SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
FACULTY OF CHEMICAL AND FOOD TECHNOLOGY

Reference number: FCHPT-19990-21598

MULTIPURPOSE AND LOW-COST ARCHITECTURES FOR AUTOMATIC CONTROL REMOTE LABORATORIES

DISSERTATION THESIS

Bratislava, 2014 **Ing. Martin Kalúz**

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
FACULTY OF CHEMICAL AND FOOD TECHNOLOGY

Reference number: FCHPT-19990-21598

MULTIPURPOSE AND LOW-COST ARCHITECTURES FOR AUTOMATIC CONTROL REMOTE LABORATORIES

DISSERTATION THESIS

Study program: Process Control
Study field number: 2621
Study field: 5.2.14 Automation
Workplace: Department of Information Engineering and Process Control
Supervisor: Prof. Ing. Miroslav Fikar, DrSc.
Co-supervisor: Ing. Ľuboš Čirka, PhD.

Bratislava, 2014 **Ing. Martin Kalúz**

Slovak University of Technology in Bratislava
Institute of Information Engineering, Automation and Mathematics

Faculty of Chemical and Food Technology
Academic year: 2014/2015
Reg. No.: FCHPT-19990-21598

STU FCHPT

DISSERTATION THESIS TOPIC

Student:	**Ing. Martin Kalúz**
Student's ID:	21598
Study programme:	Process Control
Study field:	5.2.14 automation
Thesis supervisor:	prof. Ing. Miroslav Fikar, DrSc.

Topic: **Multipurpose and Low-Cost Architectures for Automatic Control Remote Laboratories**

Assignment procedure from:	02. 09. 2010
Date of thesis submission:	02. 09. 2014

L. S.

Ing. Martin Kalúz
Solver

prof. Ing. Miroslav Fikar, DrSc.
Head of department

prof. Ing. Miroslav Fikar, DrSc.
Study programme supervisor

Acknowledgements

There is no individual success without the support from family, friends, and colleagues. This way I want to express my infinite gratitude to my supervisor professor Miroslav Fikar who gave me a lot of valuable advice and research freedom during my studies and work, to my fellow colleague, co-supervisor and office mate Ľuboš Čirka for the rich discussions, guidance, and friendly environment during last four years. My big thanks belongs to all people at Institute of Information Engineering, Automation and Mathematics, especially the professors Monika Bakošová, Ján Dvoran, Michal Kvasnica, and Ján Mikleš for their advice and help every time I needed, to people from administration Monika Mojžišová, Katarína Macušková and Andrea Kalmárová for their everyday helpfulness and patience with wrong filled forms and papers, to our technical guru Stanislav Vagač for his willingness and assistance in technical matters, and to my fellow colleagues Jakub Bendžala, Juraj Holaza, Martin Jelemenský, Mária Karšaiová, Martin Klaučo, Jana Kmeťová, Juraj Oravec, Radoslav Paulen, Ivana Rauová, Ayush Sharma, Alexander Szűcs, Juraj Števek, Balint Takács, Richard Valo, Anna Vasičkaninová, and Jana Zavacká. I am frankly grateful for the opportunity to work with such bunch of great people.

My special thanks belong to people I met during my stay at University of Deusto in Bilbao, namely to Dr. Javier García-Zubía and Pablo Orduña who adopted me to their research team, showed me an adult world of remote laboratories and gave me a lot of precious advice for my future work. My gratitude also belongs to people from WebLab-Deusto research group, which I have had the pleasure to know, namely Ignacio Angulo, Olga Dziabenko, Unai Hernández, Luis Rodriguez-Gil, Gustavo Martín Vela, Iñigo Iturrate, and other.

A big thanks is dedicated to my nearest friends who helped me to keep my sanity every time the worries overgrown my head. When I look back, I realize that the biggest force pushing me forward is the love and support from my family, for that, I will be forever grateful to them.

Martin Kalúz
Bratislava, 2014

Abstract

This thesis deals with the problems of design and realization of remote laboratories, while it is particularly focused on multipurposivness and reduction of development costs for such systems. Remote laboratory is an instrumentation apparatus with space-distribution of its functional parts and it allows to control experimental devices remotely, over the computer network. Such systems apply mostly in the area of technical education and they bring the innovative approach to practical exercising.

The work is aimed on such concept of remote laboratory architectures, which allows to separate the phase of actual technological development from the process of implementation. This leads to significant savings in development time and reduces the complexity of laboratory creation process. To fulfill these requirements, the architecture has to be considered as universal since the early development phase. The universality must apply for both, the hardware resources as well as the software. Physical parts of architecture must be able to connect and serve a wide class of different experimental devices and program parts must be designed in order to mediate the remote control of such devices.

For these purposes, two different architectures have been developed in this work. First one is the multipurpose hardware and software architecture, based on industrial-aimed control and communication devices, such as programmable logic controllers and industrial network routers. The main benefit of this architecture is that it provides a ready-made hardware and also partially software solution and it is capable to serve various types of technological processes. The second architecture uses low-cost parts, such as single-board computers and programmable micro-controllers. The purpose of this approach is to demonstrate the potential of cheap components for the use in remote laboratories and process control. The practical contribution of this work is the implementation of several different process control laboratories on both types of architecture. Another topic discussed in this work are the methods of laboratory management as well as their publication for educational purposes.

Keywords: Remote laboratories, Multipurpose architecture, Low-cost architecture, Industrial control systems, Micro-controllers, Process control education.

Abstrakt

Táto práca sa zaoberá problematikou návrhu a realizácie systémov vzdialených laboratórií so zameraním sa na viacúčelovosť a znižovanie nákladov ich vývoja. Vzdialené laboratórium je inštrumentálny aparát s priestorovo rozloženými funkčnými časťami, pričom ovládanie experimentálneho zariadenia je umožnené na diaľku, prostredníctvom počítačových sietí. Takéto systémy sa uplatňujú najmä v oblasti technického vzdelávania, pričom prinášajú inovačný spôsob prístupu k praktickému vyučovaniu.

Práca sa venuje takému konceptu architektúr vzdialených laboratórií, ktorý umožní oddeliť fázu samotného technologického vývoja od procesu implementácie, čím sa získa významná časová úspora a zníži sa zložitosť budovania nových inštancií vzdialených laboratórií. Aby boli takéto požiadavky splnené, architektúra musí byť od začiatku vývoja uvažovaná ako univerzálna, ako z pohľadu hardvérových prostriedkov, tak aj softvéru. Pevné časti architektúry musia byť schopné pripojiť a obsluhovať širokú skupinu rôznych experimentálnych zariadení a programové časti musia byť navrhnuté tak, aby dokázali bez rozdielu sprostredkovať ovládanie takýchto zariadení na diaľku.

Pre tieto účely boli navrhnuté dve rôzne architektúry. Prvou je viacúčelová hardvérová a softvérová architektúra založená na priemyselne zameraných riadiacich a komunikačných prvkoch, akými sú programovateľné logické automaty a priemyselné sieťové smerovače. Výhodou tejto architektúry je, že poskytuje hotové hardvérové a čiastočne aj softvérové riešenie a dokáže obsluhovať veľmi širokú triedu procesných zariadení. Druhá architektúra využíva nízkonákladové súčasti akými sú jednodoskové počítače a programovateľné mikroovládače. Úlohou tohto prístupu je demonštrovať potenciál lacných zariadení pre využitie v riadení procesov na diaľku. Praktickým prínosom práce je implementácia viacerých procesných vzdialených laboratórií, založených na oboch typoch architektúr. Ďalšou diskutovanou témou v práci je spôsob spravovania vzdialených laboratórií a ich publikovanie pre využitie vo vzdelávaní.

Kľúčové slová: Vzdialené laboratóriá, Viacúčelová architektúra, Nízkonákladová architektúra, Priemyselné riadiace systémy, Riadenie mikroovládačmi, Vzdelávanie v riadení procesov.

Contents

List of Figures

List of Tables

List of Acronyms

ABLA	ArPi Branched Low-Cost Architecture
AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
CMR	Compare Match Register
CSS	Cascading Style Sheets
DAQ	Data Acquisition
DOM	Document Object Model
EJS	Easy Java Simulations
FPGA	Field Programmable Gate Array
GUI	Graphical User Interface
GPIO	General Purpose Input/Output
GWT	Google Web Toolkit
HMI	Human-Machine Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
I^2C	Inter-Integrated Circuit
IDE	Integrated Development Environment
INR	Industrial Network Router
ISR	Interrupt Service Routine

JSON JavaScript Object Notation

LAN Local Area Network

LMS Learning Management System

LTI Linear Time Invariant

MHSA Multipurpose Hardware and Software Architecture

ODE Ordinary Differential Equation

OS Operating System

PC Personal Computer

PHP PHP Hypertext Preprocessor

PLC Programmable Logic Controller

PMS Power Management Server

PWM Pulse-Width Modulation

RCS Real-Time Control System

RIA Rich Internet Application

RL Remote Laboratory

RLA Remote Laboratory Architecture

RLMS Remote Laboratory Management System

SCADA Supervisory Control and Data Acquisition

SOA Service-Oriented Architecture

SoC System on Chip

SPI Serial Peripheral Interface

SSH Secure Shell

UART Universal Asynchronous Receiver/Transmitter

ULM Upper Level Management

USB Universal Serial Bus

VL Virtual Laboratory

XML Extensible Markup Language

Chapter 1

Introduction

Remote Laboratory (RL) is an instrumentation apparatus designed to provide its physical resources and their operation remotely over computer network. Although the concept of remote sensing and control is almost as old as computer science, remote laboratories, as they are known today, arisen in 1990s when the Internet was made available for masses. In early years of this period, even before outspread of the global network, the most of RL-related works were based on local communication (Herman and Aburdene, 1991), or as proposals to future usage of the Internet (Aburdene et al., 1991). First occurrences of internet-based RLs in literature are also dated to this time. Starting with the technical reports (Bohus et al., 1995), through conference proceedings (Goldberg et al., 1995), and heading to leading journals (Aktan et al., 1996), new research and application fields focused on RL development were set.

> *"In the Spring of 1994, hundreds of WWW servers were coming online every week. We conjectured that it might be possible to use this medium to allow low cost public access to a teleoperated robot, in effect providing: desktop teleoperation."*
>
> Goldberg et al. (1995)

Nowadays, applications of RL are spread among many technical and natural scientific fields, such as physics (Sievers et al., 2010), chemistry (Leal and Leal, 2013), medicine (Barros et al., 2013), engineering in general, and others. In engineering disciplines, RLs have been successfully implemented and used for: energetics and power systems (Collins, 2009), industrial systems (Aydogmus and Aydogmus, 2009), communications (Gampe et al., 2014), electronics (Tawfik et al., 2013a), robotics (Fernandez et al., 2012a), and most of all (and within a highest interest for this work) for automatic control systems (Ionescu et al., 2013; Santana et al., 2013; Vargas et al., 2011).

Web-based laboratories are commonly used as educational tools for practical exercising. Those well-designed provide at least similar features and possibilities to perform

experiments as traditional hand-on laboratories, and simultaneously extend the methodology of work in term of availability and accessibility. In fact, the RLs are considered as one of the most influencing technological enhancements in engineering education in the last 100 years (Froyd et al., 2012).

1.1 Motivation

RLs can be of various types, using different concepts of operation, and based on different hardware and software technologies. The review of the current state-of-the-art has shown that majority of RLs are the results of development methods designed directly for the specific purpose. Even these laboratories work well and provide rich features, they rely on architectures built for that one kind of usage. Many development methods are inefficient, mostly those based on approach like *"show me an experiment you want to provide remotely, and I will build a remote laboratory for it"*. This approach is good only if the desired result is single RL with no consideration of future development, reproducibility, or spread of solution over more applications.

Development of RL, especially for automation and process control, is a very time-consuming task and requires a lot of effort. It requires the analysis of appropriate procedures, design of hardware architecture and communication infrastructure, selection of software parts, and of course the realization (setting up hardware, interconnection, programming, debugging, testing, etc.). Commonly each developer must undergo these tasks for every new remote laboratory he creates. If usage requires changes, the architecture must also undergo changes on software or hardware level. Typical case is when institution owns an RL with specific experiment, and procures another one of different kind. If the developers want to connect new experiment to RL, they must create a whole new architecture and software for it, since existing RL is designed for the original one and it will not work for any other. Even if this example is the worst case scenario, it is unfortunately a very common issue.

The main motivation for this work is based on a question *"Is it possible to design such kind of architecture, which will allow people, even with average technical skills, to implement remote laboratories just by interconnection and configuration of their parts?"*. If such a goal can be achieved, it would result in reduction of complex RL development just to the acquisition of architectural parts and to implementation of experiments.

1.2 Goals of the Thesis

This work contributes to the development in area of automatic control remote laboratories in the term of the architectural design. Two main theoretical contributions are presented:

- Multipurpose Hardware and Software Architecture (MHSA) – concept of architecture based on industrial aimed devices (Industrial Network Routers and Programmable Logic Controllers),
- ArPi Branched Low-Cost Architecture (ABLA) – concept of architecture based on low-cost hardware (single-board computers and electronic development boards).

The work picks up also the following minor and practical contributions:

- Implementation of automatic control remote laboratories on MHSA and ABLA, with following experiments:
  - thermo-optical system (3 laboratories),
  - hydraulic system with two series of tanks,
  - hydraulic system with three tanks,
  - DC motor (2 laboratories),
  - air-flow heat exchanger.
- Implementation of various real-time automatic control scenarios for remote laboratories
- Incorporation of implemented laboratories into educational environment:
  - deployment of Remote Laboratory Management System (RLMS) WebLab-Deusto at the Institute of Information Engineering, Automation and Mathematics,
  - deployment of implemented remote laboratories through RLMS WebLab-Deusto,
  - arrangement of inter-institutional usage of laboratories with University of Deusto (Bilbao, Spain).

The concept of MHSA, including the early research and not yet published results, are shown in:

- M. Kalúz, L. Čirka, and M. Fikar.: Simplifying the implementation of remote laboratories in educational environments using industrial hardware. Kvasnica M. and Fikar, M., editors, In *Proceedings of the 19th International Conference on Process Control*, pages 522–527, Štrbské Pleso, Slovakia, June 18-21, 2013.

- M. Kalúz, J. García-Zubía, M. Fikar and Ľ. Čirka.: A Flexible and Configurable Architecture for Automatic Control Remote Laboratories. Submitted to *IEEE Transactions on Learning Technologies*, 2014, *submitted after revision.*

The low-cost remote laboratories based on ABLA, are described in:

- M. Kalúz, Ľ. Čirka, R. Valo, and M. Fikar.: ArPi Lab: A Low-cost Remote Laboratory for Control Education. In *19th IFAC World Congress*, Cape Town, South Africa, August 24-29, 2014, *accepted.*

The results of inter-institutional cooperation on the federation of remote laboratories between Slovak University of Technology in Bratislava and University of Deusto, Bilbao, Spain, and the deployment of Remote Laboratory Management System (RLMS) WebLab-Desuto at Institute of Information Engineering, Automation and Mathematics were published in:

- M. Kalúz, J. García-Zubía, P. Orduña, M. Fikar, and Ľ. Čirka. Sharing control laboratories by remote laboratory management system WebLab-Deusto. Sebastián Dormido, editor, In *Proceedings of 10th IFAC Symposium on Advances in Control Education*, volume 10 of *Advances in Control Education*, pages 345–350, Sheffield, UK, 2013. University of Sheffield, International Federation of Automatic Control. doi: 10.3182/20130828-3-UK-2039.00048.

The early work focused on development of on-line experimental laboratories, including both the virtual and remote laboratories, is described in:

- M. Kalúz, Ľ. Čirka, and M. Fikar.: Virtual and remote laboratories in process of control education. *International Journal of Online Engineering*, 8(1): 8–13, 2012. doi: 10.3991/ijoe.v8i1.1830.

- M. Kalúz, Ľ. Čirka, and M. Fikar.: Virtual and remote laboratories in education process at FCFT STU. Mikuláš Huba and Michael E. Auer, editors, In *Proceedings of the 14th International Conference on Interactive Collaborative Learning*, pages 134–139, Piešťany, Slovakia, International Association of Online Engineering, Kirchengasse 10/200, A-1070, Wien, Austria, September 21 - 23 2011. doi: 10.1109/ICL.2011.6059562.

1.3 Overview of the Thesis

This thesis is aimed on development and implementation of RLs for automatic control, specifically focused on the issues of operational architectures. The thesis is split into

three main parts. Part I discusses a theoretical background of experimental laboratories, defines the RLs and provides a general requirements for their development, deployment and usage. Part II is the most inclusive part of this work. It is split into three chapters. Chapters 4 and 5 provide the conceptual principles and overview of two new proposed types of architectures, which are considered to be the main contribution of this work. Additionally in Section 4.2, a discussion about common types of RL architectures and their comparison to MHSA and ABLA is provided. Chapter 6 discusses Upper Level Management (ULM) of remote laboratories and the deployment of Remote Laboratory Management System (RLMS) WebLab-Deusto in practice. Part III shows the principles of implementation of automatic control algorithms for RLs as well as the list of laboratories implemented on new architectures.

