Przemysław MAZURKIEWICZ
POZNAN UNIVERSITY OF TECHNOLOGY, INSTITUTE OF CONTROL AND INFORMATION ENGINEERING
3A Piotrowo St., 60-965 Poznań

An open source SCADA application in a small automation system

Abstract

In this paper, an application of the Open Source SCADA and a single board computer as a monitoring solution of a central heating system is presented. Description of the proposed SCADA software, hardware architecture and example data of the monitored system are reported. Application of the ScadaBR running on a SCB under Linux operating system is an effective and low cost solution.

Keywords: SCADA, IoT, home automation.

1. Introduction

For a few years a growing number of hardware and software solutions in field of Internet of Thing (IoT) has been observed. These new solutions allow connecting many different types of devices with web applications to share gathered data with users. In situation when reduction of the carbon dioxide (CO_2) becomes very important, it is important to reduce power consumption and thus reduce the fossil fuels consumption. Reduction can be accomplished if the resources to be reduced are precisely monitored. More generally, the monitoring of process data helps to lower the costs and spare the resources. Increasing costs of electric energy, water and fuels force the owners of single detached family homes to seek reduction solutions. The first step to that is to seek online media monitoring solutions. In industrial automation, this is accomplished with a Supervisory Control And Data Acquisition (SCADA) system. It is designed to acquire data from measuring devices, process them and supervise the process control [1]. At present, there are many low cost single board computers, IoT devices and M2M/SCADA projects which may adapt industry standards to small home automation systems.

In this paper, an Open Source SCADA system for the monitoring of a central heating system in a single detached family home, implemented on a Single Board Computer (SCB) is presented. The main purpose of the implementation was to get a low cost system for monitoring temperature, water and power consumption in the heating system. Due to the distribution of the monitored devices over a house area, the presented system should allow getting data from IoT devices. Additionally, it was assumed that all software should be released under the Open Source license to lower cost of the whole system.

2. Open hardware devices

The Open Source license allows the user to freely modify, use and distribute a software code, hardware designs or blueprints. There are a few Open Source license types. The most popular is GNU General Public License (GNU GPL). According to this license, software can be freely distributed (compiled version and source codes), can be used for commercial and a private purposes. Any derived code must be licensed with the same GNU GPL license and freely distributed.

An Open Hardware is a hardware whose schematics, blue prints, logic designs, CAD drawings are publicly available so that anyone can study, modify, make and distribute [2]. The Licensing is similar to the Open Source Software. The difference between the Open Source Software and Open Hardware is that the Open Hardware is not always for free. It is because some costs are connected with production or developing.

The most popular Open Hardware projects are *BeaglBone, Raspberry Pi, Arduino* or *BananaPi. BeagleBone, Raspberry Pi* and the *BananPi* are the examples of the SCB solutions. *Arduino* is the example of an Open Source rapid prototyping solution with easy to learn programming language. In the presented system, the Arduino platform was used as an IoT development platform.

The Raspberry Pi 2 model B was chosen as an SCB. Its specification is presented in Table 1. This is the most popular SCB with a good community support. As an operating system, GNU/Linux Raspbian wheezy 7.9 was used. It was assumed that the SCB would work as a network server, so only a light version (without Xorg subsytem) was installed. A kernel was updated to the version 4.1.19-v7+.

Tab. 1. Specification of the Raspberry Pi 2 [3]

CPU	Broadcom BCM2836 Arm7 Quad Core Processor
RAM	1 GB (shared with GPU)
Network	10/100 M Ethernet
I/O	4×USB, 40 pin GPIO,

3. Open Source software

There is a number of the Open Source applications which can be used in home or industrial automation. Interesting projects of home automation are OpenHab [4] and OpenRemote [5]. All these applications are the controllers of a smart building that can run on the embedded systems. In the automation industry, *RapidSCADA* [6], *Szarp* [7], *OpenAPC* [8], *ScadaBR* [9] can be used. They offer logging, monitoring and processing of the data from many data sources. These applications run on the most popular operating systems and offer the client server architecture.

