Monitoring of environmental parameters in the COMPASS experiment

T. Nechaeva[1],
Supervisor: C. M. Pires[2]
[1] *Theoretical Physics Department, Belorussian State University and Institute for Nuclear Problems, Minsk, Belarus*
[2] *Laboratory of Instrumentation and Experimental Particle Physics, Lisbon, Portugal*

August 13, 2018

1 Introduction

1.1 Overview of the COMPASS experiment

The COMPASS experiment makes use of CERN SPS high-intensity muon and hadron beams with energies from 160 to 200 GeV for the investigation of the nucleon spin structure and the spectroscopy of hadrons [1]. As for the polarized target, it requires both a strong and very homogeneous magnetic field, which is insured by a superconducting magnet, and extremely small temperatures, down to 60 mK (possibly the coldest macroscopic volume in the universe), which is possible by use of one of the biggest He3-He4 dilution refrigerators in the world. A process called dynamic nuclear polarization (DNP) is used to polarize the protons. Compared to the LHC experiments and other experiments done in colliders, the fixed target ones offer the advantage of high luminosities (because targets with solid state or liquid materials can be used).

In COMPASS, besides the naturally polarized beam of high energy muons (unique in the world), it is also possible to have polarized protons and neutrons. This allows the study of polarized cross-sections and cross-section asymmetries [2].

1.2 The Detector Control System

The detector control and monitoring systems provide a user interface to control the majority of the hardware parameters of the COMPASS apparatus and ensure that the quality of COMPASS data stays at a high level during data taking. Different aspects of the experiment are constantly monitored: the operation of the front-end electronics and the readout chain, the stability of the beam characteristics and the counting rate level of the different triggers. The consistency and correctness of the data flow produced by the front ends is not sufficient to guarantee a reliable operation of the detectors. This information has to be complemented by more detector-specific parameters, like supplied voltages and currents, hit profiles, time or amplitude information or noise spectra. The online monitoring is performed on data provided on the fly by the DAQ system.

The DCS architecture is composed of three layers. The supervisory layer provides the graphical user interface for accessing and monitoring the hardware parameters. The front-ends layer include the various software drivers specific for each hardware element. It provides the supervisory layer with a common communication interface to access the hardware. The hardware elements (crates, sensors, gas systems, etc.) form the devices layer. Such a modular architecture allows to control a large variety of devices in a coherent and transparent way:

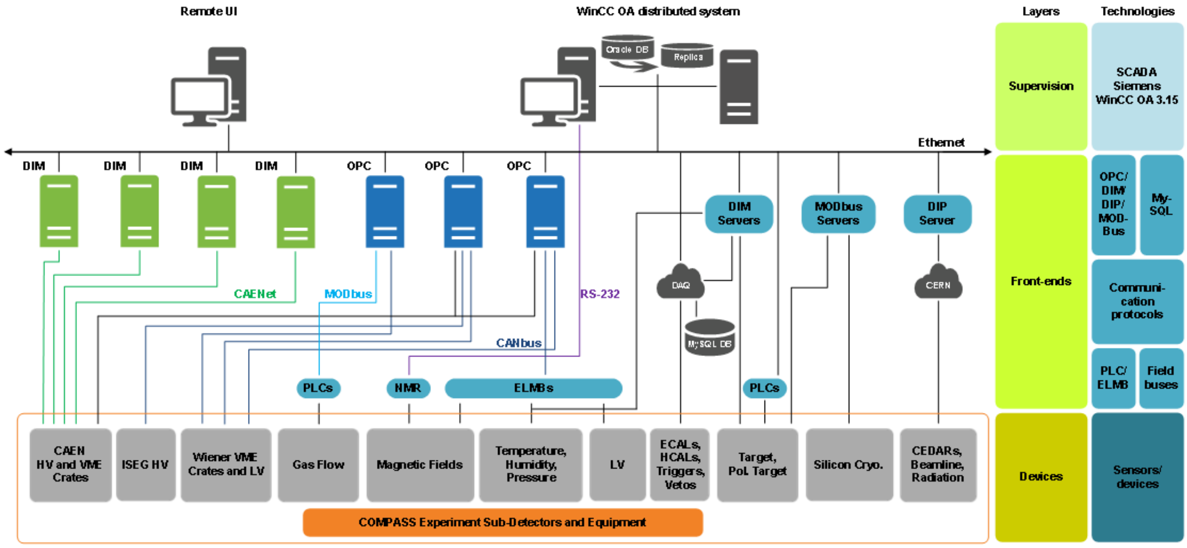

Figure 1: The architecture of the Detector Control System.

