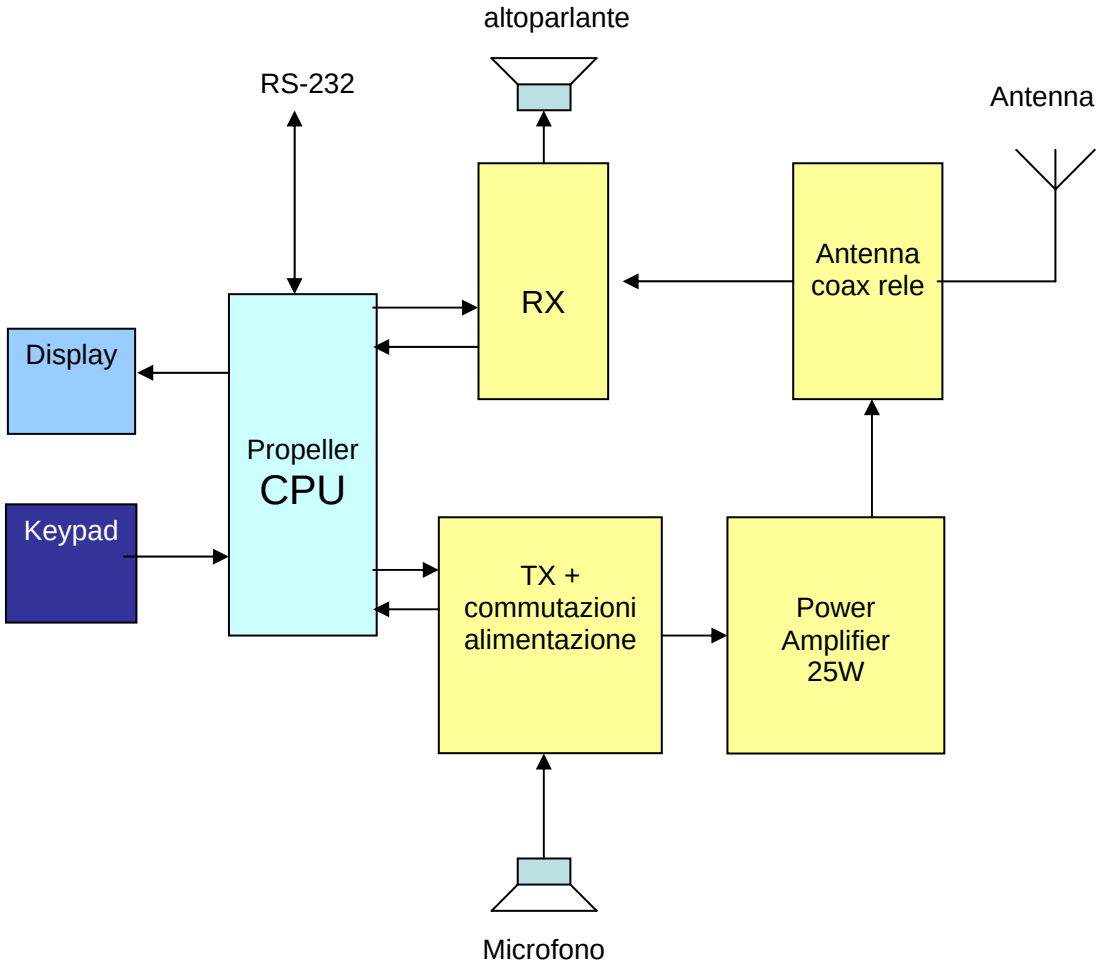


Figura 5

Scheda di controllo

La scheda si basa sul Propeller® versione DIP 40 pin. Sulla scheda è presente una eeprom in cui è memorizzato il firmware.

Il controller è così composto:

- CPU (Propeller): P8X32A-D40;
- ROM memory: 24LC256;
- time clock: DS1302;
- ADC / DAC Philips: PCF8591;
- Keypad encoder: 74C922;
- RS232 interface: Max3232;
- Sound speaker.

Il controller si interfaccia con la scheda del ricevitore e del trasmettitore mediante due piattine indicate sullo stampato con CN5 RX e CN6 TX. Il display (model: **W2004D-TMI**) lavora a 3.3 V ed è quindi facilmente interfacciabile alla CPU senza utilizzare dispositivi shift-level. Il controller può essere programmato utilizzando una connessione seriale RS-232 utilizzando il Propeller Tool della Parallax. Il Propeller possiede numerose librerie scritte dagli utenti e messe a disposizione sul sito della Parallax.

Assegnazione Linee dati della CPU

- **A0 … A5**: Display LCD
- **A6**: Modulation level
- **A7**: RX / TX commute (line out)
- **A8**: RX squelch on/off
- **A9**: RX PLL lock
- **A10**: sound beep
- **A11**: Data Available 74C922
- **A12** … **A15**: data 74C922
- **A16**: RX PLL clk
- **A17**: RX PLL data
- **A18**: RX PLL LE
- **A19**: TX PLL clk
- **A20**: TX PLL data
- **A21**: TX PLL LE
- **A22**: Push to talk (line in)
- **A23**: DS1302 reset
- **A24**: DS1302 data
- **A25**: DS1302 clock
- **A26**: PCF8591 SCL
- **A27**: PCF8591 SDA
- **A28**: 24LC256 SCL
- **A29**: 24LC256 SDA
- **A30**: MAX3232 IN
- **A31**: MAX3232 OUT

Schema elettrico scheda di controllo

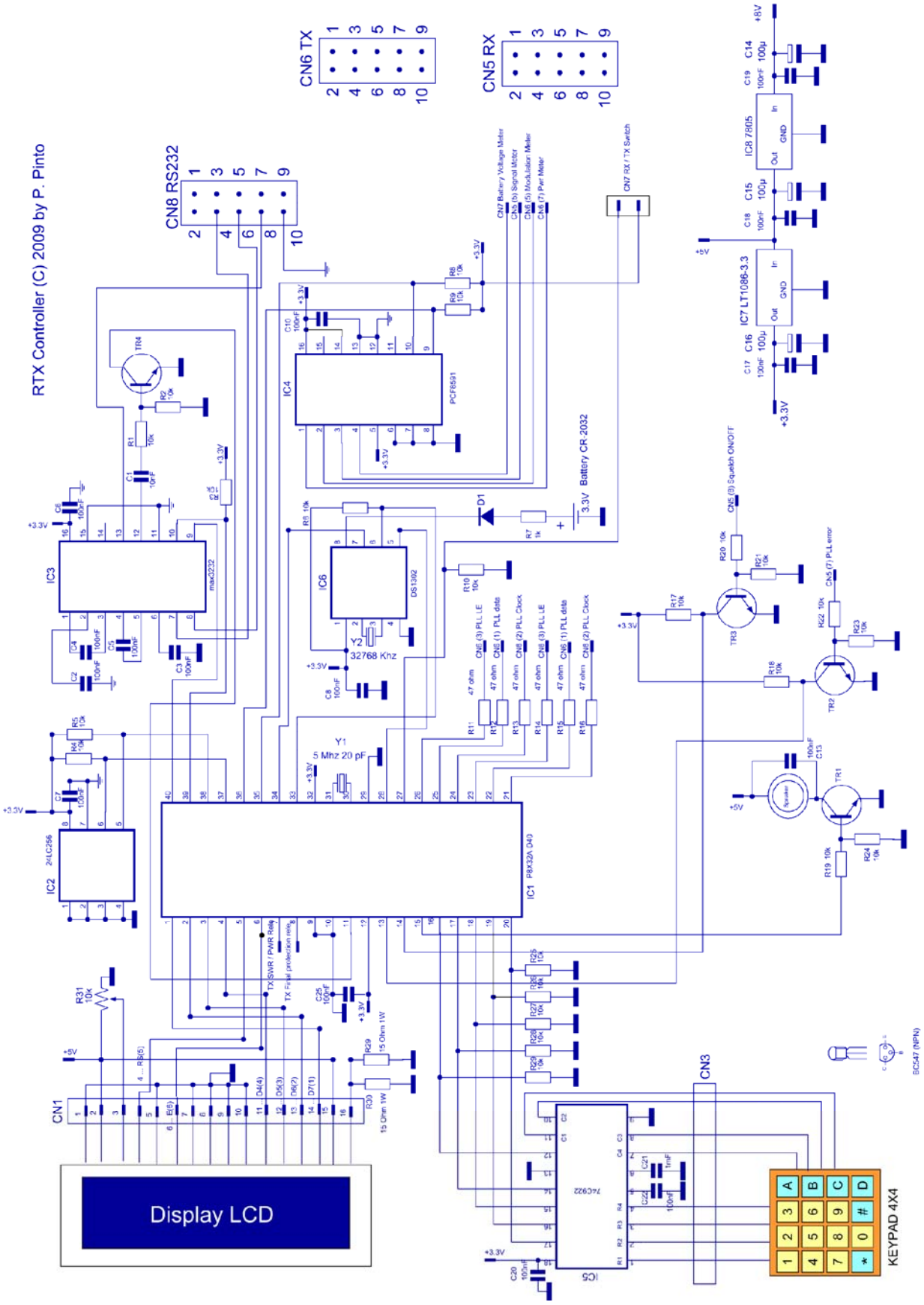
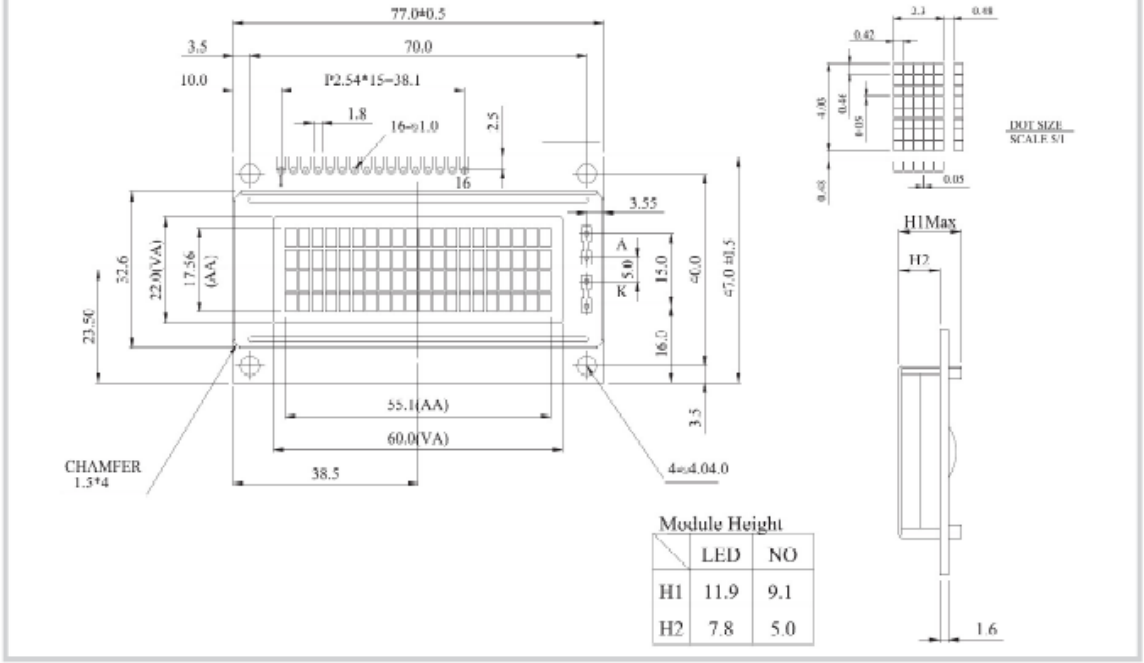


Figura 6

WH2004D Character 20x4

Dimension drawing



Module Height

	LED	NO
H1	11.9	9.1
H2	7.8	5.0

Feature

1. 5x8 dots includes cursor
2. Built-in controller (KS 0066 or Equivalent)
3. +5V power supply (Also available for +3V)
4. 1/16 duty cycle
5. LED can be driven by pin1,pin2,pin15,pin16 or A and K
6. N.V. optional for +3V power supply

Pin NO.	Symbol	Function
1	Vss	GND
2	Vdd	+3V or +5V
3	Vo	Contrast Adjustment
4	RS	H/L Register select signal
5	R/$\overline{W}$	H/L Read / write signal
6	E	H→L Enable signal
7	DB0	H/L Data bus line
8	DB1	H/L Data bus line
9	DB2	H/L Data bus line
10	DB3	H/L Data bus line
11	DB4	H/L Data bus line
12	DB5	H/L Data bus line
13	DB6	H/L Data bus line
14	DB7	H/L Data bus line
15	A	Power supply for LED 4.2 V
16	K	Power supply for B/L (0V)

Mechanical Data

Item	Standard Value	Unit
Module Dimension	77 x 47	mm
Viewing Area	60 x 22	mm
Mounting hole	70 x 40	mm
Character Size	2.3×4.03	mm

Absolute Maximum Rating

Item	Symbol	Standard Value			Unit
		min.	typ.	max.	
Power Supply	VDD-VSS	-0.3	---	7.0	V
Input Voltage	VI	-0.3	---	VDD	V

Note : VSS=0 Volt, VDD=5.0 Volt.

