

Wireless Temperature Sensors

Description

Battery powered temperature sensors are broadcasting in an adjustable interval their temperatures. A Raspberry PI+ is constantly checking for broadcasts, logs them to files and makes the data available to users in the local network.

Temperature Sensor

The sensor modul is controlled by an ATtiny84 running on 1Mhz, using a DS18B20 as the digital temperature sensor and a RFM69W modul for wireless communication. The sensor modul also has a push button to be able to set the temperature modul into receive mode for parameter setting and two Led's for status information. The RFM69W radio modul was used because its packet mode combined with sync detection reduces the work that has to be done by the microcontroller.

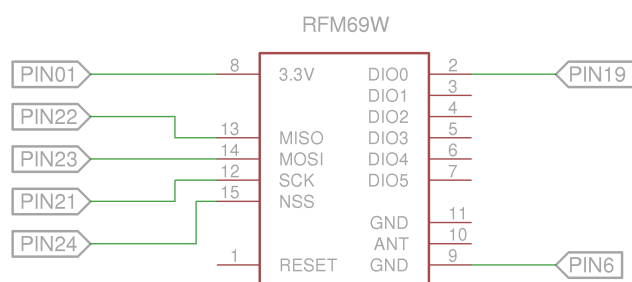
Data Receiver

The Raspberry PI+ was chosen because of its low power consumption (24h) and the possibility to directly connect a RFM69W radio modul to the GPIO pins.

Software

To be able to communicate with the digital sensor DS18B20 the Dallas 1 wire protocol is used. The RFM69W radio modul is communicating over SPI with either the microcontroller or the Raspberry PI+. To keep everything small and portable the libraries were written in C and are the same for the Atmel microcontroller as well as for the Raspberry PI+. The whole software for the microcontroller (1 Wire, SPI and RFM69W) is about 4K Bytes, so there is still 50% free program memory available on the ATtiny84.

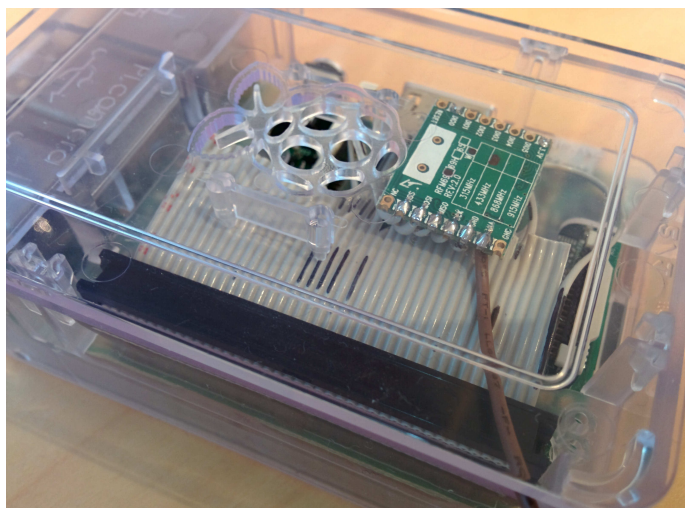
Connecting RFM69W radio Modul to a Raspberry PI+



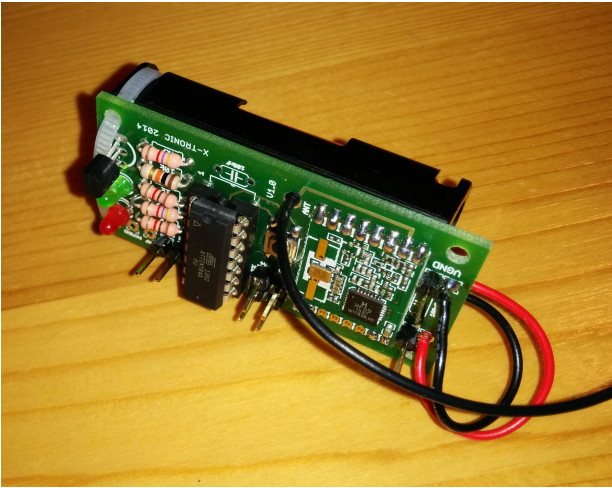
Pin numbers are connector pins on Raspberry PI

On DIO0 of the RFM69W radio modul we get a high pulse if the modul has detected a valid packet and the payload is ready to read in the FIFO buffer. So all we have to do is register an interrupt on Pin19 of the Raspberry PI+ and read the FIFO of the modul via SPI when the interrupt fires. So all the work – Sync word detection, validation via CRC - is done by the modul and the Raspberry PI+ only needs to log them. A sepearte simple PHP script is making the data available to the users.

