Arduino and Nagios integration for monitoring

V Fernández, A Pazos, J Saborido and M Seco

Instituto Galego de Física de Altas Enerxías (IGFAE), Universidad de Santiago de Compostela, Santiago de Compostela, Spain

E-mail: victormanuel.fernandez@usc.es, antonio.pazos@usc.es, juan.saborido@usc.es, marcos.seco@usc.es

Abstract. The data centre at the Galician Institute of High Energy Physics (IGFAE) of the Santiago de Compostela University (USC) is a computing cluster with about 150 nodes and 1250 cores that hosts the LHCb Tiers 2 and 3. In this small data centre, and of course in similar or bigger ones, it is very important to keep optimal conditions of temperature, humidity and pressure. Therefore, it is a necessity to monitor the environment and be able to trigger alarms when operating outside the recommended settings.

There are currently many tools and systems developed for data centre monitoring, but until recent years all of them were of commercial nature and expensive. In recent years there has been an increasing interest in the use of technologies based on Arduino due to its open hardware licensing and the low cost of this type of components. In this article we describe the system developed to monitor IGFAE's data centre, which integrates an Arduino controlled sensor network with the Nagios monitoring software. Sensors of several types, temperature, humidity and pressure, are connected to the Arduino board. The Nagios software is in charge of monitoring the various sensors and, with the help of Nagiosgraph, to keep track of the historic data and to produce the plots. An Arduino program, developed in house, provides the Nagios plugin with the readout of one or several sensors depending on the plugin's request. The Nagios plugin for reading the temperature sensors also broadcasts an SNMP trap when the temperature gets out of the allowed operating range.

1. Introduction

The data centre at the IGFAE provides computer resources to local groups working for the Auger and LHCb experiments, or doing research in QCD phenomenology and medical physics. It is also a GRID TIER-2 site serving the LHCb Virtual organization. Currently our data centre hosts five racks containing roughly 150 nodes and consuming around 25kW of electricity.

The importance of a monitoring system to control the environmental conditions inside the data centre has increased along the years. From the initial goal of checking whether those conditions meet the necessities of the equipment installed and send alarms to the appropriate operators when not, they have become vital for planning improvements on the power efficiency of the data centres.

The monitoring system implemented was a progressive design. The first step was to survey possible hardware solutions, both commercial and open hardware, and consider the pros and cons of each configuration. Of course, there were common advantages to both classes, for example the wide range of sensors available or that both classes could be tightly integrated although, for open hardware solutions where the user is doing all the development, it could require a lot of work on his part. A clear advantage of commercial systems is that they provide support for an

initial period of time that can be extended afterwards. On the minus side of the commercial solutions is their price, although for mid an large data centres this factor is not an issue since it is a small fraction of the total cost, they were, at least, twice as expensive as an Arduino based solution when the type and number of sensors intended to be installed was taken into account. Another point considered when evaluating the different solutions was how difficult was to integrate them with the already running monitoring infrastructure based on Nagios.

The organization of the paper is as follows. First we provide a brief overview of the Nagios monitoring system and the Arduino board. In section 3, both the software and hardware implementation of our system is discussed. Finally, we will show the physical distribution of the sensors and data obtained from our working system.

2. Technology overview

In this section a basic description of the main features of both Nagios and Arduino systems, used in the proposed monitoring system, is provided.

2.1. Nagios

Nagios [1] is an open source system that allows to monitor the health of machines, networks and services[2, 3]. In what follows, we will briefly describe some of the characteristics of this suite. Nagios' main task is accomplished via the use of plugins, small programs responsible for doing the tests and processing the output, since the core system does not perform any test. The status of the tests is described by one out of four states: OK, WARNING, CRITICAL or UNKNOWN. In case of monitoring numeric values these states are determined by the use of thresholds. Additional detailed information about the service is also provided in the output. The results are presented by a web interface, where the user can choose to see an overview of the full system or a more detailed view of each machine or service. The functionality of the basic system can be extended via the use of addons. Some of these addons provide interfaces for the Nagios configuration (like for example Lilac), execute plugins on remote machines (NRPE), store the data obtained by the plugins in a database (NDO, check_mk) or do graphic representations of the data (Nagiosgraph) to mention a few examples.