- control of crates and power supplies;
- monitoring of voltages and currents in crates and power supplies;
- control of HV channels;
- monitoring of temperature, humidity, pressure and magnetic field in specific points of the experimental hall, sub-detectors and magnets;
- monitoring of gas fluxes and mixtures in gaseous chambers.

The goal of the given project was to add some new temperature, humidity, pressure and radiation sensors connected via Raspberry Pi microcontroller boards installed either in the detector or in the gallery near it to the devices layer, program them in the needed way and embed them the supervisory layer using client/server approach. Then the analysis of sensors' performance and of data received from them was done.

2 Implementation

2.1 Overview of used sensors

To connect all the sensors to the DCS two microcontroller boards Raspberry Pi were used because of their reasonable price, good processor characteristics and the possibility to use several programming languages at the same time.

In this project temperature sensors of different precision were used:

1. DS18B20 – 1-Wire digital thermometer with $\pm 0.5^{o}$C accuracy from -10^{o}C to $+85^{o}$C [3],
2. TMP102 – 2-Wire low power digital thermometer with $\pm 0.5^{o}$C accuracy from -25^{o}C to $+85^{o}$C [4],
3. TMP36 – analog low voltage temperature sensor with $\pm 2^{o}$C accuracy from -40^{o}C to $+125^{o}$C, connected via MCP3008 ADC [5],
4. LM35 – analog low voltage temperature sensor with $\pm 2^{o}$C accuracy from -40^{o}C to $+125^{o}$C, connected via MCP3008 ADC [5],

5. MCP9808 – high accuracy I2C digital temperature sensor with $\pm 0.25^o$C accuracy (typical) from -40^oC to $+125^o$C [6],

as well as combined sensors:

6. DHT11 – digital humidity and temperature sensor with $\pm 2^o$C accuracy from 0^oC to $+50^o$C and ± 5%RH from 20% to 90%RH [7],

7. SI7021 – digital I2C humidity and temperature sensor with $\pm 1.4^o$C accuracy from -10^oC to $+85^o$C and ± 3%RH from 0% to 80%RH [8],

8. SHT31-D – digital humidity and temperature sensor with $\pm 0.3^o$C accuracy from -40^oC to $+125^o$C and ± 2%RH from 0% to 100%RH [9],

9. BMP280 – digital I2C pressure and temperature sensor with $\pm 1^o$C accuracy from -40^oC to $+85^o$C and ± 0.1 hPa accuracy from 300 hPa to 1100 hPa [10]

and Geiger counter

10. PocketGeiger type 5 – compact radiation detector with accuracy ± 0.016 μSv/h from 0.05 μSv/h to 10 mSv/h (Cs-137), 0.01 cpm - 300 Kcpm [11].

Their distribution between two microcontroller boards, as well as the user interface of DCS, are shown in the figure below. All the basic programming of sensors was done on Python.

