Modifiche software e hardware per gestione radiosonda RS41

I2NOS, IZ2FLY 04/04/2022

(Software base fatto da SQ5RWU, OK1TE, DF8OE… e altri)

Trasmissione APRS

Sono state effettuate significative modifiche al software per migliorare la qualità del segnale trasmesso.

Sensore di temperatura

Per poter utilizzare la sensoristica è stato necessario apportare modifiche al software.
Una parte delle modifiche è stata documentata.
Da quanto abbiamo potuto rilevare il sensore principale di temperatura della sonda è costituito da una sottile resistenza al platino connessa tra il pin 2 e il pin 13 del boom (vedi foto più avanti).
Questa resistenza è utilizzata per variare la frequenza di un oscillatore. Quindi per procedere con la misurazione della temperatura è necessario misurare la frequenza, da questa risalire alla resistenza e da questa dedurre la temperatura. Il processo è abbastanza tortuoso e ci costringe a parecchie approssimazioni e calibrazioni.

1) L'analisi empirica di una serie di sonde ci ha permesso di comprendere che ogni boom ha un valore resistivo diverso, per cui se scambiamo i boom tra due sonde (con il software originale) avremo probabilmente letture sballate.
2) Gli effetti capacitivi dei boom e dei circuiti alterano la frequenza dell'oscillatore per cui modificando le caratteristiche della sonda (ad esempio aggiungendo cavi, cambiando boom, inscatolandola, etc.) è necessario aggiustare le variabili utilizzate per i calcoli.
3) Le due conversioni temperatura>resistenza e resistenza>frequenza non sono lineari per cui è necessario intervenire sui calcoli per cercare di seguire il singolo andamento.
4) La frequenza dell'oscillatore è anche influenzata dalla temperatura degli altri componenti (in una escursione tra -52 e +150 l'effetto dei condensatori, integrati etc. si fa sentire)
5) Il processore installato sulla sonda ha una capacità elaborativa ridotta anche a causa di una frequenza di clock di appena 8mhz. Questo riduce la precisione dei convertitori utilizzati per misurare la frequenza.
6) Lo stesso oscillatore viene utilizzato, commutando elettronicamente l'input, per misurare l'umidità secondaria.

Detto ciò, ho tentato di risolvere i vari punti lavorando un po' di fantasia e procedendo, ove possibile, con aggiustamenti automatici. Rimane però la necessità iniziale di sapere come si comporta il singolo sensore (sensore che in caso di rottura del filo originale ho sostituito con un PT1000).

Per **questo è necessario misurare il più precisamente possibile il valore della resistenza a 0 gradi** (appoggiando il sensore della sonda sul ghiaccio che si sta sciogliendo; a questa temperatura può tranquillamente variare da 950ohm a 1050ohm). Per aumentare la precisione sarebbe utile riuscire ad apprezzare anche il decimo di ohm.
Ovviamente può risultare difficile, specialmente se non si dispone di un apposito connettore

effettuare questa misura**, in questo caso si può procedere più empiricamente inserendo il valore iniziale 1000 e poi procedere per successive approssimazioni.**