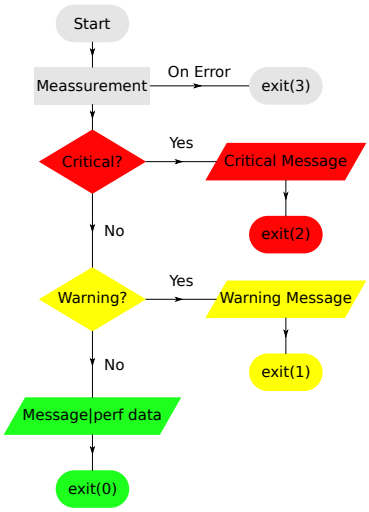

Figure 1. Basic flowchart of a generic Nagios plugin.

The typical flow chart of a Nagios plugin is shown in figure 1. In the starting phase the plugin, if necessary, sets the thresholds for the CRITICAL and WARNING states either from

command line arguments or from a set of defaults. The second step is to proceed with the actual measurement. If this step fails the sensor ends its execution with an exit code of 3. The result is then checked against the critical and warning ranges and, depending on the result, the plugin prints the proper message and finishes with the corresponding exit status. For any of this last three possibilities, OK, WARNING and CRITICAL status, the plugin could provide more detailed data about the measurement via the perfdata string, which is separated by the character '|' from the normal output. Both the normal output string and the perfdata string can be used by Nagiosgraph to obtain the data for graphic representations. The plugins can be created with any programming language, which along with the simple program flow just described, makes the creation of new plugin an extremely simple task. In addition, there are collections available providing plugins to, for example, check the state of HTTP, SNMP, SSH or CernVM-FS servers, the load of a machine, number of users logged or the amount of disk used.

2.2. Arduino

Arduino [4] is a microcontroller board and an IDE designed with the objective of being easy to use by beginners with no experience in electronics nor software, and of being affordable. Nowadays there are several Arduino boards [5]: the UNO, DUE, Duemilanove, MEGA, etc. Except the DUE which is based on the Atmega SAM3X8E 32-bit microcontroller, the others are based on various 8-bit Atmega microcontrollers with amounts of flash memory sizes ranging from 8 to 256kB and with a 20 to 70 input/output pins. The Arduino programming language is based on C/C++ and is linked against the AVR libc library allowing for access to all of its functions. The simplicity of using the system can be attributed to the high number of available libraries that enable interacting with external devices, either sensors or actuators, often with just a call to a function. All Arduino programs have, at least, two functions: the setup() function where the system is initialized, and the loop() function, equivalent to main() in a standard C program, which, as its name suggest, loops consecutively over its code. The Arduino capabilities can be extended by shields which are boards that are plugged on top or the Arduino PCB; an example is the GSM shield which was used as part of this project. They follow the same philosophy of being affordable and easy to program.

3. Implementation

In this section the software and hardware components of the system we have developed are briefly described.

3.1. Hardware

It was decided to use digital sensors in this project because they are calibrated at the factory and use digital communication protocols. The former means that the required measurement is directly read from the sensor or can be easily obtained from the read data and some parameters, provided by the vendor, that depend on the device's model . The sensors used in this project are the following:

- Maxim DS18B20 [6]: A digital thermometer with an accuracy of 0.5°C from -10°C to +85°C. It uses the 1-wire bus system [7], created by Dallas Semiconductor for its sensor devices, which requires only 1 pin for communication. Each device has a unique 64bit serial address that is used when connected in series with other devices. Although in our setup the sensors are connected to Arduino power pins, these devices can draw power from the data line when not connected to an external power supply.
- Bosch BMP085 [8]: A pressure sensor based on piezo-resistive technology; it provides an accuracy of 1hPa for a pressure between 300hPa and 1100hPa and a temperature in the range 0 to 65°C. The BMP085 uses the I^2C bus[9].

- Sensirion SHT75 [10]: A temperature and humidity sensor. It uses a capacitive technology and band-gap sensor for measuring humidity and temperature. Its accuracy is 2% relative humidity (RH) when the measured humidity is between 10% and 90% RH at 25°C. The device uses a 2-wire serial interface [11] which is compatible with the I^2C bus, therefore it can coexists with I^2C devices connected to the same wire.

