The Neutron Monitor Control Panel

O García-Población[1,2], H Ivanov[2], I García-Tejedor[1,2], J J Blanco[1,2], J Medina[1,2], R Gómez-Herrero[1,2], E Catalán[1,2] and D Radchenko[2]

[1] Space Research Group, University of Alcalá, Spain
[2] Castilla-La Mancha Neutron Monitor, Parque Tecnológico de Guadalajara, Spain

E-mail: `oscar.gpoblacion@uah.es`

Abstract.
This work presents the current status and future plans of the Neutron Monitor Control Panel (NMCP), a new software developed to aid the operator in typical station maintenance and configuration operations. This software is integrated with the new so-called NOAS data acquisition system and it can be accessed using a supported web browser. It features a visual inspection tool to help the operator to identify spikes in the data, trace the origin of the spike back to the raw readings of each counter tube and pressure reading, and mark the data as invalid in the Neutron Monitor Database if desired. The software also provides information about station operation status, some descriptive statistics about current data being recorded and, in the future, will provide an interface to configure station parameters.

1. Introduction

Neutron Monitor measurements are widely used by researchers in different research fields, such as space weather and/or geomagnetic studies. That's why the Neutron Monitor Database (NMDB)[1] puts effort into delivering data with the best quality possible. The data is also used in real-time GLE alarm systems and other real-time applications; therefore, a data quality protocol must be applied in real-time, although it will be eventually revised by a human supervisor to ensure its correct behaviour. By quality protocol we refer to all the methods and techniques used, such as:

- Detection of abnormal data, commonly referred to as spikes.
- Detection of inactivity, i.e, if there is no data uploaded in NMDB for the last minutes. This will lead to a notification to the team responsible for the Neutron Monitor station.
- Neutron Monitors are formed by several counter tubes, typically eighteen. A malfunction in a single tube shouldn't make the entire station stop reading cosmic ray measurements. While it is true that missing counters degrades the quality of the measurement by increasing its variability, if the number of working tubes is high enough the result can still be acceptable. This leads us to the need of monitoring and detection of malfunctions separately in each of the tubes.
- Detection of changes with respect to the historical activity. Since changes in the immediate environment or the instruments can cause overall changes in the measured values of a Neutron Monitor, detecting and correcting those changes is desirable.

- Comparing the data from one station to the data of different nearby stations. Detecting isolated events could indicate that there is a malfunction in that station.

Once the corrupted data is detected, the cause of such data corruption can often be traced down to its cause. Most of the time these malfunctions are caused by the instruments forming the data acquisition system, so detecting the corrupt data helps us improve and learn more about its cause. The figure 1 shows schematically the main sources of errors. Most of them are Electro-Magnetic Interferences (EMI) in the preamplifiers (A), in the transmission lines from the amplifiers to the data acquisition system (B) and in the data acquisition hardware itself (C). Also malfunctions in the software (D) can introduce undesirable artifacts in the data.

