Development of universal control platform

A new advanced fully featured control system for general public

H. Prochazka — March 9, 2015

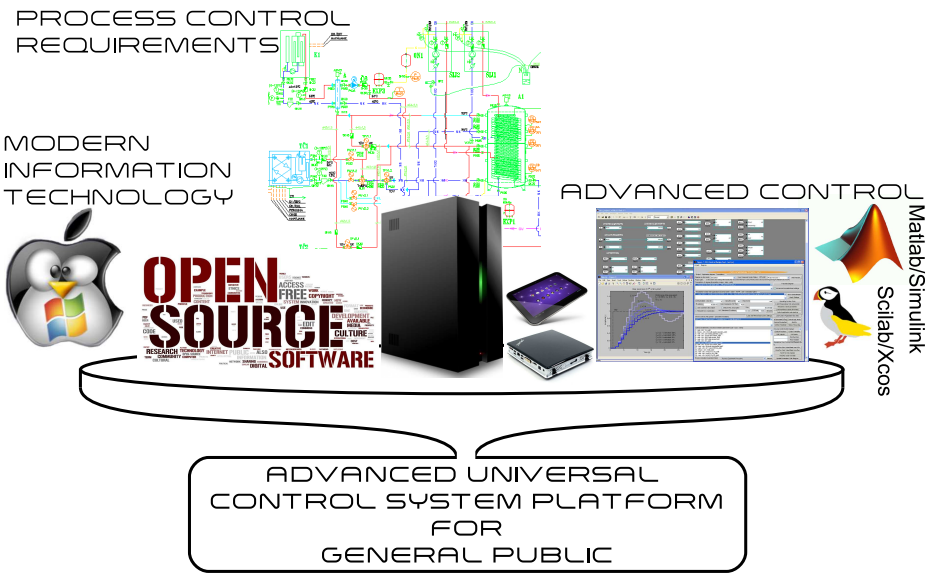

Figure 1. Development of advanced universal control system platform.

A new advanced universal control system platform available to general public is presented in this article. Essential steps of the control platform development are presented as well starting from general user point-of-view specifications and requirements down to detailed technical implementation. The description of the development aim to underline and properly motivate selection and implementation of each element of the platform. General motivation for development of a new control system platform arose from a frustrating experience of coming from

control system research into a standard industrial and building management control environment. Advanced control design techniques and algorithms are not commonly applied in practice, nor control research and development tools such as Matlab or Scilab. Smaller implementation companies and independent professionals carrying a few projects a year, not mentioning students or hobbyists, have an additional permanent issue - cost of corporate-build control platform software and hardware. All that reasons led to decision to develop a new advanced fully-featured but low-cost and open-source based control system platform. A platform that blends necessary industrial requirements, advanced control tools, modern open-source and free IT platforms, low-cost hardware, see Figure 1. A platform available at low cost to any individual or institution developing simple embedded control or highly complex large process control solution.

Note, that by **control system platform** is meant a complete hardware/software setup for building control systems. A setup that includes real-time controllers, supervisory control and data acquisition (SCADA) system, and tools for real-time control design, validation, optimization, and programming. Considered to-be-controlled technology ranges from a small stand-alone mobile device, through a compact technology such as chiller or molding machine up to a large process. Typical large control system consists, from hardware point of view, of sensors and actuators, real-time controllers, SCADA devices, and control network, see Figure 6. Real-time controllers (programmable logic/automation controllers - PLC/PAC, embedded controllers) control technology through sensors and actuators. SCADA devices are generally computers and displays with an appropriate software for programming controllers, recording process data, performing human-machine interface (HMI). Control network interconnects controllers and SCADA devices. Simple control systems are considered as well. They consist of a single small compact real-time controller and sensors and actuators. To check control status or modify control

application, a user or designer must connect to the compact controller temporally or permanently using a computer or other interfacing device.

Motivation for development

At first sight it may seem useless to develop a new control system platform with many existing control solutions available. Industry uses typically corporate process control systems such as Siemens Simatic [1], or Emerson DeltaV [2]. Corporate-built building management systems examples are Honeywell Centraline [3], or Schneider Electric TAC [4]). All these systems typically incorporate a proprietary design tools for programming real-time control and they can be completed with a SCADA system providing HMI and process data recording. SCADA systems are either proprietary to a control system or a general SCADA solution such as corporate-built InTouch by Wonderware or an open-source alternative (see [5]). In parallel with corporate process control system platforms a local-business control platforms arise that try to offer less expensive and more available alternative. Examples of these control platforms are open-source and soft-PLC based Proview [6], or Amit [7] and Domat [8] with its own PLC controllers. These alternative systems are in most aspects similar to corporate-built systems except for their lower price and accessibility to general public. They feature a proprietary design and programming tool and typically they can interact with a general or a proprietary SCADA system. All these process control systems miss or poorly support modern advanced control techniques and design tools used in control system research area. Scientific computational platforms such as commercial Matlab by Mathworks or open-source Scilab by INRIA are not directly included in the proprietary design tools. Robust controllers, optimal controllers, RST controllers, state-space or transfer functions are rarely available. If available, lack of design and optimization tools prevent its effective

use. Dynamic simulation, dynamic model identification, signal processing analysis, controller optimization techniques are never available in design and programming tools.

A few attempts to approach advanced control to process control systems described above has been made. Mathworks has developed Simulink PLC Coder toolbox for Matlab, that generates code for certain types of PLCs out of Simulink diagram (dynamic simulator of Matlab). The code is then loaded into a proprietary design and programming tool, checked and adapted to the particular input/output configuration and finally loaded into a PLC. Others has developed their own Simulink coders, see [9], [10]. Such approach brings some elements of advanced control namely dynamic simulation validation and off-line model-based controller design, if a dynamic process model is available. However, designer needs expensive Matlab/Simulink suite in parallel with another programming tool. Signal processing, model identification, or control design and optimization techniques available in Matlab remain hard to apply, since there is no direct link to process signals.

Another group of control system platforms are say experimental control systems such as National Instruments LabView [11], Matlab Real-time Windows Target [12]. Among all control system platforms, these systems are closest to advanced control, because they are implemented directly within a scientific computational and signal acquisition software. These systems have direct access to process signals and use of advanced control techniques is therefore simpler. Apart from high price which makes these systems unavailable for individuals, schools or small businesses, another drawback are missing important industrial process requirements - networking, process data recording into database system, support of alarms/events, networked visualization. These systems are meant for stand-alone PC-based experimental control in a laboratory.

Finally a large group of control systems are embedded control systems used for stand-alone control applications often without visualization or data recording. Designer programs control algorithms in C/C++ language for a specific real-time controller that is built on-measure for a specific application. Programming tools are generally assembler/C/C++ development environments for programming microprocessor or micro-controller used in the specific real-time controller. These control systems requires programming skills and often hardware knowledge, HMI or networking is poor or missing. To avoid development of a specific real-time hardware platform, designer may use a general real-time control hardware platform such as Arduino [14], or Raspberry Pi. However, these platforms requires generally adjustment according to specific requirements. Again to approach advanced control to these control systems Mathworks has developed Simulink Coder toolbox [13] that generates C code out of Simulink diagram similarly to PLC Coder mentioned above. The coder toolbox generates code for some types of micro-controllers and microprocessors. Use of this code is not straightforward, designer has to adjust the code according to its hardware configuration. Another attempt to approach embedded systems to advanced control is commercial and expensive dSpace development platform which has strong connection to Matlab/Simulink.