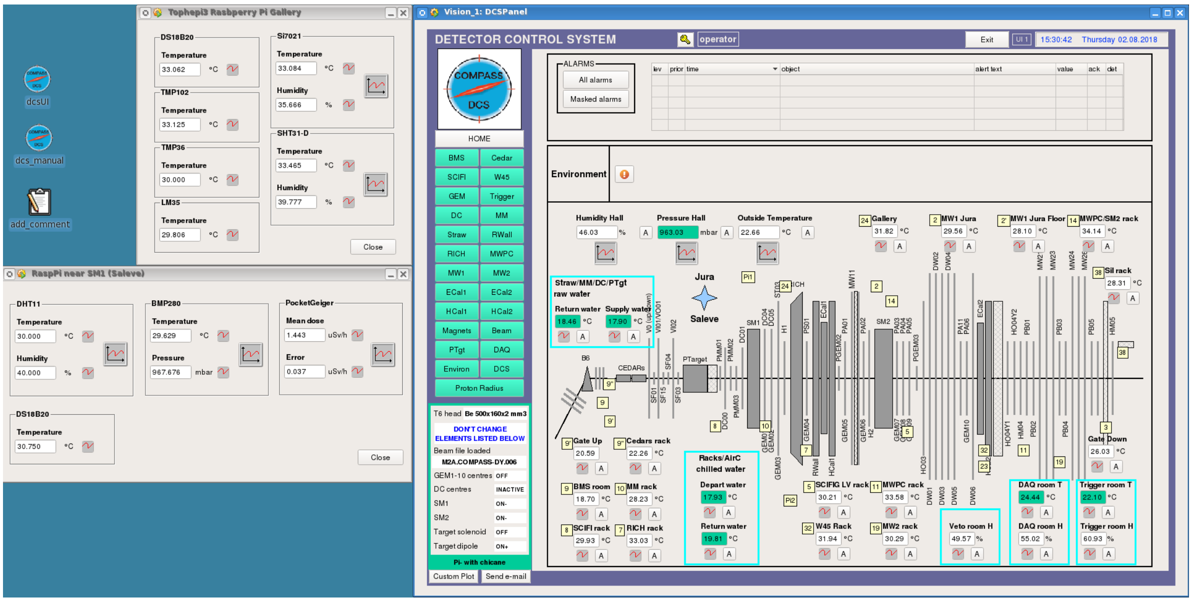

Figure 2: Distribution of sensors between microcontroller boards and the view of the DCS panel.

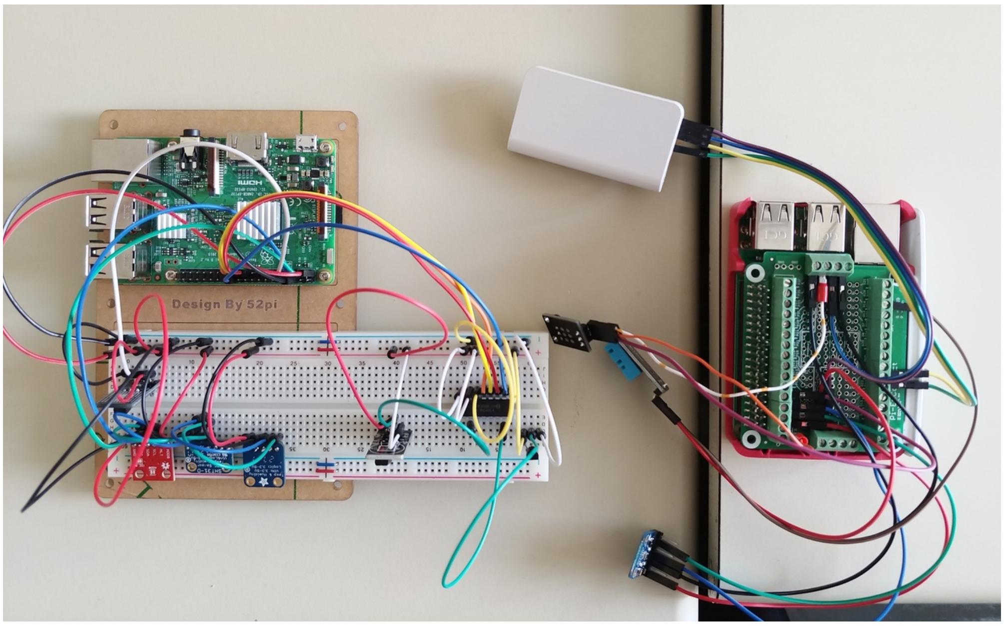

Figure 3: Assembly and connection of sensors to the boards.

2.2 Realization of client/server approach

To make microcontroller boards perform as servers the Distributed Information Management System (DIM) was used [12]. The basic concept in the DIM approach is the concept of "service". Servers provide services to clients. A service is a set of data (of any type or size) and it is recognized by a name - "named services". Services are usually requested by the client only once (at startup) and they are subsequently automatically updated by the server either at regular time intervals or whenever the conditions change (according to the type of service requested by the client).

In order to allow for transparency (i.e, a client does not need to know where a server is running) as well as to allow for easy recovery from crashes and migration of servers, a name server was introduced. Servers "publish" their services by registering them with the name server (normally once, at startup). Clients "subscribe" to services by asking the name server which server provides the service and then contacting the server directly, providing the type of service and the type of update as parameters.

The name server keeps an up-to-date directory of all the servers and services available in the system.

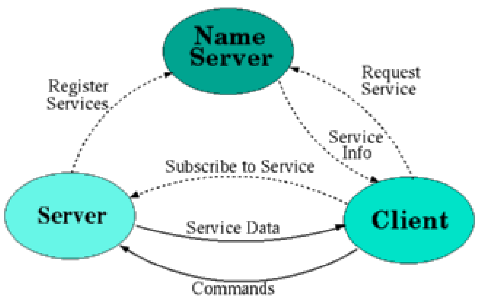

Figure 4: Interaction of DIM components.