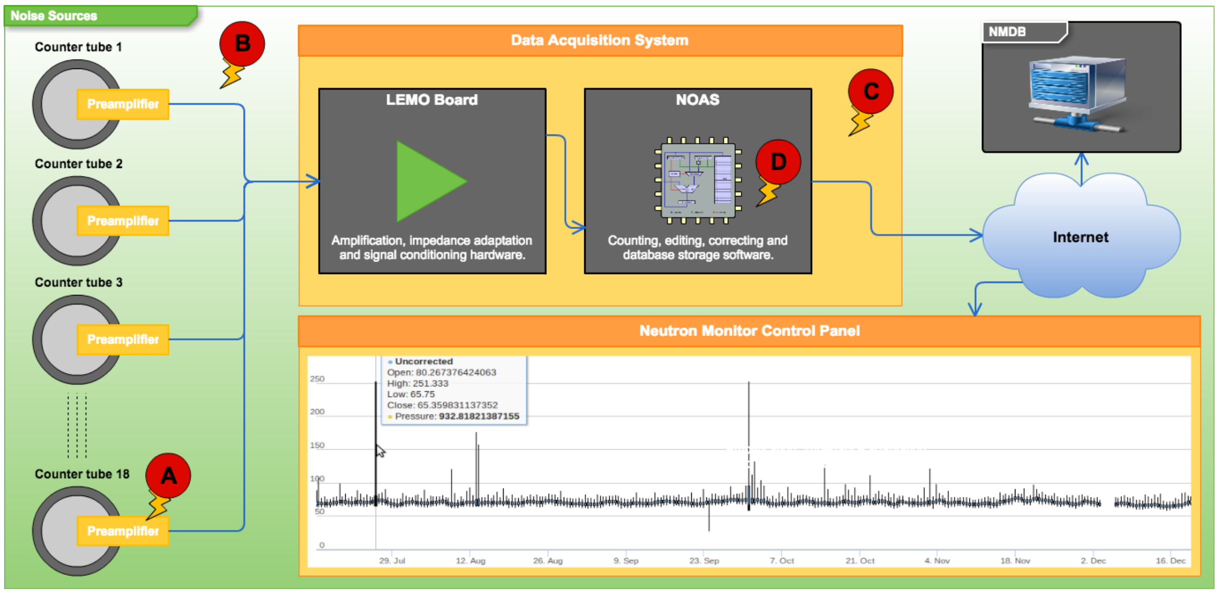


Figure 1. Possible error sources identified in our acquisition process.

In order to facilitate the detection of the corrupt data we are working in the development of a web tool which will help us detect these corruptions. The web application makes use of dynamically generated plots which are designed to highlight possible corrupted data.

This software is currently running for testing in the second release of the data acquisition system[2] designed for the CaLMa[3] and KIEL2 stations.

2. System architecture

There are two main elements in the architecture: a client that requests and consumes data using a web browser and a system that collects and serves the data, which in this system is the data acquisition system. This separation of roles follows the client-server design pattern[4]. The figure 2 shows a block diagram with all the components involved along with its relationships. Each block and relationship will be described below.

The software architecture is based on the Model-View-Controller (MVC) software design pattern[5]. This pattern encourages the separation between the data, the data processing and the data representation.

The view component is in charge of the data representation and the user interface. In our system, this component is mainly executed by the client in a web browser, which is not only a convenient graphical output device but also provides a way to interact with the user. In response to these interactions, the browser makes requests to a server which answers back with new data to be represented. All the software running in this component has been written in JavaScript, and makes intensive use of two libraries: HighStock[6] to draw plots and charts, and Sencha ExtJS[7] which is a very powerful framework that handles among other things the graphical user interface.

The model component is located on the server side, in the data acquisition system. This model comprises all the data captured from each counter tube and the algorithms used to process this data, such as atmospheric pressure corrections, sanity checks, editors to combine all the readings into a single one, etc. There are two databases involved in this component, one in the data acquisition system itself and another in a regular server. The first one is a small and light SQLite database, running in the data acquisition system on a BeagleBone Black embedded system[2]. This database stores per detector count rates, atmospheric pressure and voltage levels along with a time stamp. The data from this database are edited, corrected and sent to a second database server running MySQL[8]. This is an enterprise grade relational database server, used also by NMDB. The software needed to implement the editors, corrections, and transfer between databases has been written in the PHP programming language, using the Zend[9] as support framework.

Several controllers adapt and serve the model data as REST-like web services, using JSON[10] for data representation. REST is a well known software architecture that constraints the design of components to increase compatibility and isolation, standardising the way that services can be consumed. This web services are designed to be a documented Application Programming Interface (API) for accessing the station information and services, allowing third party applications to be further developed. The main consumer of this API right now is the spike tool, which is used to track and mark anomalous surges in the data from the station. The next section describes this tool in detail.

3. The spike tool

Currently the Spike Tool consists of three modules, `Spike`, `SpikeCorrected` and `ChannelStats`. `Spike` and `SpikeCorrected` are very similar, but `SpikeCorrected` works with a different set of data and has some extra functionality.