Electronical Characteristics

Item	Symbol	Condition	Standard Value			Unit
			min.	typ.	max.	
Input Voltage	VDD	VDD=+5V	4.7	5.0	5.3	V
		VDD=+3V	2.7	3.0	5.3	V
Supply Current	IDD	VDD=5V	---	8.0	10.0	mA
Recommended LC Driving Voltage for Normal Temp. Version module	VDD-V0	+20°C	5.0	5.1	5.7	V
		0°C	4.6	4.8	5.2	
		25°C	4.1	4.5	4.7	
		50°C	3.9	4.2	4.8	
		70°C	3.7	3.9	4.3	
LED Forward Voltage	VF	25°C	---	4.2	4.6	V
LED Forward Current	IF	25°C	---	540	1000	mA
EL Power Supply Current	IEL	Vel=110VAC;400Hz	---	---	5.0	mA

Display Character Address Code :

Display position	1	2	3	4	5	6	7	8	9	10	11	12	13	---	---	20
DD RAM Address	00	01														13
DD RAM Address	40	41														53
DD RAM Address	14	15														27
DD RAM Address	54	55														67

Figura 7

Elenco dei componenti scheda di controllo

Descrizione	ID	Distributore
P8X32A-D40	IC1	http://www.digikey.it
24LC256	IC2	http://www.digikey.it
Max3232	IC3	http://www.maxim-ic.com
PCF8591	IC4	http://www.webtronic.it
74C922	IC5	http://www.digikey.it
DS1302	IC6	http://www.maxim-ic.com
LT1086-3.3	IC7	http://www.digikey.it
7805	IC8	http://www.digikey.it
Crystal 5Mhz 20pF	Y1	http://www.digikey.it
Crystal 32768Khz	Y2	http://www.digikey.it
1N4148	D1	
BC547	TR1, TR2, TR3, TR4	
10K 1/8W	R1, R2, R3, R4, R5, R6, R8, R9, R10, R17, R18, R19, R20, R21, R22, R23, R24, R25, R26, R27, R28, R29	
1K 1/8W	R7	
47 ohm 1/8W	R11, R12, R13, R14, R15, R16	
15 ohm 1/2W	R29, R30	
Trimmer 10K	R31	
10nF	C1	
100nF	C2, C3, C4, C5, C6, C8, C10, C11, C12, C13, C17, C18, C19, C20, C22, C24, C25	
100uF 25V	C14, C15, C16	
1uF	C21	
Display LCD: 20 x 4 - 3.3 Volt model: 2004D-TMI	Display LCD	Code: **2004D-TMI** http://www.areasx.com
Keypad 16 keys	Keypad	http://www.jameco.com
CR-2032 holder battery coin	Clock Battery	http://www.digikey.it
Buzzer	Speaker	

Circuito stampato scheda di controllo

Il controllore è stato realizzato con il programma Abacom software Sprint layout. Il PCB è un doppia faccia con serigrafia.

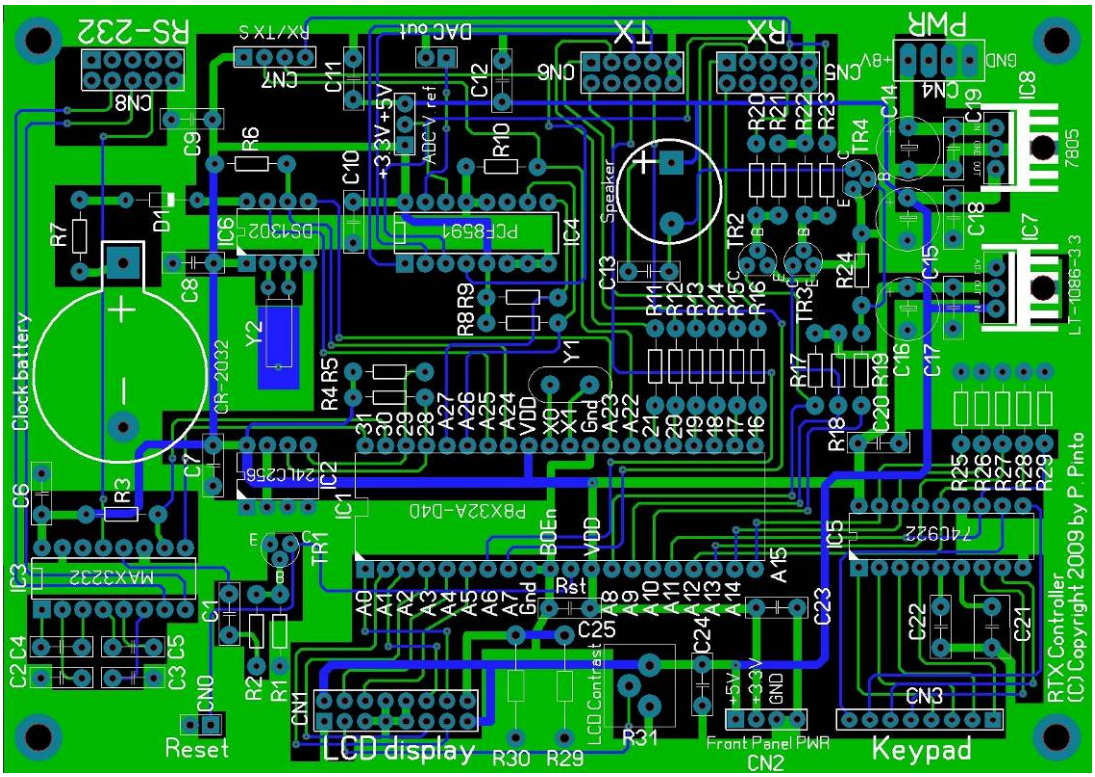


Figura 8

Ricevitore

Il ricevitore FM è un doppia conversione con la prima MF a 10.7 Mhz e la seconda MF a 455 Khz. Il tuner Mitsumi modello 407-A26, contiene un mosfet come amplificatore di antenna, un mixer ed un VCO. Il mosfet deve essere alimentato a +5V e la sua alimentazione è separata dal resto del tuner, in questo modo in fase di trasmissione è possibile spegnerlo lasciando alimentato solo il VCO.

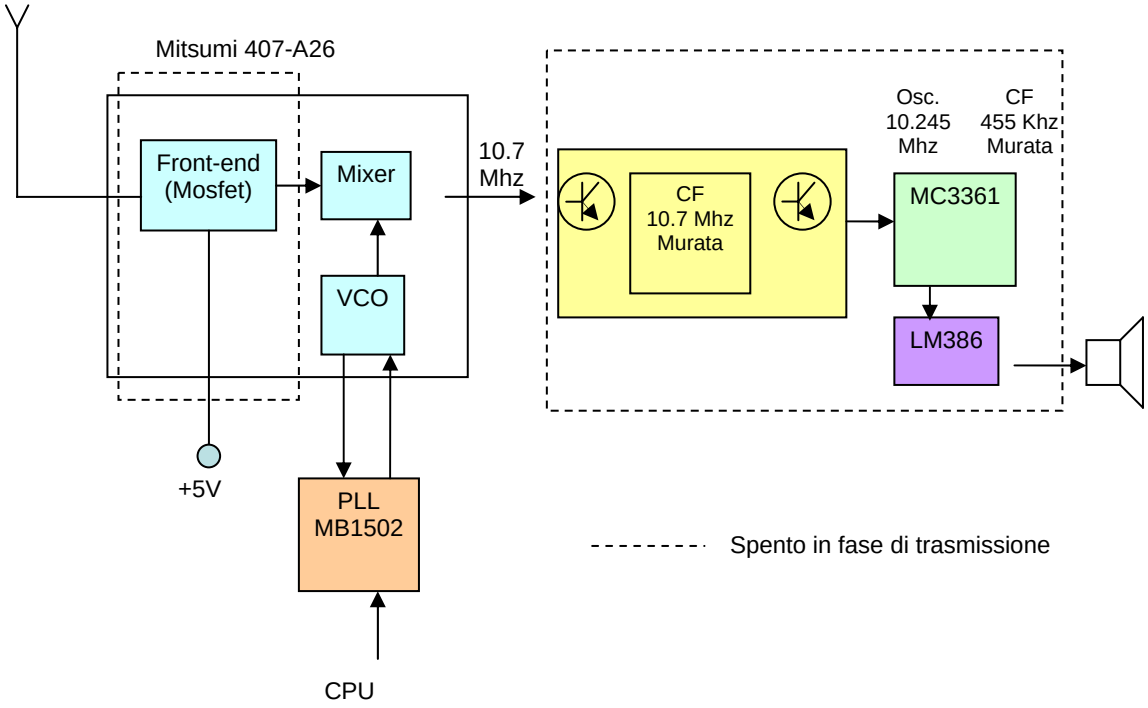


Figura 9

Il tuner (Figura 10) è in grado di sintonizzarsi da circa 60 Mhz a 150 Mhz, risulta quindi ideale per un ricevitore sui 4 metri. Il tuner è stato acquistato on line negli USA dalla ditta Alltronics: http://www.alltronics.com , il prezzo della scheda sul quale è montato è di 5 dollari. Per utilizzare il tuner è necessario dissaldarlo dalla scheda che offre anche altri utili componenti come ad esempio i connettori.

