Hardware and software modifications for RS41 radiosonde management

I2NOS, IZ2FLY **04/04/2022**

(The software source was made by SQ5RWU, OK1TE, DF8OE…and others)

APRS TRASMISSION

The software was substantially modified in order to increase the quality of the transmitted signal.

Thermal sensor

Modifications of the software were needed to make the sensors usable.
Part of the modifications has been documented.
As far as we could learn by analyzing the radiosonde, the main thermal sensor is constituted of a thin platinum resistor that is connected between the pin 2 and 13 of the boom (see the image below).
This resistor is used to vary the frequency of an oscillator. The recorded temperature is indeed derived from a frequency measurement, that gives a resistance value. The nature of the calculations made us use different approximations and calibrations.

1) The empiric analysis of some sondes allowed us to understand that each boom has a different resistive value. Thus, when two booms of different sondes are swapped (with the original software) we probably obtain values that are out of tune
2) The capacitive effects of the boom and circuits alters the oscillator frequency. Thus, when the characteristics of the sonde are modified (ex. Adding cables, changing the boom, putting it in a box etc.), it is necessary to re-set the variables used for the calculations.
3) The conversion of temperature>resistance and resistance>frequency are not linear. Thus, it is necessary to intervene in the calculations and try to follow each trend.
4) The oscillator frequency it is also influenced by the temperature of the other components (in a range of -52 to +150°C the capacitators affect the measure)
5) The chip processor present in the sonde has a reduces elaborative capacity due to a clock frequency of just 8MHz. This reduces the precision of the converters used to measure the frequency
6) The same oscillator is always used to measure the secondary humidity. This is done electronically switching the input.

We tried to fix the above-mentioned points proceeding with a little bit of imaginations and employing automatic adjustments whenever possible. Still, it is necessary to find out how the individual sensor works (in case of breaking this sensor was substituted with a PT1000).

For this reason, it is necessary to measure the value of the resistance at 0°C as precisely as possible. In fact, different resistance values are obtained from different sensors when put in a ice-water solution (the resistance ranges from 950ohm to 1050ohm). This measurement could result difficult, especially without a specific connector. In this case, an initial value (R0) of 1000 should be inserted in the firmware and then proceed with approximations changing R0 until the exact temperature value is obtained (using a original non-modified radiosonde).

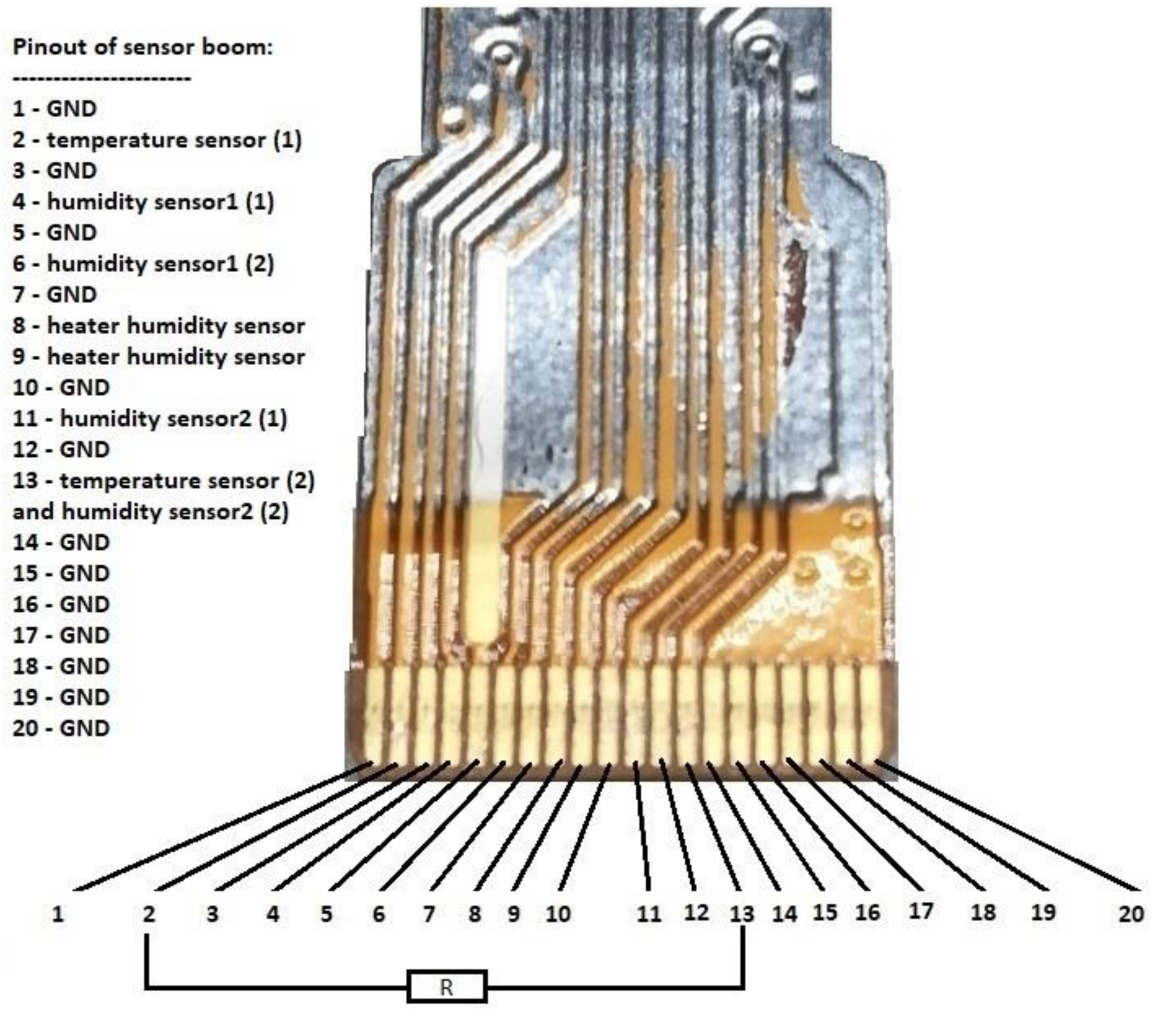
- The exact temperature measure is obtained after some transmissions. During the calibration, it is better to use only short transmission in Packet format