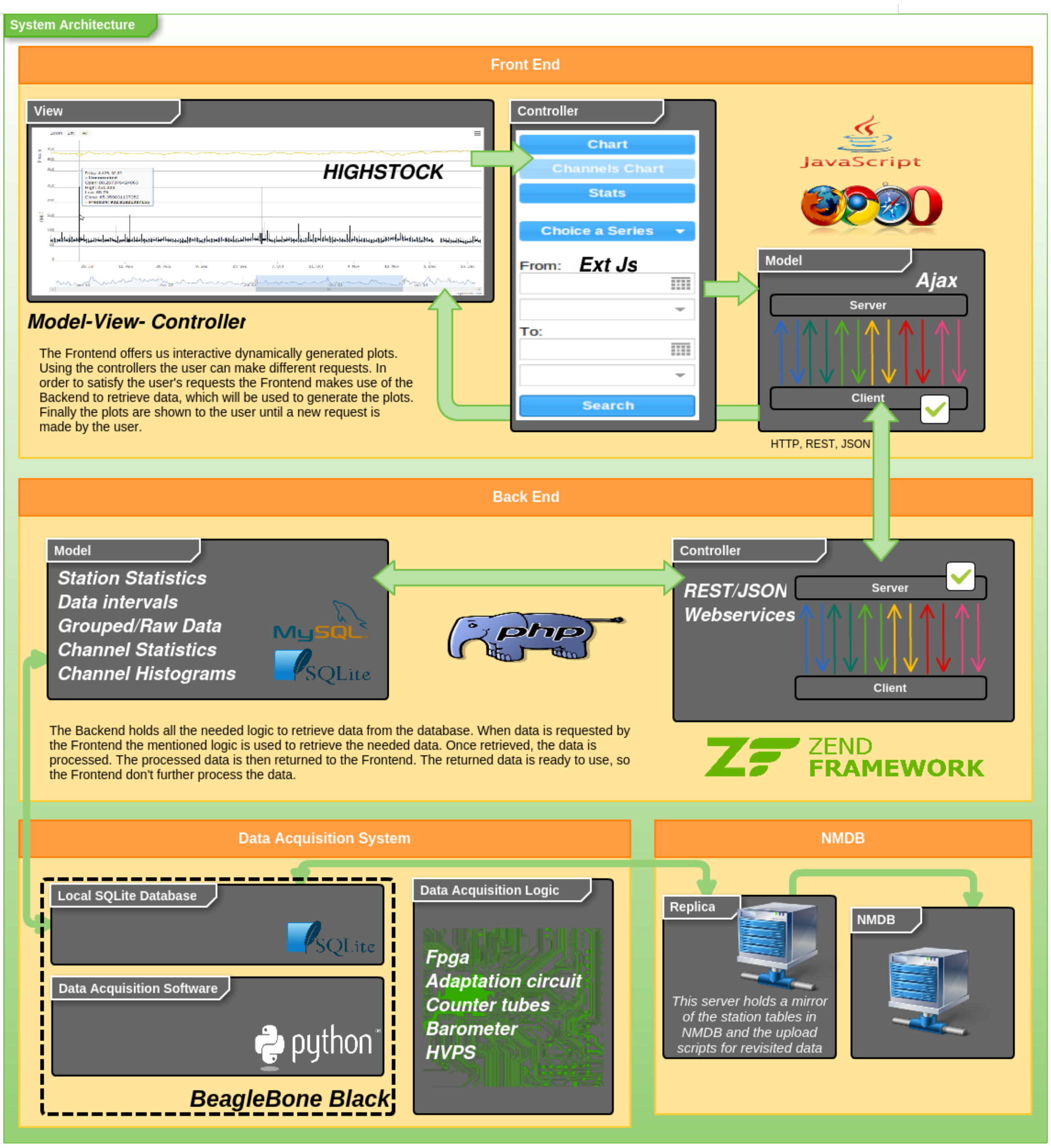
The `Spike` module offers an interactive chart in which the spikes are easy to trace. As shown in the figure 3, initially the chart displays a large set of data, and some kind of data grouping is needed. Just averaging the data flattens the spikes. The Candlestick chart is used to prevent the aforementioned problem, as this way the max and min values can be easily distinguished. Eventually, when a short interval of data is requested and no grouping is needed to display it, the chart will automatically switch to a line chart.

As mentioned above the chart is interactive, and offers a very intuitive zooming functionality. Clicking and dragging through an interval of the chart will raise a zoom event, and depending on the drag direction the event will be zoomed In or Out. The chart also offers a navigator which allows us to navigate forwards or backwards in time. Datetime input fields are available too, which allows the user to plot a specific time interval.

At first the uncorrected data are used to generate the charts, but we are given the chance to choose between Uncorrected data and Efficiency and Pressure corrected data. The atmospheric pressure is also plotted, which helps us to evaluate how the atmospheric pressure affects the measurements of our Neutron Monitor station.