The basic requirements for the software applied in this project is that it should:

- run on different operating systems (especially on the open source systems) and a hardware platform (especially on a SCB),
- easy deploy
- integrate with the IoT devices.

The above requirements are met by the *ScadaBR* application.

4. Description of the central heating system

A scheme of the monitored system is presented in Fig. 1. The main purpose of the SCADA system is to monitor temperatures, energy and water consumption in a single-family house. The temperatures are related to two subsystems. The first subsystem is a central heating system with a coal boiler, the second is a hot water solar collector subsystem. The central heating system is connected to the heaters and a hot water tank. To the same tank there is also connected the hot water solar collector. Dallas 1-wire DS18b20 sensors were used for temperature monitoring. These sensors were connected through the 1-wire bus to the Arduino MEGA 2560 board. Additionally, the output of the water meter POWOGAZ JS-90-2,5-NK were connected to this board. The software running on the MEGA 2560 collected data from the temperature sensors, the water meter and sent them through the ESP8622 wifi module to the SCB. The SCB board was connected to a Modbus network where there are two nodes. The first one was a hot water solar collector, controller Frisko ATTO2-SOL, and the second one was an energy meter Orno WE-504. The ATTO2-SOL controller controlled water heating in the water tank ERR 300. The energy meter measured the energy consumption of the coal boiler Funky Silver 16 kW. All Modbus devices communicated through the Modbus RTU. Since the SCB has only USB ports, the isolated USB to the RS-485 Mera-projekt MP01512 converter was used.

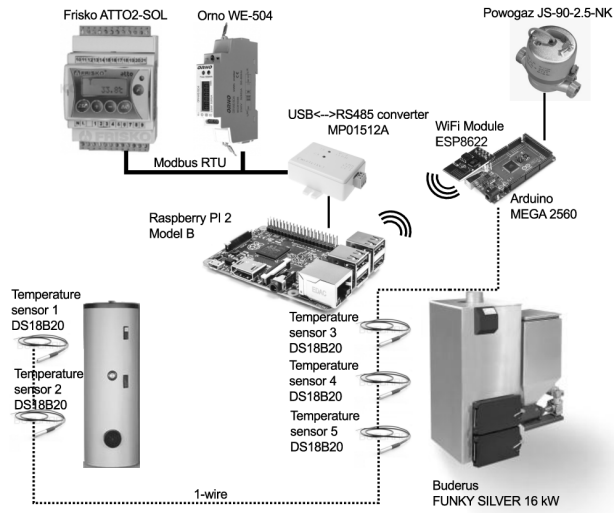

Fig. 1. Block diagram of the home automation system

5. ScadaBR description

ScadaBR is a web application written in java and released under the GNU General Public License version 3. It is a typical SCADA system offering logging, reporting and control functionality. It can communicate with the devices using many popular protocols such as Modbus RTU, Modbus TCP, Bacnet, OneWire, Http, Zwave. It allows defining the graphical views of the monitored plant. In views, the user can define buttons, animated images, charts, alarms. ScadaBR can process the collected data by applying logic operations. The Logic can be written in ECMA scripting language with a semantic similar to java. This application can run on Windows, Linux or Mac and requires only Apache Tomcat servlets server and a JDK. It can be installed locally or as a network server. All acquired date can be stored in a mysql database or locally in a file system.

The security is an important aspect in the web applications. The presented application offers the user based authentication. Every user can be assigned to two possible roles: an administrator or a regular user. The administrator of the SCADA system has the highest privileges. Only he can define the data sources and the data points and allows the access to the data sources for the regular users. Every user can define his own graphical view of the plant, define alarms and reports.

5.1. Plant configuration

The Plant configuration in ScadaBR starts with the definition of data sources which are responsible for communication between SCADA and particular measurements devices. For each data source, a communication frequency and the parameters related to the used protocols need to be defined. Each data source can have the defined data points which are process variables related to the monitored plant and acquired by the particular device. After the definition of the process variables, they can be used in setting up graphical views of the plant.