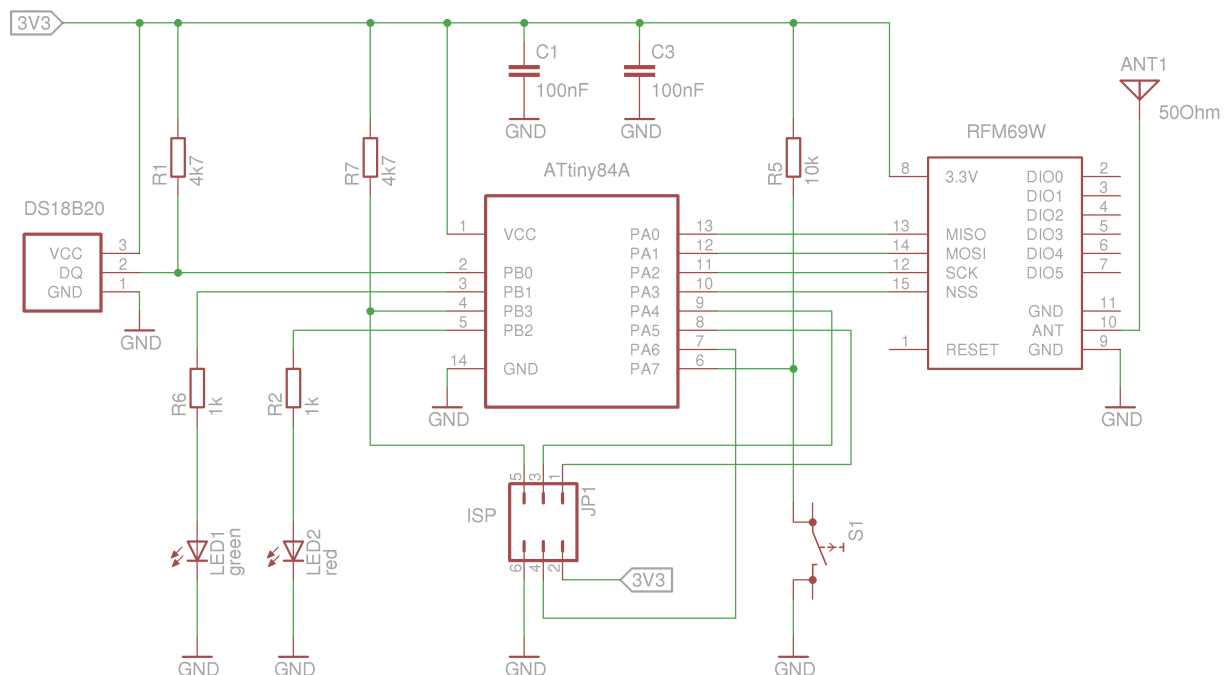
To keep it simple, I just used an old IDE cable connector which fits perfectly on the Raspberry PI+ GPIO connector, and soldered the cable wires directly to the modul. Due to the small size and weight of the modul no further work is needed, just close the original case available for the Raspberry PI+.