The value of the measurement needs to be inserted in the file **boomsens.h** of the program
#define BOOM_SENSOR_TEMP_R0 1002.357 // boom sensor temperature resistor 0°C

Once the program is uploaded, its precision can be evaluated placing the boom of the sonde in the water-ice solution (check that a temperature of 0°C is read).

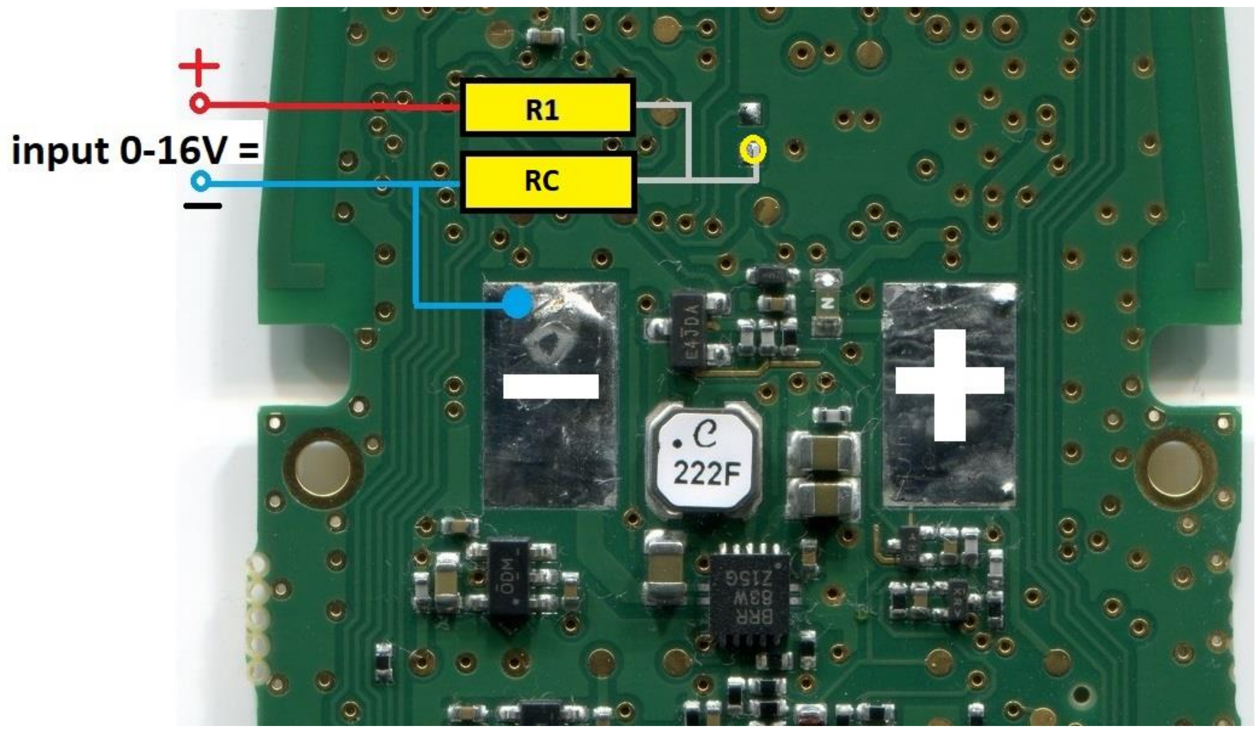
As confirmation of a good calibration the sonde can be placed in boiling water to verify a value of 100°C (considering an altitude at sea level). In case this measurement results conspicuously wrong, it is necessary to proceed with a finer calibration by detecting the values of other components (this will be described based on feedbacks).

