Modificaciones de software y hardware para la gestión de sondas RS41 I2NOS, IZ2FLY 04/04/2022

(Software básico hecho por SQ5RWU, OK1TE, DF8OE… y otros)

Transmisión APRS

Se han realizado cambios significativos en el software para mejorar la calidad de la señal transmitida.

Sensor de temperatura

Para utilizar los sensores fue necesario realizar cambios en el software.
Algunos de los cambios han sido documentados.
Por lo que hemos podido detectar, el sensor de temperatura principal de la sonda consiste en una delgada resistencia de platino conectada entre el pin 2 y el pin 13 del boom (ver foto más allá). Esta resistencia se utiliza para variar la frecuencia de un oscilador. Por lo tanto, para proceder con la medición de la temperatura, es necesario medir la frecuencia, de esta volver a la resistencia y de esta deducir la temperatura. El proceso es bastante tortuoso y nos obligó a realizar varias aproximaciones y calibraciones.

1) El análisis empírico de una serie de sondas nos ha permitido entender que cada sensor tiene un valor resistivo diferente, por lo que si intercambiamos los sensores entre dos sondas (con el software original) vamos a tener lecturas incorrectas.

2) Los efectos capacitivos de los brazos y circuitos alteran la frecuencia del oscilador por lo que modificando las características de la sonda (por ejemplo añadiendo cables, cambiando brazos, encajonándola, etc.) es necesario ajustar las variables utilizadas para los cálculos .

3) Las dos conversiones temperatura>resistencia y resistencia>frecuencia no son proporcionales, por lo que es necesario intervenir en los cálculos para tratar de seguir la tendencia única.

4) La frecuencia del oscilador también está influenciada por la temperatura de los otros componentes (en una excursión entre -52 y +150 se siente el efecto de los condensadores, integrados, etc.)

5) El procesador instalado en la sonda tiene una capacidad de procesamiento reducida también debido a una frecuencia de reloj de solo 8 MHz. Esto reduce la precisión de los convertidores utilizados para medir la frecuencia.

6) Se utiliza el mismo oscilador, conmutando electrónicamente la entrada, para medir la humedad secundaria.

Dicho esto, se intentó resolver el problema de los distintos puntos trabajando un poco con la imaginación y procediendo, donde fue posible, con ajustes automáticos. Sin embargo, queda la necesidad inicial de saber cómo se comporta el sensor (sensor que en caso de romper el hilo original ha sido remplazado por un PT1000).

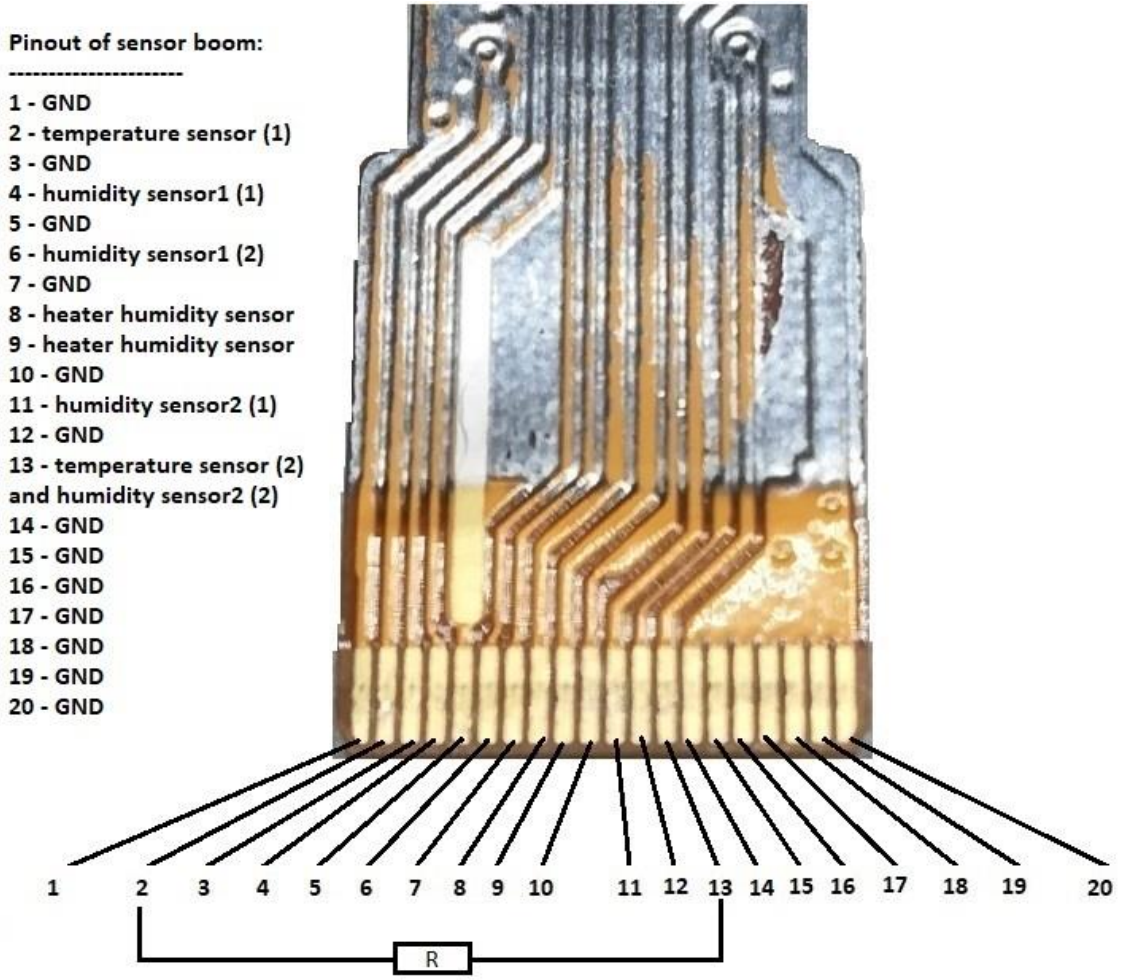
Por eso es necesario medir el valor de la resistencia a 0 grados con la mayor precisión posible colocando el sensor de la sonda sobre el hielo que se está derritiendo; a esta temperatura puede variar fácilmente de 950 Ω a 1050 Ω. Para aumentar la precisión sería útil poder apreciar incluso la décima de Ω.