Whenever one of the processes (a server or even the name server) in the system crashes or dies all processes connected to it will be notified and will reconnect as soon as it comes back to life.

Also, multithreading was implemented in Python to:

- update the values read from sensors faster and in parallel;
- check for invalid reading;
- not to get stuck if one/several sensors fail.

2.3 Integration in the DCS

To add the services to the DCS WinCC OA v3.15 (Simatic) was used [13]. First of all datapoints (device-oriented data objects representing a real device within the control system) containing datapoint elements (process variables, i.e. services from each particular sensor) were created. Then the list of available services was checked and all the necessary services were added to the datapoint elements, the elements were configured for archiving data and showing it in appropriate units. The last step was connecting datapoints to the DCS panel and creating user interface for monitoring: both boxes showing data from sensors in real time mode and plots showing changes of data for the selected time interval were made.

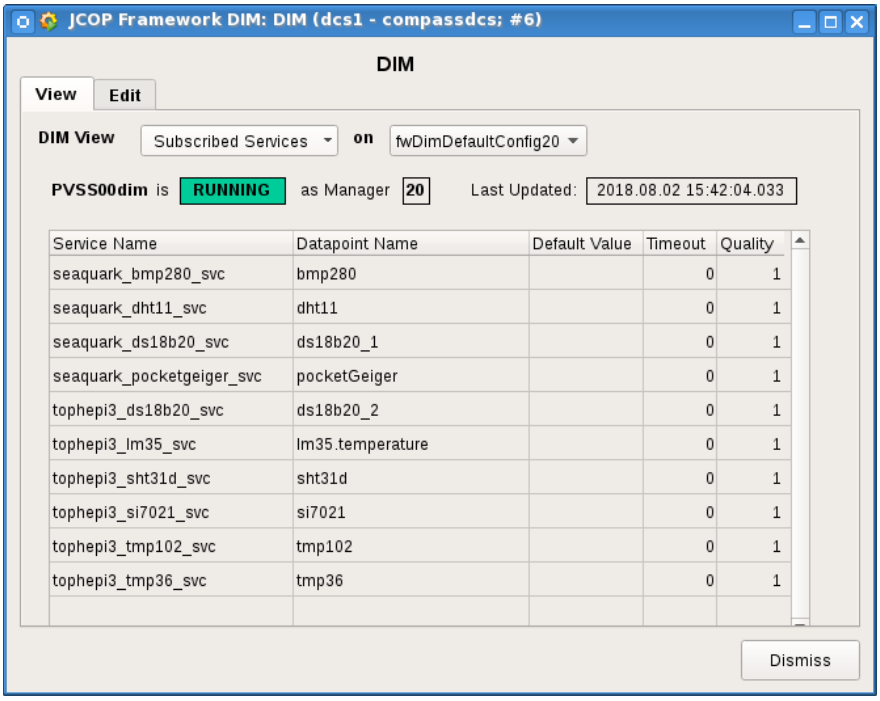

Figure 5: The list of subscribed (available) DIM services.