Another flaw of actual control system state-of-art is ineffective use of modern information technology (IT). Software tools are licensed with closed source. Most of control systems supports Windows operating system (OS) only, typically not the latest version. In terms of hardware, powerful branded Windows-only computers are usually required for SCADA systems. The reason of this state is that companies tend to protect all software elements of their control platforms and small development teams assigned to maintain, and update the software are not able to keep up the pace with the galloping IT sector. Reaction to this general trend in control systems are

some open-source based SCADA systems [5] or open-source control system Proview [6] based on Linux OS and PC control.

In conclusion, advanced control techniques and tools are in control practice available only in limited form and only to high-budget designers. General public such as small and medium companies, hobbyists, students, professional individuals, small research centers, schools developing real-time control are practically cut off the advanced control. Often even large-business control systems such as food-industry, chemical industry, or power generation avoid use of advanced control for its price and complexity of implementation. This state has its reflection at universities where students are separately learned control theory or advanced control and process control practice. In the later case, student learns how to program a PLC and set up a PI controller, tools of advanced control are omitted. However, advanced control is almost fully available to general public in form of open-source computational platforms such as Scilab/Xcos as well as many open-source codes mostly in form of Matlab or Scilab toolboxes. Toolboxes of control design, optimization, model identification, signal processing that are produced by university research. What is missing is a control system platform applying effectively all these advanced tools. A platform that takes advantage of modern IT and open-source software in order to provide at the same time advanced, flexible, powerful, and cost-optimal solution. Finally the new control platform should consider not only process control systems but also simple embedded control systems. To bring all user/designer comfort and advanced process control features to this often austere control system area.

Development stage 1 - specifications and control system platform structure

Specifications and guidelines

General specifications and objectives are defined below together with basic guidelines of further development. The specifications try to define ’ideal’ advanced control system platform for general public from user and designer point of view. The following desired specifications and objectives on a new advanced control system platform are defined:

1) Graphical, user-friendly control design and programming tool. The tool must include programming using functional diagrams, dynamic simulations, controller tuning and optimization.
2) Process data recording at the rate of at least 1sample/sec into a database system.
3) Modern human-machine interface (HMI) providing process data visualization in a comfortable graphical user interface software. The interface software must be able to display HMI on various types of actual hardware (tablets, phones, touch-screens, computers, mini-PC) over local area network and over Internet.
4) Hard real-time process control with true sampling time of 100msec. True sampling time means that entire control loop reading inputs - control algorithm computation - writing outputs is executed within 100msec. This sampling time is sufficient for a majority of standard processes. For example, in building management system highest samplings are about 1sec, in power plants 100msec, in chemical or food industry 100msec as well. **Hard** real-time means that control algorithm is executed in precise equidistant time intervals. Hard real-time feature is essential for implementation of advanced numerical control

algorithms, since digital (numeric) control theory is based on presumption of equidistant sampling.

5) Support all standard industrial/building management control elements - PID controller, alarm events, comparators, schedules/time programs, analog inputs (0-10V, 0-20mA, NTC, PT1000, Ni1000), analog outputs (0-10V, 0-20mA), digital (binary) inputs (contact sensing), digital outputs (switching contact or voltage output).
6) Support advanced control algorithms - numeric RST controller, transfer functions, state-space controllers, multi-variable controllers, mathematical functions.
7) Support advanced control design techniques - model identification, controller tuning and optimization, hardware-in-the loop validation.
8) Support controller networking.
9) Support one standard widely supported but license-free industrial communication protocol. Such protocol extends control system connectivity towards control and instrumentation devices from other producers.
10) Low price of applied real-time control hardware as well as computer hardware.
11) All software tools free, open-source, and platform independent.

Analyzing specifications and objectives, a decision was made to use existing software tools and platforms and avoid special on-measure programming whenever possible. Use, however, only open-source, reliable, actively developed, widely supported, platform-independent software that supports at least Linux, Windows, Mac OS. Elements of control system implemented with such software benefit a compelling feature. They are constantly developed and improved by a wide community of developers around the globe. This feature is rare in standard process control systems. The companies tend to develop its own, often licensed, proprietary solutions

for each sub-task (visualization, data recording, communication) to keep their know-how. Note, that use of existing software requires high modularity of the new control system platform. It was also decided, that scientific computational platforms for advanced control research and development (such as Matlab/Simulink or Scilab/Xcos) must be integral part of control design and programming tools. This condition is one of essential requirements to guarantee a full support of advanced control.

Projecting specifications and guidelines into control platform structure

Hardware/software structure of control systems is not unique. General concept is that real-time controller/PLC is executing real-time control while a SCADA system running on a computer hardware is performing process data recording, visualization, programming, and other tasks. However, in practice PLC often implements parts of SCADA tasks, such as visualization web-server, some recording service, or multiple-bus integration service. On the other hand, some control systems let computer hardware to perform real-time control. For example soft-PLC based Proview [6] relies entirely on computer hardware. The new control system platform adhere completely to the general concept. Real-time control task and remaining control system tasks are strictly separated. Real-time control is executed by real-time controllers. Remaining tasks are performed by a computer (PC, minPC, tablet, server). The following reasons lead to this strict separation of real-time control. Firstly, heavy requirements imposed on real-time control performance (support of advanced control algorithms, advanced design techniques, low cost of hardware, hard real-time) are more easily fulfilled by a special real-time-only hardware platform. Secondly, real-time control should be absolutely reliable. Adding unnecessary supplementary functionality to controller is bringing an additional source of real-time system errors or conflicts,