Temperature Sensor Modul



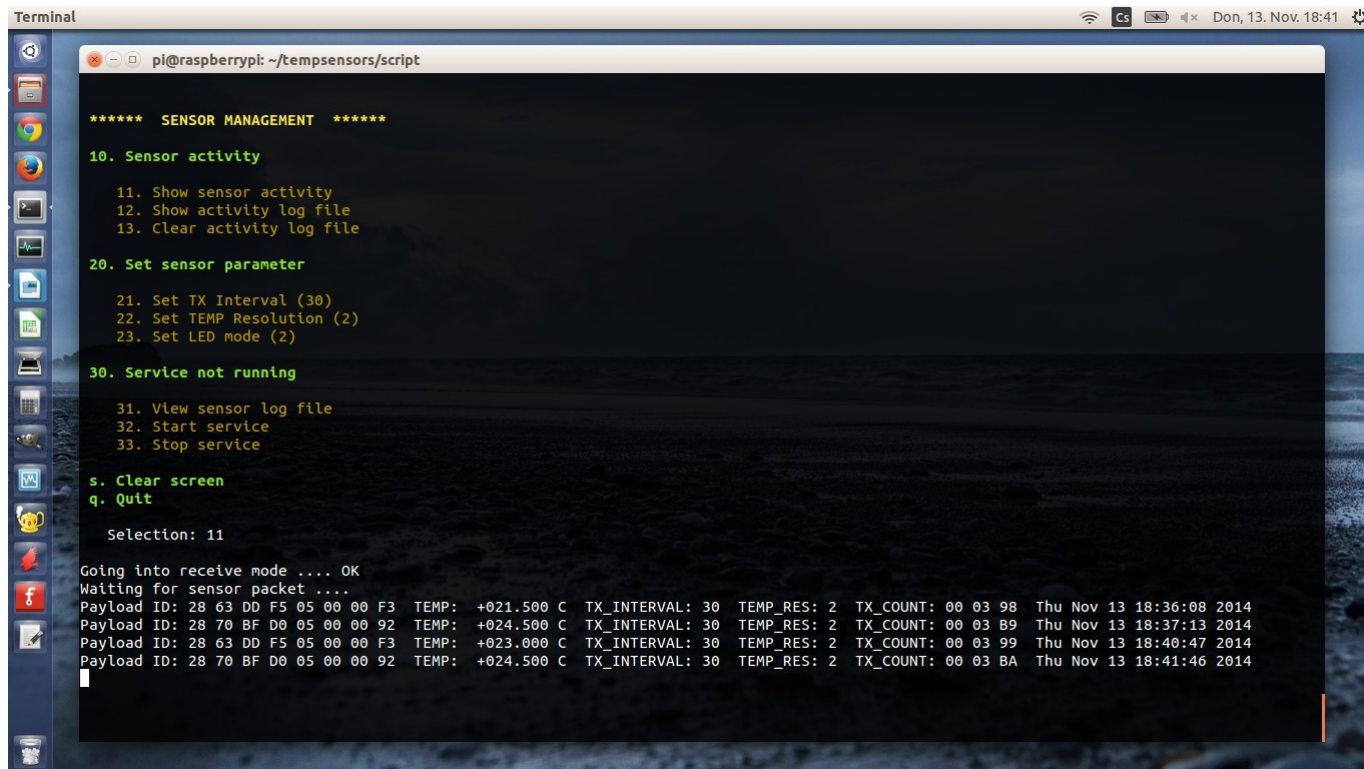
The modul is powered by two AAA batteries and therefore the pcb doesn't need to be that tiny, so we can use the ATtiny84 in the DIP version (inc. DIL socket) instead of the SMD version. No other SMD parts, except the push button, are needed which makes it also a lot easier for beginners. Furthermore the microcontroller can be programmed in circuit with a 14pin chip clip. The pins seen on this picture are not needed, but were used during testing. Of course they can also be used for connecting an ISP programmer. Also the decoupling capacitors are not mounted here, just for stress testing because this is a developer pcb.



The modul is broadcasting on 433MHz, with 4800 Bit/s and 0 dbm transmission power setting. This guaranties low power consumption and a good range. The broadcast interval is adjustable (usually between 1 and 5 minutes) as well as the transmission power, the sensor resolution and the frequency (e.g. my developer versions are running with 433,750 MHz). Each broadcast message consists of the 8 byte long sensor ID (DS18B20), the temperature value in a human readable string (+22,125 C), the sensor resolution, the tx power level and a 3 byte long tx counter. The tx counter can be used by the receiver to check if any broadcasts were missed or to check the time the microcontroller was reset (or powered up). If the push button is pressed a broadcast is triggered. This can be used for testing or to reset the interval in case we have a broadcast collision with other sensors. Never had this because they are usually not powered up (battery inserted) at the same time. If the button is pressed for 3 seconds, the modul is sending a broadcast (inRXMode) and waits 10 seconds for valid data. The receiver (Raspberry PI+) has now the possibility to alter the sensor parameters.

