A SCADA Testbed from the Toy Store

Draft version, July 23, 2018

Ramin Sadre

I. Introduction

In the past years, Supervisory Control and Data Acquisition (SCADA) systems, and the industrial control systems (ICS) they are typically part of, have been the object of intensive research in the area of safety, reliability, and security. For researchers and educators, working on SCADA systems is challenging: Unlike most traditional ICT systems, modifying a SCADA system can have immediate consequences in the physical world. For this reason, a large number of testbeds as well as approaches to accurately simulate and emulate SCADA components and the physical processes controlled by them have been presented in the literature.

In the following, we will show how the fischertechnik construction toy [1] can be used to build a SCADA system that controls a real physical process. Unlike most existing testbeds, real sensors and actuators are monitored and controlled by the system. Our approach has the interesting property that it solely relies on free software and on easily procurable parts from the fischertechnik toy line.

II. System overview

A SCADA system consists of at least three parts:

- Sensors and actuators that monitor and act on physical processes, such as the electricity production in a power plant or the heating in a building.
- One or more Programmable Logic Controllers (PLC). PLCs are small devices that process sensor data and control attached actuators. The behavior of a PLC is defined by the control program running on it.
- Supervisory computers (or SCADA servers), which are computers responsible for gathering process data from the PLCs, sending control commands to them, visualizing the current state of the system, and archiving old measurements. In small systems, all these tasks are often performed by a single server, whereas larger SCADA systems can consist of several servers in charge of specific tasks. Supervisory computers and PLCs communicate using dedicated light-weight SCADA protocols, such as Modbus [2] or DNP3.

In our implementation, we will use the following software and hardware:

- We use the ScadaBR SCADA server software [3], running as a web application on a tomcat server.
- The fischertechnik ROBOTICS TXT Controller (short TXT) acts as PLC.
- All sensors, indicator lights, and actuators are standard components from the fischertechnik toy line.

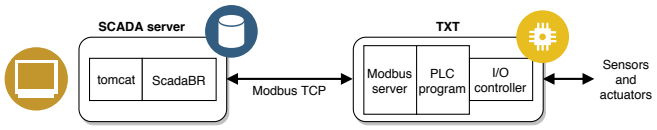

Fig. 1. System with Modbus server and PLC program on the TXT device

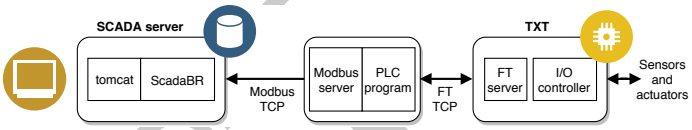

Fig. 2. System with Modbus server and PLC program on a dedicated host

ScadaBR implements various popular SCADA protocols such as Modbus TCP [2] and DNP3. Since the TXT is mainly a product targeting the educational toy market, it does not support any of those protocols by default. We have therefore written a small Modbus server that translates Modbus requests into appropriate I/O commands for the TXT. The Modbus server runs in parallel to the control program executed by the TXT. The resulting setup is depicted in Figure 1.

Alternatively, the Modbus server and the control program can run on a different host than the TXT, as shown in Figure 2. In that case, the server and the program control the TXT remotely using a proprietary TCP-based protocol by fischertechnik.

We explain the different system components in more detail in the next section.

III. System components

A. The TXT hardware

Fischertechnik is a brand of construction toy produced by the German company fischertechnik GmbH. Since its introduction in 1965, the fischertechnik toy line has put a strong focus on the construction of technical and educational models. In 1985, the company presented a computer interface that made it possible to control models from BASIC programs running on PCs and home computers.

The *ROBOTICS TXT Controller*, introduced in 2013, constitutes the fourth generation of fischertechnik computer interfaces. It contains an ARM Cortex-A8 processor, 256MB of RAM, a color touchscreen, and communication interfaces for WiFi, Bluetooth, and USB. It has 8 digital/analog inputs for switches and analog sensors, 8 fast digital-only inputs (typically used for rotary encoders), and 8 PWM outputs that can drive 9V motors, lamps, and solenoid valves produced by fischertechnik and other companies.