Part I

Theoretical Background

Chapter 2

Classification of Experimental Laboratories

Experimental laboratories play an important role both in research and education, and in last decade they have become an inseparable part of applied science. They allow researchers and students to perform experiments from different fields of science and engineering, to create an appropriate conditions for their professional growth, and to acquire a new knowledge and practical experience.

Traditional look on laboratory as the room with scientific equipment has changed with the development of modern information technologies. This resulted into branching of laboratory experimentation paradigm, and formation of two new approaches. Both are the combination of two concepts: the substitution of physical interaction between experimenter and laboratory equipment by computer-based interaction; and substitution of physical experiments by simulations. Considering these changes, laboratory instrumentation can be split into three forms as shown in Fig. 2.1.

First established paradigm uses both concepts and it is called Virtual Laboratory (VL). In these kind of laboratories, experimenters do not work with the real physical systems, but with their mathematical description projected into computer program or algorithm. A good practice in transcription of real behavior to mathematical model for VLs is that the main internal relations and causalities of system remain intact, and simultaneously the simplifications made on presented experiment should not significantly affect its nature or physical meaning.

Second paradigm uses similar computer-based interaction as in the case of VL, but it provides the resources of real laboratory systems and it is called Remote Laboratory (RL). This concept uses a distribution of services over the different locations and their interconnection through computer networks. For RLs, we distinguishes: parts that are located at the side of experimenter (client-side), often represented by software called client application; parts located at the side of remote services (server-side), serving communication between client and laboratory; and experimental equipment which may be located at

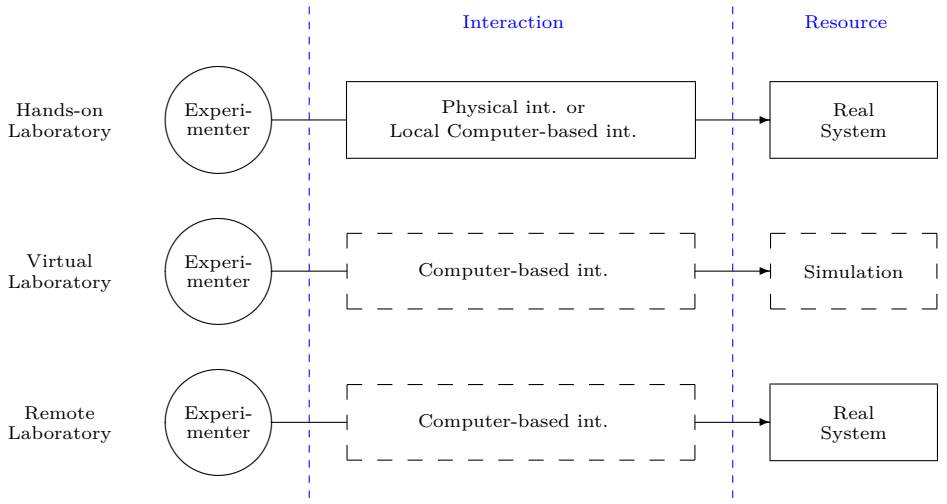


Figure 2.1: Types of experimental laboratories

different place than main server-side technology. Deployment of hardware/software and principles of communications differ from one type of architecture to other, and detailed analyses of their types will be provided in Section 4.2.

Different authors use different terms of definition and views on classification of experimental laboratories. We pick up two definitions which are used most commonly and have been presented in respected works. Dormido (2004) (Tab. 2.1a) distinguishes laboratories by nature of resource and type of access to the resource, while Gomes and Bogosyan (2009) (Tab. 2.1b) use different terms for description of, in fact, the same principles.

Table 2.1: Two different views on laboratory classification

(a) Classification of laboratories by Dormido (2004)

Access to the resourse \ Nature of the resource	Real	Simulated
Local	Hands-on laboratory	Mono-user virutal laboratory
Remote	Remote laboratory	Multi-user virtual laboratory

(b) Classification of laboratories by Gomes and Bogosyan (2009)

Type of experiment \ Type of location	Same for user and experiment	Different for user and experiment
Physical devices	Hands-on experiment	Remote experiment
Device models	Local simulation	Remote simulation

This work is exclusively aimed on those laboratories which are operated remotely and contain real instrumental devices. This form of experiment realization has several advantages and drawbacks when compared to the traditional hands-on laboratory. One of the

main benefits is the opportunity of experimenter to perform the experiments from any location with the presence of computer network, on which the RL is published. If the Internet is used as network for RL access, it can be accessed almost from any location in the world scope. Another important benefit of RL is the removal of time restrictions for experimenters. They can use experimental apparatus also in time when traditional laboratories are closed. Moreover, RLs do not require direct assistance of supervising person, as it is common in traditional laboratories. Students and other users can perform their experiment without the concerns of experiment's security, because it is (or it should be) ensured by remote laboratory itself. Security is however one of major drawbacks for remote laboratories, since a comprehensive class of problems must be taken into account. They include the security of user's hardware (Personal Computer (PC), tablet, smartphone, etc.), software (operating system), communication over network, operational hardware and software in laboratory, and mostly the security of instruments which are controlled remotely. A good practice in the realm of on-line experiments is that instruments under remote control are of nonthreatening nature. In other words, such non-supervised devices whose operation will increase a risk of situations like material ignition, electrical discharges, chemical accidents, and other types of area damage are highly inappropriate to be used in remote laboratories.

Chapter 3

Characteristics of Remote Laboratories

We have shown in previous chapter that there is no strict definition of remote laboratory in the literature, or it varies from one author to another. In general, there are several properties that remotely controlled experiment must satisfy to be labeled as remote laboratory. To address those properties correctly, three different perspectives are provided in this section: the one of user, administrator, and developer.

3.1 User's Perspective

It is a well known fact that RLs are exclusively designed and dedicated for their users. Therefore the most important requirements for their design are aimed on the satisfaction of those who use them. We can distinguish between two class of requirements, which involve features and properties. Firstly, there are those that are necessary in order to achieve the basic functionality and usability of RLs, and secondly those properties/features that make a difference between the common and good ones.

3.1.1 Awareness of Remote Laboratory

Even this is not a feature itself, the awareness of RL existence is very important. Without the information, users do not even know that they have an opportunity to perform their experiment and study tasks remotely, without the time and place restriction. There are several ways how the information about particular on-line laboratories can be released. First is information given from person to person, mostly applied in the education process, where teachers encourage students to use such tools as the supplement to standard learning methods. Another approach is the use of information sources like Internet, by publishing the information on Web pages related to the topic of interest. Generally, the awareness of RLs attracts users and results into valuable feedback and evaluation of usage, which help to keep their continuous improvements.

In many cases, the awareness of RLs also leads to the extension of user and experiment base of the educational institutions. In literature there are several collaborative systems which allow sharing of experiments, as well as provision of resources to users from different institutions. These are for example *MIT iLabs* (Hardison et al., 2008), *LabShare Sahara* (Lowe et al., 2009), *WebLab-Deusto* (Orduña, 2013), and *AutomatL@bs* (Vargas et al., 2011). For more information see Chapter 6.

3.1.2 Access Model

There are two main requirements that must be satisfied for access model of RL. The first is public availability and the second is user access control. While availability can be achieved easily, by publishing the laboratory services on the network, the access control must be supported by an advanced software system. To emphasize why is access control so important feature for RLs, the following situation is described.

The user A enters the RL and starts the experiment. During the session, user A controls experiment and gathers data from it. In some point of measurement a seconds user B accesses the same laboratory without awareness of the first user. Both users try to control the experiment but their actions may be contradictory, creating corrupted measurement, unwanted data output and overall confusion or frustration of usage.

To avoid such kind of situations, most of RLs deployed in practice allow only one user to access laboratory at a time. The concurrent attempts to enter laboratory are solved by an appropriate access model. Commonly, there are two main forms of user control in practice. First is based on booking systems, where users reserve the session in advance for the specific date, time, and duration. The booking system allows to enter only those users, which have valid booking reference in actual time. Calendar booking model is often used in practice and can be found e.g. in Uran et al. (2007) and Smrcka et al. (2012). The second model is based on putting the users into the queue, in a particular order. The order can be determined by a several factors. If some group of users has a higher priority than other users, they can be moved forward in the queue. This principle is called the priority queue and it have been deployed also in practice (Orduña, 2013).

3.1.3 Human-Machine Interface vs Graphical User Interface

Generally, Human-Machine Interface (HMI) is a composition of hardware and software that enables user to control machine and get feedback from it. In literature, this term is often confused with the Graphical User Interface (GUI). In the context of RLs the HMI is actually a whole infrastructure and architecture between the user and device (machine) under control (Fig. 3.1). As a conclusion, it can be said that RL is a special case of HMI

where information, actions, and other data are transferred through a computer network. HMI, in the case of RL, consists of several parts, and one of them is client-side software or client application. GUI is such graphical environment of client-side software that is directly designed to provide a visual environment for control of remote experiment. It is important to mention that GUI is the only part of RL with which the user comes to the contact, so the overall impression depends on GUI and its features. Therefore a useful GUI should address all user's needs and it should be as intelligible to human as possible. Users prefer those laboratories that provide easy-to-use controls and clean interfaces. The additional features, such as advanced visualizations, customizable layout, video support, augment reality and other, can attract users' attention and overall impression on RL.

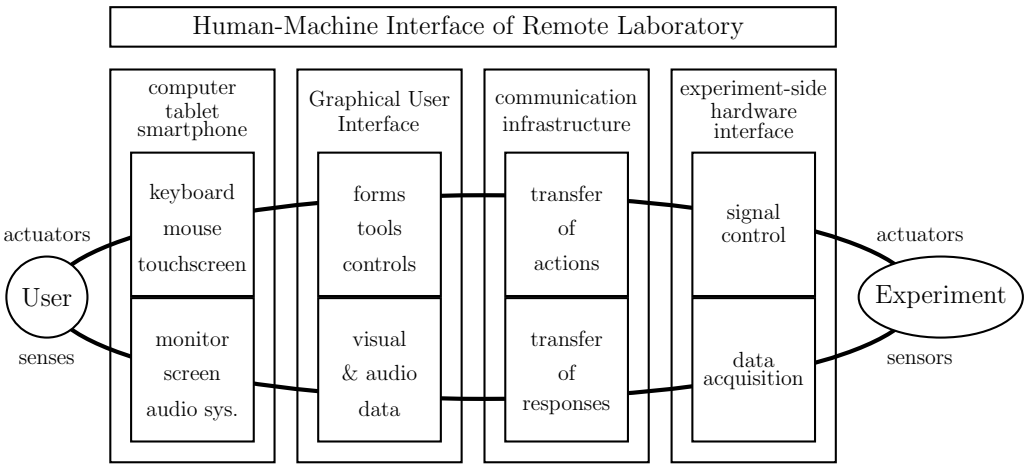


Figure 3.1: Components of HMI feedback loop

Very important aspect of RL usability is also the way how the client application is provided to the user. This strongly depends on appropriate choice of technology on which the application software is built and running. As mentioned in Gomes et al. (2007), there are two main topologies in concept of client software: dedicated desktop application and Web-based application. Desktop applications are well known for their performance and capability of running high level user interfaces. They often are marked as intrusive software, because they are based on technologies which can access the system resources in client's computer. From the usability point of view, this can be considered as unwanted security drawback of such kind of software. The biggest benefits of Web-based applications are their non-intrusiveness and universality. In other words, they can be easily used without need of any specialized software on almost all modern devices and platforms that support Web standards. On the other hand, they do not provide the performance and

interface features to such an extent as desktop applications do. Despite these facts, the use of Web-based interfaces for RLs has grown significantly in recent years due to the rapid development of Web technologies.

Another required feature of client applications is the possibility for user to acquire data of measurement in such form that is appropriate for their further processing. Actually, this requirement applies only for a specific class of RLs. We distinguish between those laboratories which are designed purely for demonstrative purposes and provide just a simple feedback for user's senses (e.g. remote control of robot movement), and those which are designed for full-featured experiments. The second mentioned must provide data in the scientific form. For example, in laboratories used for process control education, it is necessary for users to understand the nature of controlled system and to have the data for proper analysis.

3.1.4 Educational Context

Although the RLs can be designed and used also for industrial applications (e.g. remote supervisory systems), this work deals with the concept of RLs as the educational tools. This implies that remote experimentation must be linked with standard education process. In practice, RLs are often provided as a support for courses and lessons taught on universities (García-Zubía and Alves, 2011; Gomes and García-Zubía, 2007), but also on lower levels of education, such as secondary schools (Dziabenko and García-Zubía, 2013). The main idea is to provide students with higher degree of learning freedom and to give them opportunity to perform their tasks independently on their presence in institution. To achieve these goals, RLs are often closely aligned with the e-learning. Using of Learning Management Systems (LMSs) is one of most common way how to incorporate RLs into e-learning process (Fernandez et al., 2012b; Ruano-Ruano et al., 2013).

3.2 Administrator's Perspective

Administrator of RL is such a person who has right to supervise the experiments, usage of laboratories, user activities, security, and to apply users' feedback to improve the education process. This role often belongs to teachers.

3.2.1 Supervision and Management

The main role of administrator is to supervise over the usage of the system. This task often involves the evaluation of users' activities, usage of laboratories, and the general feedback from those who use labs. Based on these information, administrator apply

changes in access model, publishing methods, settings of laboratories and experiments. For example if one laboratory is under heavy load for a longer period, administrator can reduce time allowed for one session to allow more users to access laboratory in short time.

3.2.2 Security on Level of Usage

Security of RL is one of the main concerns of administrators. It is associated to an unexpected behavior of laboratory operation and possible malicious activities of users. In the first case, administrator is responsible for uninterrupted operation of laboratories. Since they are in most cases a composition of complex hardware and software parts, they can break down or act unexpectedly from time to time. These situations can be caused by hardware failures (wear of electronics), software crashes (mostly the software dependent on computer's OS), loss of power supply, connection outage, and others. In the case of such malfunction, administrator's task is to detect the fault, find its source and restore the normal operation of laboratory as soon as possible. If the fault elimination is beyond the technical skills of supervising person, a developer must handle this task.

In literature there are a few approaches of fault detection on the different levels of architecture. In Yazidi et al. (2011) the detection of failures is implemented directly on the level of experiment, and it is applied on AC electrical machines. An entirely different approach is implemented in the RLMS WebLab-Deusto (Orduña, 2013). In this system, the fault detection is carried out on the level of architecture, e.g reporting broken connections between different parts.

As mentioned before, the second concern is user's behavior. Even in the best designed RLs there is still a risk that developers have left bugs in the software or other security holes, which could be misused by malicious users. Therefore the RLs must provide at least a basic tools for tracking of users' activity and be able to interconnect unwanted event with a specific user.

3.3 Developer's Perspective

Even the RLs are primary designed for those who use them (users, teachers, ..), the developers' point of view is the main aspect which determines the difference between useful and useless laboratory.

3.3.1 Effective Approaches of Development

The development of RLs is often considered as one complex task of design, construction, and application, starting with the experimental device (laboratory instrument) and

ending with RL in ready-to-use form. This approach is often called a "creating from scratch" and resulting in *ad hoc* solutions. There is one significant drawback of such kind of approaches. Developers must repeat the whole procedure of development for each new intended laboratory. Since this task can take long time (at least weeks, but often months and more), the approach leading to *ad hoc* laboratories is highly inefficient. Therefore we, as the developers, distinguish between two main phases of RL creation. First is development without specific application (laboratory). This phase contains design and construction of multipurpose architecture and preparation of infrastructure for second phase which is implementation for particular applications. Once the first phase is completed, it does not require to be repeated for every new implementation. This principle produces some significant impacts also for other roles in RL's usage, such as that more experiments can be implemented in short time. This speeds up the overall evolution of on-line experimentation in institution applying this practice.

3.3.2 Operational Robustness

Each RL, as a set of hardware and software, must be built on appropriate technologies in order to ensure its robustness. Systems for on-line experimentation must be designed to handle uninterrupted operation for long time without frequent maintenance (e.g. for several months). This must apply for the whole architecture from client-dedicated services to experiment itself. Since the majority of RLs is implemented on Service-Oriented Architectures (SOAs), developers must choose robust hardware and software to run such services.

3.3.3 Modifiability and Reproducibility

A good RL should be designed in such way to be easily modifiable on the level of hardware and software source code. It is often necessary to make additional changes in laboratory architecture in order to solve common situation that appears during the phase of testing but also the full operation. Therefore, it is appropriate to create an open technological solution and keep it as comprehensible for other developers as possible. This also affects another important property of RLs, their reproducibility. Many educational institutions cannot afford their own development, so they rely on available ready-made solutions. If a specific concept of RL is well described, using open non-commercial technologies, low-cost hardware, and its developers provide the whole *know how*, it is likely that such concept will be popular and reproduced also by other institutions.

3.3.4 Security on Level of Technology

Each RL is only as secure as developer's methods and used technologies are. In general the security on the level of technology can be discussed for two areas, the security of hardware/software resources and the security of information. In the first case, developer must guarantee that software is safe with the respect to the self-operation, to other software parts in the same or other device, and with the respect to the hardware. In other words, software cannot allow e.g. user to cause its malfunction, allow to cause errors in operating system, or to take control over the hardware in the way that have not been intended by developer.