Evidentemente puede resultar complicado, sobre todo si no se dispone de un conector especial, en este caso se puede proceder de forma más empírica introduciendo el valor inicial 1000 y luego proceder por aproximaciones sucesivas.

- La medición correcta de la temperatura ocurre después de algunas transmisiones, es mejor, durante la calibración, configurar transmisiones más cortas solo en formato Packet. El valor de medición debe ingresarse en el programa en el archivo **boomsens.h**
#define BOOM_SENSOR_TEMP_R0 1002.357 // boom sensor temperature resistor 0°C

Una vez que se ha cargado el programa, se puede evaluar su precisión colocando el brazo de la sonda en el hielo para verificar que la lectura correcta sea 0 grados. Para una verificación adicional de una calibración decente, se puede poner la sonda en agua hirviendo para verificar una lectura de 100 ° (obviamente correcta en el caso de una altitud razonable). Si esta segunda verificación resulta notoriamente incorrecta, es necesario proceder con una calibración más fina detectando los valores de otros componentes. (fase que se describirá en base al feedback con un mayor número de sondas).

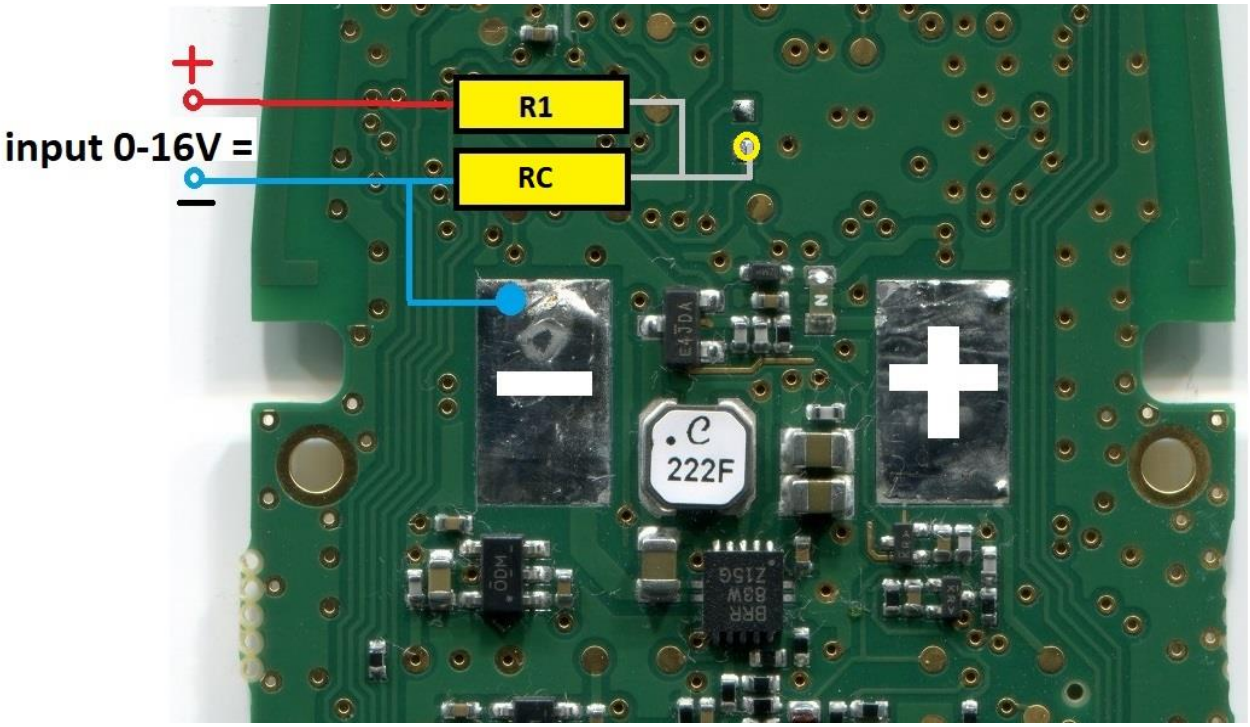
Para ver si la temperatura que se está midiendo es correcta, puede ser útil usar dos sondas, original y modificada al mismo tiempo.