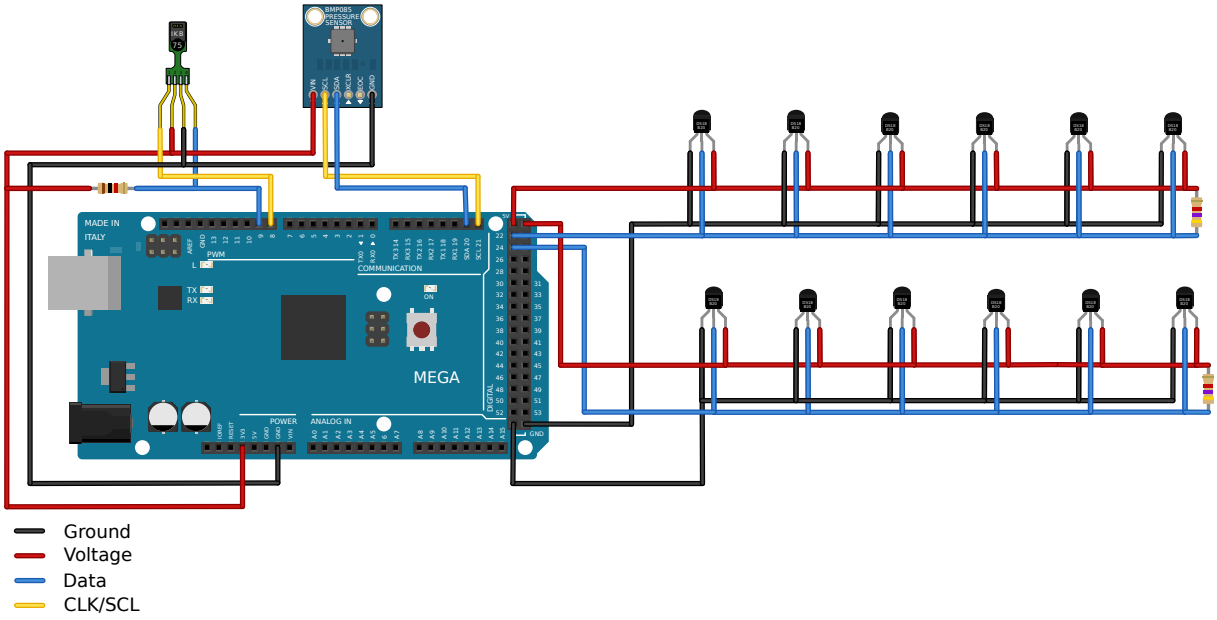

Figure 2. Connection schema of the sensors to the Arduino.

In figure 2 the wiring of the sensors to the Arduino board is shown. The SHT75 is located on the upper left part of the figure and just besides it is the BMP085. The board on which the BMP085 is assembled has all the necessary elements to connect it directly to the Arduino. However, the SHT75 needs a pull-up resistor between the power and data lines. Although the I^2C and the 2-wire serial protocols are compatible, these two sensors were not connected in series.

The final system has four sets of six temperature sensors connected in series. Three of the sensors in the same set, are located in the front a rack, and the other three in the rear. Each subset is considered as a single entity from the Nagios point of view. All the sets are connected to one of the three 5V pins provided by the Arduino with two of the sets sharing one. This type sensor also needs a pull-up resistor between its power and data lines

Hardware costs were about a third of a low-range dual-CPU server. A brief survey of some companies providing monitoring solutions with a similar funcionality, but without the possiblility of sending SMS messages, produced a price range between twice (APC) and three (ServerChecks) times the cost of our project.

3.2. Software

In figure 3 the software elements that compose the system described in this paper are shown; the information flow among these elements is also depicted. In the first step the Nagios plugin requests the status of the environment in our data centre, that is, it request the values of temperature, humidity and pressure of all installed sensors. In order to obtain this values the sensor runs, via an ssh connection to the server where the Arduino is plugged into, a program

which in turn instructs the Arduino to read the physical sensors. Once the Arduino obtains the sensors' measurement, it first checks whether the data is outside of a predefined range, in which case an SMS message is sent to the operators of the data centre. In all cases the measured value is sent back to the main server. The server provides the information obtained to the Nagios plugin, and also performs a check on the average temperature measured by a subset of the temperature sensors. If this average is above 45°C, an SNMP trap is generated to inform the worker nodes in the cluster of the critical condition.