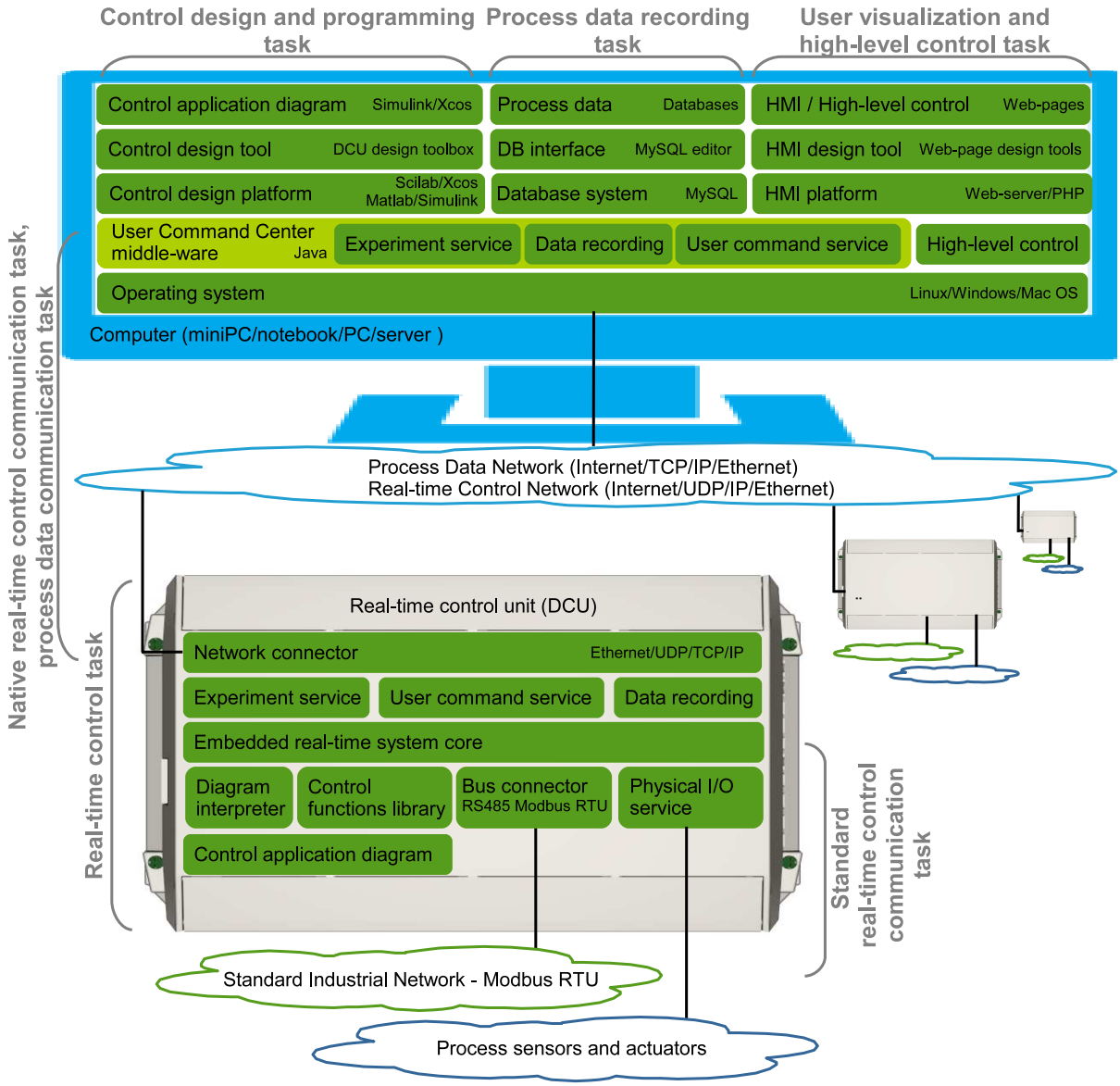

Figure 2. Modular structure of developed control system.

plus lower performance of real-time control and possible necessity of real-time system updates.

The following main control system tasks has been considered:

- Real-time control - reading inputs, executing fast control algorithms, writing outputs, sending process data into database system, communicating with other real-time controllers,

communicating with HMI and design tools.

- Process data recording - recording process data, serving data requests.
- User visualization (HMI) - displaying process data, enable user/designer to act on process control.
- High-level control - reading process data from real-time process control, executing complex control algorithms, sending resulting data into real-time control loops.
- Native real-time control communication - transfer of process variables between real-time control units
- Native process data communication - transfer of process data between real-time control units and remaining control system tasks
- Standard real-time control communication - transfer of process variables between real-time control units and other control devices.
- Control design and programming - design, programming, tuning, and optimization of real-time control algorithms.

A specific hardware/software technology is assigned to each task based on specifications and guidelines. Figure 2 shows proposed hardware/software structure of the new control system platform together with selected technologies implementing each task. From the hardware point of view, the system is composed of the following components :

- Real-time control unit - executing real-time control task.
- Computer - providing process data recording, high-level control strategy, HMI, and control design.
- Process Data/Control Network - standard local area network (LAN) as well as wide area

network (WAN) may be used. The real-time control unit has Ethernet 10/100Mb connector integrated. Data from units to computer are transferred using TCP/IP network protocol, process data are transferred between real-time controllers using simpler and faster UDP/IP streaming protocol.

- Standard Industrial Network - control and instrumentation devices from other producers may be connected using this network. Modbus RTU over RS485 serial line has been selected. The protocol is widely supported and under no license.

Concerning software implementation, the required modularity and extensive use of existing software technologies lead to multi-layer software structure inside computer. Real-time control is implemented separately in a newly developed hardware platform powered by a new embedded real-time system. Use of any existing general hardware platform was rejected because of complex specifications and requirements on real-time control task. The newly developed real-time control platform is called Dynamic Control Unit (DCU).

Designer/user specifications on the new control system platform defined previously are satisfied as follows. Specification 1) on design and programming tool is implemented using Matlab/Scilab scientific computational platforms and its toolboxes as suggested by guidelines. Process data recording specification 2) is expected to be fulfilled with MySQL database system. HMI specification 3) is implemented as Apache/PHP web-server serving visualization web-pages. Hard real-time specification 4) is implemented in a separate real-time control platform. Specification 5) and 6) on support of standard and advanced control elements is guaranteed by a complete real-time control library containing all required functions. Controller networking specification 8) is provided by Ethernet/UDP/TCP/IP protocols. The reason is

excellent performance, flexibility, low basic price of the technology, no licensing required, security techniques available, huge community developing the technology. No standard industrial network protocol (CAN, Profibus, Fieldbus) can match these features. Standard control network specification 9) is satisfied by support of Modbus RTU protocol. Specifications 10) and 11) on system cost are objectives aimed by careful selection of used hardware/software platforms.

The specification 7) on support of advanced control design techniques needs more detailed investigation. To support advanced control in control practice an instrument was developed providing direct link between advanced control design techniques and real-time process. This instrument is process experiment procedure. Sending an external excitation signal into process inputs / outputs and collecting process dynamic responses is a procedure required in many advanced control design techniques [21], [20], [19]. Process identification, controller optimization, or controller reduction methods require this type of direct process experiment. In the experiment, an excitation signal is sent into process with control loops detached (open-loop experiment) or not detached (closed-loop experiment). The model-based advanced control design methods do not require direct process experiment (for example robust control [16], multivariable control [17]). However they require a dynamic process model which is typically obtained by process identification [20] that needs process experiment. Process experiment is in the core of each control design technique, because to design a control, process dynamics must be known first. Example of process experiment is shown in Figure 3. Process experiment procedure was implemented into the control system platform by the means of experiment service. The service is integrated into DCU real-time system on one side, and into Matlab/Scilab design toolbox, and User Command Center (UCC) middle-ware on the other side.