(to see whether the measured temperature is correct or not, the use of two sondes, an original and a modified one, could be useful)

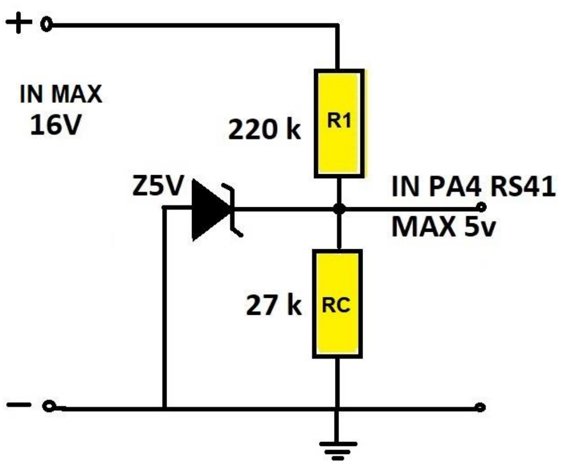

Pinout of sensor boom:
----
1 - GND
2 - temperature sensor (1)
3 - GND
4 - humidity sensor1 (1)
5 - GND
6 - humidity sensor1 (2)
7 - GND
8 - heater humidity sensor
9 - heater humidity sensor
10 - GND
11 - humidity sensor2 (1)
12 - GND
13 - temperature sensor (2) and humidity sensor2 (2)
14 - GND
15 - GND
16 - GND
17 - GND
18 - GND
19 - GND
20 - GND

To do these calibrations a 10 sec transmission cycle should be used to have faster readings.
We will adjust these notes based on your difficulties and suggestions.