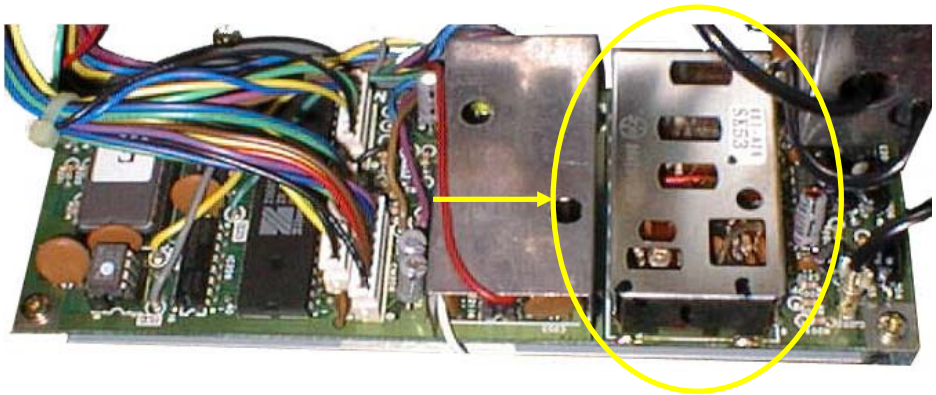


Figura 10

Il tuner Mitsumi ha queste connessioni:

- ingresso di antenna;
- alimentazione a +5V per il mosfet in ingresso;
- alimentazione a + 8V per il mixer ed il VCO;
- sintonia varicap;
- uscita segnale a 10.7 Mhz.

Il VCO viene controllato da un PLL MB1502 della Fujitsu. Il PLL viene programmato dalla CPU mediante tre linee dati: data, clock, latch enable. Il segnale in uscita dal tuner viene amplificato da due transistor 2N2222 e filtrato da un filtro murata di qualità da 10.7 Mhz. Il segnale da 10.7 Mhz arriva all'integrato MC3361 per la seconda conversione sui 455 Khz.

Per la seconda conversione è stato utilizzato un filtro murata di qualità da 455Khz. L'MC3361 fornisce inoltre il segnale per pilotare l'S-meter ed il segnale audio che viene amplificato dall'integrato LM386. Un transistor collegato all'MC3361 si occupa del funzionamento dello squelch.

La scheda del ricevitore è collegata alla scheda della CPU mediante un flat cable che trasporta questi segnali:

- PLL clock line (digitale);
- PLL data line (digitale);
- PLL Latch enable line (digitale);
- Squelch on/off (analogico);
- Signal strength (analogico);
- Ground

Comandi

Il ricevitore ha un controllo di volume ed un controllo squelch che fanno capo a due potenziometri sul frontale della radio.

Schema elettrico ricevitore

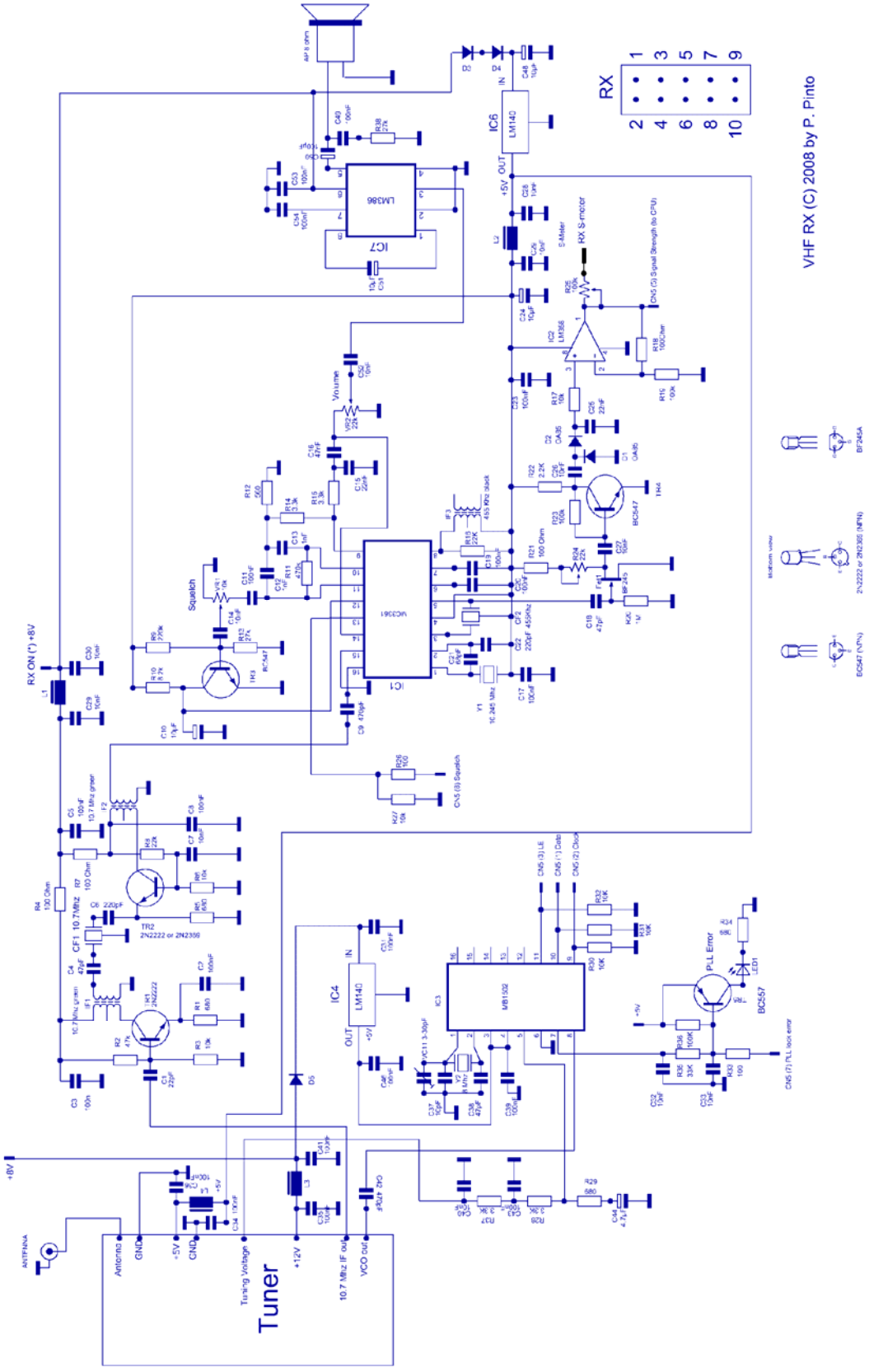


Figura 11

Elenco dei componenti ricevitore

Descrizione	ID	Distributore
Mitusmi 407-A26	Tuner	http://www.alltronics.com
MC3361	IC1	http://www.webtronic.it
LM358	IC2	
MB1502	IC3	http://www.nuovaelettronica.it
LM140	IC4	http://www.rfmicrowave.it
LM140	IC5	http://www.rfmicrowave.it
LM386	IC7	
2N2222	TR1, TR2	http://www.rfmicrowave.it
BC547	TR3, TR4	
BC557	TR5	
BF245	FET1	
Crystal 10.245Mhz	Y1	http://www.rfmicrowave.it
Crystal 8 Mhz	Y2	http://www.rfmicrowave.it
SFG10.7MA Murata Ceramic Filter 10.7 Mhz	CF1	Codice: **F-10M7-A2** http://www.rfmicrowave.it
CFG455G Murata Ceramic Filter 455 Khz	CF2	Codice: **F-455K-X1** http://www.rfmicrowave.it
OA95	D1, D2	http://www.rfmicrowave.it
IF green 10.7 Mhz	IF1, IF2	http://www.rfmicrowave.it
IF yellow 455 Khz	IF3	http://www.rfmicrowave.it
680 ohm 1/4W	R1, R5, R29, R34	
47K 1/4W	R2	
10K 1/4W	R3, R6, R17, R30, R31, R32	
100 ohm 1/4W	R4, R7, R18, R33	
22K 1/4W	R8, R16	
220K 1/4W	R9	
8.2K 1/4W	R10	
470K 1/4W	R11	
560 ohm 1/4W	R12	
27K 1/4W	R13	
3.3K 1/4W	R14, R15, R28, R37	
1Mohm 1/4W	R20	
2.2K	R22	
Trimmer 22K	R24	
Trimmer 100K	R25	
27K 1/4W	R38	
100K 1/4W	R19, R23, R36	
1N4007	D3, D4, D5	
22pF	C1	
100nF	C2, C3, C5, C8, C11, C17, C23, C31, C34, C36, C39, C41, C43, C46, C49, C54, C53	
47pF	C4, C18, C38	
220pF	C6	
10nF	C7, C14, C26, C28, C29, C32, C33, C39, C40	
100uF 16V	C10	
1nF	C12, C13	
22nF	C15	
47nF	C16	
68pF	C21	
220pF	C22	
10uF 16V	C24, C48	
10pF	C37	
470pF	C42	
4.7uF 16V	C44	

V. Capacitor 0-60pF	VC11	
6 uH inductance BFC-6	L1, L2, L3, L4	Codice: **BFC-6** http://www.rfmicrowave.it
	Signal meter	Comprato su ebay

Circuito stampato ricevitore

Il ricevitore è stato realizzato con il programma Abacom software Sprint layout. Il PCB è un doppia faccia con serigrafia.

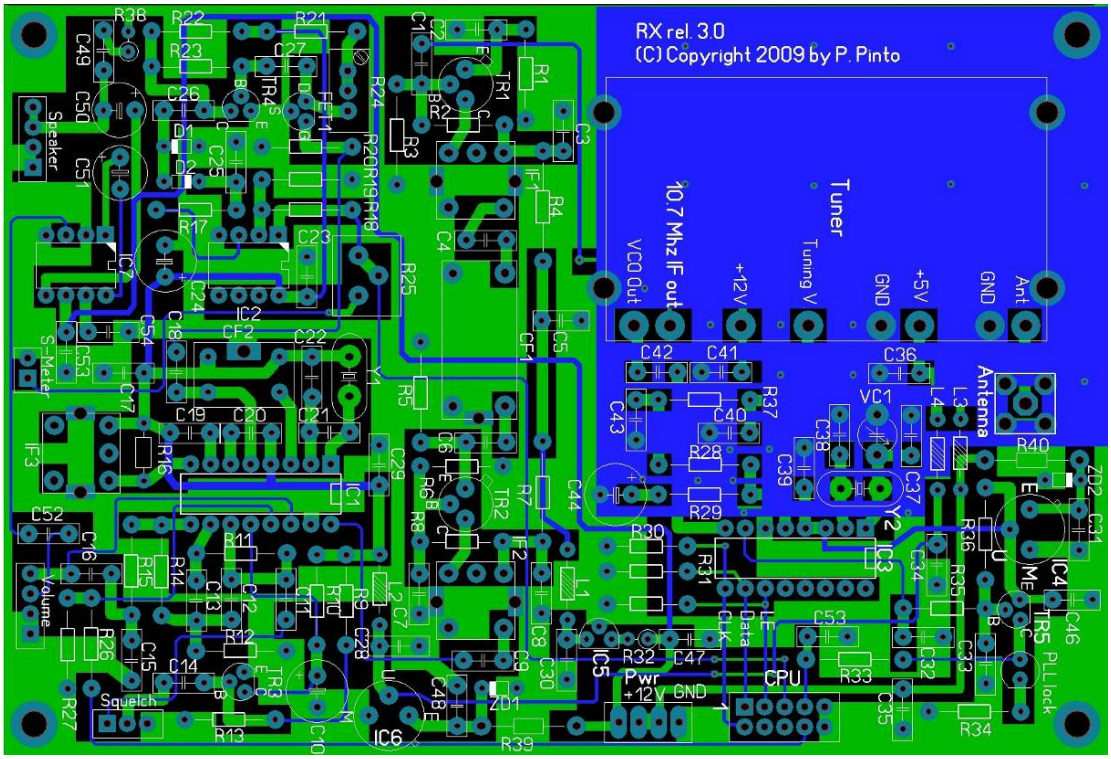


Figura 12

Trasmettitore

Per la parte trasmittente ho realizzato una scheda che comprende: il VCO, il PLL, il preamplicatore audio che amplifica il segnale proveniente dal microfono ed una sezione di commutazione delle alimentazioni. La scheda riceve i segnali di comando dalla CPU e svolge i seguenti compiti:

1) generazione della frequenza di trasmissione;
2) controllata stabilità frequenza con PLL;
3) modulazione del segnale;
4) controllo del pulsante di trasmissione del micro;
5) commutazioni delle tensioni.