Hardware structure of the real-time control was questioned at the beginning of the development. Classical PLC [1], [3], [4] controller usually uses centralized structure with one computational unit and a set of input/output (I/O) modules. One PLC controller then serves from tens up to thousands of physical I/Os. Opposite approach was selected in our case. DCU is a compact controller that integrates computational unit and certain number of physical I/Os. Each DCU compact controller then controls a part of the process (sub-process). Selecting this type of distributed control have the following reasons. The centralized structure implementing the specifications above would require powerful and expensive hardware. The centralized structure is not suitable for simple stand-alone control systems, that are considered as well. Additionally, note that the centralized structure often does not reflect real nature of process and process control. Even a complex process such as power plant is not controlled in practice as a homogeneous super-process but rather as a set of sub-processes each having its particularities. Finally, the distributed control systems are more reliable. Malfunction of one processor inside one controller does not cause fail of the entire control system. On the other hand, the distributed control systems rely more on control network than centralized systems. The reason is that not all needed process values are always present inside compact controllers. The missing process values must be supplied by another controllers through control network. To limit somewhat the dependence of control system on control network, the large version of developed compact controller DCU may have up to 56 inputs/outputs. As a result, in most cases only one DCU is needed for control of one sub-process.

Development stage 2 - real-time control implementation

Real-time system structure

Real-time system structure is shown in Figure 2 inside DCU controller. A set of specific-purpose services surround system core. Ethernet/UDP/TCP/IP service provides network interface for native process data communication and real-time control communication tasks. RS485 Modbus RTU service is an interface for standard real-time control communication. Process data recording service is permanently sending actual process data values from DCU into the process data database. Experiment service is handling real-time process experiments. User command service is interacting with HMI by providing read/write access to real-time process data. Control application interpreter is executor of loaded control application diagram. The service reads an uploaded diagram and executes the diagram using real-time control function library. Physical I/O service is reading/writing physical inputs and outputs with specified sampling time.

Hardware

In development of DCU real-time control hardware platform, emphasis was on simplicity and high performance/price ratio. Reduced instruction set (RISC) ARM-core 32-bit micro-controller with integrated RAM/FLASH memories was used to limit hardware complexity and price and to obtain maximum performance. 12-bit analog/digital and digital/analog convertors are used for analog inputs and analog outputs. 16-bit expanders are integrated for digital (binary) inputs and outputs. Real-time timer has been included with independent battery backup to keep track of time and date even in case of a long-term (up to 10 years) shutdown. A removable

EEPROM memory for control application backup has been integrated and an automatic backup service has been implemented into real-time system. In case of control unit malfunction, it is sufficient to replace unit and place into the new control unit backup EEPROM from malfunctioned unit. Application is automatically uploaded from EEPROM and so there is no need for re-programming of the new unit.

Based on the developed real-time control hardware platform, two types of compact DCU controllers have actually been developed. First is universal and modular compact DCU56IO controller with up to 56 I/Os and with selectable I/O electrical conditioning. Designer may select type of 16 analog inputs (0-10V, 0-20mA, thermistor), 8 analog outputs (0-10V, 0-20mA), and 32 digital outputs/inputs (24VDC, contact relay). Reduced-size DCU28IO controller is embedded-application oriented. Electrical conditioning of 8 analog inputs, 4 analog outputs, 8 digital inputs, and 8 digital outputs is fixed.

Control application implementation

Implementation of control application aims full support of advanced control algorithms as well as standard industrial functions. Real-time system contains diagram interpreter for interpreting and executing uploaded control application diagram. Note that control application inside control unit is not a bunch of raw binary code compiled out of a control loop diagram but directly diagram built and validated in a dynamic simulator (Simulink/Xcos). This concept brought necessity to include into real-time system an exhaustive library of control functions (blocks). The library implements all control functions necessary for industrial control loops as well as for advanced control algorithms. Control functions for industrial control loops

include: PID controller, schedules (time programs), control variable alarm state, counter, binary operations, conditional multiple-input switch and many others. Control functions for construction of advanced control algorithms include:

- Linear time-invariant discrete-time transfer function of n-th order with no limit on n.
- Multiple-input multiple-output state space system, with no limit on number of inputs/outputs/states.
- Signal processing operations: numerical integral, derivation, delay, look-up table, memory
- Basic mathematical functions.
- RST controller with integrated anti-windup functionality and transition procedure for controllers switching. No limits are imposed on sizes of R, S, or T polynomials.

Note that many advanced control algorithms such as robust, optimal controllers [16], or multi-variable controllers [17] can all be built using these control functions. All control functions are evaluated with 64-bit floating point precision. Real parameters of control functions has the same precision. Integer values and parameters have 64-bit signed integer format. For binary values, binary evaluation is used. This high precision makes implementation of advanced control algorithms straightforward. Conversion to 32-bit floating point or fixed-point precision is not needed, since scientific computational platforms Matlab/Simulink and Scilab/Xcos works both by default in 64-bit precision. Control application diagrams built in its dynamic simulators may be directly uploaded into DCU.

Up to 32 different sampling times may be defined, each sampling time can be attributed to any control function and any physical input/output. Defined sampling times may be distributed up to three separate computational interrupts (threads) - fast interrupt, medium interrupt, slow

interrupt. For example, complex computations that do no require fast sampling and that could slow remaining control loops may be placed into medium or slow interrupt.

Experiment service

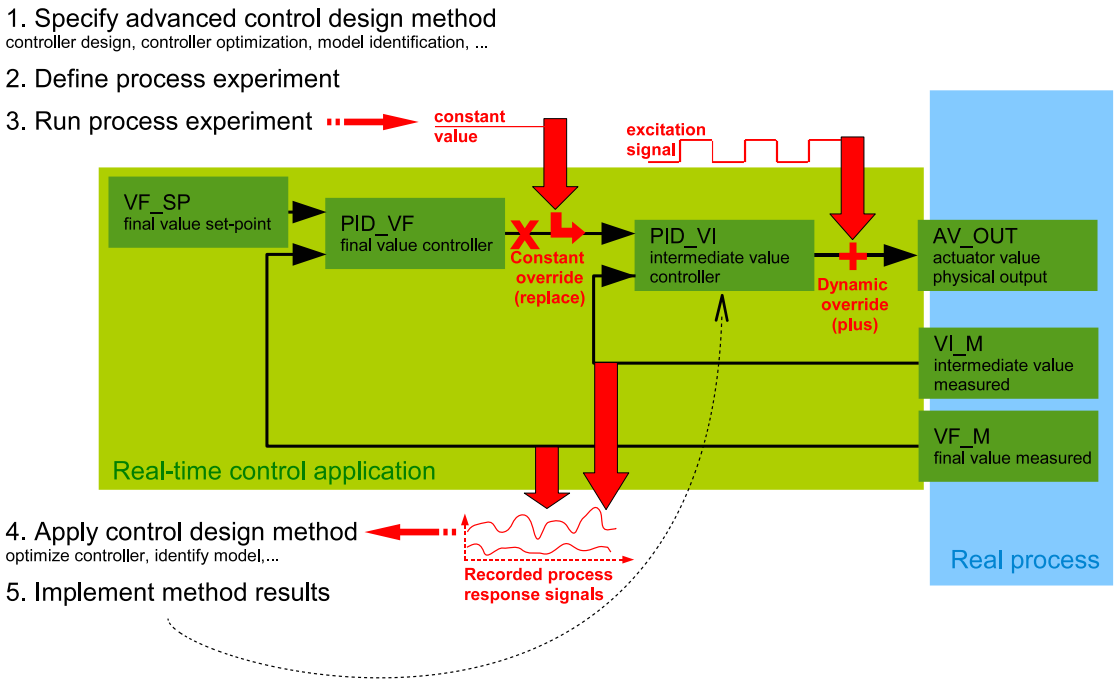

Figure 3. Example of closed-loop process experiment.