Second security concern is related to the privacy of information. RLs often transfer confidential information over the computer network, therefore they must use such kind of technologies that guarantee the protection of data against third persons. Data can be of various nature, such as user authentication, authorization keys for laboratories, and operational/process data. Security of them can be achieved in several ways, such as use of secured network protocols, two side decryption/encryption, or virtual private networks.

Part II

Architectures

Chapter 4

Multipurpose Architectures

Multipurpose architectures are those types of architecture that provide operation for at least a class of remote laboratories without need of changes in structure, hardware components or software. The idea of such architecture is to follow the concept well known as *Plug and Play*.

4.1 Problem Description and Motivation

Because the hardware architectures of remote laboratories can be very flexible, many different forms and their applications are available (Dziabenko and García-Zubía, 2013; García-Zubía and Alves, 2011; Gomes and García-Zubía, 2007). As it is, among the other interesting conclusions, mentioned in Gomes and Bogosyan (2009), the majority of available existing solutions are in the form of *ad hoc* or single-purpose labs, where creators adapt the architecture design and software development to a specific purpose of lab usage. A typical case of this kind of approach is that operational software and hardware is designed to work with a one-and-only type of experimental device, and it cannot be directly applied to a different type of purpose without interfering with hardware components and/or program source code. Therefore, one of the current trends in RL development deals with the issues of extensibility, versatility, and multi-functionality.

> *"A common wrong approach is to design first a prototype which works—it works!—and then worry about adding new features. Unfortunately, this is not a valid approach since oftentimes, there is a need to redo the whole application."*
>
> Garcia-Zubia et al. (2009)

These highly requested features are not only the matter of hardware, but also have to be supported by appropriate operational software. Because, in most cases, the operators of remote experiments are common users connected to laboratory through the Internet,

the client software must satisfy the requirements of functionality for different possible cases. The main issue of client side software development deals with the selection of the technology and operation method that developers have to choose to create client applications. These should provide users with the opportunity to use laboratories from their personal computers without the need of any specialized software. Since Web browsers are generally available for all currently used operating systems, the majority of existing solutions are operated in this way. As described in Garcia-Zubia et al. (2009), the appropriate choice of client technologies for Web browser is the essential task for creating effectively working RLs. Due to the development of new technologies and standards for Web based applications such as HTML5, and Asynchronous JavaScript and XML (AJAX) frameworks, some more traditional approaches for Rich Internet Applications (RIAs), most of all, those based on embedded technologies, like Java Applets and Flash, are slightly pushed outside of interest. Especially, JavaScript-powered Web applications have become very popular for client side implementation in recent years. The main issue with JavaScript is a lack of standardization across different Web browsers (Mesbah and Prasad, 2011). This problem is caused by a non-standardized model of native functions used by Web browsers. While JavaScript client side logic can work properly in e.g. WebKit (Apple Safari, Google Chrome), it may result in an unexpected behavior in engines like Gecko (Mozilla Firefox), Trident (Internet Explorer), Presto (Opera), and others.

Another important aspect of RL development is time and effort required to their creation and practical implementation. Even though this field is under vital development for almost 20 years, the approaches based on openness are still quite rare. Most of them are software oriented architectures that provide Application Programming Interfaces (APIs) for interconnection of different client side and experiment-serving technologies. In practice, theses architectures are often a parts of upper level frameworks that incorporate additional features of laboratory management, user management and serving of inter-institutional usage. Therefor they are called Remote Laboratory Management Systems (RLMSs). In the literature there are several RLMSs with strong impact and high reputation. These include *MIT iLab* (Hardison et al., 2008), *LabShare* (Lowe et al., 2009), or *WebLab-Deusto* (Orduña et al., 2011).

The above mentioned issues represent the main motivation for development of multipurpose architecture. However, these objectives cannot be solved by one side design strategy (only hardware or only software), but require a combination of both approaches.

In this part, two multipurpose architectures are proposed. The first Multipurpose Hardware and Software Architecture (MHSA) is based on industrial hardware, and open-source software and the second ArPi Branched Low-Cost Architecture (ABLA) is based on low-cost hardware and also open-source software. Both architectures provide several

features that are contributional to remote laboratory development. The most significant are: the possibility to connect various types of automatic control experiments without the need of changes in architecture or source code; embranchment capacity of architectures (more than one experimental node can be connected); significant time savings in implementation process; and for MHSA two control layers that allow the use of advanced process control methods.

4.2 Common Types of Architectures

Before we get to the description of MHSA and ABLA it is important to discuss the most often occurring architectures in literature and practice. Even the Remote Laboratory Architectures (RLAs) occur in various forms, using different hardware, software technologies, physical arrangement, and principles of operation, the majority of them can be classified in the following categories.

1. *Client – server with control software – experimental device.* This commonly occurring type of architecture uses a direct connection between the PC (server) and controlled device e.g. by serial or Universal Serial Bus (USB) port (Fig. 4.1 – case ①). Therefore, the PC must be equipped with proper software for communication with the client and also for control of the experimental device. In most cases, server-side PCs contains self-made software solutions based on commonly available technologies. The main advantage of this architecture type is in its low implementation difficulty and price. In Bisták (2011), the RL for control of thermo-optical plant and magnetic levitation is designed on this type of hardware architecture. The author uses the client Java application and server technology based on Java/MATLAB. Two technologically different remote control approaches for inverted pendulum are presented in Kolenčík and Žáková (2009). The authors use .NET for server-side implementation and Adobe Flash for client-side in first case, and in second case the combination of MATLAB as control software and Java-based client application. Another different approach of RL implementation on this type of architecture is given in Mohtar et al. (2008). The authors use the LabVIEW software that is designed to handle almost all required features such as communication with experimental devices, server capabilities, data acquisition and GUI design tools. The advantage of commercial programs like MATLAB and LabVIEW is that they provide easier implementation and time savings. A possible drawback is that they also increase the final cost of the RL.

2. *Client – server with control software – Data Acquisition (DAQ) device – experi-*

mental device. Remote experimental setup with very similar functionality as above mentioned. The only difference is that the data acquisition device (e.g. DAQ card) is a separate part of the architecture (Fig. 4.1 – case ②). Architecture with a DAQ device is usually used in a situation when it is not possible to directly interconnect the server-side PC and experimental device. This type of hardware setup often reduces the requirements on signal processing software included in the control PC, but also increases the price. Control lab architectures using this type of approach are proposed in Choudhary et al. (2012), and Chandra and Venugopal (2012).

3. *Client – proxy server – lab nodes (PC + control unit) – experimental devices*. This type, also known as branched architecture, provides a higher capacity of possible experimental connections than any other architecture (Fig. 4.1 – case ③). The proxy server is used for managing the connections of different clients to different experimental nodes. Each node usually consists of a local computer (lab server) equipped with appropriate software and hardware for data acquisition, and an experimental device. The main advantage of this architecture comes from its wide inter-connectivity, where experimental nodes can be located even in different countries while the user accesses are managed by proxy server (Hu et al., 2013; Lowe et al., 2009; Santana et al., 2013; Tawfik et al., 2013b). Interesting branched architecture that uses wireless experiment nodes is given in Cui et al. (2012).

4. *Client – server with Supervisory Control and Data Acquisition (SCADA) system – control unit – experimental device*. Combination of SCADA system and control unit (Programmable Logic Controller (PLC) or other hardware controller) is one of most frequently used industrial approaches (Fig. 4.1 – case ④). Among the common features like data acquisition, variable control capabilities and connectivity, most SCADA systems also provide HMIs that can be used as an alternative to client side software for RL. Gao (2010) introduces the RL for tank level control using RSView32 SCADA system with Web-based HMI. An interesting solution combining Java-based client-server communication, Easy Java Simulations (EJS) for GUI development and SCADA-PLC control mechanism is shown in Besada-Portas et al. (2012). Other applications of RLs based on PLCs and SCADA are shown in Marques et al. (2008), and Lazar and Carari (2008). Approaches in this category are characterized by their expensiveness stemming from the use of commercial software and hardware, but also by significant reduction of required implementation effort and time. Most commercial control systems contain built-in features that can be instantly used for experimental control, so the developers do not need to write processing software or user interfaces.

5. *Client – server/microcomputer – programmable electronic board – experimental device.* The very popular approaches in recent years, also often labeled in literature as *low-cost*, are based on programmable boards like Field Programmable Gate Array (FPGA), Arduino micro-controllers, and cheap alternatives to standard computers (e.g. single-board computers based on ARM architecture like Raspberry PI, Pandaboard and others) (Fig. 4.1 – case ⑤). Due to the wide spectrum of various hardware components, this category cannot be generalized as a specific type of architecture. These approaches can be described as the opposite alternatives to category 4, especially from the pricing and implementation point of view. While the purchase price of hardware is incomparably lower than the price of industrial control systems, the implementation difficulty and required effort for software development rises rapidly, because developers must work with raw hardware components. An interesting example of low-cost remote laboratory based on microcomputer and FPGA board is shown in Costa et al. (2012). Works by Hashemian and Pearson (2009), El Medany (2008), Neto et al. (2012) show similar approaches, but instead of a microcomputer, they use a standard PC (server) to provide communication between users and experimental devices.

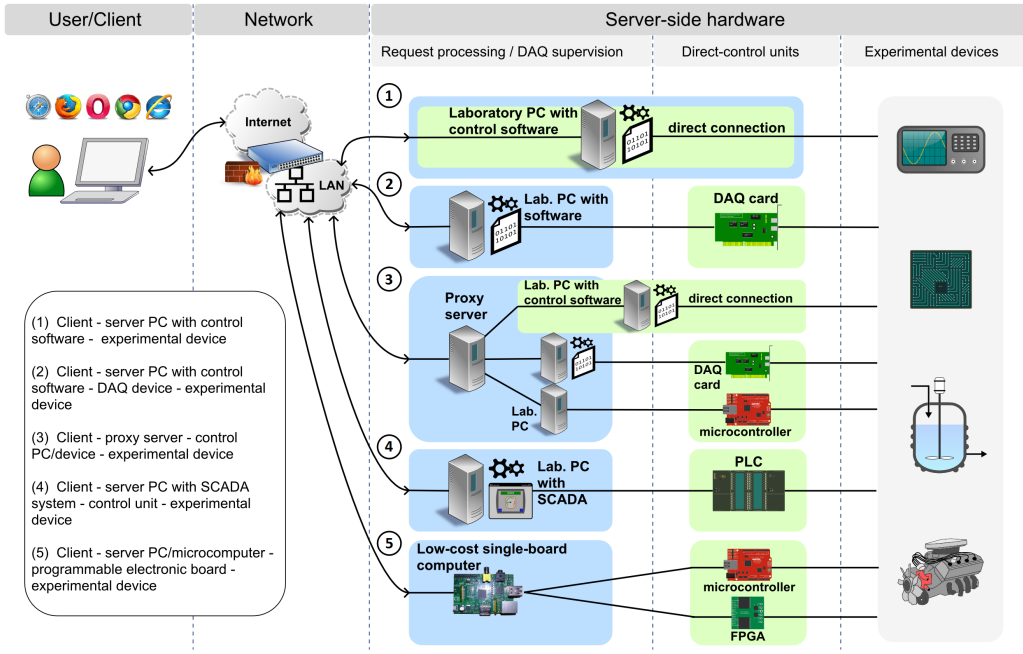


Figure 4.1: Summarizing different types of architectures.

The comparison of MHSA, ABLA and other types listed in this sections are provided in Table 4.1. To show the main benefits and drawbacks we provide evaluation of costs, requirements on: hardware implementation/software development, performance, flexibility, and extensibility/connectivity. The additional evaluation of architectures' benefits, drawbacks and application scope will be provided in Sections 4.4 and 4.5 for MHSA and in Section 5.3 for ABLA.

Table 4.1: Comparison of MHSA, ABLA, and other types of architectures

Type of architecture	Costs (€)[A]	Requirements (effort) on		Performance[B]	Flexibility[C]	Extensibility / Connectivity[D]
		hardware implementation	software development			
1*	hundreds	low	average	high	low	low
2*	hundreds - several thousands	low	average	high	average	average
3*	thousands and more	average - high[1]	low - high[2]	high	dependents on node type[3]	dependents on node type[4]
4*	thousands and more	average	low	high	high	high
5*	dozens	average	high	low	low	average
MHSA	several thousands	low	low	high	multi-purpose[5]	high
ABLA	dozens	average	high	low	multi-purpose[5]	average

* Architectures listed in this section
A. Costs contain approximate expenses of hardware components and software.
B. Computational power of hardware used in architecture.
C. Flexibility shows if the architecture is opened or closed (e.g. if implementation of new experiment requires significant changes in sw/hw).
D. How many nodes can be managed by server-side computer and how many experiments can be connected to one node.
1. Depends on number and type of components used in branched architecture. Higher number of hardware nodes results into higher implementation difficultness.
2. The requirement on software development for physical systems in branched architectures can be significantly reduced by using open-source technologies. Some of development and management systems provide direct APIs and device aimed software (iLab, LabShare, WebLab-Deusto).
3, 4. The flexibility and extensibility of branched architecture is determined by type of devices used as nodes and their operational software, which can combine approaches of other architecture types.
5. Multi-purpose architecture is opened at both, the back-end (various types of exp. devices can be connected) and front-end (client is fully configurable for all situations at back-end, e.g. use of different types of exp. devices).

4.3 Multipurpose Hardware and Software Architecture

The idea of MHSA is to provide a new type of hardware and software architecture that simplifies remote laboratory development in terms of time savings and reduction of technical skill requirements and simultaneously solves the common problem of single-purposiveness by introducing a remote laboratory structure which is opened at the back-end and configurable at front-end.

4.3.1 Hardware

MHSA is combination of hardware and operational software based on two types of industrial devices: a Industrial Network Router (INR) and a Programmable Logic Controller (PLC). Contrary to traditional types of RLAs listed in previous section, MHSA have replaced the server computer by an INR device, that is primarily designed to supervise the industrial controllers connected in its local network. PLC is used as the device for direct physical interaction with laboratory equipment, using a set of standard electrical signals to read process sensors and manipulate its actuators. Depending on the type and modularity of a particular PLC used in architecture, a wide connectivity can be achieved (analog/digital signals with high spectrum of ranges and resolutions).

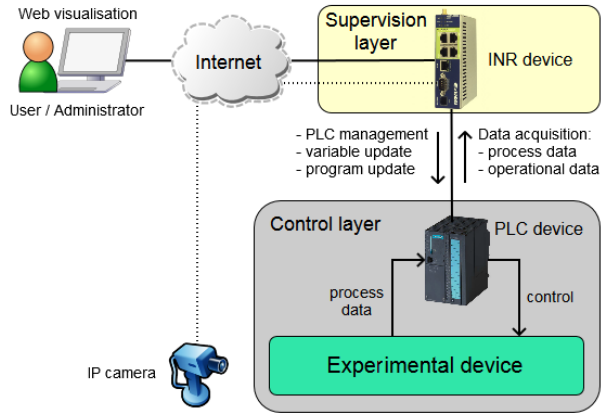


Figure 4.2: Operational setup of MHSA

As can be seen in Fig. 4.2, the INR device represents the supervision layer of operational setup. The lower operational layer consist of nodes of PLCs and experimental devices. This setup can be described as opened-at-the-end, because it allows any type of experimental device that communicates through analog and digital signals to be connected. Depending on the connectivity of specific PLC used in architecture, provided by PLC's I/O modules, and the number of signals required for controlling the experimental device, one PLC can handle several experiments in one laboratory node. To prove this statement we will show it in a practical cases of implementation (see Chapter 8). The extensional capacity of this kind of architecture is shown in Fig. 4.3.

The industrial router used in our architecture is eWON4005CD (eWON Inc., 2011), that can be described as a coupler for industrial controllers. The main features of INR are:

- Industrial protocol translation

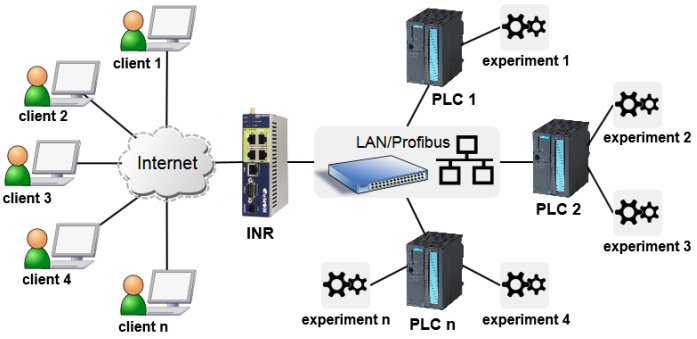


Figure 4.3: Embranchment capability of MHSA

- Direct access to PLC program variables (read/update)
- PLC program update
- Process/operational data acquisition from PLC
- Server-side script runtime environment
- Data and event logging
- Alarms
- Operational architecture management
- FTP server
- Web server

In one node of physical setup, the experimental device is connected to the PLC's I/O module. The INR device in supervision layer communicates with all experimental nodes in the local area network by their specific industrial protocols. To simplify the communication mechanism, the INR provides the feature of associated internal variables called TAGs. Each TAG is addressed to the specific memory area of the PLC and allows the user or administrator of INR to directly read and update its memory variables. TAGs are distributed through all program parts of INR like Web management screens, scripting environment and Web server services. Depending on the type of signal processing, PLC can represent three different functions in operational architecture.