5.2. Graphical views

The graphical views are the graphical schemes of the controlled process with the variables placed on them. The views are rendered in the Internet browser application. When defining the view, a number of the different types of the graphical objects is available. This can be *Simple Point, Image Chart, HTML label* or a *Binary Image.* Each graphical object can have parameters formatted according to the CSS standard. The *HTML label,* which allows defining a html frame with the references to the remote resources such as the web widgets or images, is a very interesting object. Additionally, a graphical view can be interactive by using the java scripts thus it is quite easy to define buttons, drop down lists, toggles buttons or animated objects.

5.3. Logic and processing data

ScadaBR offers a logic handling tool – scripting module thus can be a powerful controller operating on the data and processing them. Implementation of the control algorithm or the processing functions is based on creation of the ECMAScripts. The scripts are assigned to the points of the Meta Data Source. Every script has a *Name,* uniq *Id*, data typ (*Numeric, Multistate, Binary, Alphanumeric*), *script context* (list of input variables), *body script* (function code) and a definition of *update event (context update, start of minute, start of hour, start of day, start of month, start of year)*. The script is executed in the case of value update *event.* The most popular case is a *context update* which evokes script execution every time the input variable change its value. With the scripts, it is possible to implement the control functions, data processing or the fault event detections.

6. Software configuration

In the implemented automation system, version 1.12.4 of the ScadaBR, jdk-8-oracle-arm-vfp-hflt version of the java, Apache-tomcat 7.0.64 and database MariaDB v. 5.5.47 was used. The database ran on the Synology DS115j file server. Fig. 2 depicts the architecture of the system. Access to the SCB is only through ssh. ScadaBR was configured to use an external database server.

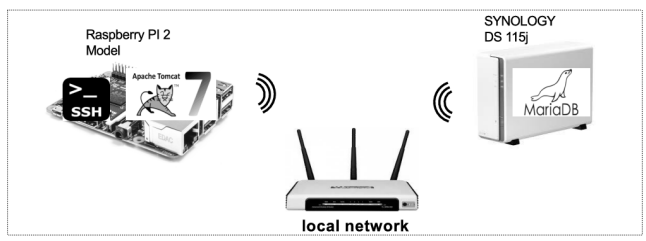

Fig. 2. System architecture

The process variables defined in the presented plant can be divided into three groups. The groups are related to the communication protocols. With a *HTTP Receiver* the SCADA system acquires the data received from the Arduino MEGA 2564 board in the HTTP PUT requests.

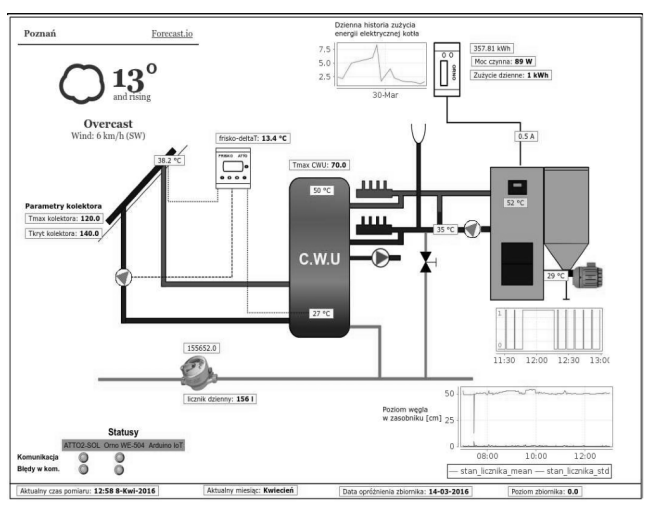

Fig. 3. Graphical view of the monitored plant

In the case of the *HTTP Retriever* data source, it is possible to gather data from the other web applications. In the presented example, the data of the particular events are retrieved from the Google docs spreadsheet, where dates of refilling the coal container are stored.

The third group are the variables related to the Modbus RTU protocol.