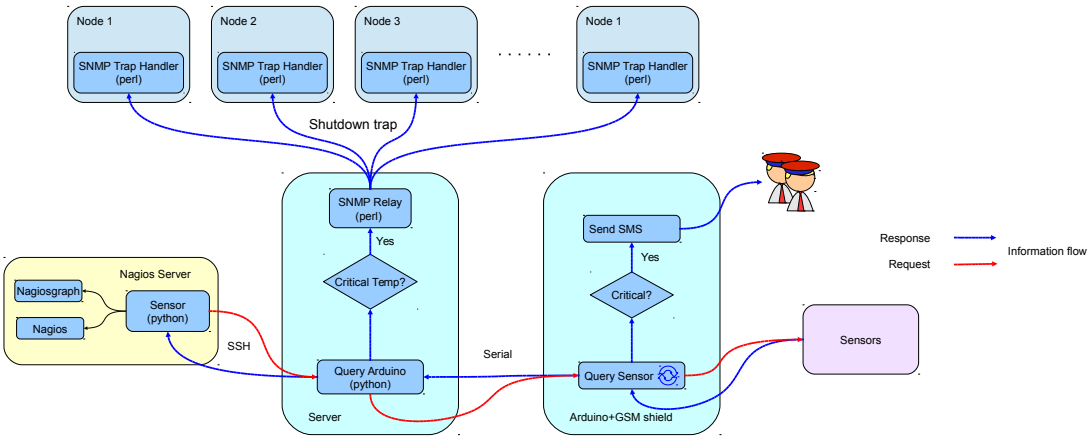

Figure 3. Information flow between the elements of the system described in the paper

Following the order in which the system responds we start with the program on the Arduino board. As stated in the introduction one of the strengths of the Arduino is the amount of libraries to control all kinds of devices. In particular, there exist libraries that hide the complexity of accessing the different sensors and the shield: OneWire [12] and DallasTemperature [13] to control the DS18B20, Sensirion for the SHT75 [14], Adafruit Unified sensor library [15] with the BMP085 driver [16] for controlling the pressure sensor and finally the GSM library [17] to access the GSM shield. These libraries allow to obtain the sensor readings with simple commands of the type getTemperature(sensor) where sensor is a label that identifies the sensor to be read. With this in mind the general behavior of the program on the Arduino board is as follows. In the setup function the GSM shield and the BMP085 are initialized. In the event they are not present an error message is printed on the serial port and a control variable is set to false to avoid calls to the missing device. Next, the loop funcion starts by reading all the sensors and checking wheter the average of the chosen subset of sensors is in a critical condition. If true, an SMS is sent to the data centre operators. At this point the program sleeps for 5 minutes before the next interaction of the loop starts. If during this period the program receives an instruction to read an specific sensor it will perform the requested measure, return the value and continue waiting until the 5 minutes are over. A command to read all sensors is interpreted as an instruction to start the next iteration of the loop. The list of the commands is provided in table 1.

Once the sensors' data is collected by the Arduino board it is sent to the program running on the main server. This program also generates the instructions for the Arduino board. Which instruction to submit is decided at execution time with the use of command line switches. Once the program receives the response, if it contains the set of temperatures used to test whether an SMS should be sent, it checks whether the average value of this set of temperatures is above 45°C. When the threshold is surpassed an SNMP trap is generated and transmitted, via an SNMP relay service, to the cluster nodes. When a node receives the trap a system shutdown is

Table 1. Commands sent to the Arduino board through the serial port. 'a' and 'b' are integers ranging from 1 to 4 and 1 to 6 respectively.

Command	Action
Tab	Read temperature sensor 'b' from line 'a'
Pt	Read temperature at the pressure sensor
Pp	Read pressure
Ht	Read temperature at the humidity sensor
Hp	Read humidity
A	Read all sensors

initiated.

The last element of the system is the Nagios plugin. In our case, the plugin behaves as if six Nagios plugins were run simultaneously meassuring the pressure, humidity and the four temperature sets. Five of these subplugins inform Nagios via the nagios' external command mechanism and the remaining one via the usual mechanism of exit code and output and perfdata strings.

4. Results

In this section we will show the sensors in our infrastructure, and the plots generated by the Nagiosgraph addon with data provided by the sensors. In figure 4 the location of the temperature sensors is shown. There are three more sensors in the rear of each rack. The rack at the farther end of the picture is offline. Figure 5 shows the location of the pressure and humidity sensors near the ceiling of the data centre.

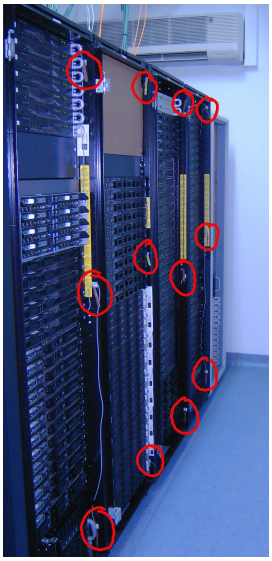

Figure 4. Location of the temperature sensors in the racks.