In the section on projecting specifications into system structure, process experiment procedure has been introduced as a corner stone of advanced control design methods. Process experiment procedure is launched by designer from control design tool, and it is executed by UCC middle-ware in cooperation with experiment service of real-time system. Experiment service particular task is to receive excitation signals from design tool and apply this signal into specified process variables. This task executed by experiment service was called dynamic override because of its relation to override operation. Classical override or forcing (or switch to manual) value is

a standard process control operation. User (operator) replace automatic process value computed by control algorithm with a user-selected constant value. This concept has been extended in two directions. Firstly, user may not only replace but also add or multiply selected constant value with automatic value. Secondly, user may impose not only constant value but also a signal (a series of values-samples). The second option was called dynamic override and it is used to apply excitation signals into process according to requirements of process experiment procedure.

Notion of process experiment procedure has been generalized for use inside control design tool. A process experiment is defined by 3 groups of process variables: variables in constant override, variables in dynamic override, variables to be recorded during experiment. Constant overrides are typically used to open control loop, or cut parts of control that would cause an undesirable disturbance. Dynamic overrides define external excitation signals of the experiment. Recorded variables represent variables used to collect process response. Process experiment example is shown in Figure 3.

The process experiment can be defined as described above directly by user/designer. A special graphical interface is provided inside design tool. Launching process experiment defined directly by user in this interface will produce a set of recorded signals that represent process response to the experiment. The signals are available in Matlab/Scilab workspace for further processing. Another way of using process experiment implementation is to define and use the experiment procedure directly inside an advanced design technique programmed in Matlab/Scilab. A developer/researcher may include easily its own advanced control design technique into DCU design tool and integrate process experiment into this technique. Example is provided in DCU design tool with Zigler-Nichols PID tuning method. This method needs to apply a step into

open-loop process input and based on the process response the method computes coefficients of PID controller. The method is thanks to process experiment completely automatic. User only chooses PID controller to tune, selects amplitude and minimal length of step, and launch the tuning procedure.

Control modes

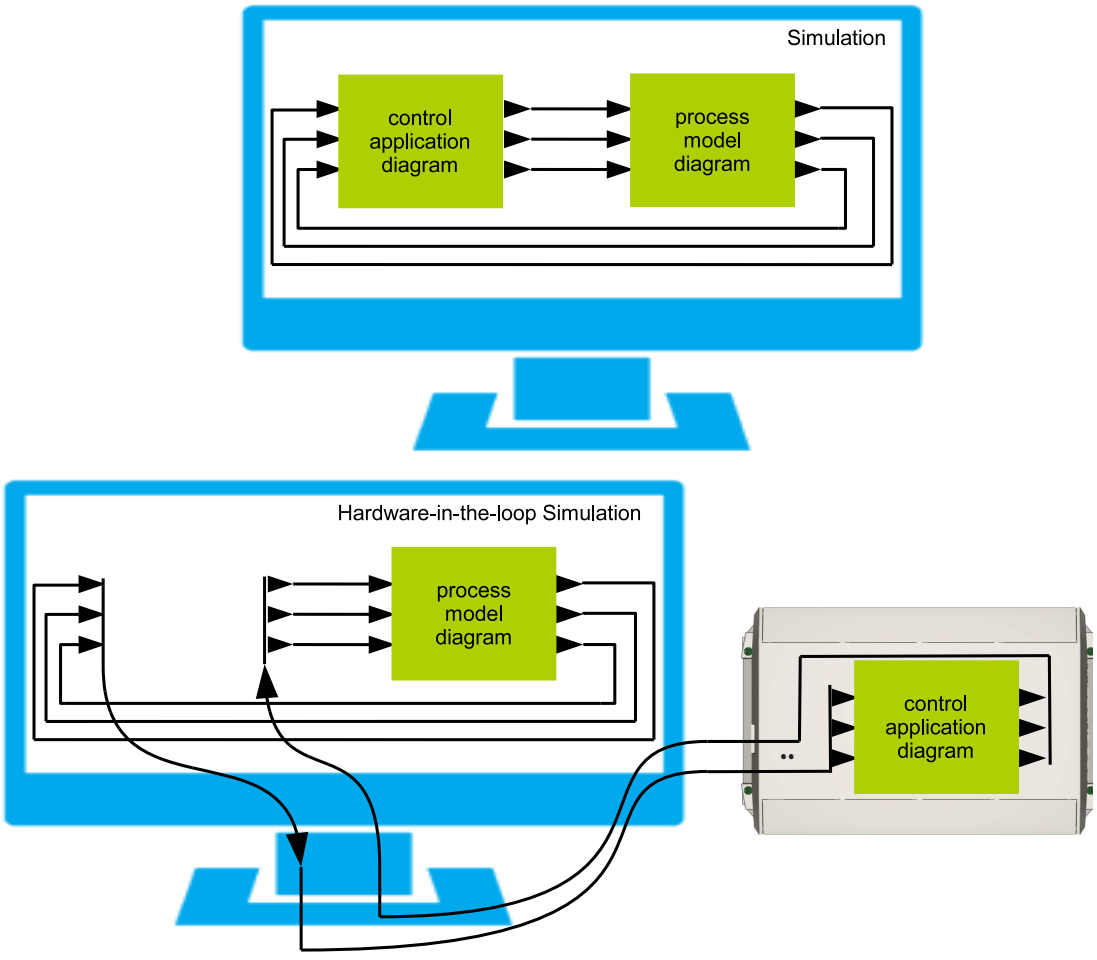

Figure 4. Control application validation procedures.

Actual control mode of DCU controller determines control application behavior with

respect to physical I/Os. Four control modes are considered. Two classical control modes are Mode 1 - on-chip real-time control and Mode 2 - control application execution stopped. Two additional special control modes has been implemented to enhance advanced control support: Mode 3 - on-chip simulation control and Mode 4 - on-PC real-time control. In Mode 1, real-time control application is executed inside DCU controller and its physical I/Os are processed. Application is not executed at all in case of Mode 2, physical I/Os remains in last state. Mode 3 - on-chip simulation control provides support for hardware-in-the-loop (HIL) like validation technique. Control application is executed inside DCU controller, but instead of physical I/Os dynamic model simulated I/Os are processed. Last Mode 4 is an experimental mode of real-time process control. Application is not executed inside DCU controller but inside the dynamic simulator of designer's computer, however real physical I/Os are affected. In other words, modes 1 and 4 are real-time process control modes with control application running once inside DCU and once inside designer's computer, see Figure 5. Mode 2 is stopped application and mode 4 serves for application validation.