The Graphical view of the presented example is shown in Fig. 3. This view consists of the *Simple Points*, *Html labels, Image Chart* and *Binary Image objects.* The *Simple Points* are assigned to the data points of the defined data sources. The *Image Charts* renders the historical values of the particular variables. In the presented example, the *Image Charts* are used to show the historical values of the electric energy consumed by the coal boiler, the heating pumps and the activity of the coal boiler fan.

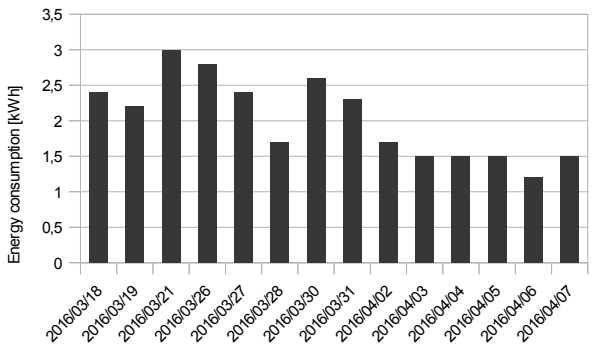

Fig. 4. Example of daily energy consumption

On the presented *Graphical view* the weather information from f*orecast.io* site is displayed. It was achieved with a *Html label* which includes the html code described on f*orecast.io* [10].

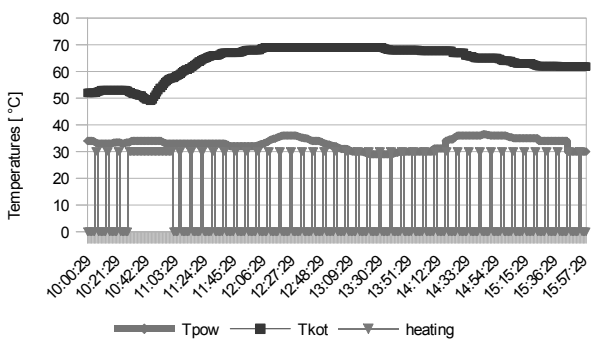

Fig. 5. Temperatures in the heating system observed during 5 hours' period

The examples of the data exported from the SCADA system are presented in Figs.4 and 5. Fig. 4 shows the daily electric energy consumption of the coal boiler on a particular day. Fig. 5 presents the temperatures related to the central heating system exported from a particular day and recorded in a range of 5 hours. In this example, the *Heating* curve is scaled to be shown in the figure. This curve depicts the ON/OFF activity of the coal boiler fan. This activity is detected by a current monitoring meter.

7. Summary

The Application of ScadaBR running on the Raspberry Pi 2 under the Linux operating system is an effective and a low cost solution for the process data monitoring and control in a small automation system.

This solution can be easily scaled and configured to the distributed SCADA system with a central ScadaBR server running e.g. on a VPS server and a number of the remote subsystem with ScadaBR on a SCB or an IoT devices.

By combining the proposed SCADA approach with a standard Linux tools such as *ssh, iptables* and *pound* proxy server one can build a stable and secure system.

8. References

[1] Bailey D., Wright E.: Practical SCADA for industry. Elsevier 2003.
[2] http://www.oshwa.org/definition/
[3] https://www.raspberrypi.org/products/raspberry-pi-2-model-b/
[4] http://www.openhab.org/
[5] http://www.openremote.org/
[6] http://rapidscada.org/
[7] http://www.szarp.org/pl
[8] https://www.openapc.com/
[9] https://sourceforge.net/projects/scadabr/
[10] http://blog.forecast.io/forecast-embeds/

Received: 21.03.2016 *Paper reviewed* *Accepted: 02.05.2016*

Przemysław MAZURKIEWICZ, PhD, eng.

Przemysław Mazurkiewicz received the MSc degree in Electrical Engineering from the Poznan University of Technology, in 2001 and the PhD degree in biocybernetics from the same University in 2010. His research work includes biosignals processing, IoT. Since 2002 he has been employed by Institute of Control and Information Engineering at Poznań University of Technology.

e-mail: przemyslaw.mazurkiewicz@put.poznan.pl