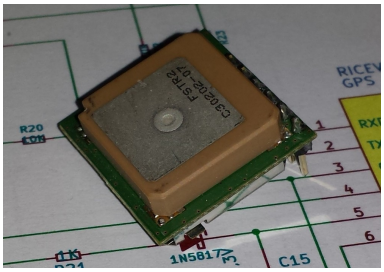
TRACKER APRS

(BASATO SUL LAVORO DI WB8WGA ed altri)

Antefatto:

Dopo un lungo periodo di inattività avevo riassemblato il mio sistema APRS utilizzando un GPS eMap della Garmin, un tracker basato sul progetto di WB8WGA ed un vecchio palmare a contravers. L'uso dell'attrezzatura oggettivamente non era molto comodo, un “tre pezzi”, con relativi cavi di alimentazione, pile, antenna non erano poi il massimo della trasportabilità. Poi c'è stato il guasto del ricevitore GPS.... acquistarne uno nuovo con interfaccia RS232 non è al momento attuale ne facilmente reperibile ne a buon mercato.



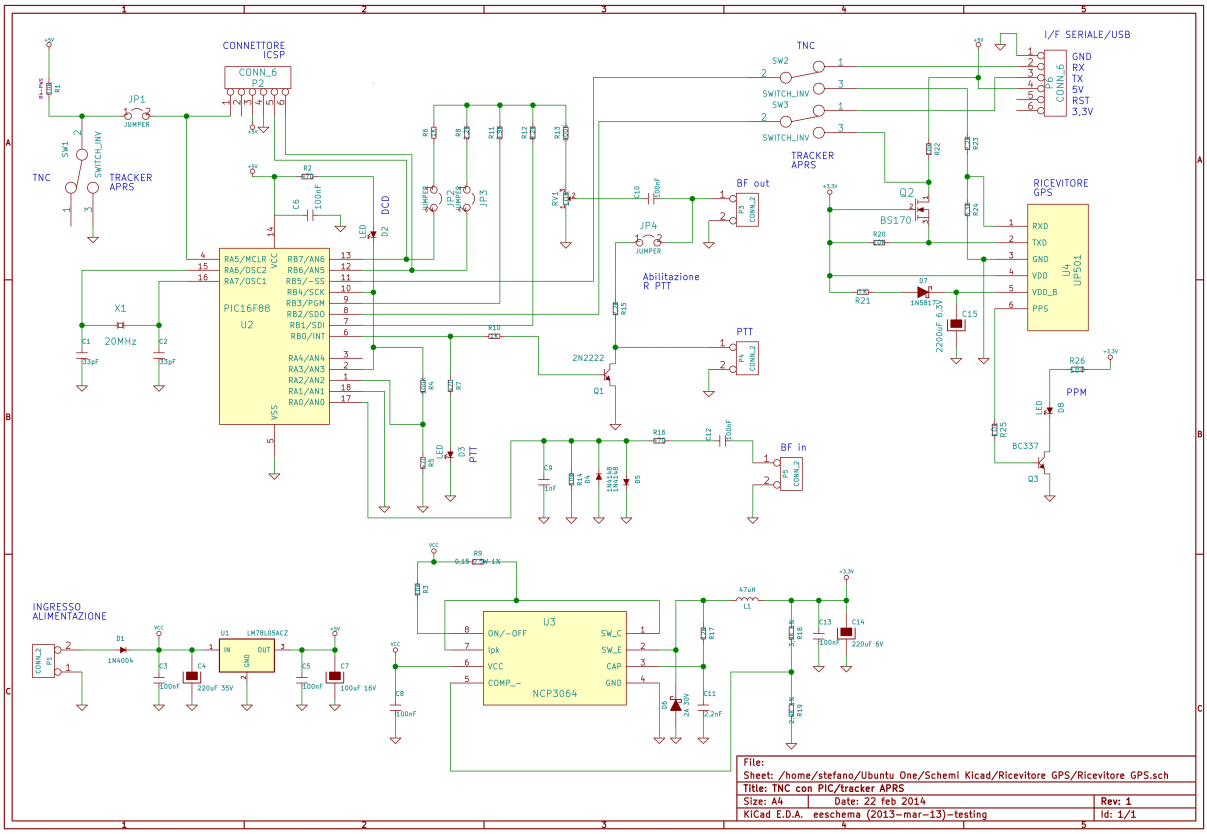
Parlando con un mio collega vengo a conoscere l'esistenza di un modulo ricevitore a basso costo che lui utilizza per la telemetria di modelli navali radiocomandati, si tratta del Fastrax UP501.