Il VCO che ho utilizzato è il modulo SMD LX.1235/3 di Nuova Elettronica con escursione di frequenza da 70 Mhz a 150 Mhz. Questo modulo produce un ottimo segnale sinusoidale con una buona potenza in grado di pilotare direttamente il finale ibrido BGY45A. Il PLL

MB1502 è lo stesso utilizzato nella parte ricevente, a questo è collegato un operazionale LM358 che fornisce la tensione ai Varicap del VCO. Il segnale audio viene amplificato da un preamplificatore a due transistor, la profondità di modulazione può essere regolata mediante un trimmer. La scheda ospita inoltre la sezione di commutazione delle alimentazioni. Questa parte del circuito alimenta il ricevitore, la cpu e quando viene premuto il tasto di trasmissione anche il VCO, il PLL ed il finale BGY45.

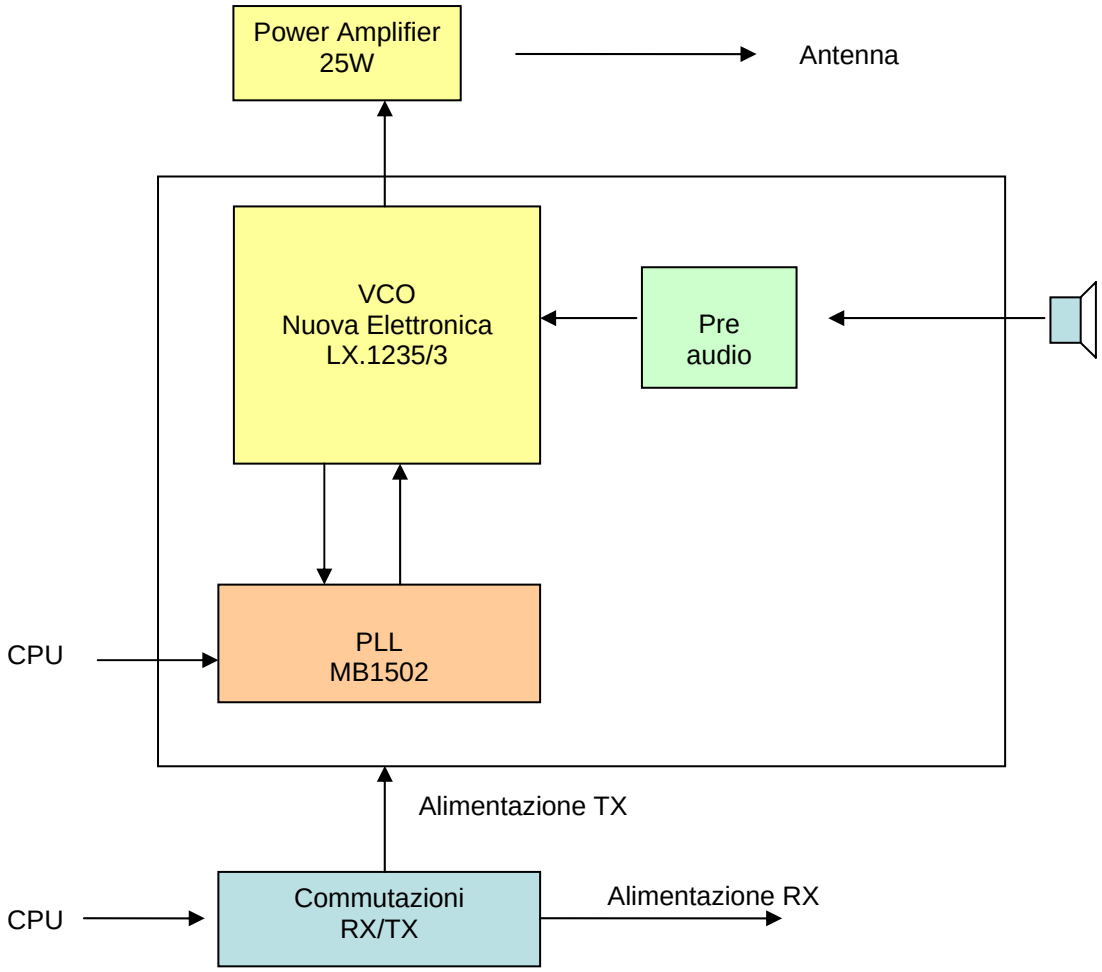


Figura 13

Schema elettrico trasmettitore

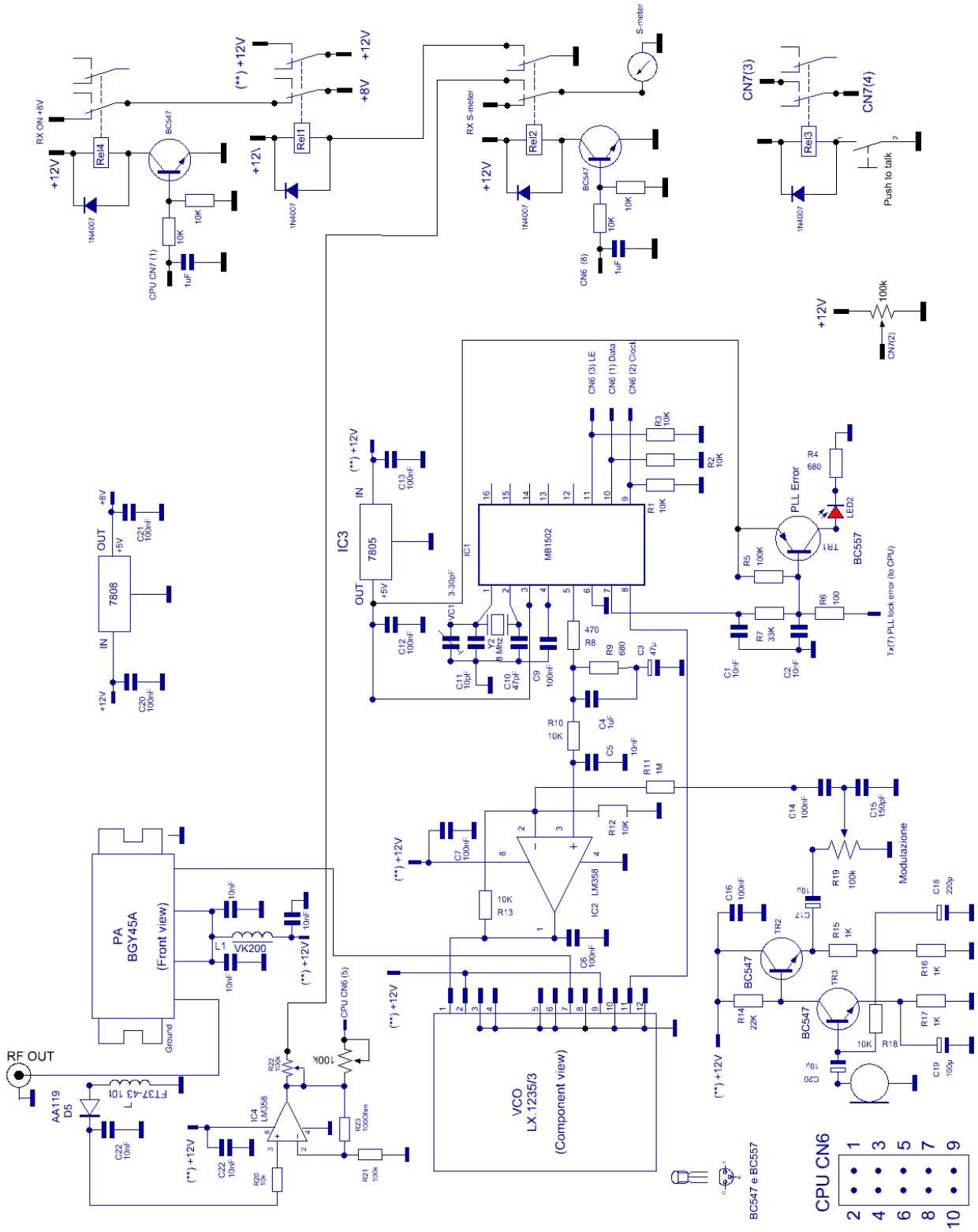


Figura 14

Circuito stampato trasmettitore

Il trasmettitore è stato realizzato con il programma Abacom software Sprint layout. Il PCB è un doppia faccia con serigrafia.

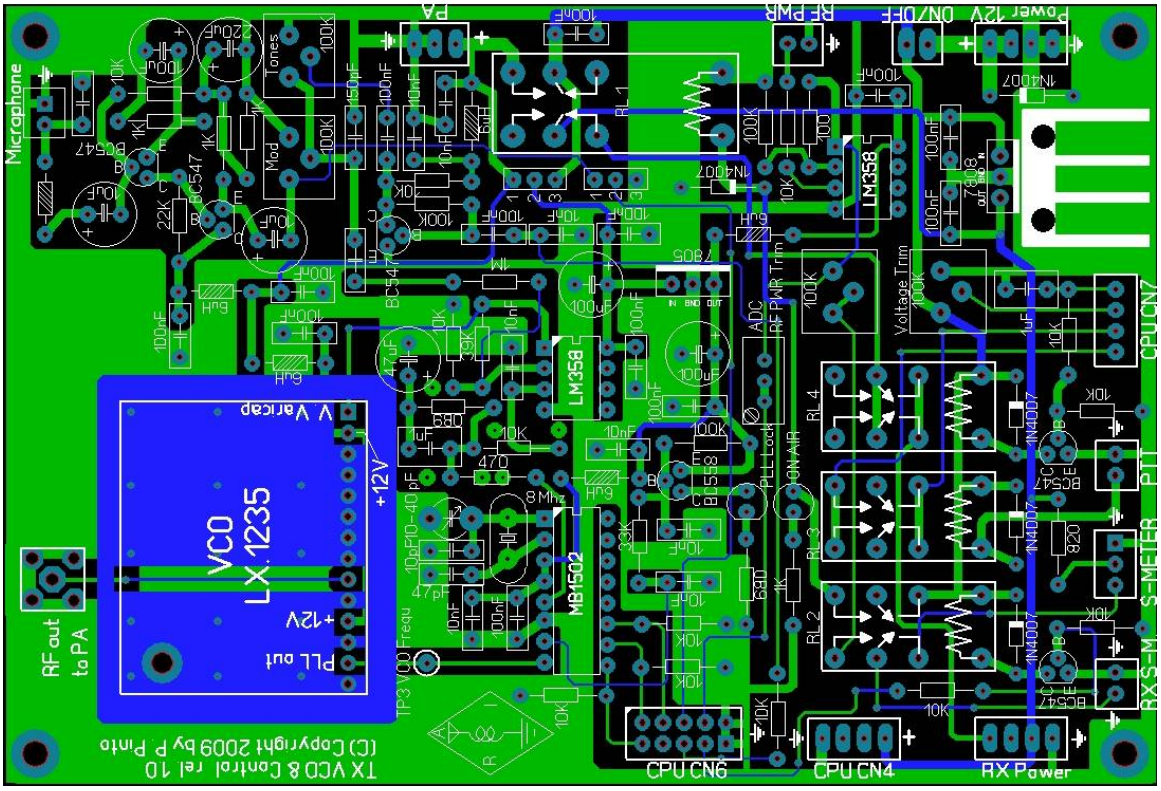


Figura 15

Commutazioni segnali ed alimentazione

La commutazione delle tensioni viene gestita dalla CPU. Il processore controlla tramite una delle sue CPU (cog) lo stato della linea A22.