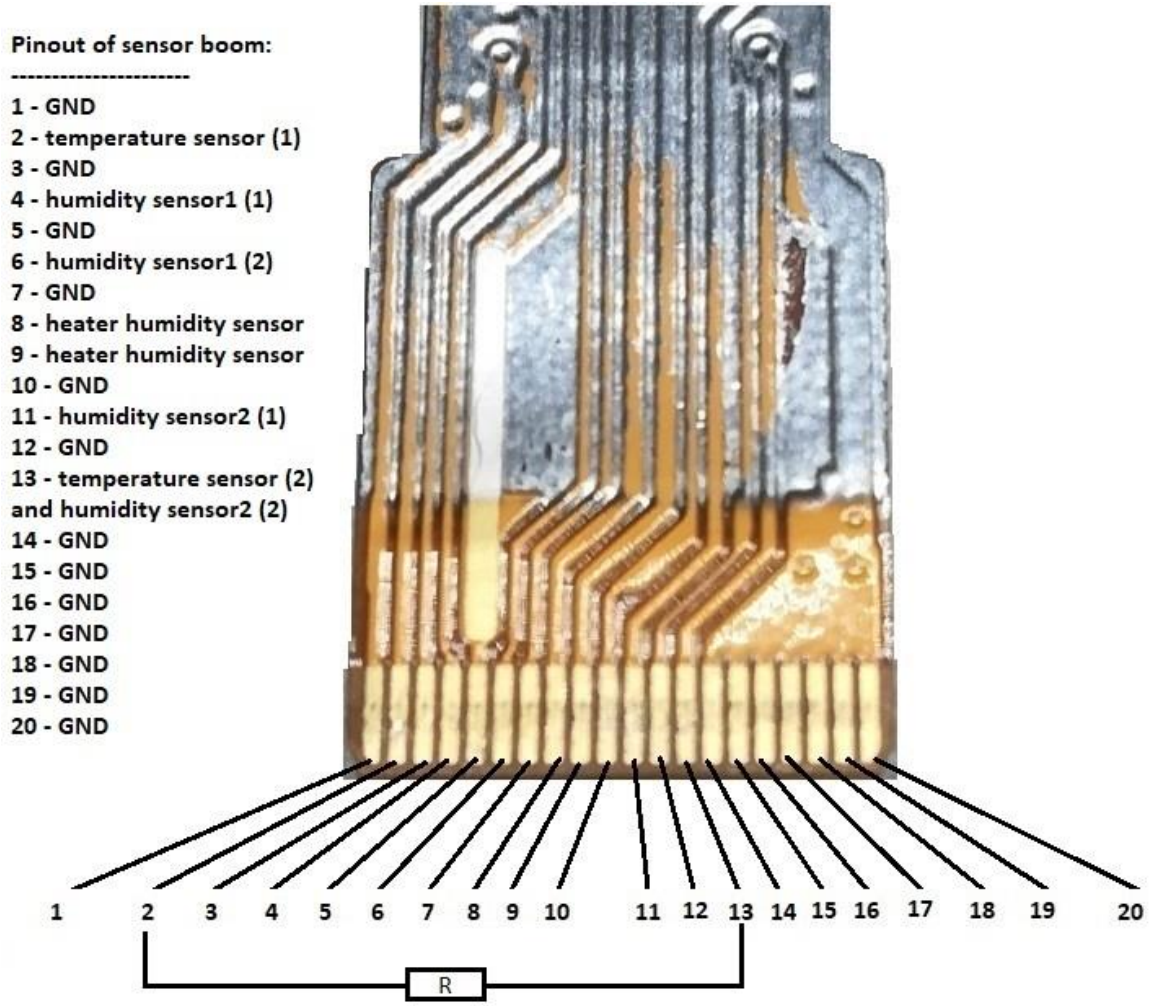
- La misura corretta della temperatura si ha dopo alcune trasmissioni, è meglio, durante la calibrazione, impostare trasmissioni più brevi solo in formato Packet.

Il valore della misurazione va inserito nel programma nel file **boomsens.h**
#define BOOM_SENSOR_TEMP_R0 1002.357 // boom sensor temperature resistor 0°C

Una volta caricato il programma si può valutarne la precisione appoggiando il boom della sonda nel ghiaccio per verificare che dia la corretta lettura di 0 gradi.

Per una ulteriore verifica di una decente calibrazione si può inserire la sonda nell'acqua bollente per verificare una lettura di 100° (ovviamente corretta in caso di discreta altitudine). Nel caso questa seconda verifica risultasse vistosamente errata, bisogna procedere con una calibrazione più fine rilevando i valori di altri componenti. (fase che descriverò in base ai feedback su un numero maggiore di sonde).

(Per vedere se la temperatura che si sta misurando è corretta potrebbe essere utile usare contemporaneamente due sonde, originale e modificata).