Once we have a spike located, we can see the raw reading of the counter tubes, which allows us to better track the spike. This option is only activated when representation time scale is small enough to allow plotting every single value for each counter without neither overplotting nor cluttering the chart.

As said before, the `SpikeCorrected` module is very similar to the `Spike` module which was explained above. First of all, this module makes use of a different set of data. The `Spike`



The Frontend offers us interactive dynamically generated plots. Using the controllers the user can make different requests. In order to satisfy the user's requests the Frontend makes use of the Backend to retrieve data, which will be used to generate the plots. Finally the plots are shown to the user until a new request is made by the user.

The Backend holds all the needed logic to retrieve data from the database. When data is requested by the Frontend the mentioned logic is used to retrieve the needed data. Once retrieved, the data is processed. The processed data is then returned to the Frontend. The returned data is ready to use, so the Frontend don't further process the data.

This server holds a mirror of the station tables in NMDB and the upload scripts for revisited data

Figure 2. System architecture

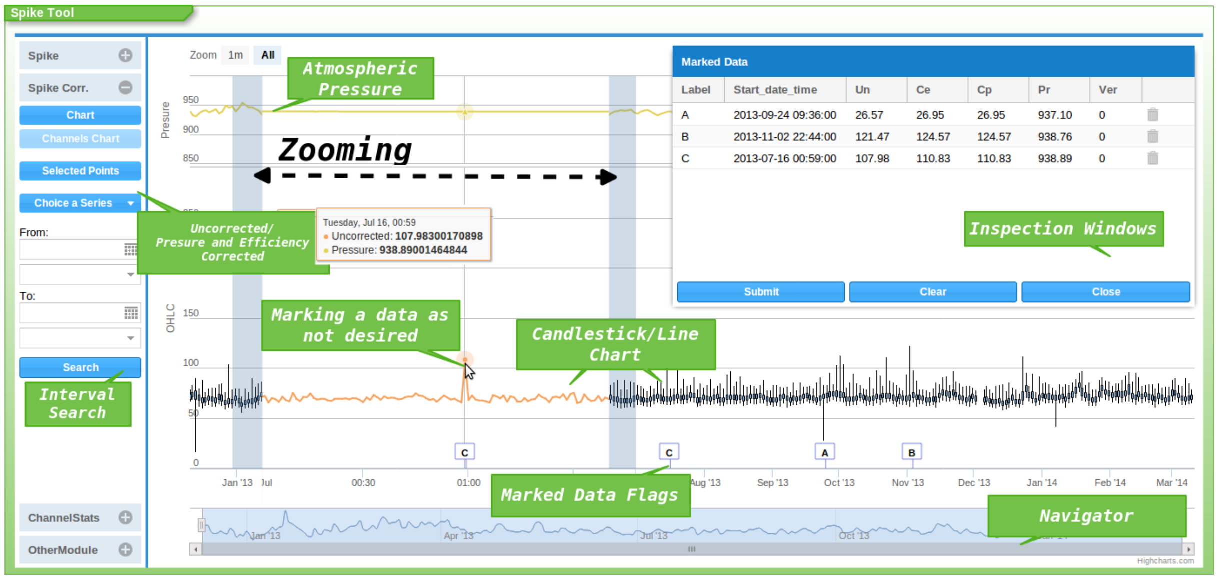
module uses the raw data which comes from the data acquisition system while `SpikeCorr` uses the revised data, and also provides the ability to update the revised data.

Clicking on a value in the chart will mark the clicked data, which is stored in a table. The data in the table can be later submitted to the revised set of data, which means that the mentioned data will be considered invalid and treated as void in the future.

The table which contains the marked data can be seen on a separate window, which can be shown and hidden when needed. The window also contains all the controls needed for submitting and unmarking the marked data.

The last module, `ChannelStats`, is intended to give information about the different channels separately, so that we can identify malfunctions on isolated channels. The module offers various statistics for user defined time intervals, among which the simplest are the average value, minimum/maximum value and standard deviation.

The ChannelStats module offers distribution charts, too. Histograms with the channels' raw data are plotted in the requested time intervals. The chart is highly interactive, and the series can be shown, hidden and highlighted. This module is currently under development.