1. *Non-interfering communication channel.* In this kind of situation the PLC is used as the simple communication channel, without any internal logic implemented in it.

The TAGs in INR are addressed directly to I/O memory of PLC and work with the raw signals. This setup fits only for situations when no unit conversion or signal processing is required, e.g. digital state control (signals can reach only values 0 or 1).

2. *Signal processor and unit converter.* In this case, the internal logic of the PLC contains algorithms for signal conversion and processing like filtering, standardization, and unit conversion. These procedures are applied mostly for raw analog signals collected from and sent to the experiment.

3. *Control processing environment.* In industrial applications, PLCs are primary aimed at base level control of technological processes. The versatility of their functional usage is wide, containing various types of control algorithms and predefined features. All of these are also available for use in operational architecture of remote lab. In this case, PLC can provide additional computational power for advanced control algorithms like those based on optimization tasks (e.g. model predictive control).

4.3.2 Communication Services of INR

By the communication principles, the MHSA can be classified as XML-based Service-Oriented Architecture (SOA). The INR device can be, from the usage point of view, characterized as a common network router with a set of administrative Web pages and a set of internal services which process the users' actions. Although the INR is not directly designed to work as remote laboratory serving unit, a detailed inspection of provided services has shown that INR can be operated by a Web application in the same way as by the user logged in the administrative interface. The list of Web services provided by INR that are important for RL's operation are following.

Original services pre-implemented in INR by default:

- `rcgi.bin/UpdateTagForm` - is the service that performs changes in TAG variables. It accepts the Hypertext Transfer Protocol (HTTP) POST data in the form of `TagName1=[name]&TagValue1=[value]&TagName2=[name]&TagValue2=[value]`.

- `rcgi.bin/ScriptEditForm` - service that allow upload script for internal computing environment of INR. These scripts consist of initial part (executed once) and cyclic part (executed in loops). Internal scripts are defined in BASIC language and they can read/update TAGs. These can be used for implementation of control algorithms on the supervision layer of architecture.

- `rcgi.bin/RunStopProgForm` - is a simple service that enable user to run or stop the execution of internal scripts.

- `rcgi.bin/ScriptCtrlForm` - service that allows batch execution of short scripts inside BASIC environment.
- `rcgi.bin/EditUserForm` - allows to create and change the account of INR administration. This service is used by the ULM system (see Chap. 6), to grant permissions to the specific user of client application.

Services implemented for MHSA operation:

- `tagXML.shtm` - is the active server page that returns Extensible Markup Language (XML) document containing current values of all predefined TAGs. This is the configuration file which is written by an implementer of laboratory as the process of laboratory configuration.
- `pollingInfo.shtm` - is optional active server page that returns the information about actual presence of user in laboratory. This file can be used by and ULM system to determine whether the user session finished or not.
- `clientConfig.xml` - is a static XML file that provides the full configuration of client application, as well as the paths to other services.

Each service, depending on the purpose, is accepting a different set of parameters sent in HTTP request's body (POST method).

4.3.3 Client Application

One of the most important parts of MHSA is client application *ControlApp*, which provides GUI and a set of services for communication with INR. It is built as a JavaScript powered HTML5 Web page with semi-dynamic Document Object Model (DOM) and event-driven internal logic. In the comparison with other Web-based laboratories, the *ControlApp* provide a whole new concept of GUI construction. This concept is called on-fly content generation of RL interface and it has been designed for specific needs of MHSA. The *ControlApp.html* file itself do not have almost any static content in its initial form, and contains only required structure of Web document and reference to main JavaScript file. Whole GUI is automatically generated during the load of application, reflecting the specific configuration for particular RL. The initial layout of application, communication, and event handling is processed by an optimized JavaScript logic. This concept have been chosen for two main reasons. Firstly, to have one unified application for all laboratories developed at MHSA, and as the result, to save development effort for each new instance of experiment, connected to architecture. Secondly, this concept reduces the amount of data that needs to be stored in INR's Web server. In traditional approaches of Web-based

RLs, a whole content of GUI (ready-to-use Web page) is transferred to user. The MHSA's *ControlApp* transfers only a very small static Web page and minimized JavaScript file. Moreover, it does not carry any specific information about layout. It can be described as the constructor for GUI. The generation of specific components is performed in Web browser and it depends on information provided by a set of configuration files loaded by client application. The principle of GUI generation is shown in Fig. 4.4.

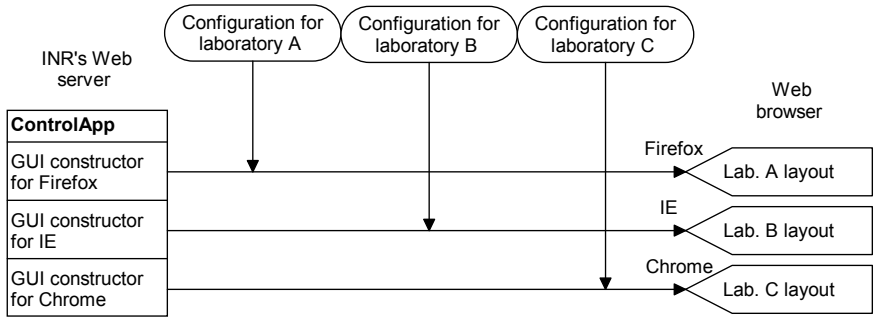


Figure 4.4: Principle of GUI construction depending on Web browser type and specific laboratory configuration

The *ControlApp* has been developed in Google Web Toolkit (GWT) development framework [1]. GWT provides a set of tools for building client-to-server AJAX Web applications. The main benefit of this framework is that client side implementation is written in Java language and it is compiled into optimized JavaScript. This allows developers to use a full spectrum of programming paradigms, which makes Java so popular. Moreover, GWT compiler uses a cross-compile feature and can produce JavaScript variations for all major types of Web browsers. As the result, developers do not need to pay attention to compatibility issues with some of JavaScript native interfaces, because GWT will handle it automatically. Produced Web application in default contains several different JavaScript files, each dedicated for different Web browser core, and it is fully supported by the MS Internet Explorer, Mozilla Firefox, Google Chrome, Chromium, Apple Safari, and Opera.

ControlApp's GUI (Fig. 4.5) is designed as user-friendly workspace with draggable windows (similar to working desktop of common operating systems) and can contain several types of predefined components, which are: main control panel located at top of Web page; tables of inputs, outputs and variables ①, located at fixed position in left side of interface; a set of charts ② with visualization of signals and variables; window with selection of control algorithms ③; a set of video streams from remote video devices ④;

[1] http://www.gwtproject.org/

window for download of data ⑤; and logging window ⑥.

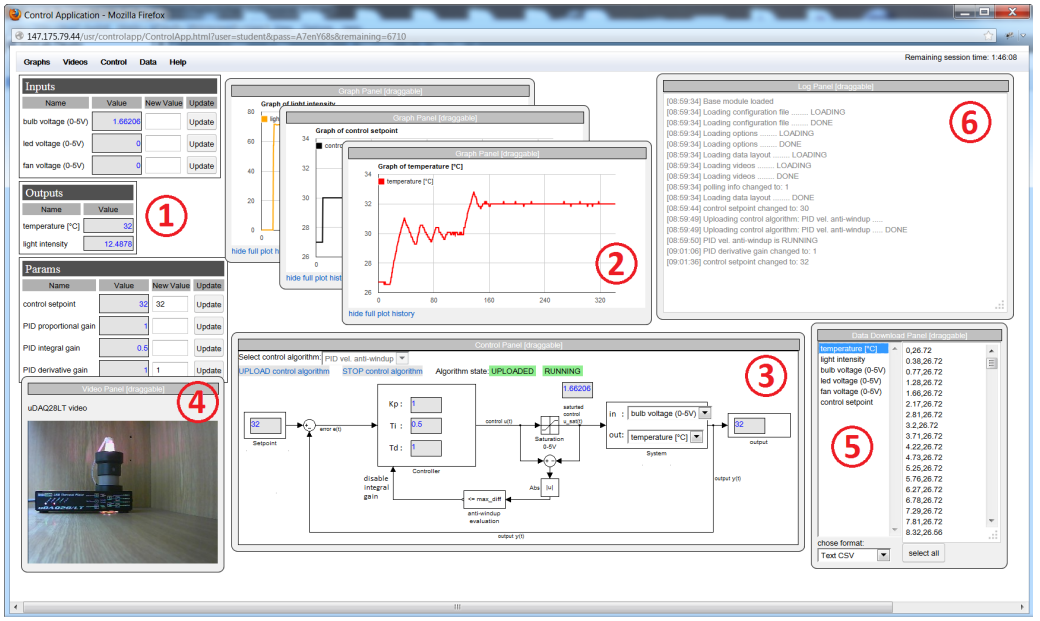


Figure 4.5: Graphical layout of *ControlApp* for thermo-optical laboratory

In the main control panel, user can select which components are displayed in working desktop. This feature is useful especially for situations when *ControlApp* is used for laboratories with many signals or multiple video streams. The window-based style of application also provides the user with possibility to organize graphical layout for his/her own needs in order to ensure comfort and good orientation in usage of RL. Application provides three types of tables. The first table shows the list of input signals of remote system and it is intended to provide manual control of actuators. Each row contains name of signal with the information about engineering unit and range in which signal can be changed, current value, input field where user can type a new value, and submit button. The second table is used to display current values of system's outputs (measurements from senors). The third table contains a list of additional variables used in RL, such as control setpoint, parameters of controller, or other operational variables, which are linked from INR's TAG bank to client application.

Each signal that is displayed in GUI's tables can be also displayed in graphical form as an interactive chart. This feature allows users to observe trends of variables during the remote session. Interactive charts are embedded in draggable windows and they can be show or hidden on demand, using the controls on main panel. Each chart plots the values of one signal. User can display value of plotted data points by moving the mouse

cursor over the specific point.

The client application performs the layout of tables and charts based upon the information supplied by the configuration file. A fragment of `clientConfig.xml` responsible for signals is show in Example 4.1.

Example 4.1: XML fragment of configuration file for definition of signals

```
<SignalsAndVars>
 <eWonInputs>
  <input tag="bulb_tag" name="bulb voltage (0-5V)" minAllowVal="0"
    maxAllowVal="5" defaultVal="0" showGraph="onDemand"
    graphLineColor="green"/>
  <input tag="fan_tag" name="fan voltage (0-5V)" minAllowVal="0"
    maxAllowVal="5" defaultVal="0" showGraph="onDemand"
    graphLineColor="gold"/>
 </eWonInputs>
 <eWonOutputs>
  <output tag="temp_real" name="temperature [\degreeC]"
    showGraph="always" graphLineColor="red"/>
 </eWonOutputs>
 <eWonVars>
  <var tag="w_sp" name="control setpoint"  minAllowVal="0"
    maxAllowVal="100" defaultVal="30" showGraph="onDemand"
    graphLineColor="#000000"/>
  <var tag="Kp" name="PID proportional" />
  <var tag="Ti" name="PID integral" />
  <var tag="Td" name="PID derivative" />
 </eWonVars>
 <eWonHidden>
  <hidden tag="time" name="server time" />
 </eWonHidden>
 <eWonTime useSpecialTimeSignal="true">
  <signal tag="tn" format="millis" />
 </eWonTime>
 <eWonSpecialized>
  <scriptStateVar tag="script_state" />
 </eWonSpecialized>
</SignalsAndVars>
```

The definitions of signals for system's inputs are enclosed in `<eWonInputs>` tag. Each `<input>` tag contains mandatory attributes for: name of TAG variable associated to signal; name that is shown in GUI; minimum and maximum allowed values, which are used

to restrict user to manipulate signal and actuators only inside their physical boundaries; default value which is applied when client application is loaded and, optionally, after user leaves the session; information whether the application generates the chart for signal; and line color for chart. The `<output>` tags enclosed in `<eWonOutputs>` use the similar attributes, except those for value restrictions. The `<var>` enclosed in `<eWonVars>` requires only attributes for TAG name and GUI name and other mentioned attributes can be defined as optional. The `clientConfig.xml` file also contains an `<eWonHidden>` tag that is used for definition of variables which are hidden from user, but necessary for an appropriate work of client application. The implementer of laboratory can set up the time signal which will be used for plotting of data series in charts. In this case, two options are available: to use server side time of INR; or the reference time from Web browser. The last optional tag `<scriptStateVar>` enclosed in `<eWonSpecialized>` is used to determine whether the BASIC language environment inside INR is active (executing the algorithms in loops) or not. This option is used for those laboratories, which provide possibility to upload and run control algorithms directly from GUI.

ControlApp allows implementers to attach multiple video streams to GUI of remote laboratory. Currently it supports every type of IP camera that provide Motion JPEG stream and/or series of static images over the Web services. If both options are available client application automatically displays streamed video for Web browsers that support Motion JPEG (e.g. Mozilla Firefox, Google Chrome, Apple Safari, etc.) and simulated pseudo-stream (put together from single images) for other browsers (e.g. MS Internet Explorer). Data for video configuration is shown in Example 4.2. Each video stream is defined by tag `<frame>` inside the `<videoFrames>` tag. Each `<frame>` tag contains attributes with information about name that is show in GUI, resolution in pixels, and appearance option. Additional two tags `<source>` are nested inside `<frame>` tag of each video. These two tags define the types and sources of available services. The tag for pseudo-stream contain additional attribute, which defines the image refresh rate in milliseconds.

Example 4.2: XML fragment of configuration file for definition of video frames

```
<videoFrames>
  <frame name="uDAQ28LT video" width="320" height="240"
   showFrame="always">
    <source type="mjpeg" url="http://147.175.79.43/video.mjpg"/>
    <source type="jpg" url="http://147.175.79.43/jpg/image.jpg"
       refreshMillisec="1000"/>
  </frame>
</videoFrames>
```

The settings for charts are defined in `<graphConfig>` tag (Example 4.3). Implementer of laboratory can chose the size of charts (width and height), maximum number of point rendered in one chart, and number of points displayed in floating mode (e.g. for 100 points and data sampling time of 0.5s, the chart will display range of 50s).

Example 4.3: XML fragment of configuration file for definition of charts' properties

```
<graphConfig dataPointLimit="100" pointLayoutLimit="1000"
  width="500" height="250" allowFullDataPointView="true" />
```

ControlApp uses an simple model of user control based on asynchronous timer invocation in INR. This feature has been implemented in order to avoid situations when user unexpectedly exits laboratory during the measurement or control task procedure. Once the user closes the browser, the *ControlApp* is terminated and cannot perform any tasks in order to reset laboratory to its default state. There are two tags in configuration file (Example 4.4) which implementer can set up to handle these kind of situations. First one `<timerBasedUserControl>` activates the timer in INR which handles the predefined procedure and defines the period in which the client application invokes the reset of this timer. The predefined procedure consists of commands which set TAGs listed in second tag `<resetToDefaultBeforeHalt>` to its default values and function that halts the execution of BASIC scripts in INR (e.g. control algorithms). If user enters the laboratory which uses the configuration with user control from Example 4.4, the client application will generate the command `TSET 1,30:ONTIMER 1,'bulb_tag@=0:led_tag@=0:fan_tag=0:HALT'`. This batch command is then sent to `rcgi.bin/ScriptCtrlForm` service. The timer 1 will execute the code after 30 seconds, but if the user is still on-line, the client application periodically (each 10 seconds) sends the command `TSET 1,30`, which resets the timer's timeout. As the result, the final procedure (reset of laboratory to default setting) is applied only if client application does not contact the INR for longer than 30 seconds. The last optional tag inside the user control configuration (`<rlmsWdPolling>`) defines the hidden TAG variable that informs ULM system (in this case RLMS WebLab-Deusto) that user is still present in laboratory.

Example 4.4: XML fragment of configuration file for definition of user control

```
<userControl>
  <timerBasedUserControl active="true" timeoutSec="30"
    checkIntervalSec="10"/>
  <resetToDefaultBeforeHalt tagList="bulb_tag,led_tag,fan_tag"/>
  <rlmsWdPolling active="true" tag="wd_polling"/>
</userControl>
```

A list of pre-implemented services (described in Sec. 4.3.2) of INR and file paths are

provided in `<comConfig>` tag of configuration file (Example 4.5).