Se questa linea è alimentata (a 3.3V) la CPU interpreta questo segnale come pressione del pulsante “push to talk” del microfono. La linea A7 viene utilizzata per polarizzare un transistor che apre e chiude il rele RL2. Il Rele RL2 commuta il segnale dell’S-meter, e alimenta un altro rele RL1 che fornisce tensione alla parte trasmittente o alla parte ricevente.

Sulla scheda del trasmettitore è presente un transistor preamplificatore collegato alla linea A6 della CPU. Poiché il “Propeller” è in grado di produrre suoni o sintetizzare la voce umana, questa linea potrà essere utilizzata per inviare in trasmissione un “roger beep” di fine trasmissione o addirittura in automatico il proprio nominativo.

La commutazione delle tensioni poteva semplicemente essere manuale senza passare dalla CPU mentre in questo modo ho la possibilità di trasmettere un suono prima di ritornare in ricezione.

Calcolo dei fattori di divisione del PLL

Ho trovato molto utile questo documento per comprendere il funzionamento dei PLL:

http://www.webalice.it/max4000/r3valsusa/Documenti/PLL%20Per%20Radioamatori.pdf

Esempio di calcolo dei fattori N ed A del PLL MB1502 per la frequenza di 75 Mhz:

RIF = 12.5 Khz = 12500 Hz
P = 64
F = 75 Mhz = 75000000Hz

$$F_{VCO} = RIF \cdot \left[\left(P \cdot N \right) + A \right]$$

Per calcolare N ed A:

Fattore di divisione complessivo: $F_{DC} = \frac{F_{VCO}}{P \cdot RIF} = \frac{75000000}{64 \cdot 12500} = 93.75$

N=parte intera(F_{DC})=parte intera(93.75) = **93**
A=P * parte decimale(F_{DC}) = 64*0.75 = **48**

Per impostare il PLL sulla frequenza di 75 Mhz il controllore deve inviare sulle linee Data, LE, Clock il valore N seguito dal valore A. Nel sorgente la funzione **`PLL(N, A, clockPin, dataPin, LEPin)`** si occupa di inviare questi due parametri:

```
' imposta i registri del PLL
PRI PLL(N, A, clockPin, dataPin, LEPin)
  BS2.SHIFTOUT (dataPin, clockPin, 1, BS2#MSBFIRST, 1)     ' BIT SW
  BS2.SHIFTOUT (dataPin, clockPin, 640, BS2#MSBFIRST, 14)  ' carica il registro R
  BS2.SHIFTOUT (dataPin, clockPin, 1, BS2#MSBFIRST, 1)     ' pone il bit C=1
  BS2.PULSOUT (LEPin, 1)
  BS2.SHIFTOUT (dataPin, clockPin, N, BS2#MSBFIRST, 11)    ' carica il registro N
  BS2.SHIFTOUT (dataPin, clockPin, A, BS2#MSBFIRST, 7)     ' carica il registro A
  BS2.SHIFTOUT (dataPin, clockPin, 0, BS2#MSBFIRST, 1)     ' pone il bit C=0
  BS2.PULSOUT (LEPin, 1)
```

Per il calcolo on-line dei parametri relativi ai PLL della Fujitsu visitate questa pagina:

http://f5soh.free.fr/montages/mb150x/mb150x.html

Firmware

Funzioni

Il firmware presenta queste funzioni:

1) Controlla il display LCD;
2) Legge il tastierino;
3) Controlla le line dati del Propeller connesse al modulo RX ed al modulo TX;
4) Permette l'impostazione della frequenza di lavoro;
5) Effettua la scansione delle frequenze con uno step di 25 Khz;
6) Controlla lo stato del muting (squelch);
7) Controlla la commutazione ricezione / trasmissione;
8) Visualizza l'ora dall'orologio interno.

Note: la parte relative al convertitore ADC /DAC è in fase di sviluppo.

Comandi

Tasto	Azione
A	Imposta una frequenza;
D	Conferma di dati impostati (enter);
1	Diminuisce la frequenza di 25 Khz;
2	Incrementa la frequenza di 25 Khz;
B	Effettua una scansione delle frequenze;
*	Visualizza l'ora.

Descrizione

L'istruzione `_XINFREQ = 5_000_000` indica alla CPU che il quarzo dell'oscillatore è da 5 Mhz. Le due costanti LFREQU e HFREQU impostano il limite inferiore e superiore di ricezione e trasmissione.

Nella sezione Obj sono indicate le librerie richieste per il funzionamento: La routine principale è "Init", dopo la visualizzazione del messaggio iniziale parte un loop infinito con il quale la CPU controlla lo stato dell'encoder di tastiera, del muting, se l'utente ha premuto il tasto di trasmissione.

Ho scritto delle piccole funzioni che controllano alcune linee dati della CPU. Ognuna di queste funzioni viene mandata in esecuzione su un diverso processore. Un cog viene dedicato alla lettura della linea dati "squelch", un cog viene dedicato al controllo del lock del PLL, un cog si occupa di restituire il codice di un tasto del tastierino e un cog controlla se l'apparato è in ricezione o in trasmissione.

Ognuna di queste funzioni restituisce un valore che viene memorizzato in una variabile pubblica @s, @l, @k, @rt. Il loop principale viene eseguito su un altro cog. Quindi al momento il software prevede l'utilizzo di cinque CPU, ne restano tre libere di svolgere altri compiti.

```
cognew(SQUELCH(@s),@stack[0])    ' controlla il muting
cognew(PLL_LOCK(@l),@stack[10])  ' controlla il lock del PLL del ricevitore
```

cognew(keypad(@k),@stack[20]) ' restituisce il tasto premuto sul tastierino
cognew(RXTX(@rt),@stack[30]) ' controlla se l'apparato è in ricezione o in trasmissione

La funzione PRI PLL_PARAMS(F) calcola i parametri N ed A del PLL del ricevitore e del trasmettitore. Il parametro di input è F (la frequenza). Si noti, nel sorgente della funzione, come la frequenza del VCO del ricevitore deve essere superiore al quella del VCO del trasmettitore di 10.7 Mhz.

La funzione PRI FREQU_SCAN effettua una scansione delle frequenze controllando lo stato dello squelch. Se lo squelch va in off interrompe la scansione sulla frequenza corrente.

La funzione PRI keypad(k2) legge da tastierino i tasti premuti e va a cercare in una lookup table i valore corrispondente a tale tasto che viene restituito.

La funzione PRI RXTX(rt2) controlla se viene premuto il tasto di trasmissione sul microfono.

La funzione PRI SET_RX_PLL invia al PLL del ricevitore collegato alle line 16, 17 e 18 i parametri NRX ed ARX calcolati dalla funzione PRI PLL_PARAMS.

La funzione PRI SET_TX_PLL invia al PLL del ricevitore collegato alle line 19, 20 e 21 i parametri NTX ed ATX calcolati dalla funzione PRI PLL_PARAMS.

La funzione PRI PLL(N, A, clockPin, dataPin, LEPin) invia al PLL identificato sulle linee dati passate come parametri i parametri N ed A (utilizza la libreria BS2).