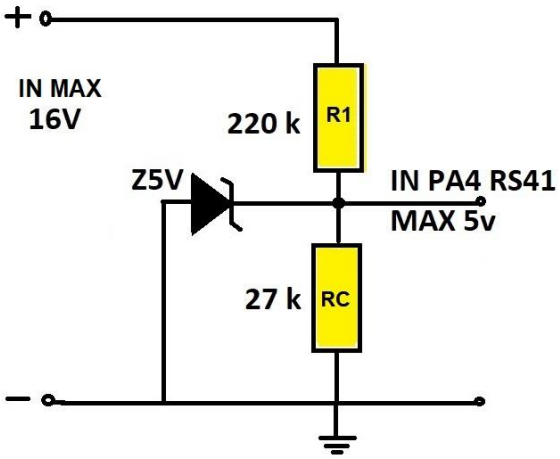
Para realizar estas calibraciones es útil que el ciclo de transmisión sea de unos 10 segundos para poder tener lecturas rápidas

Ajustaremos estas notas en función de sus dificultades y sugerencias.

Medida de tensión externa



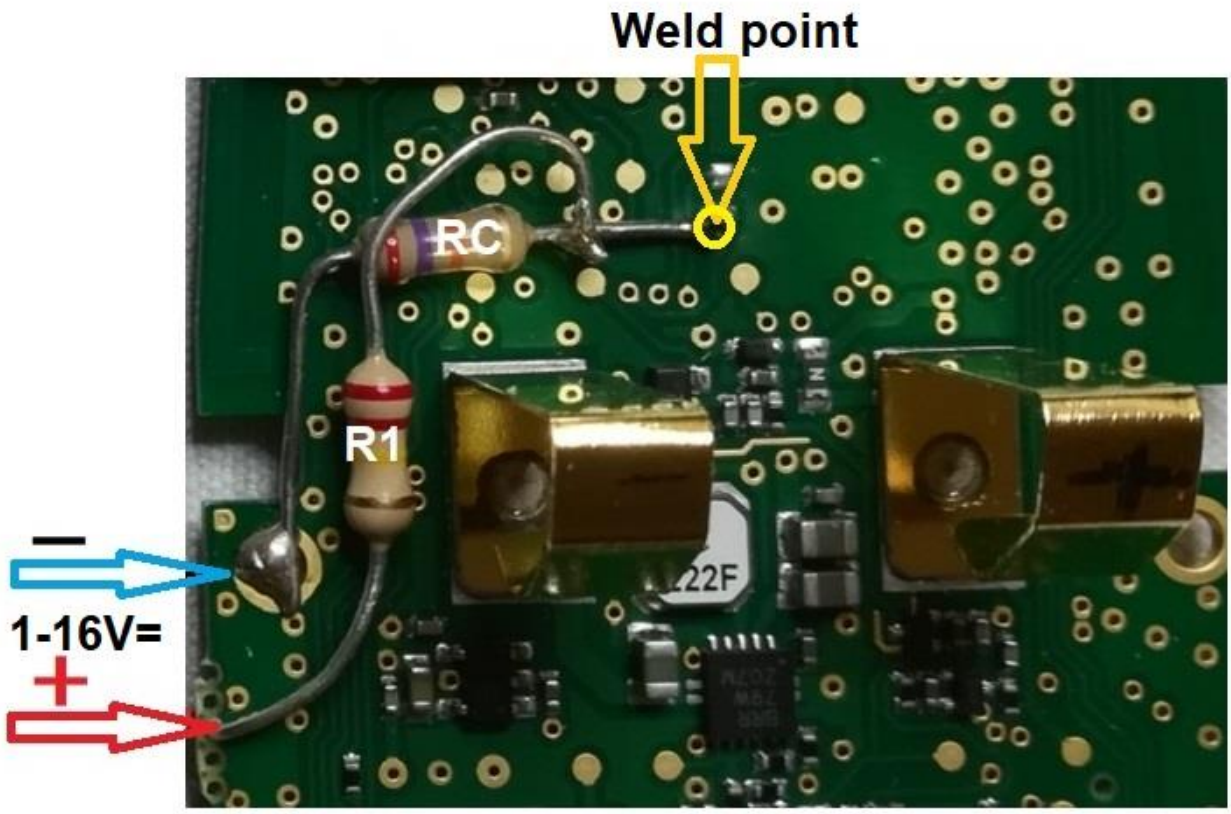
Necesitábamos instalar la sonda en condiciones de alimentación desde una batería recargada desde un panel solar. Por eso nos interesaba que también se transmitiera el nivel de voltaje de la batería. Desafortunadamente, los pines no utilizados no están disponibles en el conectore, por lo que hemos abordado la reutilización de un pin (PA4) disponible en un punto de la placa de circuito impreso probablemente por una posible expansión.