Marked Data

Label	Start_date_time	Un	Ce	Cp	Pr	Ver
A	2013-09-24 09:36:00	26.57	26.95	26.95	937.10	0
B	2013-11-02 22:44:00	121.47	124.57	124.57	938.76	0
C	2013-07-16 00:59:00	107.98	110.83	110.83	938.89	0

Figure 3. Spike Tool

4. Future work

The Neutron Monitor control panel is a natural add-on to the new data acquisition system designed for Neutron Monitors and is under active development to add new useful features in the near future. Some of these future features are summarized below.

Online statistics The control panel will include a module for descriptive statistics that will be calculated in real-time from the counter tubes raw readings. This will help to identify drifts in the behavior of the detectors and to apply the appropriate actions to maintain data quality.

Station reconfiguration This will allow the operator to disconnect a counter from the station without affecting global station count. This can be useful to perform maintenance operations such as counter tube diagnostics.

Station parameter setup New electronics allow the remote control of some parameters, for example the high voltage power supply set point, the remote database for data upload, backup policies, etc.

Alarms and push notifications When connectivity allows it, it will be possible to configure alarms and notifications to recipients using the new push mobile technologies. For example, if the station is no longer uploading data to NMDB or if its data is out of quality standards.

Detector response histogram A new design from University of New Hampshire of the front-end amplifiers provides a pulse from which the width is proportional to the energy of the incident particle. The new data acquisition system NOAS features a different programmed core running in its Field Programmable Gate Array (FPGA) device. This core enables the system to read these pulses and to generate a histogram with the distribution of the pulse energy over a given time interval. This will enable a mechanism for non-intrusive diagnostics of the counter tubes. This means that diagnostics can be carried out without disconnecting the detector, changing its biasing voltage or, in general, altering its normal operational parameters. Therefore there is no need to interrupt the radiation measurements while the tests are being conducted.

Even though this tool is focused on spikes, there are many other data anomalies that affect data quality, like data drifts, steps, missing data, and so on. Dealing with each one of these requires a very different approach and further study is therefore required. When available, the software designed to address these problems could be easily integrated in the NMCP as a new service.

Acknowledgments
The authors would like to thank the Guadalajara Science and Technology Park for its support for this project.

References

[1] Mavromichalaki H, Papaioannou A, Plainaki C, Sarlanis C, Souvatzoglou G, Gerontidou M, Papailiou M, Eroshenko E, Belov A, Yanke V, Flückiger E O, Bütikofer R, Parisi M, Storini M, Klein K L, Fuller N, Steigies C T, Rother O M, Heber B, Wimmer-Schweingruber R F, Kudela K, Strharsky I, Langer R, Usoskin I, Ibragimov A, Chilingaryan A, Hovsepyan G, Reymers A, Yeghikyan A, Kryakunova O, Dryn E, Nikolayevskiy N, Dorman L and Pustil'Nik L 2011 *Advances in Space Research* **47** 2210–2222

[2] Población Ó G, Blanco J J, Gómez-Herrero R, Steigies C T, Medina J, Tejedor I G and Sánchez S 2014 *Journal of Instrumentation* **9** T08002

[3] Medina J, Blanco J J, García O, Gómez-Herrero R, Catalán E J, García I, Hidalgo M A, Meziat D, Prieto M, Rodríguez-Pacheco J and Sánchez S 2013 *Nuclear Instruments and Methods in Physics Research A* **727** 97–103

[4] Wikipedia Client–server model — wikipedia, the free encyclopedia [Online; accessed 22-January-2015] URL `http://en.wikipedia.org/wiki/Client-server_model`

[5] Wikipedia Model–view–controller — wikipedia, the free encyclopedia [Online; accessed 22-January-2015] URL `http://en.wikipedia.org/wiki/Model-view-controller`

[6] Highcharts js [Online; accessed 23-March-2015] URL `http://www.highcharts.com`

[7] The extjs framework [Online; accessed 23-March-2015] URL `http://www.sencha.com`

[8] The mysql database server [Online; accessed 23-March-2015] URL `http://http://www.mysql.com`

[9] The zend framework [Online; accessed 23-March-2015] URL `http://http://framework.zend.com`

[10] Wikipedia Json — wikipedia, the free encyclopedia [Online; accessed 23-March-2015] URL `http://en.wikipedia.org/wiki/JSON`