Codice sorgente Spin

```
CON
  _CLKMODE      = XTAL1 + PLL16X
  _XINFREQ      = 5_000_000
  RIF=12500
  P=64
  MIF=10700
  LFREQU=70000
  HFREQU=72000

VAR
  LONG k, s, l, rt, hh, mm, ss, rtf, nrx, arx, ntx, atx
  LONG FREQU, FREQU_TEMP, stack[60]

  long  ore, minuti, secondi, pin[1]
  long storeData[3]


    ' coordinate LCD
    '1,1 PRIMA RIGA
    '2,1 SECONDA RIGA
    '1,21 TERZA RIGA
    '2,21 QUARTA RIGA


OBJ
  LCD : "LCDDriver"
  BS2 : "BS2_Functions"
  rtc : "DS1302"
  eeprom  : "Propeller Eeprom"
  fm      : "FloatMath"
  fs      : "FloatString"
```

```
PUB Init
'eeprom.VarRestore(@storeData, @storeData[0] + 1)   ' carica dalla EEPROM i canali di
ricezione e di trasmissione salvati
'FREQU:=storeData[0]
IF FREQU==0
  FREQU:=LFREQU

' imposta il PLL dell'RX
PLL_PARAMS(FREQU)
SET_RX_PLL

LCD.START
LCD.CLEAR
LCD.MOVE(1,1)
LCD.STR(STRING("MarconProp ver. 1"))
LCD.MOVE(2,1)
LCD.STR(STRING("4m - 70Mhz RTX"))
LCD.MOVE(1,21)
LCD.STR(STRING("(C) 2009 by P.Pinto"))
WAITCNT(50_000_000 + CNT)
LCD.CLEAR

s:=0
l:=0
k:=0
rt:=0

hh:=0
mm:=0
ss:=0
rtf:=0


' assegna alle diverse CPU compiti diversi
cognew(SQUELCH(@s),@stack[0])    ' controlla il muting
cognew(PLL_LOCK(@l),@stack[10])  ' controlla il lock del PLL del ricevitore
cognew(keypad(@k),@stack[20])    ' restituisce il tasto premuto sul tastierino
cognew(RXTX(@rt),@stack[30])         '  controlla  se  l'apparato  è  in  ricezione  o  in
trasmissione

'visualizza la frequenza
FREQU_SHOW(FREQU)


rtc.init(25, 24, 23)

REPEAT
' visualizza il messaggio relativo all'aggancio del PLL

    IF l==0
          LCD.MOVE(2,21)
          LCD.STR(STRING("PLL ERROR *"))
          WAITCNT(5_000_000 + CNT)
    ELSE
          LCD.MOVE(2,21)
          LCD.STR(STRING("PLL LOCK  *"))



    IF rt==1 ' rx/tx flag
          LCD.MOVE(1,21)
          LCD.STR(STRING("*** ON AIR ***"))
          LCD.MOVE(2,33)
          LCD.STR(STRING("TX MODE"))

          IF rtf==0 ' imposta il PLL del TX una sola volta
            ' imposta il PLL del TX
            SET_TX_PLL
            rtf:=1
    ELSE
          rtf:=0
          ' visualizza il messaggio relativo allo squelch
          IF s==1
             LCD.MOVE(1,21)
             LCD.STR(STRING("Muting OFF      "))
          ELSE
             LCD.MOVE(1,21)
             LCD.STR(STRING("Muting ON       "))

          LCD.MOVE(2,33)
```

```
          LCD.STR(STRING("RX MODE "))


          if k==1  ' decrementa la frequenza di 25 khz
            IF FREQU=>LFREQU+25
                BEEP
                FREQU:=FREQU-25
                FREQU_SHOW(FREQU)
                PLL_PARAMS(FREQU)
                SET_RX_PLL
                k:=16

          elseif k==2 ' incrementa la frequenza di 25 khz
            IF FREQU=<HFREQU-25
                BEEP
                FREQU:=FREQU+25
                FREQU_SHOW(FREQU)
                PLL_PARAMS(FREQU)
                SET_RX_PLL
                k:=16

          elseif k==3 ' tasto 3

          elseif k==4 ' imposta la frequenza al limite inferiore
                FREQU:=LFREQU
                FREQU_SHOW(FREQU)
                PLL_PARAMS(FREQU)
                SET_RX_PLL
                k:=16

          elseif k==5 ' imposta la frequenza al limite superiore
                FREQU:=HFREQU
                FREQU_SHOW(FREQU)
                PLL_PARAMS(FREQU)
                SET_RX_PLL
                k:=16

          elseif  k==7    ' imposta  la  frequenza  sulla  prima  concessione  sperimentale:
70.100 Mhz
                FREQU:=70100
                FREQU_SHOW(FREQU)
                PLL_PARAMS(FREQU)
                SET_RX_PLL
                k:=16

          elseif  k==8  ' imposta  la  frequenza  sulla  seconda  concessione  sperimentale:
70.200 Mhz
                FREQU:=70200
                FREQU_SHOW(FREQU)
                PLL_PARAMS(FREQU)
                SET_RX_PLL
                k:=16

          elseif k==9 ' imposta la frequenza sulla terza concessione sperimentale: 70.300
Mhz
                FREQU:=70300
                FREQU_SHOW(FREQU)
                PLL_PARAMS(FREQU)
                SET_RX_PLL
                k:=16

          elseif k==10 ' tasto A imposta la frequenza
                LCD.CLEAR
                FREQU_TEMP:=FREQU

                LCD.MOVE(2,21)
                LCD.STR(STRING("F.range: "))
                LCD.DEC (LFREQU/1000)
                LCD.STR(STRING("-"))
                LCD.DEC (HFREQU/1000)
                LCD.STR(STRING("Mhz"))

                repeat
                    FREQU:=SET_FREQU
                    if(FREQU<LFREQU  or  FREQU>HFREQU  or  ((FREQU//1000//25)<>0))  '  se  la
frequ. non è nel range ed i khz non sono di 25 khz
                        LCD.MOVE(1,21)
                        LCD.STR(STRING("Frequency Error !"))

                        FREQU:=FREQU_TEMP ' ripristina la frequenza precedente
```

```
                    WAITCNT(100_000_000 + CNT)
                    LCD.CLEAR
                    quit

              while (FREQU<LFREQU or FREQU>HFREQU)

              LCD.CLEAR

              FREQU_SHOW(FREQU)
              PLL_PARAMS(FREQU)
              SET_RX_PLL

              beep
              k:=16

        elseif k==11 ' tasto B: scansione delle frequenze
              s:=0
              FREQU_SCAN
              k:=16

        elseif k==13 ' tasto asterisco visualizza l'ora
              beep
              rtc.readTime( @ore, @minuti, @secondi )

              LCD.MOVE(2,1)
              LCD.STR(STRING("Time: "))

              LCD.MOVE(2,8)
              if(ore<10)
                 LCD.STR(STRING("0"))
              LCD.DEC (ore)

              LCD.MOVE(2,10)
              LCD.STR(STRING(":"))

              LCD.MOVE(2,11)
              if(minuti<10)
                 LCD.STR(STRING("0"))
              LCD.DEC (minuti)

              LCD.MOVE(2,13)
              LCD.STR(STRING(":"))

              LCD.MOVE(2,14)
              if(secondi<10)
                 LCD.STR(STRING("0"))
              LCD.DEC (secondi)
              WAITCNT(100_000_000 + CNT)
              LCD.MOVE(2,1)
              LCD.STR(STRING("                        "))

              k:=16



PRI PLL_PARAMS(F) | N,N1,N2,A,F2
  F2:=F
  F:=F*1000
  NTX:=(F/(RIF*P))
  N1:=F/(RIF*P/1000)
  N2:=N1-(NTX*1000)
  ATX:=(P*N2)/1000
  IF (ATX//2)==1
      ATX:=ATX+1

  F:=F2+MIF
  F:=F*1000
  NRX:=(F/(RIF*P))
  N1:=F/(RIF*P/1000)
  N2:=N1-(NRX*1000)
  ARX:=(P*N2)/1000
  IF (ARX//2)==1
      ARX:=ARX+1


PRI FREQU_SCAN
   repeat
      if(FREQU<HFREQU AND s==0) ' non è stato raggiunto il limite superiore
```

```
         FREQU:=FREQU+25  ' incrementa la frequenza di 25 khz
         FREQU_SHOW(FREQU)' visualizza la frequenza
         PLL_PARAMS(FREQU)
         SET_RX_PLL' imposta il PLL dell'RX

      if(FREQU=>HFREQU)    ' se ha raggiunto il limite superiore riporta la frequenza al
limite inferiore
         FREQU:=LFREQU
      if s==1
       ' se il muting va in off ha trovato un segnale ed esce dal loop
         longbeep
         WAITCNT(5_000_000 + CNT)
         longbeep
         QUIT
      WAITCNT(20_000_000 + CNT) ' imposta uno stato di attesa per la visualizzazione
della frequenza
   while(k<>15)         ' finquando non viene premuto il tasto D continua

' legge lo stato dell'aggancio del PLL del ricevitore
PRI PLL_LOCK(l2)
  repeat
    long[l2]:=INA[9]

' legge lo stato dello squelch
PRI SQUELCH(s2)
  repeat
    long[s2]:=INA[8]

' legge il tasto premuto sul tastierino
PRI keypad(k2) : i
   repeat
    IF INA[11]==1
      i:=INA[12..15]  ' tasto premuto
      long[k2]:=lookupz(i:1,2,3,10,4,5,6,11,7,8,9,12,13,0,14,15)

      ' 1,2,3,A,4,5,6,B,7,8,9, C,*,0,#,D
      ' Codifica dei tasti
      '
      ' TASTO | Valore restituito
      ' -------------------------
      '   1   |        1
      '   2   |        2
      '   3   |        3
      '   A   |        10
      '   4   |        4
      '   5   |        5
      '   6   |        6
      '   B   |        11
      '   7   |        7
      '   8   |        8
      '   9   |        9
      '   C   |        12
      '   *   |        13
      '   0   |        0
      '   #   |        14
      '   D   |        15
      ' -------------------------


' legge il pin 22 relativo alla modalità di rx/tx
PRI RXTX(rt2)
   repeat
      if(INA[22]==1)
        ON_AIR
      else
        RECEPTION
      long[rt2]:=INA[22]



' legge l'ora dal DS1302
PRI OROLOGIO(hh2, mm2, ss2)
  rtc.init(25, 24, 23)
  repeat
     ' read time from DS1302
    long[hh2]:=ore
    long[mm2]:=minuti
    long[ss2]:=secondi

' imposta il PLL del ricevitore
```

```
PRI SET_RX_PLL
   PLL(NRX,ARX,16,17,18)


' imposta il PLL del trasmettitore
PRI SET_TX_PLL
  ' invia i dati al PLL del TX
  PLL(NTX,ATX,19,20,21)


' imposta i registri del PLL
PRI PLL(N, A, clockPin, dataPin, LEPin)
  BS2.SHIFTOUT (dataPin, clockPin, 1, BS2#MSBFIRST, 1)     ' BIT SW
  BS2.SHIFTOUT (dataPin, clockPin, 640, BS2#MSBFIRST, 14)  ' carica il registro R
  BS2.SHIFTOUT (dataPin, clockPin, 1, BS2#MSBFIRST, 1)     ' pone il bit C=1
  BS2.PULSOUT (LEPin, 1)
  BS2.SHIFTOUT (dataPin, clockPin, N, BS2#MSBFIRST, 11)    ' carica il registro N
  BS2.SHIFTOUT (dataPin, clockPin, A, BS2#MSBFIRST, 7)     ' carica il registro A
  BS2.SHIFTOUT (dataPin, clockPin, 0, BS2#MSBFIRST, 1)     ' pone il bit C=0
  BS2.PULSOUT (LEPin, 1)


' visualizza la frequenza di ricezione e di trasmissione
PRI FREQU_SHOW(F) | x
  x:=f//1000
  LCD.MOVE(1,1)
  LCD.DEC (f/1000)     ' visualizza le prime due cifre (Mhz)

  LCD.MOVE(1,3)
  LCD.STR(STRING(".")) ' visualizza il punto decimale

  LCD.MOVE(1,4)        ' visualizza le cifre dopo il punto decimale (khz)
  if(x<100)
    if(x==0)
      LCD.STR(STRING("0"))
    LCD.STR(STRING("0"))
    LCD.DEC (x)
  else
    LCD.DEC (x)

  LCD.MOVE(1,8)
  LCD.STR(STRING(" Mhz"))

' attiva rele trasmissione
PRI ON_AIR
  DIRA[7]~~
  OUTA[7]~~

' disattiva rele trasmissione
PRI RECEPTION
  OUTA[7]~

' emette un beep
PRI beep
  DIRA[10]~~
  OUTA[10]~~
  WAITCNT(1_00_000 + CNT)
  OUTA[10]~

' emette un beep
PRI longbeep
  DIRA[10]~~
  OUTA[10]~~
  WAITCNT(5_00_000 + CNT)
  OUTA[10]~

' imposta la frequenza
PRI SET_FREQU : j | v,x,y,z,w
  LCD.MOVE(1,1)
  LCD.STR(STRING("Frequency set:"))

  v:=0
  j:=0
  x:=0
  y:=0
  z:=0
  w:=0
  LCD.MOVE(2,1)
  LCD.STR(STRING("["))
```

```
LCD.MOVE(2,7)
LCD.STR(STRING("]"))
repeat
    if (j==0 and INA[11]==1 and k<10)
           v:=k*10000
           j:=1
           LCD.MOVE(2,2)
           LCD.STR(STRING("     "))
           LCD.MOVE(2,2)
           LCD.DEC(v/10000)
           WAITCNT(10_000_000 + CNT)
    if (j==1 and INA[11]==1 and k<10)
           x:=v+k*1000
           j:=2
           LCD.MOVE(2,2)
           LCD.STR(STRING("     "))
           LCD.MOVE(2,2)
           LCD.DEC(x/1000)
           WAITCNT(10_000_000 + CNT)
    elseif (j==2 and INA[11]==1 and k<10)
           y:=x+k*100
           j:=3
           LCD.MOVE(2,2)
           LCD.STR(STRING("     "))
           LCD.MOVE(2,2)
           LCD.DEC(y/100)
           WAITCNT(10_000_000 + CNT)
    elseif (j==3 and INA[11]==1 and k<10)
           z:=y+k*10
           j:=4
           LCD.MOVE(2,2)
           LCD.STR(STRING("     "))
           LCD.MOVE(2,2)
           LCD.DEC(z/10)
           WAITCNT(10_000_000 + CNT)
    elseif (j==4 and INA[11]==1 and k<10)
           w:=z+k
           j:=0
           LCD.MOVE(2,2)
           LCD.STR(STRING("     "))
           LCD.MOVE(2,2)
           LCD.DEC(w)
           WAITCNT(10_000_000 + CNT)
    WAITCNT(10_000_000 + CNT)
while k <> 15
return w
```