Example 4.5: XML fragment of configuration file for definition of communication services and files paths

```
<comConfig>
  <ewGeneral url="http://147.175.79.44/" />
  <ewMainDirectory url="usr/controlapp/" />
  <ewMainFile url="ControlApp.html" />
  <ewTagXML url="usr/controlapp/config/tagXML.shtm"/>
  <ewTagForm url="rcgi.bin/UpdateTagForm" />
  <ewBasicEditForm url="rcgi.bin/ScriptEditForm" />
  <ewBasicRSForm url="rcgi.bin/RunStopProgForm" />
  <ewBasicExecForm url="rcgi.bin/ScriptCtrlForm" />
</comConfig>
```

The last part of configuration file contains information about the restrictions of laboratory usage. They define which situations must be separately handled. Each restriction type is defined by a single tag with three attributes. Attribute `active` accepts boolean value `true` or `false` which determine whether is restriction active or not. Additional `message` and **img** define the text information and image that appear in pop-up window in GUI. Situations that are handled by these restrictions are detected in source code of *ControlApp*. Example 4.6 shows the restriction for input signals, which ensures that user is allowed to update their values only in valid boundaries. Otherwise the action is ignored and warning window appears (Fig. 4.6).

Example 4.6: XML fragment of configuration file for definition of restrictions

```
<restrictions>
  <onInputOutOfLimits active="true" message="Input is out of allowed
    boundaries." img="image/no_sign.png"/>
</restrictions>
```

MHSA allows developers and implementers to create automatic control scenarios in two different ways. Controllers can be implemented in control layer directly in PLC. In that case, the controllers are hard-coded and users can tune those parameters, which are linked to GUI as TAG variables. One PLC can provide several different control tasks for one experiment, or in the case when one PLC is used to control more systems, even to control several of them. In such case, some logical switches must be implemented in PLC and INR to avoid conflict situations.

The second scenario is the control algorithm running on supervision layer inside the INR. As mentioned before, INR is able to execute user-defined scripts in BASIC lan-

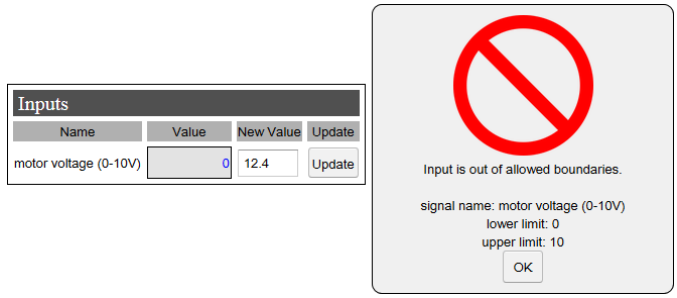


Figure 4.6: Warning window appears when user tries to set invalid input

guage. This function was used to develop a feature which allows user to choose from predefined control algorithms, to upload them to INR, and to run them directly from client application.

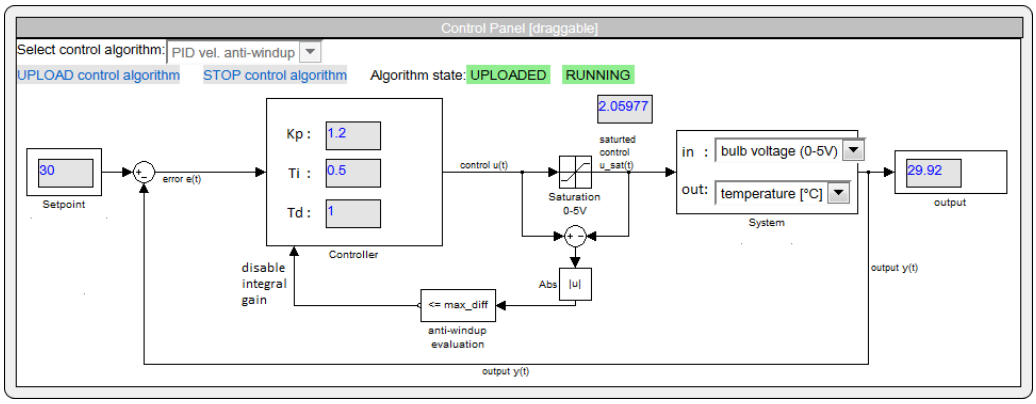


Figure 4.7: Control window of client application with selected controller

The control window (Fig. 4.7) is the main tool in client application that allows to user to apply automatic control algorithms to remote process. In similar way as other parts of application, it also depends on specific configuration of experiment. Implementers of laboratory can define control algorithms in configuration files with predefined structure. During the construction of GUI, client application loads the file `listOfControlAlgorithms.xml`, where information about available algorithms is provided (Example 4.7).

Example 4.7: Configuration file with a list of control algorithms

```
<listOfControlAlgorithms>
  <algorithm name="Simple relay"
    path="controlAlg/simple_relay/simple_relay.xml"/>
  <algorithm name="PID velocity"
    path="controlAlg/pid_velocity/pid_velocity.xml"/>
  <algorithm name="PID vel. anti-windup"
    path="controlAlg/pid_velocity_aw/pid_velocity_aw.xml"/>
</listOfControlAlgorithms>
```

This file does not contain definitions of algorithms, just their names and paths to their particular files. Each control scenario is located in a separate directory containing two files. The first is XML file with definition of signals used in algorithm, appearance setup, and BASIC script for algorithm definition. The second is the image file with the picture of schema that represents the control scenario. This picture is used in client application as the background for control window layout. This file determines which control algorithms are provided to user in client application (Fig. 4.8).

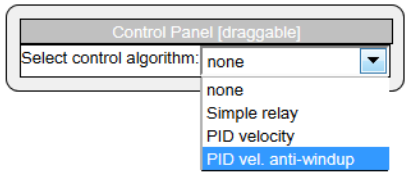


Figure 4.8: Drop-down list with control algorithms

Configuration file for definition of control algorithm contains general information such as author's name, date of creation, path to image file and it's layout resolution, and tag with a short description (Example 4.8).

Example 4.8: Fragment of file for control algorithm definition: general information

```
<generalInfo author="Martin Kaluz" date="24.10.2012"/>
<schema img="pid_velocity/schema.png" width="817" height="284"/>
<description>
  <![CDATA[description of algorithm]]>
</description>
```

This file also contains the list of signals and variables used by an algorithm (Example 4.9). Client application distinguishes three types of them, the `<systemInput>` and `<systemOutput>` which define the symbolical variables for interconnection of algorithm

with specific input and output signals, and `<tag>` for definition of additional non-signal variables used by algorithm. Symbolic input and output variables are used to define a place in BASIC script (as unique string `::input::` or `::output::`), where real definition of signal's TAG is placed before the algorithm is uploaded into INR's program environment. These two tags also creates the drop-down lists in control scheme and provide user with opportunity to select which of available signals will be used by controller as inputs and outputs of system.

Example 4.9: Fragment of file for control algorithm definition: signals and variables

```
<signals>
  <systemInput string="::input::" showX="500" showY="95"
    displayValue="false" />
  <systemOutput string="::output::" showX="500" showY="130"
    displayValue="true" showX="730" showY="110"/>
  <tag name="w_sp" showX="24" showY="100"/>
  <tag name="u_sat" showX="385" showY="165"/>
  <tag name="Kp" showX="260" showY="70"/>
  <tag name="Ti" showX="260" showY="100"/>
  <tag name="Td" showX="260" showY="130"/>
</signals>
```

This way, user can selects the specific signals which will be connected to control scheme (Fig. 4.9). The main benefit of this approach is that one controller can be used to control each subsystem of connected plant separately. Each signal and variable tag can contain the optional attributes with the coordinates in scheme, where the box with actual value will be displayed.

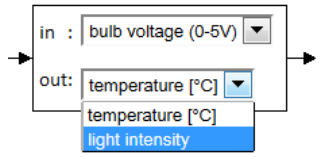


Figure 4.9: Drop-down lists with selection of signals to be connected to control algorithm

The last part of control definition file is the algorithm itself. It is split in two scripts, first for initialization (`<initial>`) and second for in-loop execution (`<cyclic>`).

ControlApp also provides a tool for download of measured data (Fig. 4.10). User can open the window where a selection of important signals and variables is provided. This list contains data that were measured during the whole period of laboratory session. By selecting a specific series, *ControlApp* will print out data in one of available formats. These

are comma and semicolon separated values, XML structure, and matrix in MATLAB syntax.

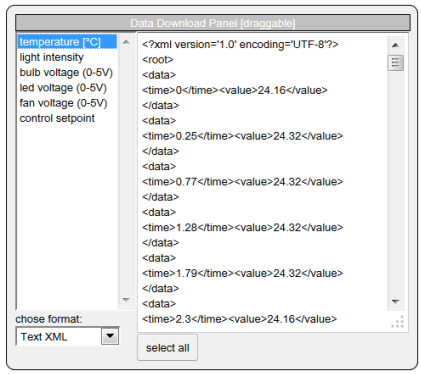


Figure 4.10: Data window with selected series in XML format

Another part of GUI is the logging window (Fig. 4.11), where all major events of *ContorlApp* and user actions are recorded. This component does not have the important role in operation of client application, but can be used to collect information about unexpected behavior that can be reported to the administrator of laboratory. Among the logs of normal operation, this window also provides an information about errors and exceptions if they occur.

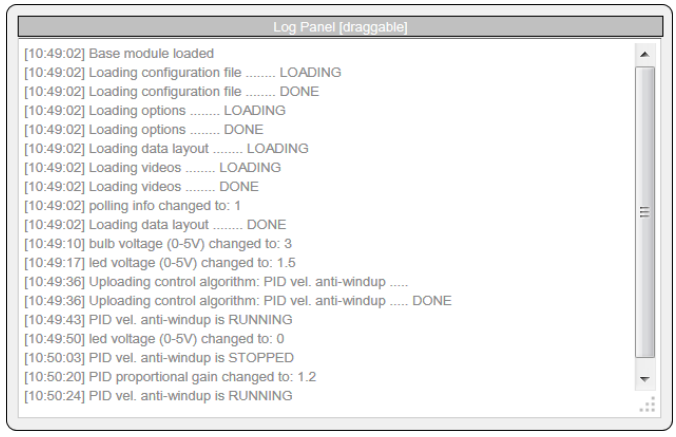


Figure 4.11: Window with usage information of client application

4.4 Application Scope

As mentioned above, the MHSA is capable to serve any kind of device that can be operated by PLC. This fact directly indicates the main scope of applications for which the architecture is generally aimed. Since the realm of remote laboratories is quite broad, nowadays covering most scientific and educational areas, the nature of experiments and their applications can significantly differ. In this section we discuss and pick up those applications for which the MHSA is generally suitable and also those where architecture can be used with some limitations.

The MHSA is generally suitable for implementation of remote laboratories for technological processes that use sensors and actuators operated by standard electrical signals. These include processes of chemical technology like chemical reactors, distillation columns, hydraulic systems (example of a coupled tank system is given in Section 8.2), thermal transfer processes (example of an air-flow heat exchanger is given in Section 8.4), and other types of laboratory training devices (example of laboratory implementation with thermo-optical device is provided in Section 8.1).

Even though the MHSA is based on devices aimed on industrial usage and with some additional operation-based security modifications can be used also for smaller industrial applications, the architecture itself is mainly designed for implementation of remote laboratories in academic scale.

Other types of experiments that can easily be connected to this architecture are those based on electro-mechanical actuators (example of DC motor implementation is given in Section 8.3). This extends the usability of architecture also for some specific applications in robotics area, mostly for robotic arms and other types of manipulators. Although we do not show the example of application in robotics, available works in literature provide some examples of PLC-based control of robotic devices (Stankovski et al., 2010; Zhong, 2012). As MHSA is PLC-based architecture it is capable to serve this kind of devices as well. On the other side, PLCs are unsuitable for other robotic applications, especially those operated by structured instructions and also those operated wirelessly.

4.5 MHSA Advantages and Limitations

The MHSA provides several features that are contributional to remote laboratory development. The most significant are: the possibility to connect various types of automatic control experiments without the need of changes in architecture or source code; embranchment capacity of architecture (more than one experimental node behind INR); significant time savings in implementation process; and two control layers that allow the use of advanced process control methods like adaptive control, model predictive control, etc.

The main advantages/limitations of MHSA are summarized in the following lists.
Advantages of MHSA:

- INR substitutes computer (server machine) in architecture,
- reduced time and effort of implementation,
- no server-side programming required (services already implemented in INR),
- allows direct use of AJAX to industrial hardware,
- fully configurable cross-browser client application,
- supports most of PLC brands on the market (also industrial protocols),
- two hardware layers for control algorithms (INR, PLC),
- SCADA system not required,
- one INR can manage up to 16 PLCs.

Limitations of MHSA:

- architecture is bound to specific type of INR (eWON COSY 141, 2005CD/4005CD, 2101CD/4101CD, 2104/4104),
- laboratory node must contain PLC type supported by INR,
- limited INR's memory (14MB for Web server, 11MB internal/script memory),
- limited PLC's operational memory (depends on type),
- limited possibilities of scripting in INR (BASIC language – only one script at a time),
- communication between client application and INR is limited to the use of standard HTTP request-response model.

Chapter 5

Low-Cost Architectures

This chapter shows a different approach to the development of RLA than the one described in the previous chapter. While in the case of MHSA, the main objective was to put a strong emphasis on usability for laboratory experiments with no consideration of their type and nature, the low-cost approach follows a different objective – minimization of overall expenses related to RLA's composition. Even in this case, the main idea of multipurposiveness can be retained, but obviously, it requires a lot more development effort from the side of RLA creators. These architectures use cheap electronic components like single-board computers, programmable integrated circuits, micro-controllers and electronic development platforms.

> *"To invent, you need a good imagination and a pile of junk."*
>
> Thomas A. Edison

Unlike the approaches using ready-made solutions, low-cost architectures are built on cheap and in its initial form often featureless hardware components. Therefore, this low-cost concept requires incorporation of following operational parts (ordered from back-end to front-end).

- *Signal interface* – is used for direct physical interaction between device under control and hardware controller. This interface ensures the transfer of process information from laboratory system to back-end services of architecture, and the control actions in the opposite way. In this work we will consider and work only with signal interfaces based on electrical variables. Signal interface, in the case of electrical variables, uses a set of A/D converters to transfer analog signals to their digital representation required by the processing unit.
- *Experiment-side processing unit* – is a small computer, micro-controller or any kind of hardware device with programmable logic that is directly capable to control signal interface and simultaneously to communicate with other parts of architecture.

This component is called experiment server, because it provides services for control and observation of laboratory equipment. If the architecture is branched on one of the upper levels located before the experiment server, this device coupled with the experiment is called a laboratory node. In the literature, the use of branched structure of RLA is very popular (Section 4.2 – case 3). The reason is the efficiency of communication and reduction of development costs in the case of multiple experiment laboratory setup. The main differences between single-node and branched low-cost architecture are shown in Fig. 5.1. In practice, most of applications of low-cost RLAs use electronic development boards, equipped with micro-controllers and additional communication interfaces (Ethernet, WiFi, UART, SPI, I^2C, etc.).

- *Communication infrastructure* – is a specific communication network between laboratory node and front-end client application in the case of single-node architecture, or between several nodes and laboratory server in branched architecture.
- *Laboratory server* – is a computer that serves the communication between user and particular node. In low-cost architectures, standard desktop computer can be substituted by a cheaper alternative such as single-board computer. Laboratory server performs several essential tasks for properly working RL. They are:
  - provision of client side application (for Web-based RLs)
  - provision of communications services for client application
  - forwarding of communication to and from specific node
  - ensuring the network security of laboratory
  - provision of credentials and permissions for users and services
  - supervision of laboratory nodes
- *Client application* – is the front-end software implementation of RL. A role of client application is to provide remote user with an appropriate set of features/tools for:
  - observation of the remote physical system's behavior in the form of numerical data, graphs, or optionally live video from remote camera;
  - interaction with the remote system on the level of signal interface (manual control of actuators) and program (update of control parameters);
  - acquisition of measured data in order to their further processing;
  - application of automatic control strategies;
  - provision of experiment booking system or interconnection with upper level management system (e.g. LMS or RLMS).

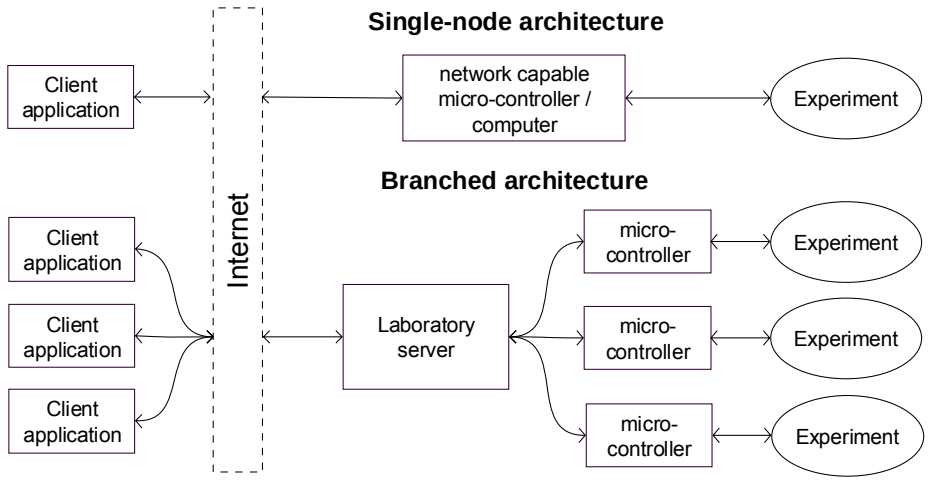


Figure 5.1: Low-cost architectures based on micro-controllers

5.1 ArPi Branched Low-Cost Architecture

The ArPi Branched Low-Cost Architecture (ABLA) is the architecture, based on two types of hardware devices: the single-board computer Raspberry Pi and Atmel AVR 8-bit micro-controller development board Arduino UNO/Arduino Yún. This architecture uses a branched structure with top layer communication coordinator which is implemented on Raspberry Pi computer (representing laboratory server), and a set of experimental servers on the bottom layer (control layer), implemented in AVR micro-controllers (Fig. 5.2).