Questo GPS ha interfaccia RS232 che di default fornisce ogni secondo i dati di posizione in formato NMEA alla velocità di 9600 baud, il dispositivo è reperibile su internet ad una cifra intorno ai 16-20 € più spedizione.

Vi sono però dei problemi, il ricevitore funziona a 3,3V quindi i livelli dei segnali TTL non sono compatibili con il PIC funzionante a 5V utilizzato dal tracker, il GPS va poi scatolato per renderlo trasportabile senza che subisca danni.

Ho così deciso di integrare insieme il microcontrollore ed il GPS, semplificando così anche le operazioni in portatile.

Di seguito lo schema del circuito.



Riferimento	Valore	Riferimento	Valore	Riferimento	Valore
C1	33pF	JP2	JUMPER	R12	8,2K
C2	33pF	JP3	JUMPER	R13	100K
C3	100nF	JP4	JUMPER	R14	10K
C4	220uF 35V	L1	47uH	R15	2,2K
C5	100nF	P1	Connettore 2 poli	R16	470
C6	100nF	P2	Connettore 6 poli	R17	12K
C7	100uF 16V	P3	Connettore 2 poli	R18	3,9K 1%
C8	100nF	P4	Connettore 2 poli	R19	2,4K 1%
C9	1nF	P5	Connettore 2 poli	R20	10K
C10	100nF	P6	Connettore 4 poli	R21	1K
C11	2,2nF	Q1	2N2222	R22	10K
C12	100nF	Q2	BS170	R23	2,2K
C13	100nF	Q3	BC337	R24	3,3K
C14	220uF 6V	R1	10K	R25	10k
C15	2200uF 6,3V	R2	470	R26	180
D1	1N4004	R3	10K	RV1	10K
D2	LED	R4	100K	SW1	Jumper o dev. 3 vie
D3	LED	R5	470	SW2	Jumper o dev. 3 vie
D4	1N4148	R6	1K	SW3	Jumper o dev. 3 vie
D5	1N4148	R7	470	U1	LM78L05ACZ
D6	2A 30V	R8	2,2K	U2	PIC16F88
D7	1N5817	R9	0,15 0,5W 1%	U3	NCP3064
D8	LED	R10	1K	U4	UP501
JP1	JUMPER	R11	3,9K	X1	20MHz

Sezione alimentazione:

Sono necessarie due tensioni, la 5V per il PIC e la 3,3V per il modulo GPS. La prima è ottenuta come nel progetto di WB6WGA con un regolatore 78L05 (U1), più che sufficiente visto il ridotto assorbimenti del PIC. Per la seconda ho impiegato un regolatore di tipo a commutazione basato sull'integrato NCP3064 (U3), questo perché a causa della ridotta caduta, mettere un ulteriore stabilizzatore in cascata sulla 5V non forniva una corretta regolazione. Se invece si partiva dai classici 13,8V si otteneva un una certa dissipazione di potenza sul regolatore.

Sezione microcontrollore:

Il cuore basato sul PIC 16f88 è quella del progetto originale di WB8WGA, ho solo predisposto il circuito per un triplo deviatore (o tre jumper) che permettano agevolmente di selezionare il funzionamento come TNC o tracker. Nella prima modalità è poi possibile impostare i parametri personali della stazione come il proprio nominativo, beacon, e tutti i successivi parametri per il funzionamento in APRS. Nella seconda modalità il PIC si aspetta di ricevere i dati dal GPS per poi trasformarli in pacchetti da trasmettere via radio.

Nel progetto originale questi dati vengono scambiati originale tramite un adattatore RS232 basato sull'integrato MAX232 a cui collegare o il PC o il ricevitore GPS esterno.