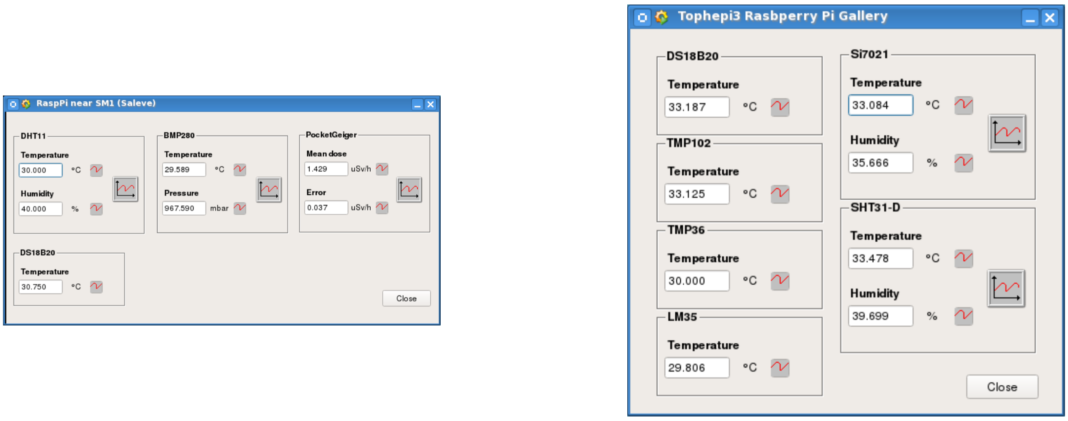

Figure 6: *Left panel:* User interface for the board located between the RICH detector and the Superconducting magnet (SM1); *Right panel:* User interface for the board located in the gallery on the opposite side. Values in the boxes are updated every 10s, buttons to the right of the boxes open plots of the data for the chosen time interval.

3 Installation and running

As it was mentioned before the boards were installed in two points of the experimental area: between the RICH detector and the SM1 and in the gallery on the opposite side. The data were taken from both of them for more than three weeks; the data for the last two weeks for all the sensors are displayed on the plots below. X-axis displays the time period of data taking, y-axes display the values of each sensor given in the corresponding units; y-axes are slightly offset from each other for clarity.

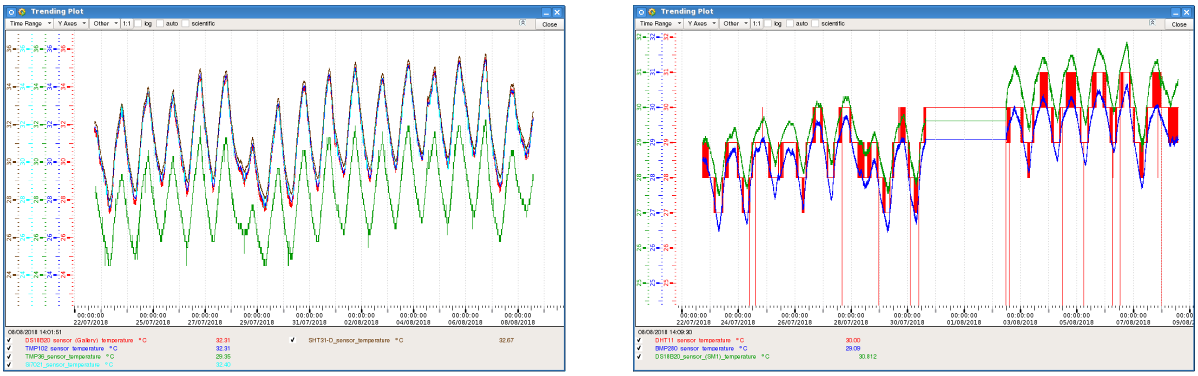

Figure 7: *Left panel:* Values of temperature from the sensors installed in the gallery taken for the two-week period; *Right panel:* Values of temperature from the sensors installed in the SM1 point taken for the two-week period.

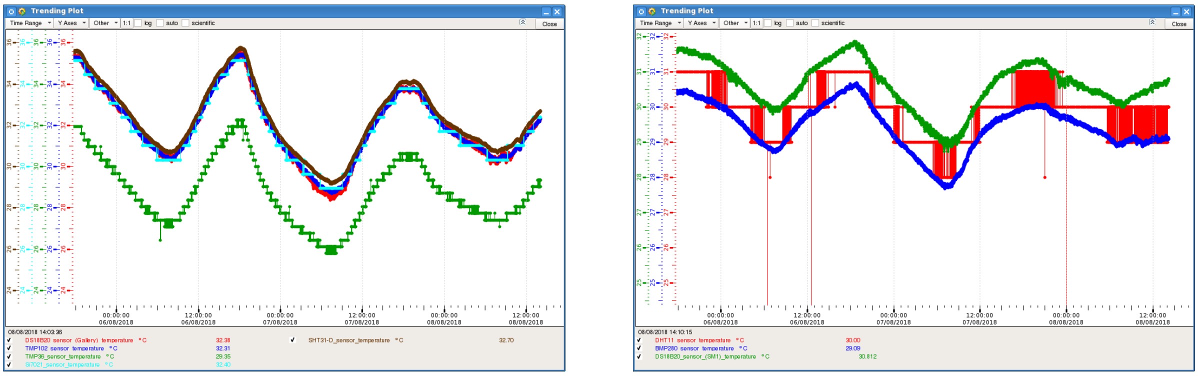

Figure 8: *Left panel:* Values of temperature from the sensors installed in the gallery taken for three days; *Right panel:* Values of temperature from the sensors installed in the SM1 point taken for three days.