5.1.1 Hardware

Raspberry Pi (model B, revision 2) is a single-board computer with the market price about $35. It is based on System on Chip (SoC) Broadcom BCM2835[1] with embedded ARM1176JZ-F applications processor (clock rate of 700 MHz), graphics processing unit Broadcom VideoCore IV (clock rate of 250 MHz), 512 MB of SDRAM, and one integrated USB. The storage is solved through the SD card slot, which supports SD, SDHC and SDXC cards up to class 10 and capacity of 128GB[2]. Moreover, Raspberry Pi provides two USB 2.0 ports (through integrated hub), 10/100Mbps Ethernet interface, composite video output, HDMI video output and display serial interface. Low level communication interface includes a set of GPIO pins, UART, SPI, and I^2C. A detailed information about Raspberry Pi hardware equipment can be found in Broadcom BCM2835 peripherals data-

[1] http://www.broadcom.com/products/BCM2835
[2] http://elinux.org/RPi_SD_cards

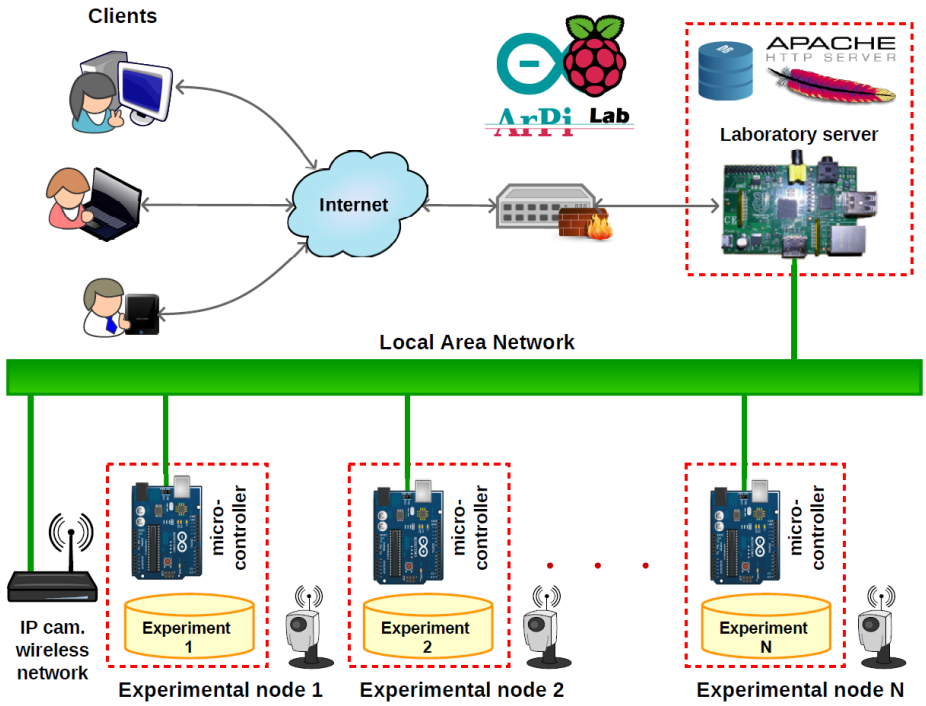


Figure 5.2: Structure of ArPi Lab

sheet[3].

Market provides a wide spectrum of competing alternatives to Raspberry Pi, varying in hardware features and price. The main reasons to use Raspberry Pi in ABLA was based on the lower price than competing systems and the high level of support from developers and community. Despite these reasons, the ABLA do not strictly requires a specific type of computer used as laboratory server since its operational software is based on open and multi-platform technologies. For the implementation of architecture with high number of experiment nodes and therefore higher traffic load on laboratory server, some more powerful low-cost alternatives of single-board computers can be considered as well.

As mentioned before, ArPi Lab architecture uses two different types of control devices as experiment servers, the Arduino UNO equipped with add-on Ethernet board (Ethernet Shield), and Arduino YÚN. Even if both platforms use slightly different types of micro-controllers (ATmega328p for UNO and ATmega32u4 for YÚN), main difference is that YÚN contains additional embedded computer Atheros AR9331 running lightweight Linux distribution OpenWRT Linino, WiFi chip, Ethernet port, hardware serial interface

[3] http://www.raspberrypi.org/wp-content/uploads/2012/02/BCM2835-ARM-Peripherals.pdf

between micro-controller and embedded computer ensuring the internal communication.

5.1.2 Signal Interface

Both platforms provide the same physical interface for electrical signals operating on voltage of 5 V. They contain 14 digital pins while 6 of them can be configured to work in Pulse-Width Modulation (PWM) mode and generate signals useful for control of analog actuators. First two digital pins are connected to hardware buffer and they can be used as transmitter (TX) and receiver (RX) for serial communication interface. PWM pins have 8 bit resolution with the internal digital representation of 0-255.

The average value of PWM signal $\bar{y}$ is expressed as

$$\bar{y} = \frac{1}{T} \int_0^T f(t)\,\mathrm{d}t. \tag{5.1}$$

We assume that the waveform of pulse function $f(t)$ reaches only the values $y_{\max}$ for $0 < t < TD$ and $y_{\min}$ for $TD < t < T$ (pulse shape), where D is the duty cycle. Therefore, the equation (5.1) can be written as

$$\bar{y} = \frac{1}{T} \left(\int_0^{TD} y_{\max}\,\mathrm{d}t + \int_{TD}^{T} y_{\min}\,\mathrm{d}t \right) \tag{5.2}$$

which gives the algebraic form

$$\bar{y} = D y_{\max} + (1 - D)\, y_{\min}. \tag{5.3}$$

The equation (5.3) shows that average value of PWM signal $\bar{y}$ is linear function of duty cycle D, while D can reach values from 0 to 1 ($0 - 100\%$). Waveforms of PWM signals for different duty cycles are shown in Fig. 5.3.

The definition of PWM duty cycles of Arduino boards is in the form of integer value. The ATmega328p micro-controller provides 3 timers that are used for operation of Arduino's 6 PWM pins. The `Timer 0` (8-bit) serves pins 5 and 6, `Timer 1` (16-bit) pins 9 and 10, and `Timer 2` (8-bit) pins 3 and 11. The default resolution for all 6 PWM pins is 8-bit and therefore they can produce 256 different pulse signals, represented by 8-bit integer $I = \{0, 1, \ldots, 255\}$ in the internal C++ code. Then the minimum step of average value is defined as

$$\Delta\bar{y} = \frac{y_{\max} - y_{\min}}{I_{\max} - I_{\min}}. \tag{5.4}$$

Depending on the operating voltage ($y_{\min} = 0\,V$, $y_{\max} = 5\,V$), and internal representation ($I_{\min} = 0$, $I_{\max} = 255$), the minimum step of average value of PWM signal is

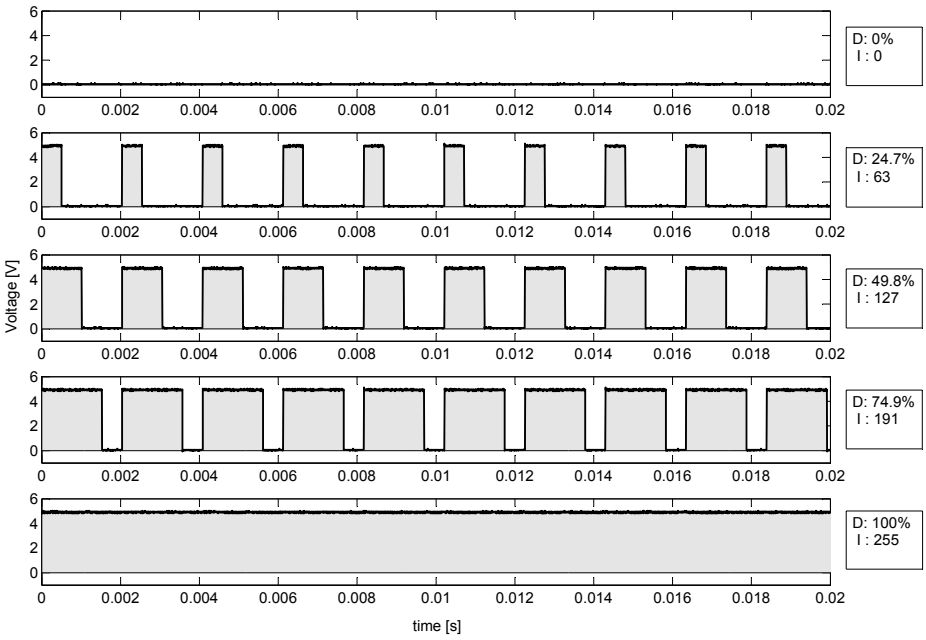


Figure 5.3: Measured PWM signals of Arduino UNO pin 3 with period $T = 0.0204s$ (frequency $f = 490Hz$) for different duty cycles $D = \{0\%, 24.7\%, 49.8\%, 74.9\%, 100\%\}$

$\Delta\bar{y} = 0.0196\,\mathrm{V}$. This value also represent the maximum precision which can be used for control of signal interface on 8-bit pins.

Another characteristic of PWM signals that has influence on the quality of interaction with the physical system is the frequency of signal itself. This is very important property in relation to control. In the ArPi Lab, signal interface is a part of the control loops between the control algorithms inside micro-controller and physical system. Default PWM frequencies provided by Arduino UNO boards are 590 Hz on pins 3, 9, 10, 11 and 977 Hz on pins 5, 6. These frequencies are sufficient for most control applications in ArPi Lab, which require sampling period no lower than dozens of milliseconds. However, some systems require higher frequency of control loop executions in order to preserve their stability or qualitative characteristics. If the sampling time of control loop is lower than sampling of PWM control signal, these requirements cannot be guaranteed. For example, if the system under control has a fast dynamics and discrete control loop ensures the stability with sampling frequency of 1 kHz (e.g. magnetic levitation system in Chapter 7.4.2), neither the 590 Hz or 977 Hz PWM can be used to preserve the information of control action. In this case Theorem 5.1.1 by Shannon (1998) can be applied.

Theorem 5.1.1. (Shannon's sampling theorem) *If a function $f(t)$ contains no frequencies*

higher than ω, it is completely determined by giving its ordinates at a series of points spaced $1/2\omega$ seconds apart.

Based on this information, the sufficient PWM frequency for discrete control with sampling of 1 kHz must be no lower than 2 kHz to keep the consistency of transferred action to system.

AVR micro-controller ATmega328p used in ArPi Lab allows code-based changes of PWM frequencies. By default, `Timer1` and `Timer2` have the base frequency f_B for digital pins 31250 Hz and `Timer0` base frequency 62500 Hz. By choosing the value of divisors $d = \{1, 8, 32, 64, 128, 256, 1024\}$, the produced PWM frequency $f_{PWM} = \frac{f_B}{d}$ can be scaled to values from ∼31 Hz to ∼31 kHz on `Timer1` and `Timer2` and two times higher on `Timer0`. A detailed overview of available PWM frequencies for ATmega328p (Atmel Corp., 2009) is shown in Table 5.1. The ATMega32u4 (Atmel Corp., 2010) uses different internal timer architecture, but principles of frequency scaling are retained.

Table 5.1: List of available frequencies on PWM pins of Arduino UNO (real values measured by oscilloscope are shown in brackets)

Divisor	f_{Timer0} [Hz]	f_{Timer1} [Hz]	f_{Timer2} [Hz]
1	62500 (62409.3)	31250 (31326.5)	31250 (31327.2)
8	7812.50 (7801.1)	3906.25 (3915.9)	3906.25 (3916.0)
32	N/A	N/A	967.56 (979.0)
64	976.56 (975.2)	488.28 (489.5)	488.28 (489.5)
128	N/A	N/A	244.14 (244.7)
256	244.14 (243.8)	122.07 (122.4)	122.07 (122.4)
1024	61.04 (60.9)	30.52 (30.6)	30.52 (30.6)

As the addition, both boards used in ArPi Lab provide 6 analog input pins with the 10-bit resolution (internal digital representation of 0-1023). They allow to interconnect the micro-controller with the analog sensors of controlled system.

5.1.3 Services of Laboratory Server

A laboratory server role is to physically and communicatively separate the local parts of architecture (experimental nodes) from the outer network in order to secure laboratory resources. In ArPi Lab architecture, Raspberry Pi computer is equipped with Web server Apache 2, PHP processor and MySQL database system. Based on these technologies, two main services are implemented: the configuration provider; and communication for-

warder. Both services share the resources in database, where information about particular experiment settings are stored. The main ArPi Lab database `arpi_lab` contains $2n + 1$ tables, while n is the number of experimental nodes in laboratory. One separate table `arpi_nodes` contains the list of all configured and ready-to-use experimental nodes in laboratory setup. This table stores:

- `node_id` – the key index referring to specific validation table;
- `name` – the name of experimental node (e.g. `dev_uDAQ28/LT_01` for first instance of thermo-optical RL);
- `description` – text information about experimental node;
- `node_url` – the network address of experiment server in Local Area Network (LAN);
- `verification_key` – public identification hash code of experimental node (used as external credential in data exchange between client application and laboratory server);
- `node_key` – private identification hash code of experimental node (used as internal credential in data exchange between laboratory server and experiment server);
- `ui_video_setup` – condensed text string with optional information about video services of IP camera/s (containing name of camera, resolution, appearance setup, and network addresses of video and image capture services);
- `ui_appearance_setup` – condensed text string with settings for visual layout of client application. The structure of entry is *[CSS style path]|[width of graphs]|[height of graphs]|[default num. of point in graphs]|[max num. of points in graph]|[allow full view]*, e.g. *default.css|500|250|100|1000|true*.

Other tables use the generic names in the form `arpi_node_validator_`n. These contain information about each signal and variable of physical laboratory device linked to internal logic of RL. Each record contains:

- `name` – defines the full name of signal that is used in client application for information purposes;
- `ext_rep` – is a self descriptive short text string used for identification of signal/-variable in the external communication between client application and laboratory server (e.g. `temp` for temperature);

- `int_rep` – is the integer number used for identification of signal in internal communication between laboratory server and experiment server (in the case of signal, it is the number of pin, and for internal variables, index of array, where variable is stored);
- `type` – is a short text string, identifying the operational type of data transferred through internal communication (`PWM` – PWM-operated output, `AI` – analog input, `DO` – digital output, `DI` – digital input, `VAR` – internal variable, `CS` – controller switch);
- `min_resolution` and `max_resolution` – are information about minimum and maximum integer value that specific type of signal can reach (0-255 for 8-bit `PWM`, and 0-1023 for 10-bit `AI`). This values are used for proper conversion of signal values from internal to external representation and the opposite way. For types of `DO`, `DI`, `VAR` and `CS`, these entries are set to 1 (no conversion);
- `min_val` and `max_val` – are information about minimum and maximum real values that signal can reach. They represent the physical boundaries (e.g. 0.0-5.0 for operational voltage of signal). As well as the resolutions, these entries are set to 1.0 for transferred information that do not require conversion;
- `unit` – is text string, holding the information about engineering unit of signal (e.g. V – volts, % – percentage value, etc.);
- `ui_setup` – is condensed text string containing optional initial definition of signal value and settings for visualization of each signal in client application. The structure of this entry is *[default value]|[show graph switch]| [graph color]*, e.g. *0|1|green* for signal of bulb voltage in thermo-optical plant laboratory. Default value of signal is automatically applied on start of client application. The switch for graphs can be set to three different values: 0 – always display graph on start-up, 1 – display on demand (graph is constructed in DOM of client application, but shows only after user's request), 2 – never display graph (graph is not even constructed in DOM);
- `control_bind` – is the informative integer representing the interconnection of signal with one or more control algorithms provided in remote laboratory. Signals and variable which have no relation to control scenarios are marked with number -1. Those which are available for all control scenarios (such as inputs and outputs of system) are marked with number 0. If the specific signal/variable is defined for a specific algorithm, the `control_bind` is set to integer value that represents identifier of such algorithm. E.g. if PID controller is defined in database table `arpi_node_control_`n with `id` entry set to 1, the associated variables of controller

parameters Kp, Ti, and Td, have `control_bind` entry also set to 1. This informs the operational software of remote laboratory that these three variables have to be provided together with PID controller;

- `bind_type` – is additional text information of signal type in relation to controller (input, output, parameter, setpoint, switch, etc.).

The `arpi_lab` database stores only the information of available control algorithms in tables with generic names `arpi_node_control_`n, where n is the identifier of experiment node (same as for validator tables). Each table contain a list of algorithms with identifier, name, description, and path to image file with control schema. The actual algorithm of specific controller is defined in micro-controller of experiment node.