External voltage measurement

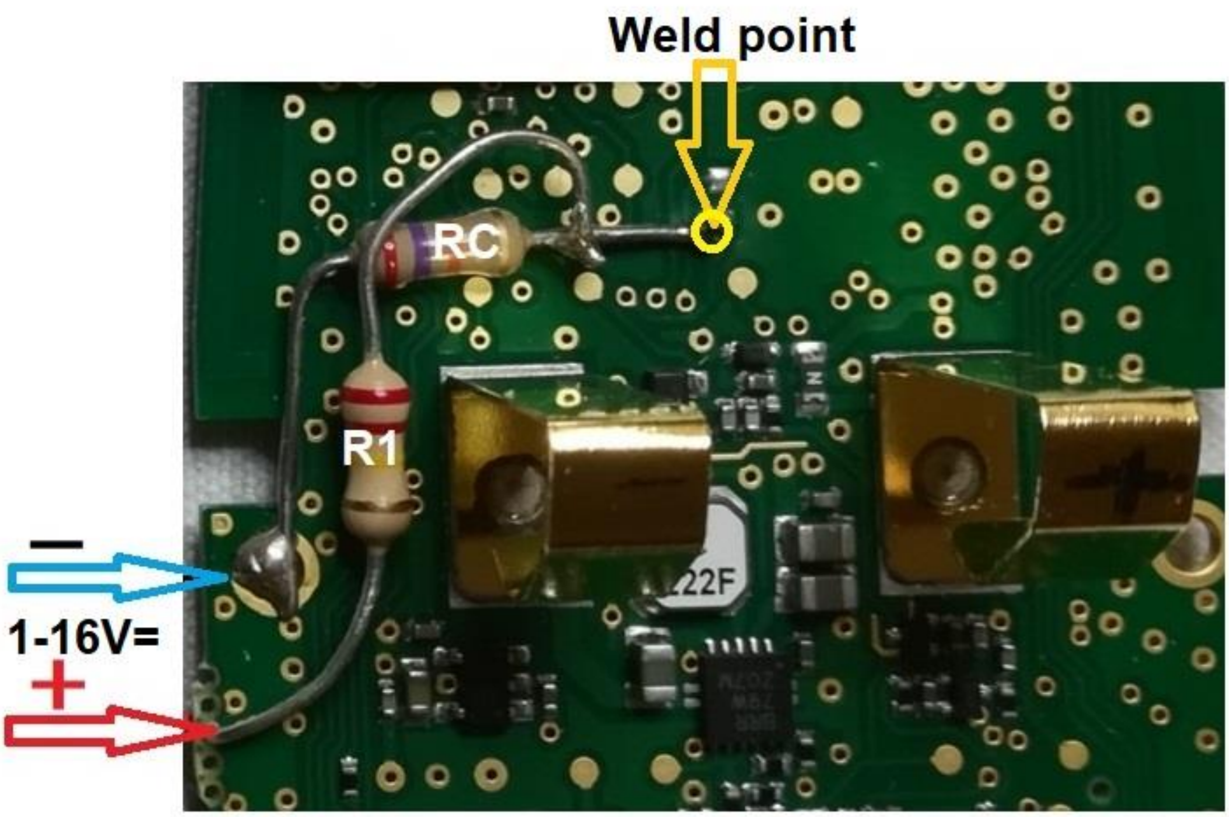


A battery alimentation by solar energy was our chosen method of recharge for the radiosonde. Thus, the energy level of the battery needs to be monitored through time. Unfortunately, no unutilized pins were available to achieve this task. So, we used a pin (PA4) located in a secondary plate designed for an eventual expansion.



It is good to mention that other spots are available in order to establish the connection of the two resistors. However, we used the aforementioned because it was already soldered with tin.
Once the welding is performed, it is useful to further fix the pcb resistors with hot glue or other suitable solutions.

(As an ulterior safety measurement, 5V Zener diode may be useful to install)



Resistance values can be different depending on the proportions of the divider and they have to be imported in the **main.c** file

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

NOTES for APRS

(The file **aprs.cpp** allows to modify the sign that is used on the map aprs.fi acting on:

O/A = Sonde balloon, **[/A** = walking man, **_/A** = weather station, **>/A** = car, **r/A** = tellis

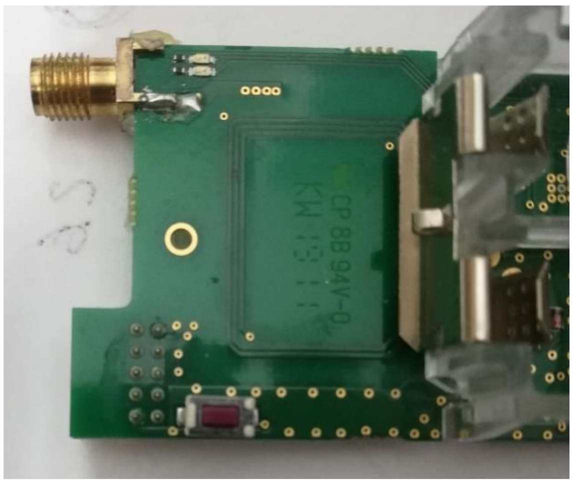
ROW 67

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

Unfortunately, a very few number of weather stations receive UHF packet signals in aprs.info.

I manage to find one located in the zone 2, where I did some test at higher altitudes.

It is useful to weld a SMA connector instead of an antenna for a more suited one.



Short guide to the config.h file

config.h can be customized by changing text and parameters; here, few examples are reported

#define RTTY_CALLSIGN "IZ2FLY" // **put your name.**

#define APRS_CALLSIGN "IZ2FLY" // **put your name.**

#define MORSE_PREFIX "VVV TEST DE IZ2FLY/B" // **put your name, the lessor will be automatically added during the transmission, taking the info directly from the satellite data.**

#define APRS_FREQUENCY 432.500f // **Do not modify to access the APRS circuit**

#define TX_DELAY 9750 // **approximate transmission time of the CW tone after the text**

#define RTTY_FREQUENCY 432.450f // **put the desired frequency that is the same for the CW, in the default settings only CW is active** e Packet 432.500.
#define SEND_RTTY 0 **// #define ………. 0 = not active state, #define ………. 1 = active state**
This is applicable to all the other commented parameters.

File main.h

Remove **//** to enable the frequency debug in the **main.h** file
//#define BOOM_DEBUG

The RTTY software is still not completely optimized and functional to duty.

- In the future the application of humidity and pressure sensors on **RS41-SGP** radiosonde will be investigated
- We look forward to receiving your feedbacks with suggestions and comments.

I2NOS, IZ2FLY