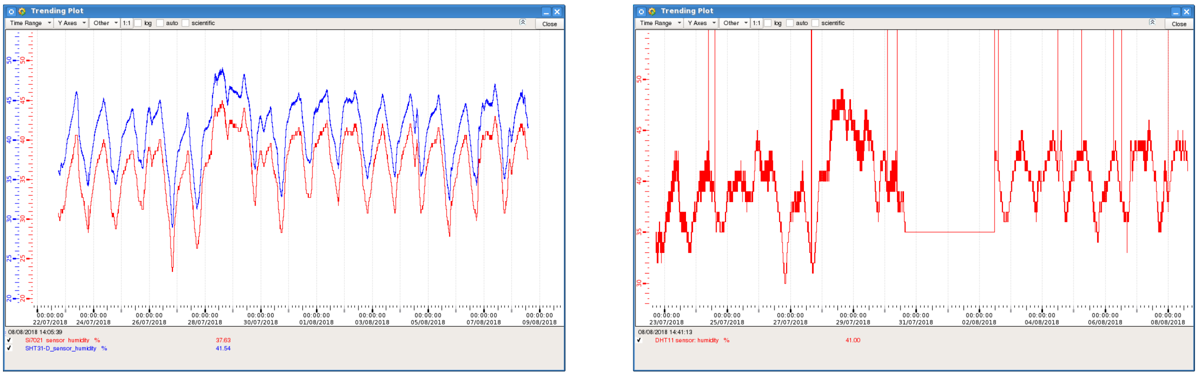

Figure 9: *Left panel:* Values of humidity from the sensors installed in the gallery taken for the two-week period; *Right panel:* Values of humidity from the sensors installed in the SM1 point taken for the two-week period.

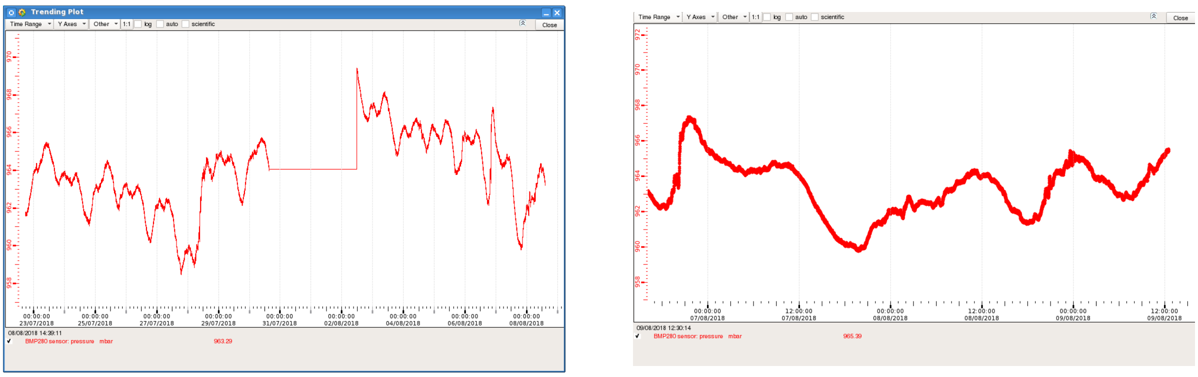

Figure 10: Values of pressure from the sensor installed in the SM1 point taken for: *Left panel:* the two-week period; *Right panel:* three days.