Complex validation procedure consists of two steps. First, the developed control algorithm is tested in dynamic simulation using a process model. Next, the control algorithm is implemented into a real-time controller and its behavior is again validated using the process model and dynamic simulation (control mode 3), see Figure 4. This advanced technique for control application validation is used in case of expensive and complex processes. The entire complex validation procedure is supported in DCU design tool. First validation step is pure simulation. The control application is designed (drawn) in Simulink or Xcos dynamic simulators. To simulated process control, process model may be easily included/built inside control application diagram and connected to control loops. Next, with some pre-defined scenarios the simulation may be started.

To execute HIL validation, user must translate control application diagram and load the translated diagram into control unit. Next, DCU design tool is switched into on-chip simulation mode and simulation of the diagram prepared in the previous step may be started. DCU design tool automatically switches DCU into on-chip simulation control mode and starts sending process model outputs into DCU. Sent values are used inside control unit instead of real physical inputs. Controller evaluates control loops with received input values and sends computed output values into process model instead of writing them to physical outputs. DCU controller is controlling process model instead of real process.

Concerning control mode 4, real process is controlled, but control application is not running inside DCU controller but inside computer within simulation platform. Controller is only receiving computed values of outputs to be applied and sends into computer measured physical inputs. This type of control requires longer sampling times. In one sample, output values must be computed inside simulation platform, send into the controller and new measured inputs have to be received. Advantage of this mode is that control algorithm is evaluated inside simulation platform. Any block or function built or programmed inside simulation platform may be used. Designer is not limited by real-time control library defined for DCU controller. Only physical input/output blocks need to be used to send/receive physical output/input values into/from DCU controller.

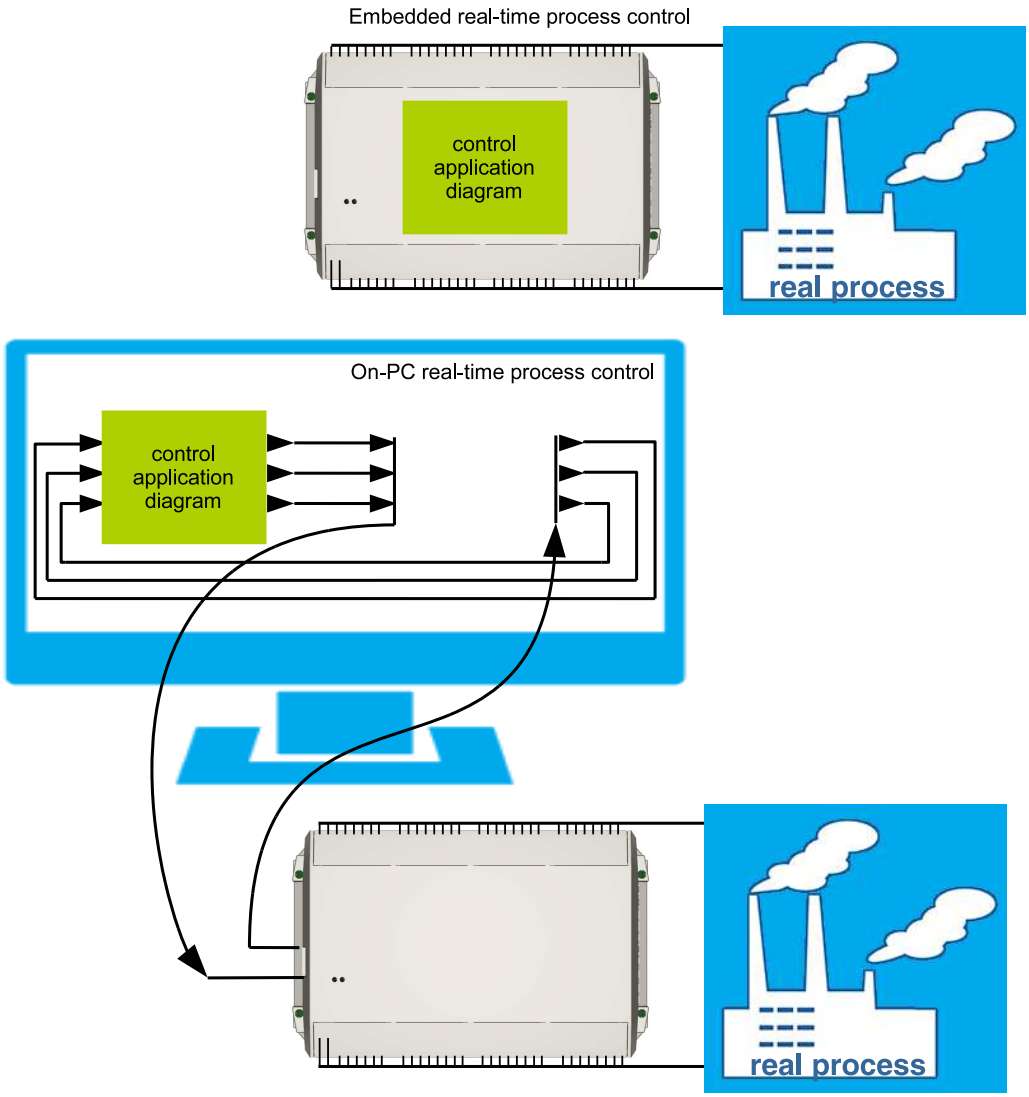

Figure 5. Control system real-time control modes.

Computer software development

Control design, data recording, visualization

Control design, data recording, and HMI software have all multi-layer structure, see Figure 2. Bottom layer, called also platform, is a foundation on top of which sits user/designer

interface application and final product - control application, process data, and web pages. In case of control application design, the platform is Matlab/Simulink or Scilab/Xcos and user-interface is an on-mesure built design toolbox called DCU design tool. For HMI the selected but not imposed platform is PHP/Apache web-server. Process data recording is implemented using MySQL database system. User/designer may choose among many user-interfaces for MySQL database as well as thousands of web-page design tools.

The reasons for employing Scilab and Matlab for control design task has already been discussed. The major role that plays these scientific platforms in advanced control design research and development lead to decision to use this platforms as primary design and programming software. Selection of PHP/Apache web-server for HMI and MySQL database system for data recording was guided by the fact that widely used software suit LAMP / WAMP / MAMP stack (Linux/Windows/Mac-Apache-MySQL-PHP) contains all needed platforms in one installation package. The stack is widely supported and open-source based. Many companies and programmer communities are participating on development of different variants of the stack. At the same time, the platforms are confirmed, reliable, and efficient software solutions. For example, MySQL is used by huge web-services such as Facebook, YouTube or Wikipedia that handles millions of queries per day. The Apache web server is the most popular web-server in the world. No company-developed data recording or HMI solution can mach this performance or keep up the pace in constant development.