Foto

Figura 16: Vista frontale

Figura 17: Vista interna dall'alto

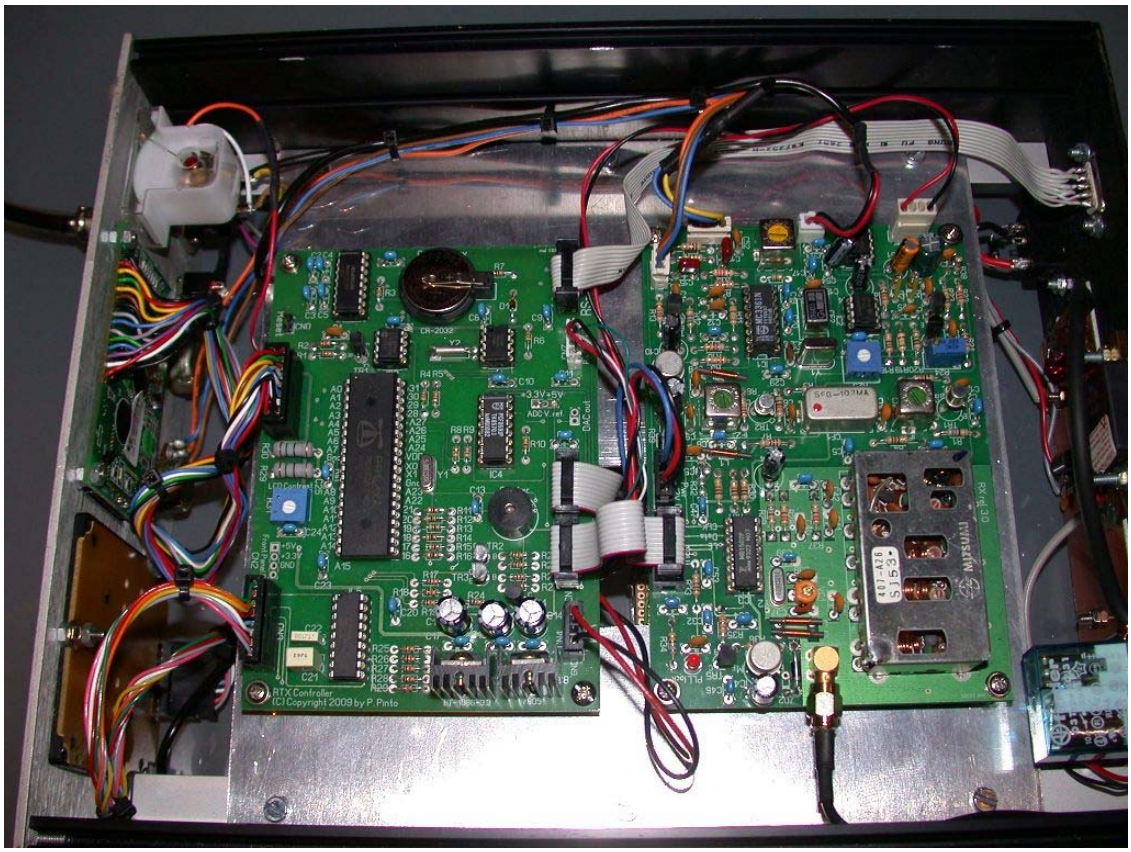


Figura 18: Vista interna superiore: Controller + Ricevitore

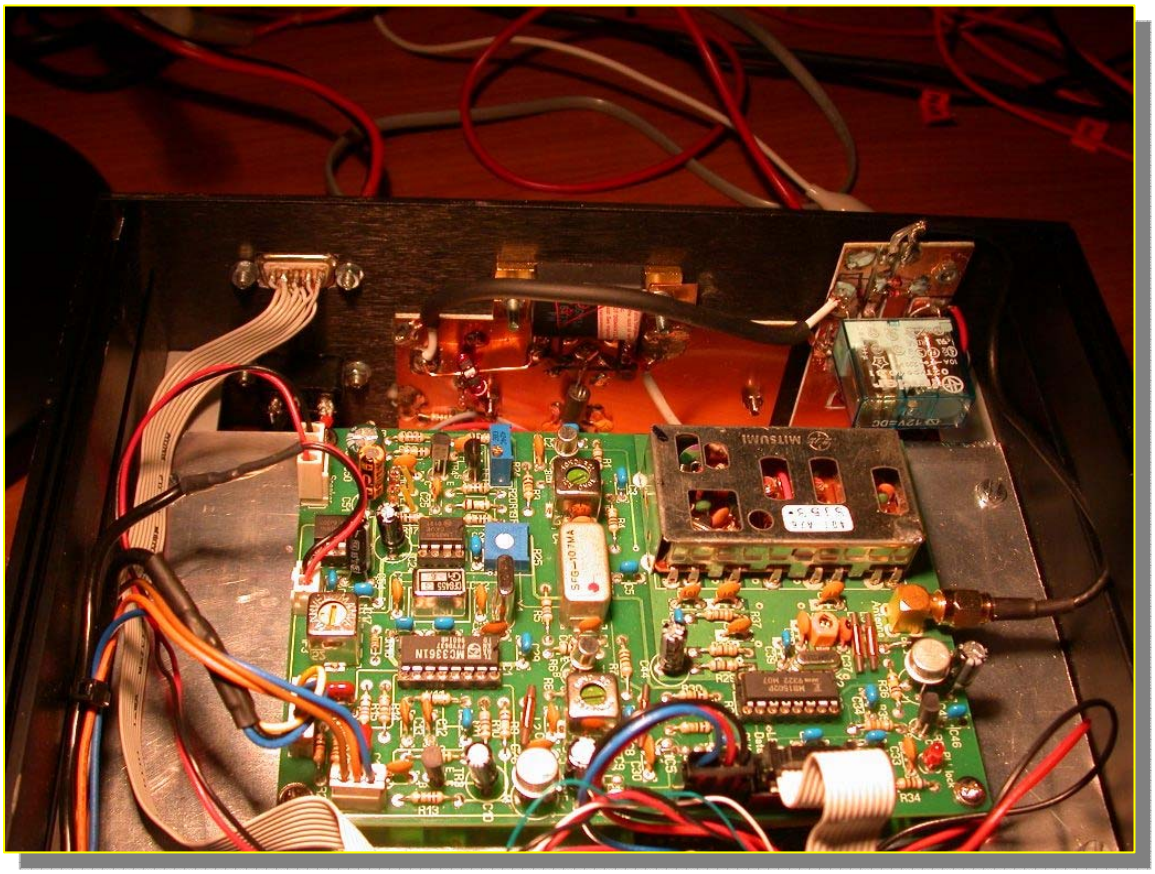


Figura 19: Dettagli del ricevitore

Figura 20: Vista posteriore

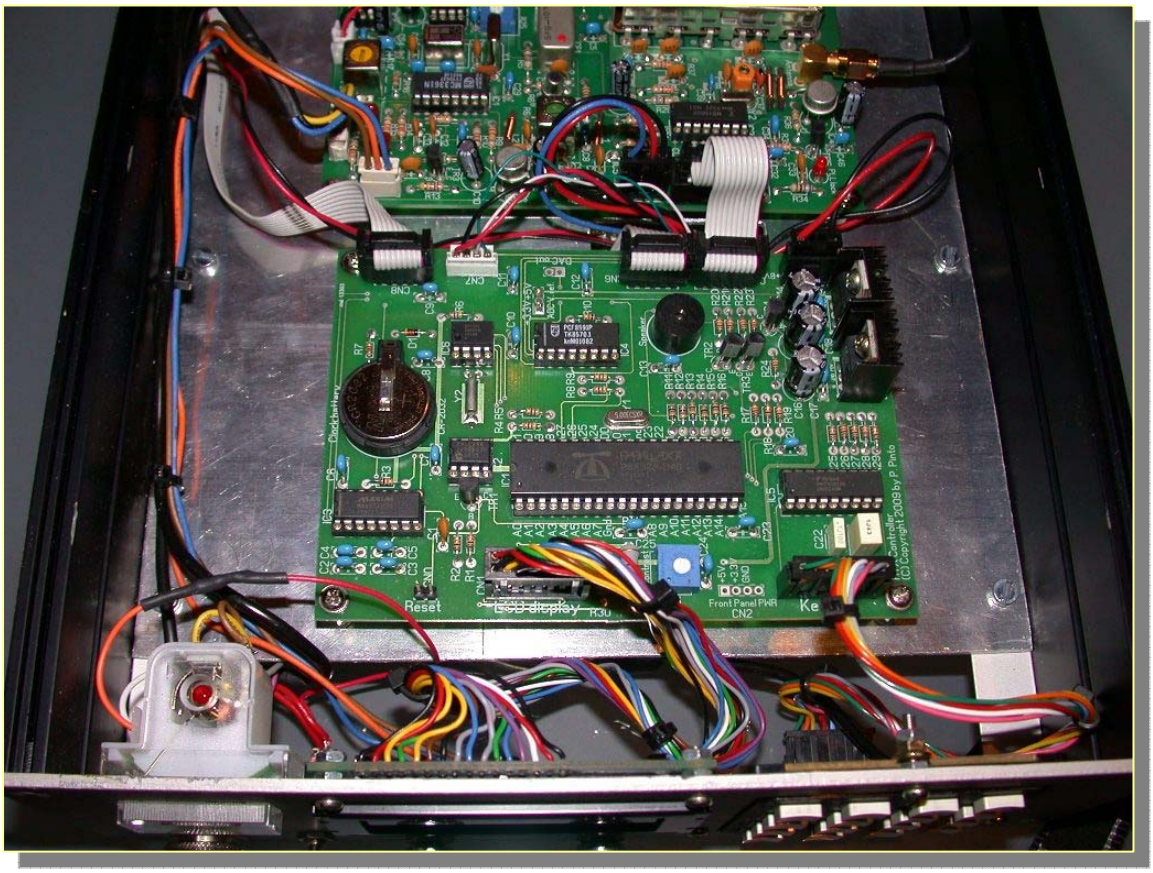


Figura 21: Dettaglio del controllore

Figura 22: Dettaglio del display

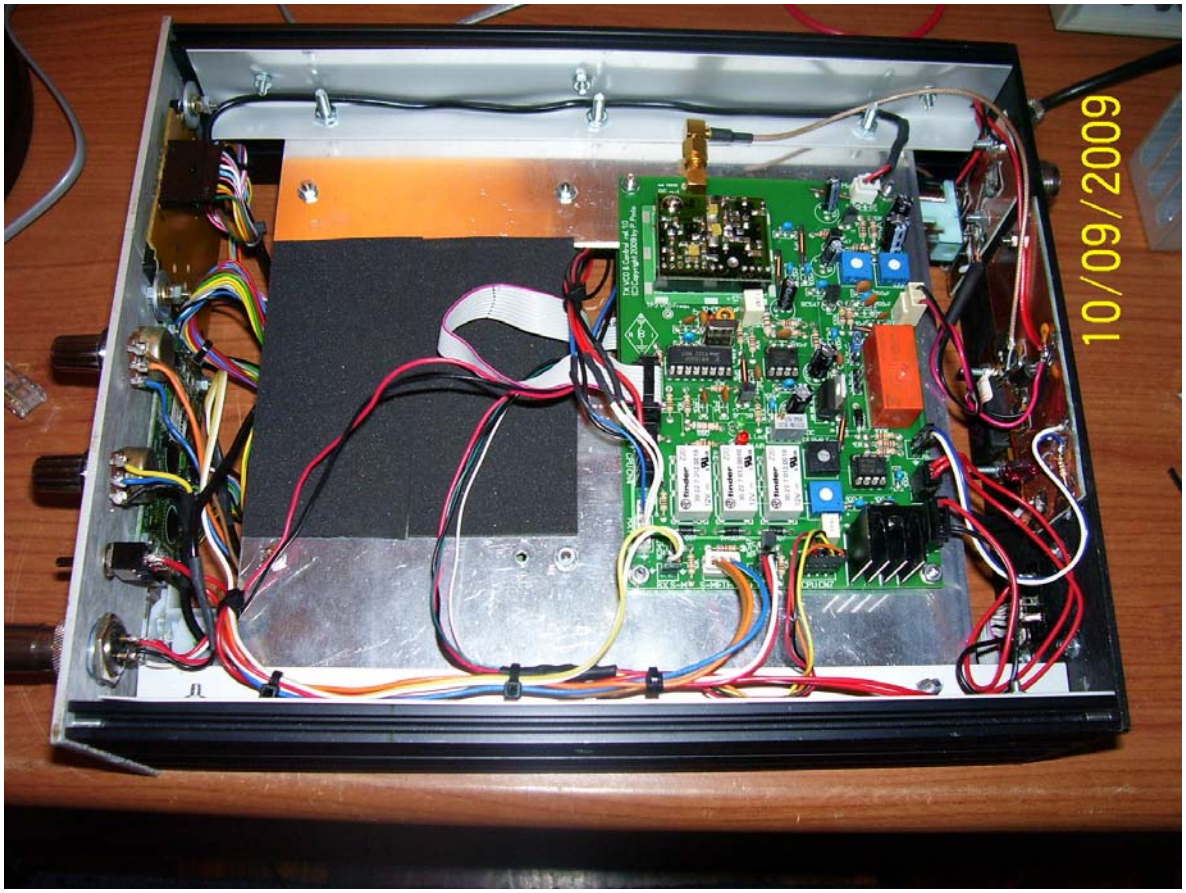


Figura 23: Vista inferiore - trasmettitore

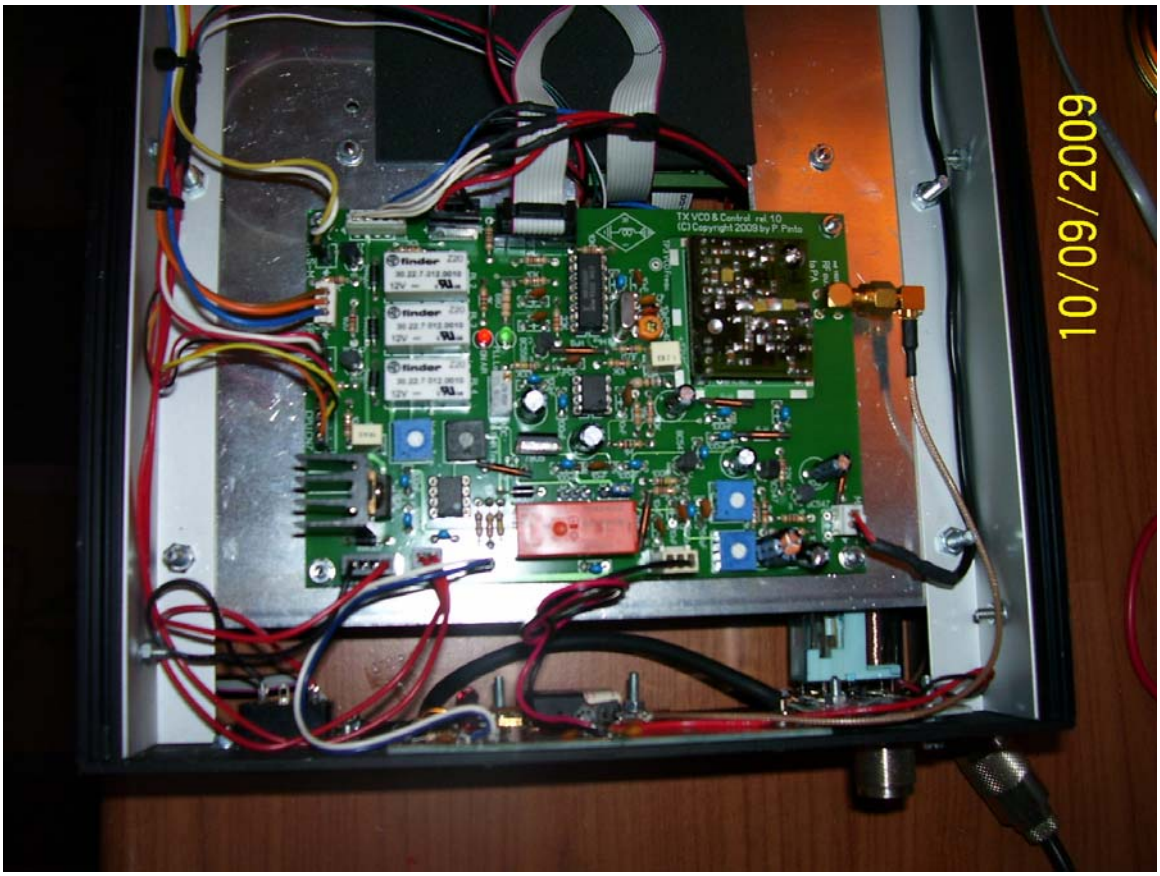


Figura 24: Dettaglio del trasmettitore

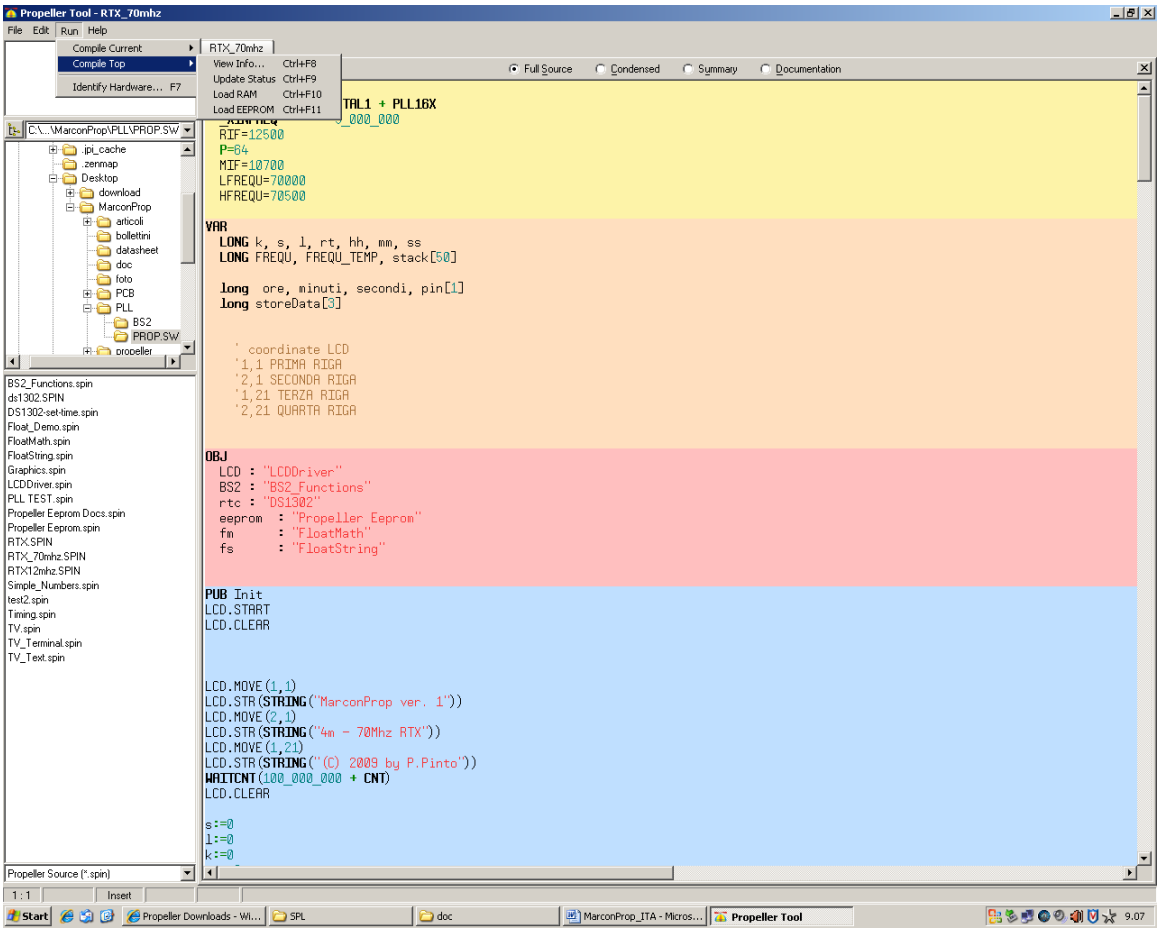


Figura 25: Ambiente di sviluppo “Propeller Editor Development System”

Bibliografia

Documentazione Propeller

http://www.parallax.com
http://www.parallax.com/dl/docs/article/Propeller_Firmware_magazine_ITA.pdf
http://www.elettronicamente.com/engine/default.asp?page=projectview&pid=701

Componenti elettronici

http://www.digikey.it	(Propeller IC e quarzo 5 Mhz 20pF)
http://www.alltronics.com	(Tuner Mitsumi 407-A26)
http://www.jameco.com	(Tastierino)
http://www.nuovaelettronica.it	(VFO TX)
http://www.webtronic.it	(ICs e connettori)
http://www.rfmicrowave.it/	(componenti HF: filtri, bobine, quarzi)
http://www.hifi2000.it/	(Contenitore)
http://www.elettroshop.it	(Propeller IC)
http://www.areasx.com	(Display LCD: 20 x 4 - 3.3 Volt model: **W2004D-TMI**)
http://www.maxim-ic.com	(un ringraziamento a Maxim per i campioni dell'integrato DS1302 e MAX3232)

Produttore circuiti stampati

Per realizzare gli stampati mi sono rivolto alla ditta **Millenium Dataware**: http://www.mdsrl.it che devo dire è molto precisa e puntuale nelle sue produzioni.