DRAFT

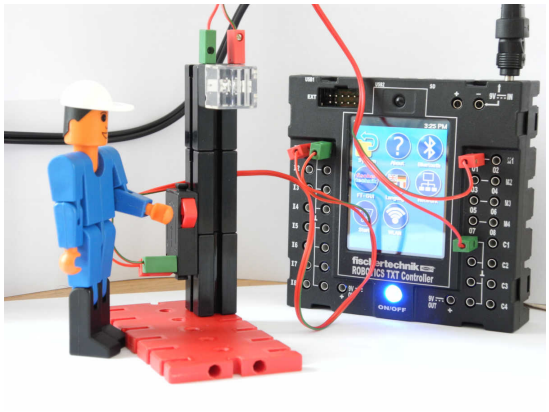

Fig. 3. TXT device with connected switch and lamp

Most customers program the TXT in *Robo Pro*, fischertechnik's visual programming language and integrated development environment. Robo Pro allows running programs in two different modes: In *offline mode*, the Robo Pro program is first compiled on a Windows PC and uploaded to the TXT. Once uploaded, the TXT can be disconnected from the PC and autonomously control physical models. In *online mode*, the device stays connected to the PC, which allows, amongst others, to debug programs and to access the I/O ports from the PC.

The TXT runs the GNU/Linux operating system, so it is not surprising that versed members of the fischertechnik community quickly started to look for ways to directly program the TXT instead of using the Robo Pro environment. Their activities resulted in the development of the Linux-based *community firmware* (CFW) [4] which gives users the capability to install their own software on the device and to download apps from CFW's app store. Most conveniently, CFW can be booted from an SD card and no modification of the shipped firmware is required.

B. The physical world

We use standard components from the fischertechnik toy line for our experiments. A simple test setup is shown in Figure 3, exhibiting a switch (next to the left hand of the toy figure) and a lamp (the transparent box above the switch). The switch and the lamp are connected to an input port, respectively output port, of the TXT.

C. The PLC software

The PLC software consists of two components: (a) The Modbus server that enables the TXT to communicate with the SCADA server, and (b) the real-time PLC program that defines how the TXT should react to input signals and what actions it should perform on the output ports. As explained in Section II, both components can either run on the TXT device itself (Figure 1) or on a dedicated host (Figure 2) that controls the TXT remotely using a proprietary TCP-based protocol by fischertechnik. Obviously, the former configuration corresponds more to the traditional design of commercial PLCs found in industry. However, running the PLC software on a dedicated host, such as a workstation, can be more convenient for educational experiments since it allows to quickly modify the PLC program without the need to upload it to the TXT after each modification.

We have written the Modbus server and the PLC programs in Python, mainly for the reason of convenience and short turnaround time during experiments. We heavily rely on the module *ftrobopy* [5], which provides an easy to use and rather complete Python API for the TXT hardware. We have implemented the server such that PLC programs can access the contents of the Modbus holding registers. A small example using this feature is given in Section III-D.

D. The SCADA server

ScadaBR [3] is a free open-source SCADA server. Deployed as a web application in tomcat, it is responsible for the communication with the PLCs as well as for preparing the HTML-based human machine interface (HMI). Events and measurements can be logged into a SQL database.

ScadaBR includes a web-based tool for the design of graphical HMIs. Figure 4 shows a simple HMI that we have created for the test setup presented in Figure 3. The PLC is executing a small program that turns the light on resp. off when the switch is pressed. The program stores the number of times the switch has been pressed in a Modbus holding register where it can be retrieved by the server. The yellow rectangle in the upper right corner of the image visualizes the state of the lamp (or more correctly: the state of the output port the lamp is attached to). The number in the center of the image shows the value of the previously mentioned holding register.

Fig. 4. HMI for a test setup with a switch and a lamp

IV. Conclusion and Future Work

We have shown how the fischertechnik construction toy [1] can be used to build a SCADA system. It solely relies on free software and on parts from the current fischertechnik toy line. In particular, it uses fischertechnik's TXT controller as PLC.

DRAFT

The software running on the PLC is written in Python. As an alternative, we envisage to port *OpenPLC* [6] to the TXT. OpenPLC is written in C and has the advantage that it is compatible to code generated from Ladder Diagrams (LD) and Structured Text (ST) by the MATIEC compiler [7]. In that way, people familiar with those languages would not need to learn Python to program the PLC.

REFERENCES

[1] fischertechnik. [Online]. Available: https://www.fischertechnik.de/en

[2] (2012) Modbus application protocol specification v1.1b3. [Online]. Available: http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf

[3] ScadaBR. [Online]. Available: http://www.scadabr.com.br/

[4] fischertechnik TXT community firmware. [Online]. Available: http://cfw.ftcommunity.de/ftcommunity-TXT/de/

[5] ftrobopy. [Online]. Available: https://github.com/ftrobopy/ftrobopy

[6] T. Alves. OpenPLC runtime version 3. [Online]. Available: https://github.com/thiagoralves/OpenPLC_v3

[7] M. de Sousa. MATIEC - IEC 61131-3 compiler. [Online]. Available: https://bitbucket.org/mjsousa/matiec

DRAFT