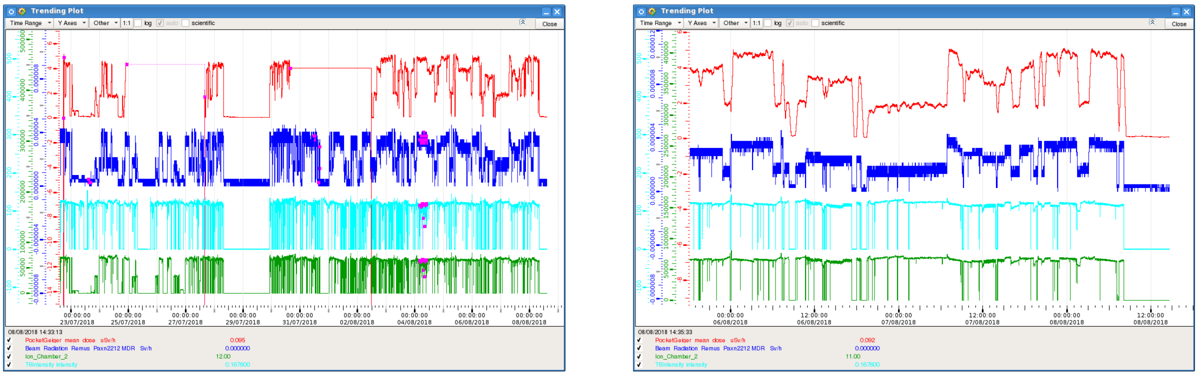

Figure 11: Values of the mean dose from the radiation sensor installed in the SM1 point taken for: *Left panel:* the two-week period; *Right panel:* three days; comparison with the values taken from other sensors/systems.

4 Conclusion

One of the main goals of the project was to compare the performance of different types and brands of sensors and to show, which can be suitable for the precise environmental monitoring as well as for the running in radiation conditions.

From the analysis of the plots made for different time periods (two weeks and three days) one can see both the stability of sensors' performance and the precision of each sensor. Smooth lines show almost continuous changes with a small step, hence, the precision of the corresponding sensor is higher. Absolutely straight line stands for absence of incoming data, i.e. either a crash (incorrect performance) of the sensor or a crash of the hosting board.

It can be seen (Fig.8; Fig.9 and 10 – right panel), that the most precise sensors within the given set are SHT31-D (temperature and humidity), TMP102 (temperature) and BMP280 (temperature and pressure), while the most inaccurate is DHT11 (temperature and humidity). Usage of the analog sensors (e.g. TMP36) for the precise measurements requires a high-resolution ADC. The data from LM35 and MCP9808 aren't displayed on the plots since these sensors broke down almost at the beginning of running.

The PocketGeiger radiation sensor values are displayed with ones from other sensors already been a part of the DCS. The PocketGeiger and the Remus paxn2212 MDR (CERN radiation and environment monitoring) show both the mean dose (the scales are quite different) and beam dynamics: the peaks correspond to the spills, the straight lines mean either the absence of the beam or the crash of the sensor (or the board). The other two sensors (named Ion Chamber 2 and T6Intensity) show the beam intensity as well as the beam dynamics. One can see, that the Remus and the PocketGeiger plots look quite similar, what may indicate a rather high accuracy of the last one, but the stability of its performance is lower – the straight line in the period from 25.07 to 27.07 shows the crash of this sensor, which restarted without any external actions afterwards. The data for the period from 28.07 to 29.07 correspond to the absence of the beam.

Straight line on the plots for SM1 installation point corresponding to the period from 31.07 to 02.08 shows the crash of the board. The reason was, most likely, a system error; after the restarting the board continued to work.

Furthermore, it was shown that the Raspberry Pi microcontroller boards can be a good commercially available solution, since they have good processor characteristics and all the necessary for any data taking (hence, for any measurements) protocols preinstalled within the Raspbian operating system. Also, the boards proved to be relatively stable in radiation conditions: for the duration of their running (without any protection shells) near the detector no significant damage and crashes were noticed.

References

[1] Christine A. Aidala, Steven D. Bass, Delia Hasch, and Gerhard K. Mallot Rev. Mod. Phys. 85, 655 (2013)

[2] "The COMPASS experiment at CERN", NIMA 577 (2007) 455518

[3] https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf

[4] https://www.sparkfun.com/datasheets/Sensors/Temperature/tmp102.pdf

[5] http://www.analog.com/media/en/technical-documentation/data-sheets/TMP35-36-37.pdf

[6] https://cdn-shop.adafruit.com/datasheets/MCP9808.pdf

[7] http://robocraft.ru/files/datasheet/DHT11.pdf

[8] https://cdn-learn.adafruit.com/assets/assets/000/035/931/original/Support-Documents-TechnicalDocs-Si7021-A20.pdf

[9] https://cdn-shop.adafruit.com/product-files/2857/Sensirion-Humidity-SHT3x-Datasheet-digital-767294.pdf

[10] https://cdn-shop.adafruit.com/datasheets/BST-BMP280-DS001-11.pdf

[11] http://www.radiation-watch.org/p/english.html/type5

[12] http://dim.web.cern.ch/dim/

[13] https://readthedocs.web.cern.ch/download/attachments/21178115/0407PVSSIIGettingStartedBasicsen.pdf?version=1modificationDate=1299054760000api=v2