Middle-ware integration service

Scientific computational platform (Matlab/Scilab), database system (MySQL), web server (Apache/PHP), and real-time control system are independent software platforms. A software service providing interface between all these platforms is needed. In the modular structure diagram in Figure 2 the service is entitled User Command Center (UCC). The module is on-measure software service programmed in Java. MySQL database is used by UCC as an application interface data exchange tool. Java and MySQL database support multiple operating systems, so UCC service supports Windows, Linux, and Mac OS operating systems.

A general file format for control application diagram has been defined. The file has a form of XML file containing only necessary information about control application diagram. A control application diagram may actually be developed in Matlab/Simulink (.mdl file format) or in Scilab/Xcos (.xcos file format) To upload a control application diagram into real-time control unit, the diagram is first translated by UCC into the general diagram format (file extension .capp). Next, the general-format diagram can be uploaded into real-time unit.

High-level control

High-level control algorithms are not considered to implement fast hard real-time control loops. High-level control is intended to overlook entire process control and based on some analysis/optimization act on it. Typical example is an algorithm for optimizing process control set-points based on some process efficiency analysis, fault-detection algorithms, or complex optimization-based control. In the newly developed control system platform, high-level control

is executed inside computer. No special platform is deployed to perform high-level control. Two different ways of implementation are available. Firstly, designer may program high-level control directly inside HMI platform. Actually implemented PHP/Apache platform disposes with constantly-growing set of libraries including linear algebra library. Second possibility is to program a local application in a selected language (C/C++, Java, Python, .Net) that has a connector to MySQL database. A special real-time service is provided inside UCC to transfer process values between real-time controller and this local application. The service creates and updates a real-time database table with selected actual process data. Local application of high-level control may at any time access this table to send needed value into DCU or to read actual values from DCU.

Openness

All information on control system platform structure, data formats, software structures are available to general public. Use of Matlab/Scilab for control design and MySQL for data recording and Apache/PHP for visualization is not mandatory. Alternative platforms may be applied as well by general public. For example, LabView can be implemented as new control design platform. However, the implementation requires two non-trivial elements. A design tool has to be programmed inside LabView platform similar to DCU design tool. Next, a translator of LabView diagrams has to be programmed inside UCC.

Instead of MySQL database system an alternative database systems may be implemented. This time, only slight modifications are required inside UCC. The new database system has to have Java connector supporting standard SQL language to be easily implemented into UCC. All

widely used databases fulfill this condition.

Simplest module replacement is for HMI. The platform PHP/Apache can be replaced without any modifications in control system platform. Alternative web-servers and high-level languages that are able to read MySQL database may be used (Microsoft IIS, NGINX, Perl, Python, Java, C/C++,...). A web-based visualization as well as a local application visualization may be created.

New control system platform in practice

Application example

The development of the new control system platform has been carried out during the last 4 years. The platform is actually at the final stage of development - tests by implementation on real processes. In this stage, possible hidden errors and conflicts, and improving basic user/designer comfort are aimed. Multiple real control systems has already been implemented including DCU test bed, simple one-controller systems, up to complex process control. Complex control systems combining multiple controllers networking, SCADA system, and other producer's control devices have been the last and the most demanding test implementations. One example of such complex control system implementation is shown in Figure 6. Cooling/heating/ventilation technology of a large printed board circuit (PCB) manufacturing site is controlled, monitored, and visualized by four DCU units and one micro-server. Part of the system (ventilation units) is controlled by other producer controllers, but this sub-system is integrated into SCADA system through one DCU using Modbus RTU protocol.

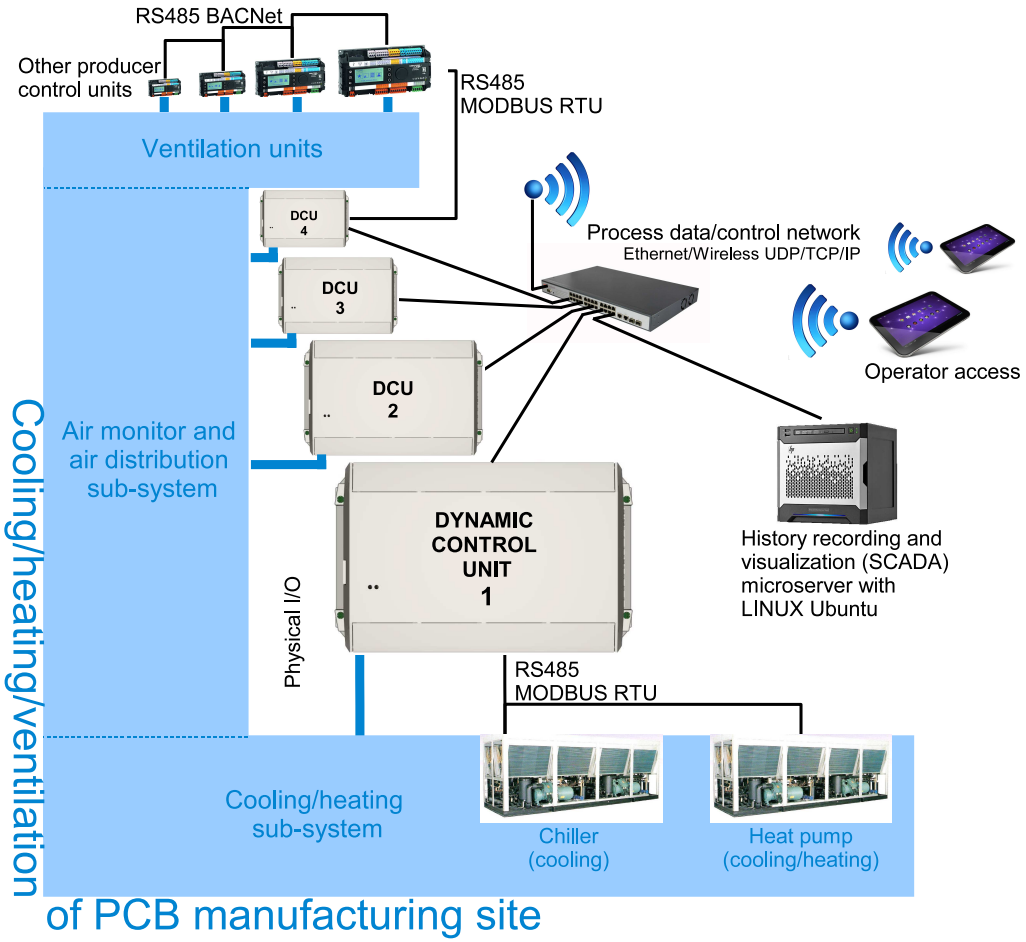

Figure 6. Example of DCU control system.

Performance limits

Performance tests showed the following limits of DCU control system. Serving all inputs and outputs requires 1.2msec. Limit is imposed mainly by analog inputs, which has slowest communication rate. A standard control application running in one thread and serving all inputs/outputs required about 11msec of computational time. Minimal true sampling time for a larger application running in one thread and serving all 56 inputs and outputs is about 13msec. True sampling time for smaller applications serving only some inputs / outputs starts from 1msec.