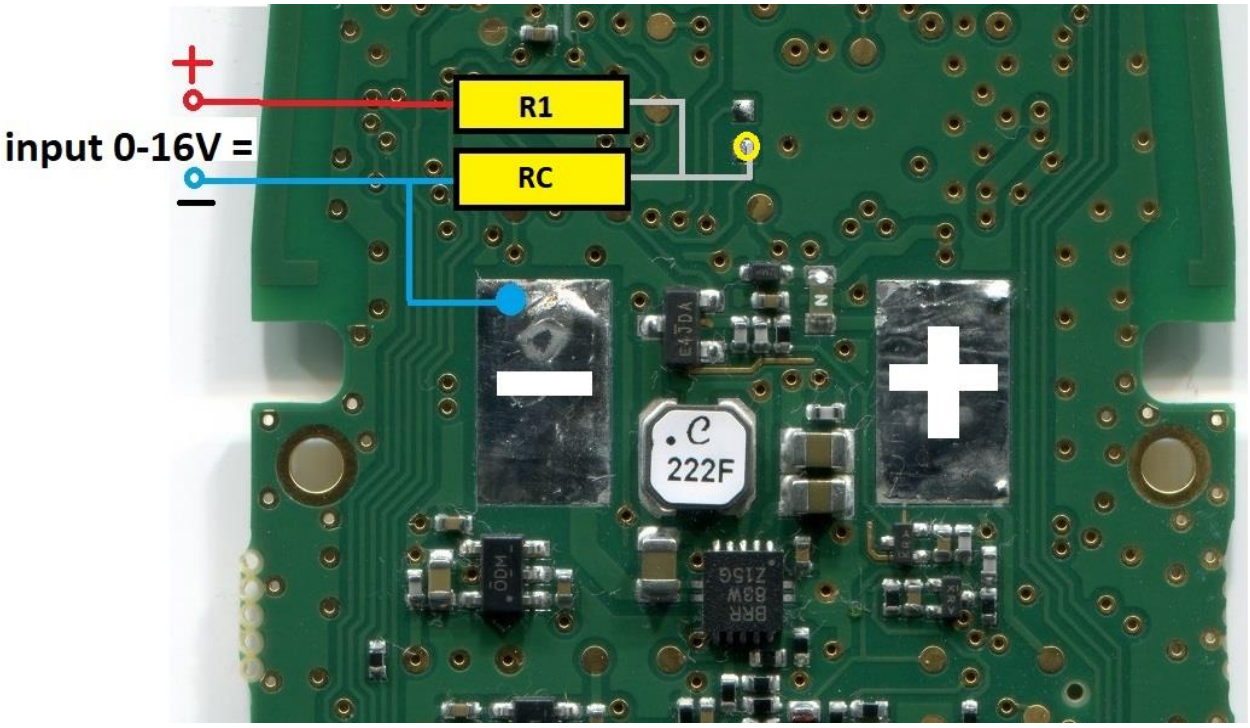


Per fare queste calibrazioni è utile che il ciclo di trasmissione a circa 10 secondi in modo di disporre di letture rapide

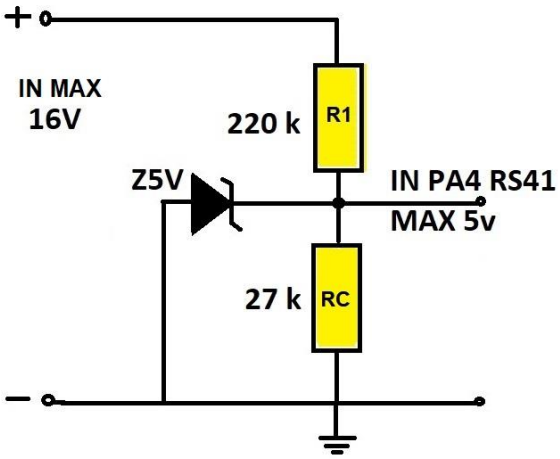
Aggiusteremo queste note in base alle vostre difficoltà e suggerimenti.

Misura di tensione esterna

Avevamo la necessità di installare la sonda in condizioni di alimentazione da batteria ricaricata da