Nel mio progetto ho in pratica eliminato dallo stampato il convertitore RS232 in quanto i moderni PC molto difficilmente hanno questo tipo di interfaccia, evitando in questo modo l'acquisto di un cavo di conversione USB/RS232 esterno. Ho così preferito lasciare un connettore a cui collegare una semplice interfaccia TTL/USB (P6) reperibile su internet o alle fiere ad un prezzo di alcuni euro, l'integrarla sulla scheda sarebbe costata di più tra integrato, connettore ed altro.

Rimane l'adattamento dei livelli tra il PIC e GPS; nel caso in cui il segnale provenga dal dispositivo che lavora a 5V per farlo dialogare con il dispositivo funzionante a 3,3V senza danneggiarne gli ingressi è sufficiente utilizzare un partitore di tensione (R23, R24).

Per l'operazione inversa sono ricorso ad un MOSFET (Q2) per poter effettuare la traslazione del livello. Questi quando il pin2 del GPS è a livello alto il è interdetto, nel drain non scorre corrente cosi come in R22, di conseguenza la tensione sul pin 8 del PIC collegato al drain è pari a 5V.

Quando invece la tensione in uscita del pin 2 del GPS è 0V ovvero collegato a massa la tensione tra gate e source è 3,3V sufficiente a portare in conduzione il mosfet che porterà praticamente a massa il drain.

Per la programmazione iniziale del PIC ho predisposto un connettore ICSP (in circuit serial programming) e tre jumper necessari all'isolamento della circuiteria in modo che questa non interferisca al momento dell'upload del programma.

Sezione GPS:

Non impiegando un classico GPS commerciale dotato di batteria autonoma che mantenga le effemeridi, ogni volta che si toglie l'alimentazione questi dati vengono persi ed è necessario che il ricevitore li acquisisca nuovamente, ovvero che effettui un fix a freddo. Questa operazione anche se rapida dura almeno un minuto, in un tipico utilizzo in auto con alimentazione fornita dalle instabili prese per accendisigari, non è il massimo per avere un buono e costante tracciamento della stazione.

Il modulo UP501 è dotato di un ingresso per un batteria tampone o comunque una

tensione che gli permetta di mantenere alimentate le memorie, ho così dotato il circuito un condensatore elettrolitico di discreta capacità (C15) più che sufficiente a mantenere i dati per alcuni minuti.

Software:

Su internet esistono varie versioni derivate dall'originale di WB8WGA, la presenza in spirito radioamatoriale anche dei sorgenti ha permesso ad altri di portare avanti lo sviluppo implementando nuove funzioni o migliorando il software.

Per il mio progetto ho dovuto tenere conto dei parametri di default del modulo GPS, sul sito del riferimento (1) ho trovato la versione 2.20 completa di sorgenti così ho potuto trovare il passo sul software che permette di impostare correttamente la velocità di comunicazione

```
;-------------------------------------------------------------------
; Some Key Defines
;

#define   XMIT_BUFFER_SIZE          .125              ;Buffer to store chars during converse mode
#define   MAX_BEACON_TEXT           .100                        ;max size of beacon text
#define   BEACON_EE_START           .100      ;where in EE data does beacon text start

;#define BAUD_VALUE        .129              ;For 9600 baud, set to .129 , BRGH is set
#define BAUD_VALUE  .129            ;For 4800, set to .255
                                                                  ;For 19200, set to .65

;=====================================================================
```

Per caricare il firmware sul PIC, dopo aver tolto i 3 jumper in modo da isolare la circuiteria, ho utilizzato il mio Propic 2 e gli strumenti software messi a disposizione direttamente da Microchip anche su piattaforma linux.