In laboratory server, two main PHP Hypertext Preprocessor (PHP) services operate with data stored in `arpi_lab` database. The `config_loader.php` reads database tables to generate initial configuration for client application. The `forwarder.php` ensures the transfer of communication from client application to experimental node and the opposite way and simultaneously performs signal/variable conversion and validation. There are three main reasons for this concept of services used in laboratory server. Firstly, the services ensure that automatically generated content of client application is always adapted to specific experiment. Secondly, they provide translation between human comprehensible data to data usable by machine (micro-controller). For explanation, it would be very difficult for developers and implementers to create new laboratory configurations based just on simplistic data directly dedicated for micro-controllers. Thirdly, they ensure the internal security of laboratory. Since these services are publicly available on the Internet, they answer only to requests containing valid authentication key, predefined structure of data and their correctness. Only in the case when request is evaluated as valid, it is translated to internal data structure and sent to experiment node.

5.1.4 Services of Experiment Server

The internal logic of each experiment server is written in C++ language and implemented in embedded micro-controller of Arduino board. Standard Arduino's Integrated Development Environment (IDE) contains a set of built-in APIs, additional libraries, and C++ to AVR compiler `avr-gcc`. The programs written for ArPi Lab nodes uses the same logic and differs only in network settings and internal identifiers. The network service of experiment server processes the requests from laboratory server. The data are transferred by the synchronous HTTP in the JavaScript Object Notation (JSON) structure. The structure of request body is shown in Example 5.1.

Example 5.1: JSON structure of internal representation request (double-slash comments have only informative purpose and they are not contained in transferred data)

```
{"node_key": "03ae9ec81b",
 "mode": 1,  // 0 - read signals, 1 - write and read signals
 "PWM": {"3": 255, // Internal representation of signal
         "5": 27,  // pins and 8-bit integer values 0-255
         "6": 154},// put as PWM outputs.
 "DO": {"8": 1}  // digital output on pin 8 set to high value
 }
```

The services of experiment server are designed to provide two types of operation, the manual control of experiment and automatic control. In manual operation, all requests for signal update/acquirement are directly transferred to signal interface. Automatic control uses algorithms with tunable parameters to control connected system. Each experiment provides several types of controllers which user can choose and set up. The micro-controller, which executes the program in experiment server, contains two computational parts. These are executed in one thread but with different priority levels. The main program contains the initial setup, communication, data processing, code for manual control of experiment, and second program defined as routine which is executed with the higher priority than main part and it is invoked by Timer-based events. The algorithms for automatic control are placed inside of this routine in order to be applied in real-time mode. More information about real-time execution of control scenarios is provided in Chapter 7.4.

The program in experiment server is after compilation stored in the flash memory of Arduino UNO and it can take up to 32kB. It is executed on power up of the board and it remains stored until a new program is uploaded to the device. The main program is defined in two sections, `setup()` which is executed once at start-up and `loop()` executed in cycle. The `setup()` contain initialization of network communication, configuration of signal interface, and timing of routines based on interrupts. The `loop()` section contains the code for communication handling, signal interface control (manual mode), and additional data processing. The program flow is as follows. All required variables and objects are created and then initial setup is executed. Program periodically checks the communication buffer for incoming requests from laboratory server. If data appears in buffer they are read into temporary buffer of program as array of characters and decoded into JSON structure by *aJSON* library[4]. Next procedure is the security check if the request came from trusted source by comparison of `node_key` entry with key defined for this particular experiment server. If validation is successful, program reads the `mode` entry

[4] https://github.com/interactive-matter/aJson

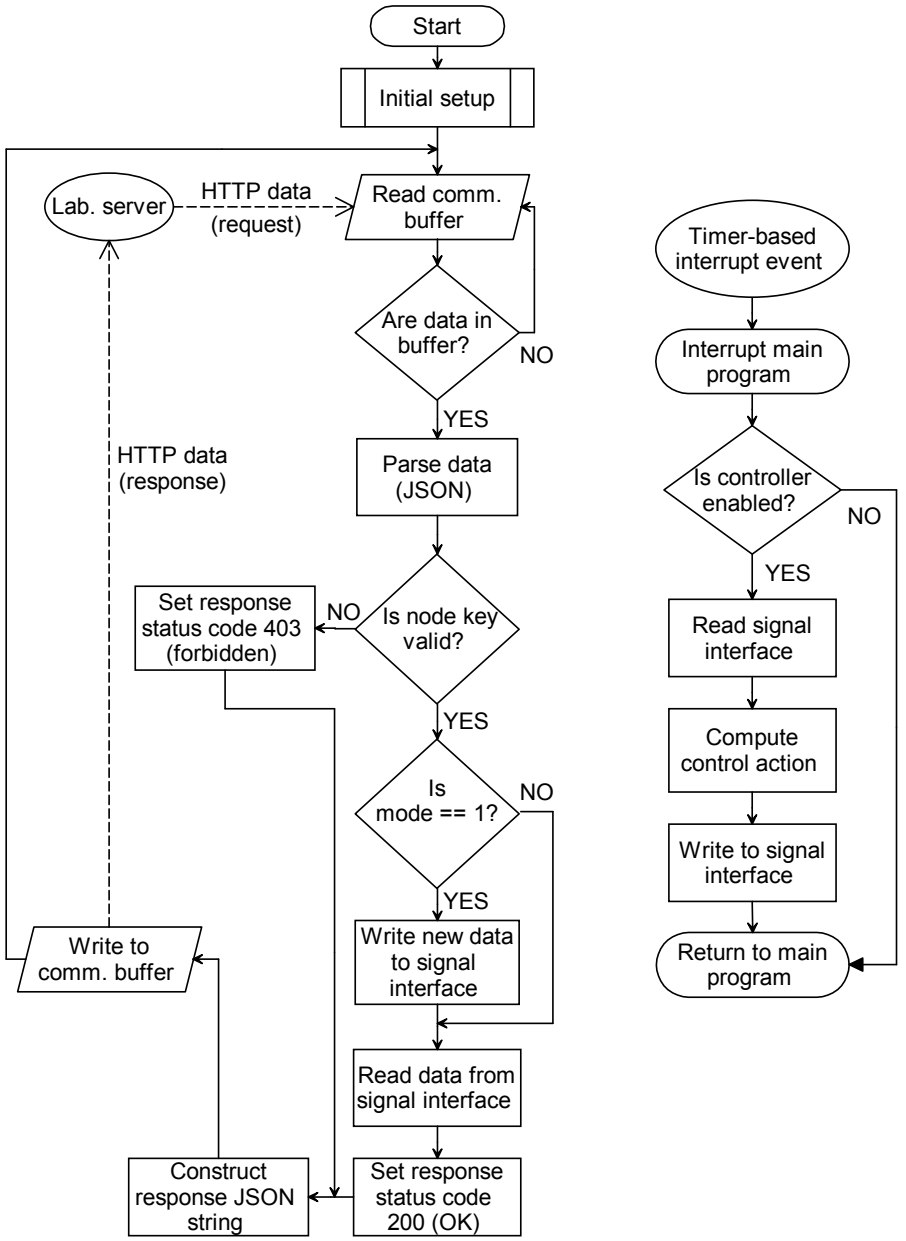


Figure 5.4: Experiment server program flow

to get the information if the client has sent new data which have to be written into the signal interface, or if just requested the reading. Once the operation on signal interface is finished, program generates the HTTP response header with status 200 (OK) and encodes

the signal data into JSON structure which is along with the HTTP header written back into communication buffer. This generates the HTTP response for the laboratory server. The program flowchart is shown in Fig. 5.4.

The execution of automatic control algorithms is handled separately from the main program. They are enclosed in the special function called Interrupt Service Routine (ISR) which is invoked by internal events based on Timers/Counters. Predefined controllers are represented by an integer number (control switch), which is used by user to control which algorithm is computed. If no controller action is required (manual control – default), the control switch is set to 0. Otherwise it uses the controller with the selected identifier. More information about their implementation is given in Chapter 7.

5.1.5 Communication Principles

The implemented communication scenario is based on Arduino UNO development board. The board itself does not provide Ethernet network interface. Therefore, in ArPi Lab architecture planning we made the decision to equip each board with additional communication module (Ethernet Shield Rev-3[5]). Since the communication service of experiment server is implemented directly in micro-controller, and it affects the execution speed of internal logic (signal processing and control algorithms), this scenario is used for any system that do not require data sampling lower than 0.1 s. The second development board, Arduino ÝUN is already equipped with Ethernet module and also the WiFi module, so it can be used for implementation of wirelessly operated experimental nodes.

ArPi Lab setup contains two main network layers (Table 5.2): the external, between client-side communication services and PHP forwarder service on laboratory server operated through Internet; and internal, between laboratory server and experimental nodes operated through private LAN.

Table 5.2: ArPi Lab communication layers

Layer	Internet	Local network
Net. character	public	private
Protocol	HTTP	HTTP
Data structure	JSON	JSON
Comm. model	asynchronous	synchronous
Back-end serv.	Apache + PHP	Embedded HTTP
Credentials	authorization key	private node key

[5] http://arduino.cc/en/Main/ArduinoEthernetShield

The best way how to explain the communication scenario in ArPi Lab architecture is to track the user's request from client application to physical system and then track the response. There are two types of requests produced on the side of the client. First are standard predefined periodic calls to services in order to keep the data flow through architecture. These requests are sent with the constant sampling period and their purpose is to show actual process data in client application (e.g. the data used for dynamic graphs). Second type of request is generated on user's action and its purpose is to apply changes on the side of experiment. For example, if user of laboratory performs the manual change of system's parameter/input, this action is handled by a separate request. Both types are served by an asynchronous calls using the AJAX method from the internal JavaScript logic of client application and they are processed through the same program threads.

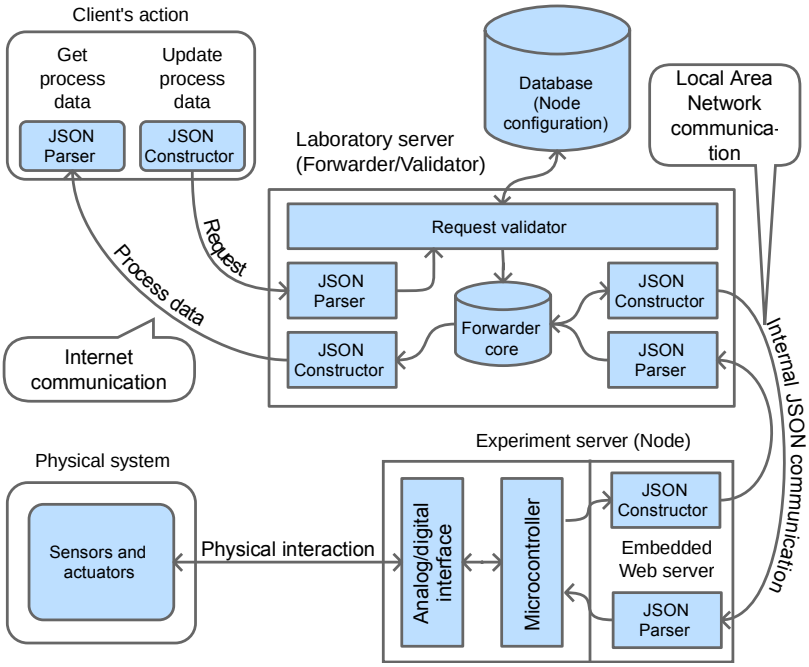


Figure 5.5: Communication scenario of ArPi Lab's architecture

The entire communication scenario is shown in Fig. 5.5. Requests are transferred through HTTP and they are addressed to `forwarder.php` service in laboratory server. The data in request body are structured in JSON representation (Example 5.2).

The `forwarder.php` script parses the body of request to associative array. If the structure cannot be parsed by PHP's method `json_decode`, script generates the JSON response with error status code (list of codes is show in Table 5.3).

Example 5.2: JSON structure of request sent by client application

```
{"client_auth_key" : "3ea68b4c4c",
 "mode" : 1,  // 0 - read signals, 1 - write/read signals
 "signals" : {"bulb_voltage" : 2.14,  // Define changes of
              "led_voltage"  : 5.0,    // signals.
              "test_digital8" : 1
             }
 }
```

Table 5.3: List of error codes

Code	Text code	Meaning
01	MYSQL_CONN_ERR	Database connect error.
02	MYSQL_QUER_ERR	MySQL query error.
03	CLIENT_AUTH_KEY	Authorization error.
04	INVAL_AUTH_KEY	Invalid authorization.
05	JSON_PARSE_ERR	JSON parse error.
([6])	HTTP_STAT_CODE	Exp. sever HTTP status.

If parse procedure is successful, the script creates the connection to MySQL database `arpi_lab` and from table `arpi_nodes` loads the record where field `verification_key` matches the authorization key sent from client application (otherwise, the error code is produced). Then the record `node_id` (which has value n) is used to load the validation data from table `arpi_node_validator_`n. For each signal/variable in JSON structure sent from user (Example 5.2), a validation and numerical conversion procedure is performed. The validation lies in comparison of signals/variables names with those collected from validator table and correctness of numerical data. If names match and requested values fulfill predefined requirements, they are transcribed from external to internal representation (Example 5.3) dedicated for micro-controller of experiment server.

Example 5.3: JSON structure of request for experiment server

```
{"node_key" : "03ae9ec81b",
 "mode" : 1,
 "PWM" : {"3" : 109, "5" : 255},
 "DO" : {"8" : 1}
 }
```

[6] http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

From this point, transferred data are no longer represented by couples of variable name and value in real engineering units, but they are directly dedicated for micro-controller and its signal interface. Signals and variables are split into groups according to the type of their usage. Signals physically operated by pins with pulse modulation are grouped under the `"PWM"` key, digital outputs under `"DO"` key, other variables under `"VAR"` key, and controller switches under `"CS"` key. Internal keys for PWM and digital signals are transcribed from signal name to number of pin, where signal is connected to physical interface. E.g. in Example 5.2 the signal controlling voltage of bulb has variable name (JSON key) `"bulb_voltage"`. For this signal, `arpi_node_validator_`n table provides information that signal is of type PWM, it is connected to the digital pin 3 with 8-bit resolution (0-255), and it can reach values between 0 and 5 V. This information is used for numerical conversion of signal value 2.14 V to integer number 109. The equation used for computation of integer value is

$$V_{\text{int}} = \left\lfloor V_{\text{real}} \frac{R_{\max} - R_{\min}}{V_{\max} - V_{\min}} \right\rfloor, \tag{5.5}$$

where V_{int} is computed integer representation of real value V_{real}, $R_{\max}$ and $R_{\min}$ are integer boundaries of V_{int}, given by pin's resolution (0 and 255 for 8-bit), and $V_{\max}$ and $V_{\min}$ are maximum and minimum values which V_{real} can reach (determined by its physical boundaries). Symbols $\lfloor . \rfloor$ denote integer part of the expression.

When the experiment server receives the data, it performs the processing described in Fig. 5.4. The generated response is transferred through the same way as the request, but since it comes from the trusted source, not additional validation is required on the side of laboratory server. The only task performed before the transcription back to external representation is the conversion of numbers to engineering units. Then the laboratory server generates the response for user.

5.1.6 Client Application

ArPi Lab uses the client side application *ArPiLabClient* as the user interface. *ApPiLabClient* is the general purpose Web-based application written in JavaScript. The application provides a similar concept of on-fly content generation as the application for MHSA-based laboratories. The main difference between *ControlApp* and *ArPiLabClient* is in technologies used for development and communication. While *ControlApp* source codes are written and provided in Java and for implementation compiled into JavaScript, the *ArPiLabClient* is written directly in JavaScript, which simplifies the development phase, debugging, and possible improvements. As mentioned before, ABLA uses the JSON as unified structure of transferred data between different parts of architecture. JSON is native form of object notation for JavaScript and therefore it is used in *ArPiLabClient*

for each communication thread.

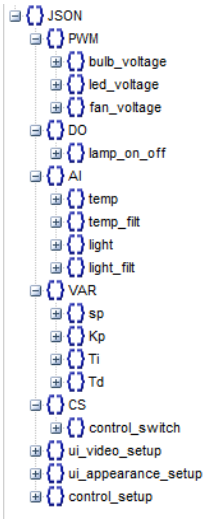
The client application is built on the top of free-to-use and open source technologies. It uses the HTML5 for the static structure of Web page provided to the user. However it contain only a minimal amount of HTML content required for attachment of style sheets and main JavaScript processing files. A set of Cascading Style Sheets (CSS) definitions, inherited from jQuery UI library, are provided along with the client application. From these, user or implementer can choose a specific one that will be applied on Web page. Additional style sheet can be imported through configuration of laboratory. JavaScript source codes of *ArPiLabClient* consist of external libraries and main program. Libraries are used mostly for improvement of user interface in order to add high level of interactivity and responsiveness. Client application uses following JavaScript libraries.