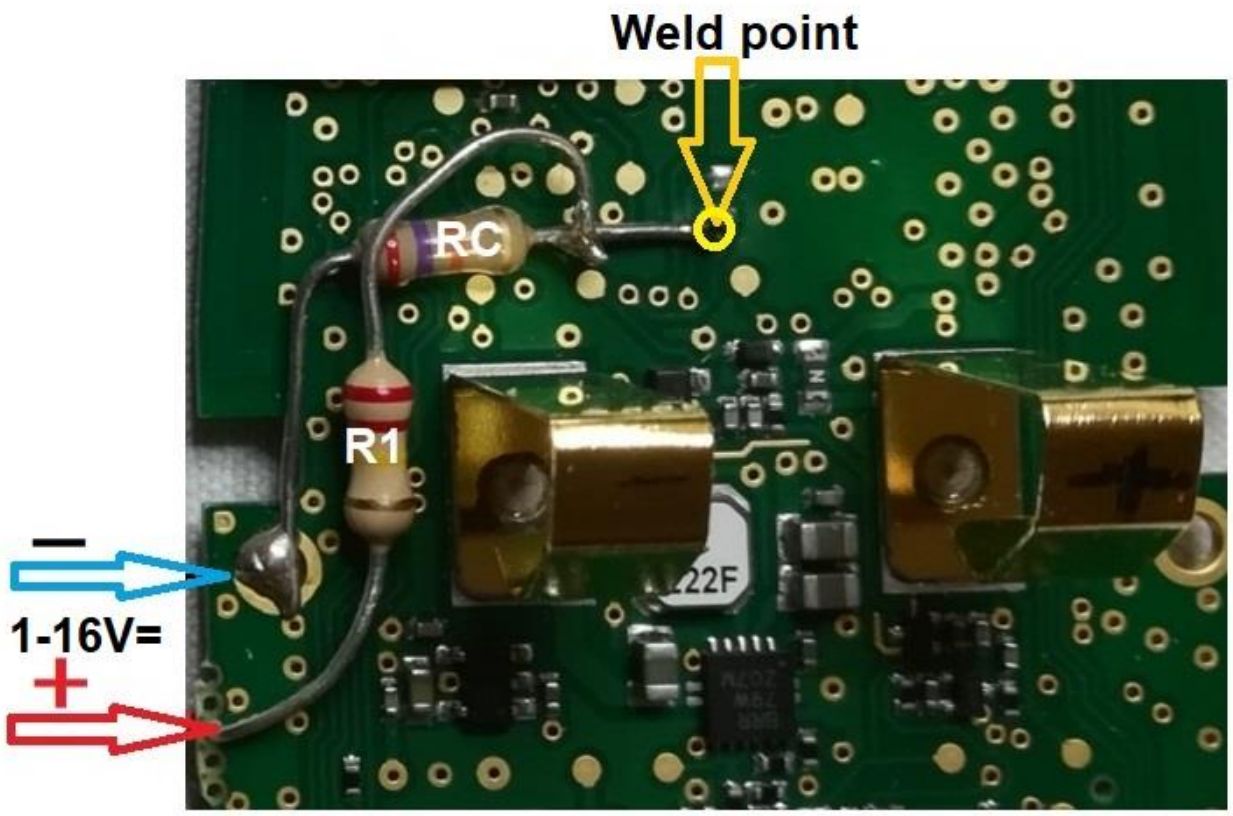


pannello solare. Per questo motivo ci interessava che venisse anche trasmesso anche il livello di tensione della batteria. Purtroppo, sui connettori non sono disponibili pin non utilizzati, per cui ci siamo indirizzati al riutilizzo di un pin (PA4) disponibile su una piazzola prevista per una possibile espansione.



Ci sono altri punti sulla scheda per la connessione comune alle due resistenze, questa scelta pare quella più facilmente utilizzabile essendo già stagnata. Effettuate le saldature, per una maggiore tenuta meccanica, è utile fissare ulteriormente le resistenze al pcb con colla caldo o altre soluzioni più adatte.

(Per una ulteriore sicurezza potrebbe essere utile inserire un diodo zener da 5V)



Le resistenze possono essere di valori diversi rispettando le proporzioni del partitore e vanno inserite nel file **main.h**

```
// EXTERNAL VOLTAGE MEASURE
#define ADC_BAT_R1 220000 // Voltage divider resistor
#define ADC_BAT_RC  27000 // Voltage divider common resistor
```

NOTE per APRS

(Nel file aprs.cpp si può modificare il simbolo che viene visualizzato sulla mappa in aprs.fi agendo su:
O/A = Pallone sonda, **[/A =** uomo a piedi, **_/A =** stazione meteo, **>/A =** automobile, **r/A** = traliccio
ROW 67

…sprintf(packet_buffer,("!%02d%02d.%02u%c/%03d%02u.%02u%c**O**/A=%06ld/P%d/S%d/T%d/V%d"

Purtroppo sono molto poche le stazioni che ricevono segnali packet UHF e messi in aprs.info.
In zona 2 ne ho trovata solo una, sono riuscito a fare dei test andando in altura.

È utile saldare un connettore SMA al posto dell'antenna per un'antenna adeguata.

Breve guida al file config.h

Il file **config.h** può essere personalizzato modificando testo e parametri, ecco alcuni esempi.

#define RTTY_CALLSIGN "IZ2FLY" // **mettere il proprio nominativo.**

#define APRS_CALLSIGN "IZ2FLY" // **mettere il proprio nominativo.**

#define MORSE_PREFIX "VVV TEST DE IZ2FLY/B" // **mettere il proprio nominativo, durante la trasmissione verrà aggiunto automaticamente il locatore preso dai dati satellitari.**

#define APRS_FREQUENCY 432.500f // **Lasciare questa per entrare nel circuito APRS**

#define TX_DELAY 9750 // **tempo di trasmissione approssimativo del tono CW dopo il testo**

#define RTTY_FREQUENCY 432.450f // **mettere la frequenza desiderata che è la stessa per CW, di default attivo solo CW** e Packet 432.500.
#define SEND_RTTY 0 **// #define ……. 0 = stato disattivato, #define ……. 1 = stato attivato**
Questo vale per tutti gli altri parametri che sono ben commentati.

File main.h
Per abilitare debug frequenze **nel file main.h** togliere **//**
//#define BOOM_DEBUG

La parte software RTTY non è ancora completamente ottimizzata e funzionale.

- Prossimamente si cercherà di studiare la possibilità di utilizzare il sensore di umidità, e di pressione per le sonde **RS41-SGP**
- Attendiamo suggerimenti dati dai test effettuati.

I2NOS, IZ2FLY