Raspberry PI+ as data receiver

A interrupt controlled server is running on the Raspberry PI+. It logs the temperature data along with the timestamp and the tx counter to a file with the sensor id as the filename. Also found errors like missed broadcasts and wrong temperatures are logged to a separate error file. Temperatures are considered wrong if they belong to DS18B20 “reserved” ones, especially -127 C (communication error) and +85 C (power up) or are far out of range. The parameters of the temperature moduls can be set via a interactive bash script and the temperature data are visualized with the help of an Apache webserver, together with php and javascript.



```
Terminal
pi@raspberrypi: ~/tempsensors/script

***** SENSOR MANAGEMENT *****

10. Sensor activity
    11. Show sensor activity
    12. Show activity log file
    13. Clear activity log file

20. Set sensor parameter
    21. Set TX Interval (30)
    22. Set TEMP Resolution (2)
    23. Set LED mode (2)

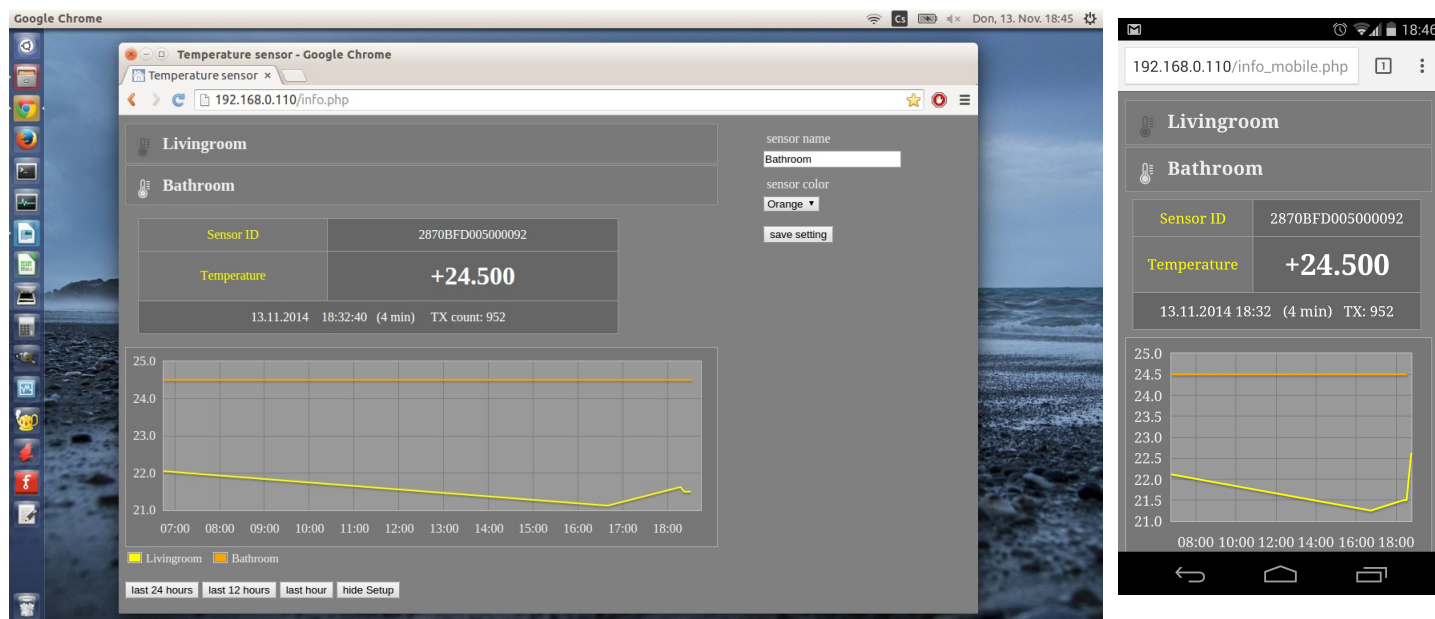
30. Service not running
    31. View sensor log file
    32. Start service
    33. Stop service

s. Clear screen
q. Quit

Selection: 11

Going into receive mode .... OK
Waiting for sensor packet ....
Payload ID: 28 63 DD F5 05 00 00 F3  TEMP: +021.500 C  TX_INTERVAL: 30  TEMP_RES: 2  TX_COUNT: 00 03 98  Thu Nov 13 18:36:08 2014
Payload ID: 28 70 BF D0 05 00 00 92  TEMP: +024.500 C  TX_INTERVAL: 30  TEMP_RES: 2  TX_COUNT: 00 03 B9  Thu Nov 13 18:37:13 2014
Payload ID: 28 63 DD F5 05 00 00 F3  TEMP: +023.000 C  TX_INTERVAL: 30  TEMP_RES: 2  TX_COUNT: 00 03 99  Thu Nov 13 18:40:47 2014
Payload ID: 28 70 BF D0 05 00 00 92  TEMP: +024.500 C  TX_INTERVAL: 30  TEMP_RES: 2  TX_COUNT: 00 03 BA  Thu Nov 13 18:41:46 2014
```

Sensor management is done in a ssh terminal session on the Raspberry PI+



Temperature data visualized for the user in the browser