If unsatisfactory, control application execution speed may be optimized by distributing control loops into up to 3 computational threads. Loops requiring fast sampling may be placed into fast interrupt thread and remaining loops into medium or slow interrupt thread. Note, that defined specification was at least 100msec true sampling time.

Data recording from one DCU controller on low-cost mini-PC (Intel NUC Kit DCCP847DYE) was proved to work smoothly without any data loss at the rate of 160000 records/min. All 56 inputs/outputs of one DCU may be recorded with sampling of 25msec without loss of data. Note, that specification for recording was 1sec.

Scilab Xcos simulation platform (version 5.4.1) proved to be slow for simulation of large control applications on a standard personal computer. In case of larger diagrams, the recommendation is to execute simulation of each control loop in a separate diagram. New versions of Scilab and more powerful computers can certainly limit this drawback. Commercial alternative Matlab/Simulink does not have this problem.

Use of low-cost mini-PC (Intel NUC Kit DCCP847DYE) as a data recording and visualization server for handling two DCU controllers with recording sampling rate 1sec proved to work without any problem. Linux Ubuntu 12.04.3 operating system was installed together with LAMP stack. Response of visualization system to operator action is about 3-4seconds. Response time may be easily improved by using more powerful hardware.

Control application update inside control unit such as changing control function parameters or control variable override requires to interrupt control application execution. Time of interruption is about 60msec. Therefore, application update may have certain disturbing effects

(delay in evaluation) on control algorithms with sampling times under 70msec.

Conclusion

A new control system platform for general public and its development has been presented. The main objective of the development was to built a complete advanced, but low-cost control system platform available to anyone. The developed control system platform aims small mobile stand-alone control solutions as well as complex process control systems. For real-time control task a new hardware platform - DCU controller - has been developed to satisfy imposed high-performance / low-cost specifications. All remaining control system tasks are entrusted to computer hardware. The main innovation resides in strong support of advanced control algorithms and design techniques. Real-time control programming is performed directly in most advanced scientific computational software for control design (Scilab/Xcos, Matlab/Simulink). Direct link is established between the scientific software and real-time control data. Advanced control functions are integrated into real-time control. Integration utilities for new advanced control design techniques are available for control system researcher.

At the same time, the control system platform is trying to satisfy most of industrial specifications and requirements on control and SCADA system. The developed control platform is not a unique compact software suit typical for corporate control systems. Modular structure is used instead with existing software solutions used for each particular control system task. This may be seen by some designers as a disadvantage because of certain ’discomfort’ caused by passing from one application to another. However compact software suit needs to be constantly maintained and updated by a team of dedicated developers. Modular structure, on the other hand,

is self-maintained, permanently updated, cost-effective, highly flexible, and reliable provided open-source, confirmed, widely supported software is chosen. Modular structure required an integration middle-ware application to be developed. A positive consequence of multi-application structure is that a designer/user learns and uses modern information technology (MySQL, web-page design, Matlab/Scilab) and not some narrowly-specialized corporate-built software.

The developed control system platform is actually tested on real-life applications. Multiple simple as well as complex industrial/building management control system implementations are helping to uncover many errors and inefficiencies. These implementations are helpful to test networking, interoperability, designer comfort, visualization, and other SCADA elements. However, from the advanced control point of view, they are trivial. That is why universities are addressed to participate on tests of advanced control capabilities. These collaborations are bringing a mutual benefit. Firstly, university researchers obtain a tool for direct implementation and validation of their model identification and control design methods in practice. Secondly, the control system platform has significant educational value as it presents entire control design field within one frame. Starting with control theory and dynamic simulations a student can pass fluently to real-time control programming while staying in Matlab/Simulink or Scilab/Xcos and finish by setting up a SCADA system. Moreover, a student may work on the control system platform at home, and use it for its own purposes. No licenses are required if open-source computational software Scilab/Xcos is used for control design.

References

[1] “SIMATIC Controllers Overview” Available online: http://www.automation.siemens.com/salesmaterial-as/brochure/en/brochure_simatic-controller_overview_en.pdf

[2] “DeltaV Distributed Control System” Available online: http://www2.emersonprocess.com/en-US/brands/deltav

[3] “Centraline Building Management System” Available online: https://www.centraline.com/

[4] “TAC I/A Series - Enterprise-wide control solutions” Available online: http://www.schneider-electric.com/products/ww/en/1200-building-management-system/1210-building-management-systems/7756-tac-i-a-series/

[5] “Linux SCADA List” Available online: http://linuxscada.info/

[6] “Proview Open Source Control System” Available online: http://www.proview.se/

[7] “Amit Control System” Available online: http://www.amit.cz/

[8] “Domat Control System” Available online: http://domat-int.com/en/

[9] “BaR Automation Studio Target for Simulink” Available online: http://www.br-automation.com/en/products/software/automation-studio-target-for-simulink/

[10] “Rex Controls” Available online: http://www.rexcontrols.com/rex

[11] “LabView System Design Software” Available online: http://www.ni.com/labview/

[12] “Real-Time Windows Target” Available online: http://www.mathworks.com/products/rtwt/

[13] “Simulink Coder” Available online: http://www.mathworks.com/products/simulink-coder/

[14] “Arduino” Available online: http://www.arduino.cc/

[15] H. Prochazka,J.M. Legros, “Iterative Feedback Tuning Applied to Power Plant Control.”, International Conference Technical Computing 2009, Prague, Czech Republic.

[16] K. Zhou, “Essentials of Robust Control.”, Prentice Hall, New Jersey, 1998.

[17] S. Skogestad, I. Postlethaite, “Multivariable Feedback Control Analysis and Design.”, John Wiley and Sons, 2005.

[18] E.F. Camacho, C. Bordons, “Model Predictive Control.”, 2nd edition, Springer-Verlag, 2004.

[19] I.D. Landau, A. Karimi, L. Miskovic, H. Prochazka, “Design and optimization of restricted complexity controllers benchmark.” European Journal of Control, 9(1), 2003.

[20] L. Ljung, “System Identification Theory for the User”, 2nd edition, Prentice Hall, New Jersey, 1999.

[21] H. Prochazka, M. Gevers,B.D.O. Anderson,C. Ferrera. “Iterative Feedback Tuning for robust controller design and optimization”, CDC-ECC Conference 2005, Seville, Spain.

[22] C.W. Taft,H. Prochazka,D. Faille,A. Hussey,J.N. Sorge Automatic Simultaneous Tuning of Multiple PID Control Loops, 54th ISA POWID Symposium 2011, Charlotte, USA.

Author Information

H. Prochazka is C.E.O. and research and development engineer in Prosystemy, Slovak Republic. He received his PhD at Institute National Polytechnique de Grenoble, France in the domain of multivariable controller optimization. He worked as a post-doc at Universite Lovain la Neuve on optimization of industrial controllers for power plants. He is co-founder of a small company Prosystemy, s.r.o. providing consultancy and services in control system design, optimization, and implementation. Company experience cover design and implementation of building management systems, research and development in controller optimization for power plants, thermodynamic modeling and simulation.