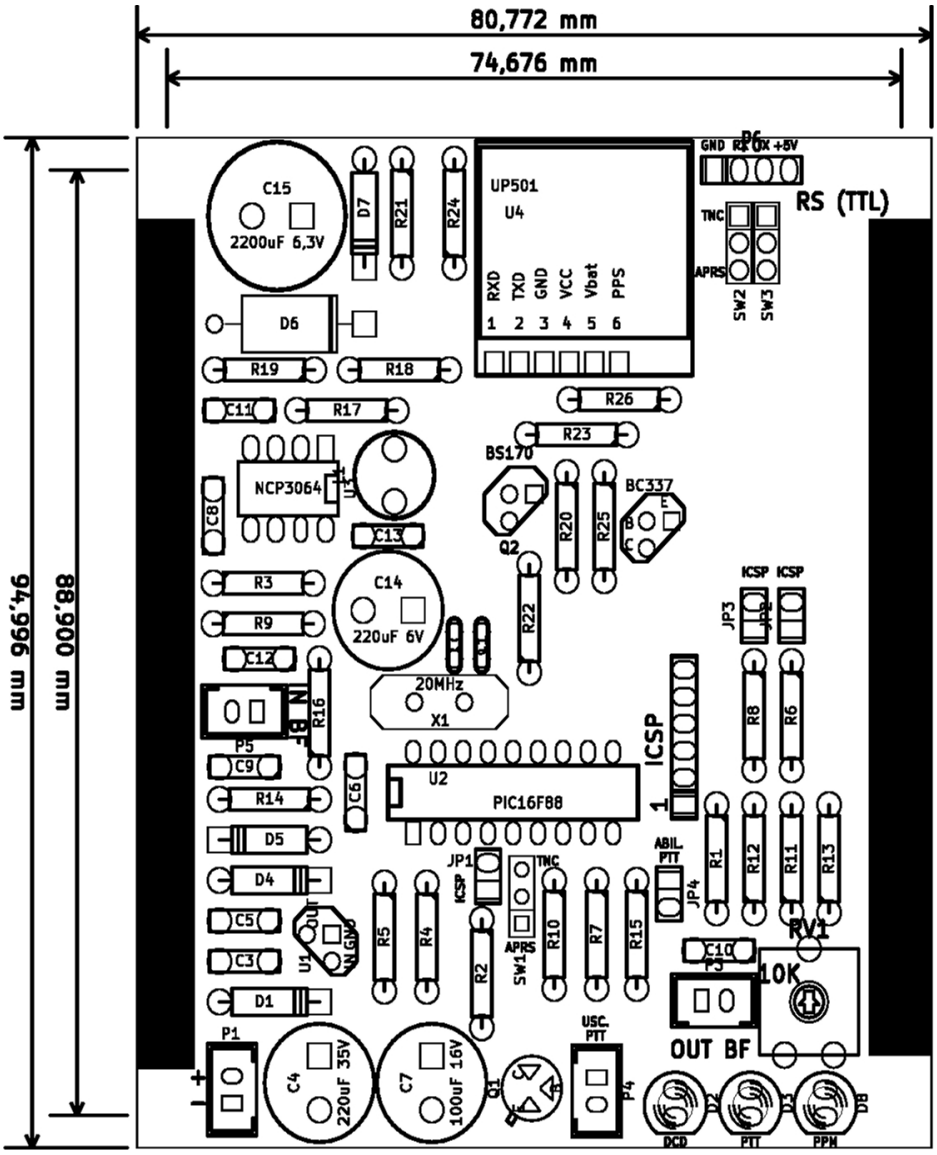
Ovviamente al termine dell'operazione ho scollegato il programmatore e reinserito i jumper.

Costruzione:

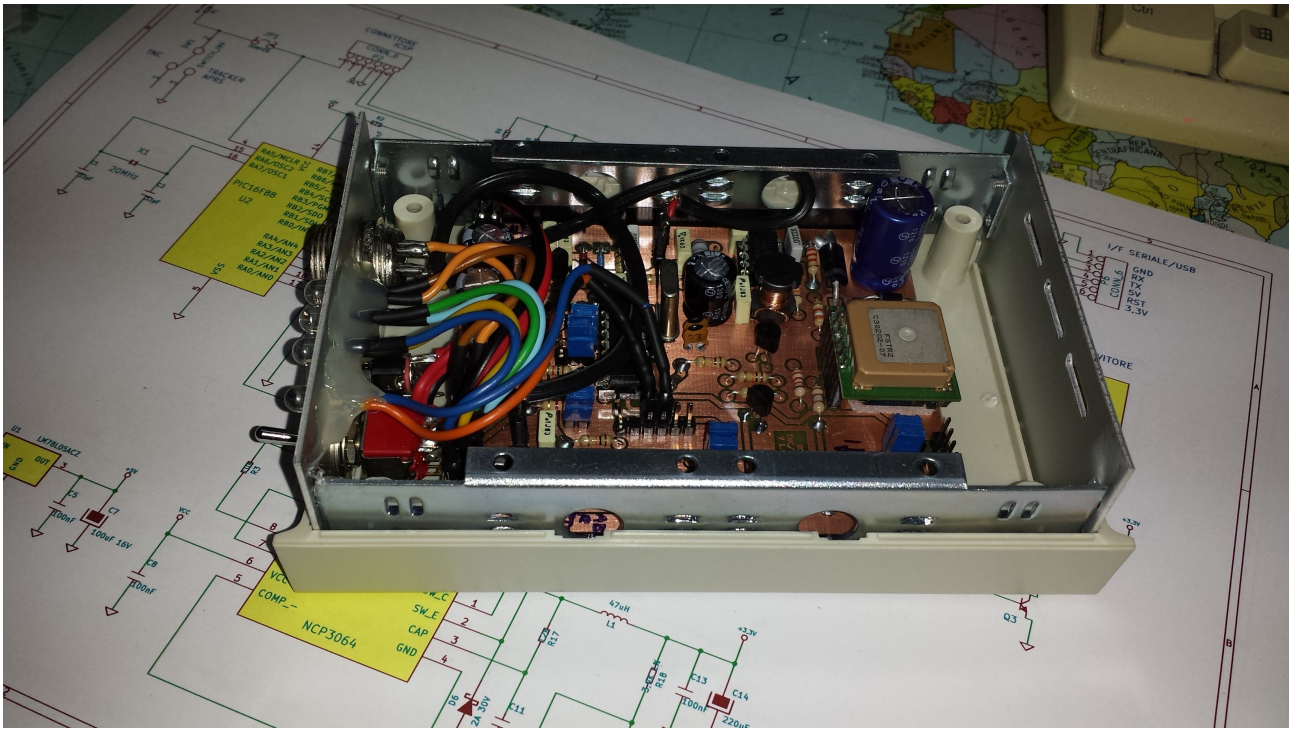
Ho progettato un circuito stampato con il programma Kicad, un programma multi piattaforma rilasciato liberamente con licenza GNU GPL v2, ed alcune librerie di componenti predisposte da con licenza Creative Commons (by-sa) da Walter Lain IW3IDN, (consiglio a tutti di dare un'occhiata al suo sito in quanto ha fatto un'opera direi monumentale).

Dai master su carta lucida ho ricavato una basetta doppia faccia delle dimensioni adatte ad una scatola RETEX in plastica che aveva disponibile il negozio di componenti cittadino.

La serigrafia dello stampato è la seguente:



Qui sotto potete vedere il tutto assemblato nel contenitore.



Per chi è interessato i file completi, in formato Kicad, sono disponibili tramite semplice richiesta all'email della Sezione ARI di Lucca ad info@arilucca.it

Configurazione:

Deve essere eseguita collegando un'interfaccia TTL/USB o TTL/RS232 in base al PC di cui uno dispone, ed utilizzando un programma ci comunicazione come CuteCom o Hyperterminal configurati con i seguenti paramteri: 9600 bps 8/N/1.

Tratterò la sola configurazione come tracker GPS, se si vuole utilizzare il dispositivo come TNC sulla linkografia troverete tutta la documentazione.

Innanzitutto va impostato il proprio nominativo con il seguente comando

- mycall ix9xxx-9 [invio] (sostituire ix9xxx con il proprio call e farlo seguire dal suffisso -9)

Impostare l'indirizzo di destinazione dei pacchetti UI con il comando UNPROTO

- unproto GPSC30 v wide1-1 v wide2-2 [invio]

Impostare il periodo di ogni quanto viene trasmesso il beacon, per le stazioni portatili generalmente ogni 1 o 2 minuti.