Ho potuto inoltre verificare che gli stampati prodotti sono estremamente robusti per quanto riguarda le sollecitazioni termiche delle piste. Può infatti capitare che un componente debba essere sostituito, e le tracce non si staccano con il calore del saldatore anche dopo numerosi interventi.

Spesso ordinando solo due stampati si ha la piacevole sorpresa di riceverne quattro o più. Devo infine fare presente la gentilezza dei tecnici sempre disponibili a risolvere eventuali problemi nel progetto dei PCB.

Su richiesta sono disponibili i file Gerber.

Taratura

La progettazione, l'assemblaggio e la taratura della radio richiedono un minimo di strumentazione elettronica. Per la taratura e l'allineamento dei circuiti ho utilizzato:

- un tester digitale;
- un oscilloscopio HP 54601A doppia traccia digitale da 100 Mhz;
- un frequenzimento Atten F2700-C da 2.7 ghz;
- un generatore di segnale Marconi 2018 da 80 khz - 520 Mhz;
- un alimentatore stabilizzato 0-15V;
- un induttanzimetro digitale di Nuova Elettronica;
- un carico RF da 52 ohm 30W.

Software per PCB

Il programma utilizzato per la realizzazione dei circuiti stampati è Sprint-layout 5.0 mentre quello per il disegno degli schemi elettrici è sPlan 6.0 entrambi sono acquistabili on-line sul sito:

http://www.abacom-online.de

Fra i tanti software provati a mio parere questi due sono i più semplici e completi per un uso hobbystico. Sprint-layout permette, inoltre, di creare i file Gerber e di esportare in altri formati i circuiti stampati.

Note

La maggior parte dei componenti elettronici sono stati acquistati in Italia. Il contenitore è un prodotto HIFI2000: http://www.hifi2000.it