Figure 5. Location of the pressure and humidity sensors in the data centre.

The next two figures show the temperatures over a 24 hour period, figure 6, and in 8 days, figure 7. In our setting, Nagiosgraph also provides quarterly and annual graphs, but the periods for the plots can be chosen at configuration time.

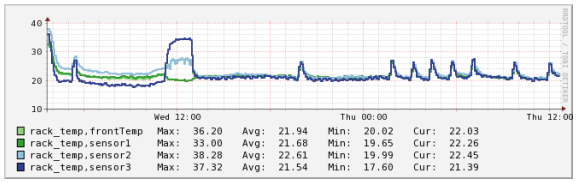

Figure 6. Temperatures in a 24h period.

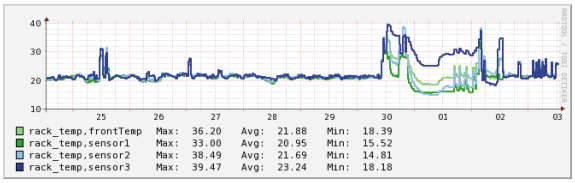

Figure 7. Temperatures during a week.

Finally we show the daily plots for pressure, figure 8, and humidity, figure 9.

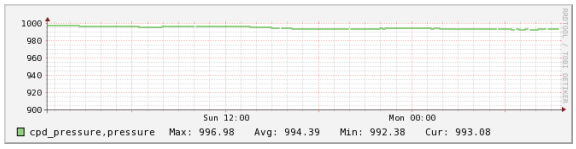

Figure 8. Daily pressure.

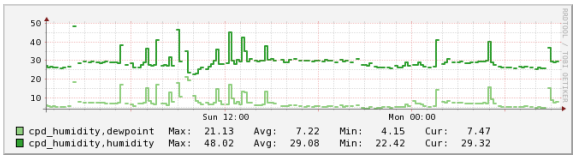

Figure 9. Daily humidity.

5. Conclusions

In this paper, a functional system that monitors the enviromental conditions inside a data centre is described. To perform the required task a solution that integrates an Arduino microcontroller with the Nagios monitoring system was developed. Because of the open nature of both systems it has been possible to create an affordable solution for small data centres.

Acknowledgments

This work was supported by grant FPA2010-21885-C02-01/02 from the Ministerio de Educación y Ciencia, Spain.

References

[1] Nagios website: `http://www.nagios.org`
[2] Josephsen D 2007 *Building a Monitoring Infrastructure with Nagios* (Prentice Hall) ISBN: 978-0-13223-693-5
[3] Schubert M, Bennet D, Gines J, Hay A and Strand J 2008 *Nagios 3 Enterprise Network Monitoring* (Syngress books) ISBN: 978-1-59749-267-6
[4] Arduino website: `http://www.arduino.cc`
[5] Arduino website: `http://www.arduino.cc/Products`
[6] Maxim DS18B20 Datasheet: `http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf`
[7] 1-Wire Bus: `http://www.1wire.org/Files/Articles/1-Wire-Design%20Guide%20v1.0.pdf`
[8] Bosch BMP085 Datasheet:
`http://ae-bst.resource.bosch.com/media/products/dokumente/bmp180/BST-BMP180-DS000-09.pdf`
[9] I^2C Protocol: `http://www.nxp.com/documents/user_manual/UM10204.pdf`
[10] Sensirion SHT75 Specs:
`http://www.sensirion.com/en/products/humidity-temperature/humidity-sensor-sht75/`
[11] 2-Wire Serial Interface:
`http://www.jennic.com/files/support_documentation/JN-AN-1041-2-Wire-Serial-Interface-1v1.pdf`
[12] OneWire Library: `http://playground.arduino.cc/Learning/OneWire`
[13] DallasTemperature Library: `http://milesburton.com/Dallas_Temperature_Control_Library`
[14] Sensirion Library: `http://playground.arduino.cc/Code/Sensirion`
[15] Adafruit Sensor Library: `https://github.com/adafruit/Adafruit_Sensor`
[16] Adafruit BMP085 unified driver: `https://github.com/adafruit/Adafruit_BMP085_Unified`
[17] GSM library `http://arduino.cc/en/Reference/GSM`
[18] PySNMP Library: `http://pysnmp.sourceforge.net`