- beacon every 1 [invio]

Impostare quale stringa del GPS utilizzare per ottenere i dati da trasmettere

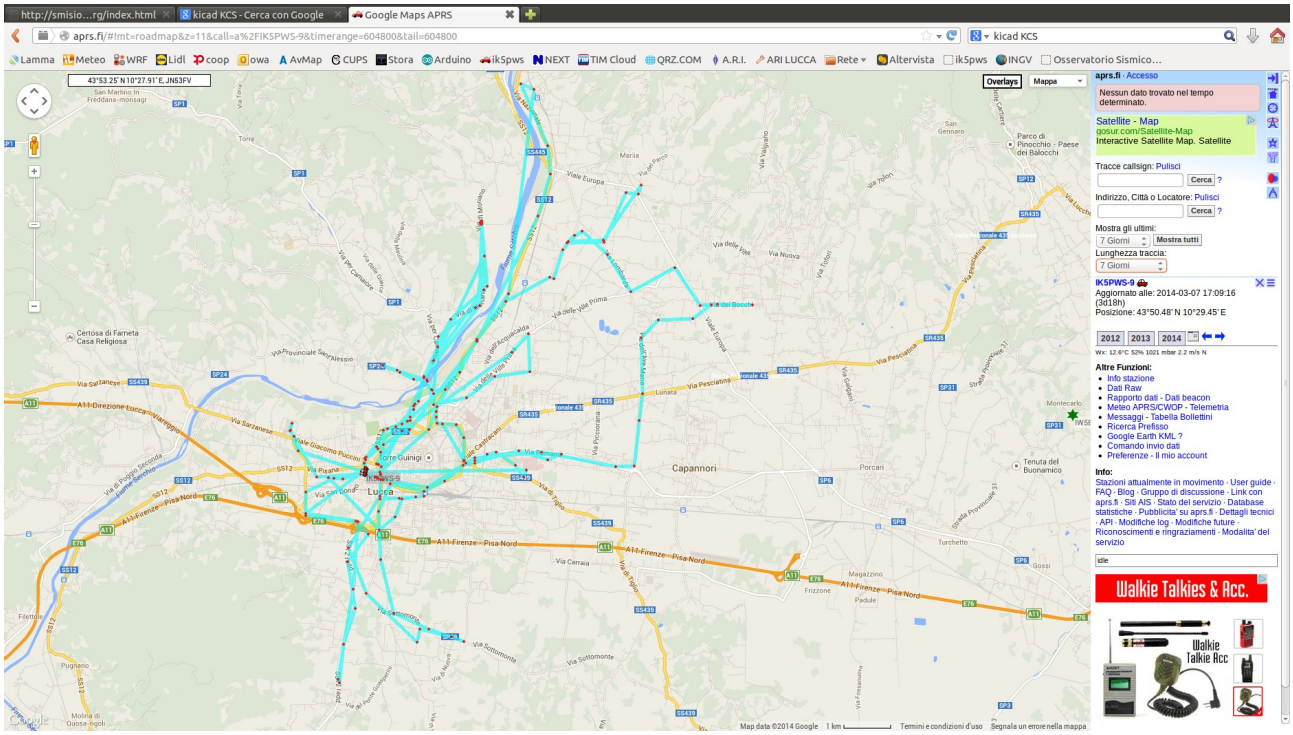
- gps $GPRMC [invio]

Salvare nella memoria non volatile del PIC i parametri appena inseriti

- perm

Tracciamento della stazione:

C'è poco da dire, esiste il sito *apsr.fi* di uso intuitivo, qui sotto potete vedere il risultato del tracciamento della mia stazione mobile installata sulla mia auto di servizio.



Linkografia:

1. http://www.enide.net/webcms/index.php?page=wb8wga-tnc
2. http://www.fantino.it/TNC.htm traduzione in italiano e versione di **IZ1DNJ**
3. http://www.pianetaradio.it/progetti/minitnc.htm versione di IZ8EWD
4. http://www.kicad-pcb.org/display/KICAD/KiCad+EDA+Software+Suite sito ufficiale KiCad
5. http://smisioto.no-ip.org/elettronica/kicad/kicad.htm Librerie componenti di Walter Lain IW3IDN