Hay otros puntos de la placa de circuito impreso para la conexión común a las dos resistencias, esta opción parece ser la más fácil.

Después de la soldadura, para un mejor sellado mecánico, es útil fijar más las resistencias a la placa de circuito impreso con cola caliente u otras soluciones más adecuadas.

(Para mayor seguridad puede ser útil insertar un diodo zener de 5V)



Las resistencias pueden ser de diferentes valores respetando las proporciones del divisor y deben ser insertadas en el archivo **main.h**

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

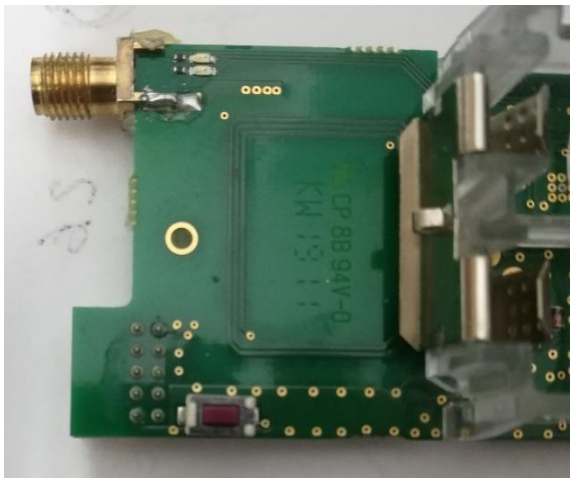
NOTAS para APRS

En el archivo **aprs.cpp** se puede cambiar el símbolo que se muestra en el mapa en aprs.fi:
O/A = globo sonda, **[/A** = hombre a pie, _/A = estación meteorológica, **>/A** = coche, **r/A** = pilón

ROW 67

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

Desafortunadamente, muy pocas estaciones reciben señales de packet en UHF y las ponen en aprs.info. Es útil soldar un conector SMA en lugar de la antena para una antena adecuada.



Archivo config.h

El archivo config.h puede personalizar cambiando el texto y los parámetros, aquí hay algunos ejemplos.

#define RTTY_CALLSIGN "IZ2FLY" // tu call

#define APRS_CALLSIGN "IZ2FLY" // tu call

#define MORSE_PREFIX "VVV TEST DE IZ2FLY/B" // inserta tu call.
La posición de la sonda (WW Locator) se ingresa automáticamente con los datos recibidos de los satélites.

s#define APRS_FREQUENCY 432.500f // Deja esto para ingresar al circuito APRS

#define TX_DELAY 9750 // tiempo de transmisión del tono CW después del texto

#define RTTY_FREQUENCY 432.450f // frecuencia deseada que es la misma para CW,

por defecto activo solo CW y Packet 432.500.

#define SEND_RTTY 0 // #define ……. **0 = OFF**, #define ……. **1 = ON**

Esto se aplica a todos los demás parámetros que están bien comentados.

Archivo main.h

Para habilitar las frequencias de debug en el archivo main.h, elimine **//**

// # define BOOM_DEBUG

La parte RTTY del software aún no está completamente optimizada y funcional.

- En un futuro próximo intentaremos estudiar la posibilidad de utilizar el sensor de humedad, y el sensor de presión para las sondas RS41-SGP

- Esperamos sus sugerencias después de sus pruebas.

I2NOS, IZ2FLY