- jQuery[7] – is an open source extension of JavaScript APIs which simplifies code writing and provides the optimized methods for DOM manipulation, handling of events, and AJAX. In most aspects of client side Web development, jQuery is considered a multi-browser library with wide support. *ArPiLabClient* uses jQuery as the main API for application development.
- jQuery UI[8] – is an extension library for jQuery, which provides a set of components for GUI improvement. *ArPiLabClient* uses several widgets from jQuery UI such as dialog windows, menu, progress bar, slider, and drag- and-drop behavior.
- jQuery DialogExtend[9] – is a third party extension for jQuery UI, providing additional features for dialog windows.
- jQuery Flot[10] – is a plotting library for JavaScript. In client application, it is used for generation of online charts.

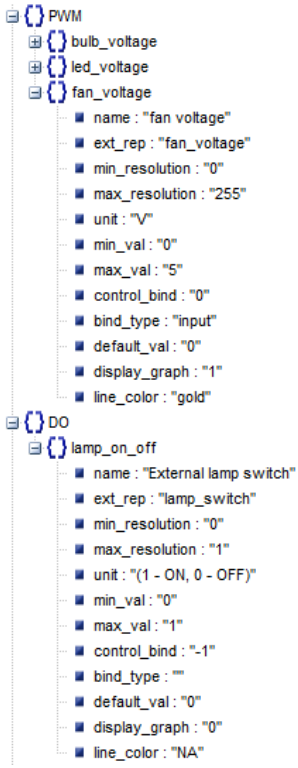
The processing scripts of client application can be divided into several parts, each ensuring different task.

- *main* – is a functional part that controls overall program flow of client application.
- *basicConfig* – is an object that provides the URL to two main services of laboratory server (`config-loader.php` and `forwarder.php`)
- *language* – is an object with description text of GUI components in different languages. The language selector is collected from URL string, where variable *lang* is defined. Currently, the client application supports English and Slovak language.

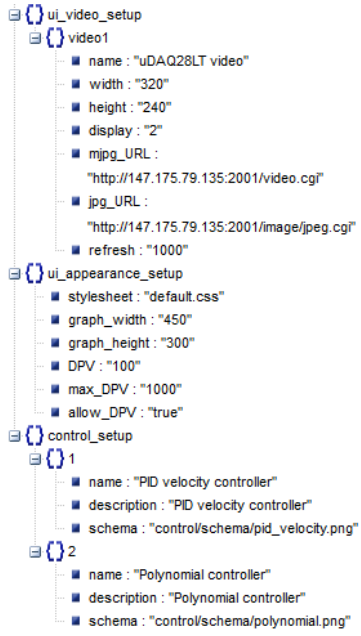
[7] http://jquery.com/
[8] https://jqueryui.com/
[9] https://github.com/ROMB/jquery-dialogextend
[10] http://www.flotcharts.org/



(a) Full configuration of remote laboratory



(b) Configuration of signals and variables



(c) Configuration of appearance, video, and control algorithms

Figure 5.6: JSON structure of laboratory configuration, returned by laboratory server

- *configLoader* – is a functional part gathering all necessary information for *ArPi-LabClient* operation and GUI construction. During the initiation of client application, it collects the configuration data for particular laboratory setup from `config-loader.php` service.

- *DataCollector* – is a class used for acquisition of operational and process data from experiment. It consists of data container and methods for storing and loading data. During the laboratory session, each sample collected form experiment is stored in object of this class.

- *DataTables* – is a set of classes used for construction of tables in GUI. Each table contains a different type of signals or variables, therefore each has its own class and constructor. Client application distinguishes four types of tables. *Input table* allows user to directly manipulate input signals of controlled process. *Output table* shows actual values of system's outputs. *Var table* allows user to change additional variables associated to controlled process (e.g. controller parameters). *Digital table* stores additional signals/variables which can reach only binary values. Each digital signal/variable can be manipulated by toggle switch button.

- *menu* – is a functional part that creates a menu bar on the top of GUI.

- *videoFrames* – is a functional part that constructs windows with attached video streams from remote IP cameras.

- *GraphConstructor* – is a class that ensures layout of online charts with process data. Each chart is attached in separate draggable window.

- *ControlWindow* – is a class responsible for layout and operation of window that provides automatic control algorithms.

The program flow is similar to *ControlApp*. First of all, client application reads the authorization key that is provided by a third side (ULM system) in the URL. Then the application collects the URLs of services of laboratory server. It performs the AJAX request to `config-loader.php` with attached authorization key. The request data uses the same structure as the example 5.2, but contains only `"client_auth_key"` entry. Configuration loader will return the full set of configuration data for that particular laboratory, since the authorization key is bound to specific configuration tables in database. The example of configuration for laboratory equipped with thermo-optical device is shown in Fig. 5.6.

After the *configLoader* successfully receives data, a whole JSON structure is parsed into the JavaScript object, which is later used by particular functional parts to create

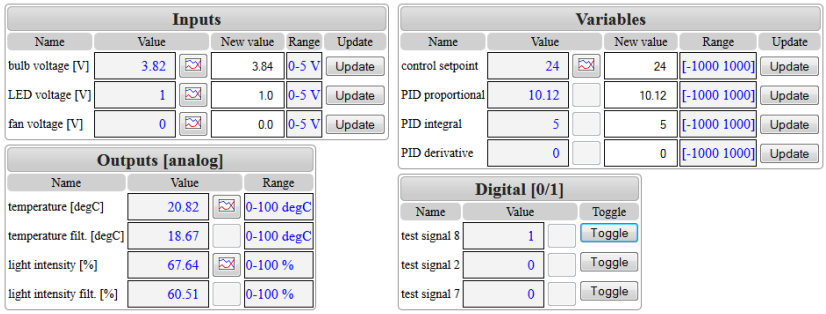


Figure 5.7: Layout of tables with signals and variables

user interface. The menu is first graphical component that is created because it holds the references for other components. Using the information about signal/variables and their properties, for each type of them a separate table is constructed. A set of tables for particular GUI is shown in Fig. 5.7.

When data tables are successfully created, the main data holder object is created from `DataCollector` class. This object holds all data samples continuously collected from experiment, which are used also in tables and graphs.

Then `GraphConstructor` class creates windows with graphs of selected signals and variables and attach them to the GUI (Fig. 5.8).

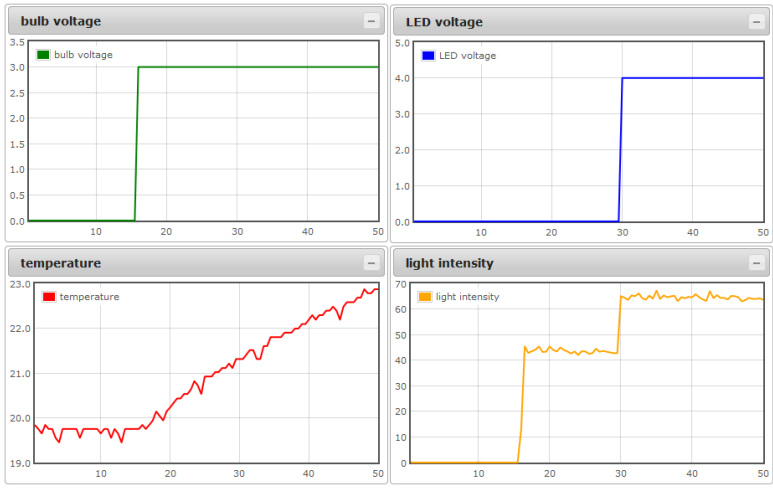


Figure 5.8: Layout of windows with graphs

Video streams are displayed also in separated windows (Fig. 5.9). In most cases, remote experiments do not require more than one video stream. However, *ArPiLabClient*

does not limit the number of video streams that can be displayed in GUI. Currently, *ArPiLabClient* supports each type of IP camera or streaming service that can provide video as Motion JPEG or single-image snapshots.

Figure 5.9: Wireless IP camera used in ArPi Lab (left) and its image displayed in GUI (right)

The last two components that are constructed in GUI initiation before the client application is put into the operation are the control window and data download window. First one (Fig. 5.10) allows user to switch from manual mode into automatic control mode, by selection, configuration and execution of specific controller. In this window, each controller provides an scheme of control loop, where user can select which input/output signals will be connected to it. Controller parameters appear in table with variables (Fig. 5.7) once the user selects a specific controller.

When the initiation of GUI and configuration of client application is finished, the main operational script is run with predefined frequency of execution. In each execution, application gathers an actual data sample from experiment and updates tables, graphs, and control window. Sampling timing can be set up in by implementer to achieve collection of appropriate amount of data that are necessary for user to understand nature of experiment and analyze it properly. Sampling of client application could be selected with the respect to the smallest time constant of the experiment that affects measured data.

Data are provided to the user in the same way as in MHSA's *ControlApp*, by a data download window. However as the part of future work, we plan to develop an external data-logging system with database that will allow user to access the measured data anytime.

While data collection is automated and performed with specific timing, the requests of user (manipulation of experiment) are based on real actions handled by JavaScript

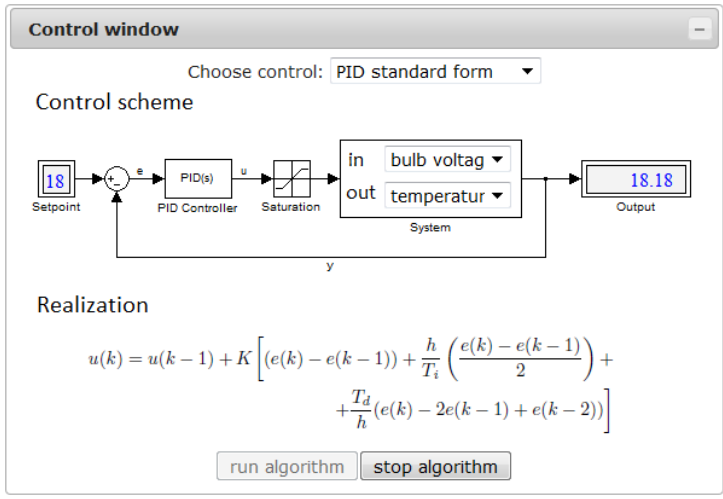


Figure 5.10: Control window with selected PID controller

events. They are sent to controlled system immediately after user performs the action.

5.2 Power Management

In laboratory practice, but also in common life, we meet situations when electronic equipment fails and the only option is restart. This issue is typical for networking devices like routers and switches, but can occur for electronics in general. In remote laboratories, this issue is even more significant. Failure of networking device can cause the outage of connection between students and labs, and in worse case, the overall loss of control/administration over the labs.

ArPi Lab uses a sophisticated method of power management to deal with hardware failures and power related issues. Each functional part of hardware architecture is powered by electric source that can be managed and monitored remotely (Fig. 5.11). For this purpose we use the sets of programmable power outlet strips (Gembird Silver Shield), which are controlled by Power Management Server (PMS) through USB. PMS is the Apache based HTTP Web server running on Raspberry Pi computer. Unlike other hardware parts of ArPi Lab architecture, the PMS is the only device in laboratory that uses separate power and network line for its operation.

If an outage of hardware is reported to administrator, he/she can log-in to PMS's terminal through Secure Shell (SSH) and can restart or power off the faulty device.

Even if the power management is fully operable through SSH, we plan to extend this system by a secured Web interface for power lines control. Moreover, we consider

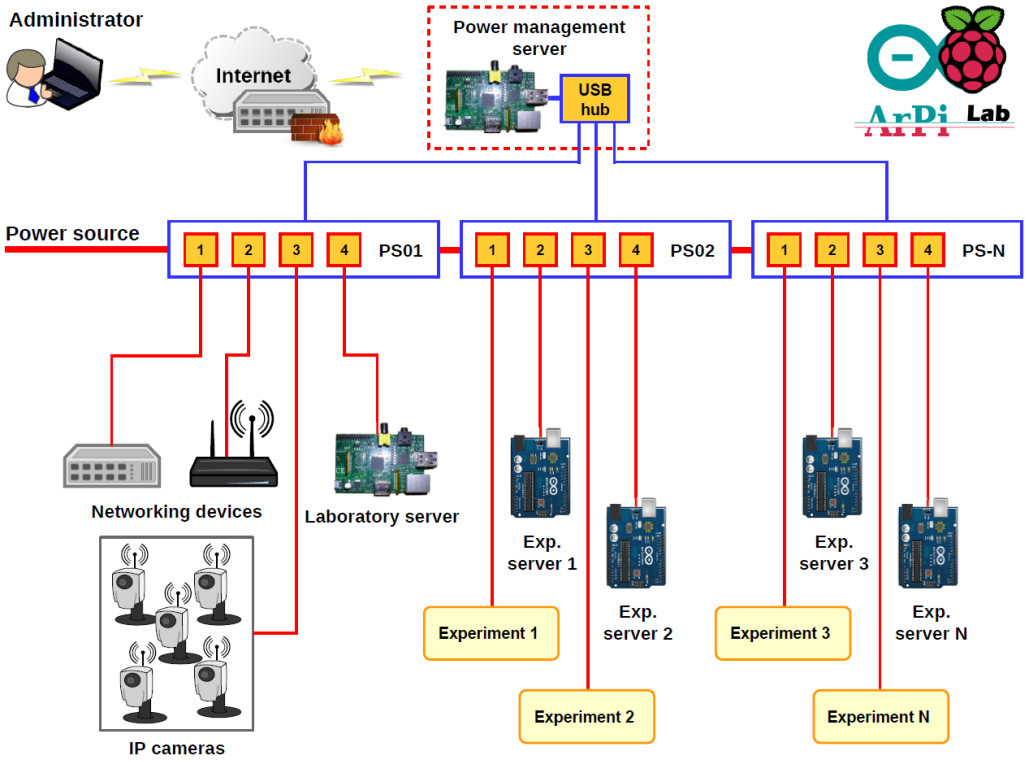


Figure 5.11: ArPi Lab power management

to implement automatic fault detection to run in PMS for scanning both hardware and communication issues.

5.3 Application Scope, Advantages, and Limitations

The application scope of ABLA is very similar to MHSA. While the implementation of specific experimental system in MHSA is dependent on PLC connectivity, for ABLA it relies on the connectivity of experiment server (Arduino board). The experiment node equipped with Arduino UNO is capable to control laboratory equipment by low-current voltage signals in range of 0-5 V. It can manipulate with 6 PWM-driven actuators, another 8 digital signals, and read 6 analog sensors. The Arduino boards are generally suitable for control of various electro-mechanical systems such as DC motors (Neto et al., 2012), stepper motors (Barber and Crespo, 2013), or more complex systems such as robots (Al-Busaidi, 2012).

The main advantages/limitations of ABLA are summarized in following lists.

Advantages of ABLA:

- experiment server is a low-cost device (Arduino board),
- multipurpose concept reduces implementation time and effort,
- fully configurable and universal client application,
- high level of branching capability without significant grow of costs per each new laboratory,
- reproducibility,
- laboratory server runs on low-cost device (Raspberry Pi),
- usage of open-source technologies.

Limitations of ABLA:

- low-cost hardware does not provide enough performance for computational tasks of high complexity,
- choice of hardware for experiment server is limited to Arduino boars with Ethernet capability,
- limited memory of micro-controller (32kB for program and 2kB for dynamic variables in ATMega328p),
- control algorithms must be hard-coded into micro-controller,
- communication is a part of micro-controller's program (for Arduino UNO),
- signal interface of Arduino boards does not provide such wide capabilities as industrial solutions.

Chapter 6

Upper Level Management

The Upper Level Management (ULM) is a concept of overall control of one or a set of RLs. It is dedicated to manage experimental resources, users, access model, usage, permissions and most of all, the publishing of RLs. In the realm of information technologies designed for educational purposes, two main classes of ULM systems are available: the Learning Management Systems (LMSs) and Remote Laboratory Management Systems (RLMSs).

6.1 Learning Management Systems

The LMS are Web-based information systems that allow to merge information sources and activities focused on similar topic into groups. They provide all of above mentioned features, but they are mainly aimed on educational purposes. The most renown LMSs currently available are Blackboard Learn[1] and JoomlaLMS[2] from commercial systems and Moodle[3] from open source and free-to-use systems. The main advantage of open source systems is that they can be extended by new modules and plug-ins developed by community. In the realm of on-line experimentation LMSs are widely used for publishing of virtual simulation and also remote laboratories. In work by Fernandez et al. (2012a), the authors show an interoperability platform, which consists of modules, intended for interconnection of various types of virtual and remote laboratories with LMS Moodle.

6.2 Remote Laboratory Management Systems

The RLMSs are specific-purpose management systems designed for usage on the top of RLs. The main idea of such systems is to ensure following features.

[1] http://www.blackboard.com/Platforms/Learn/Overview.aspx
[2] http://www.joomlalms.com
[3] http://moodle.com

- *On-line publishing of laboratories.* Since RLs are provided as Internet tools, they should be both, visible and accessible for every person interested in their usage. While the choice of accessibility level is up to the providers of laboratories, visibility is necessary for creation of awareness about their existence. The publishing model should allow users to find the RLs though search engines or at least as the reference from topic-related Internet sources. In contrast with single-laboratory publishing, the RLMSs are able to attract a high numbers of users, and as the result, increase the self indexing in search engines like Google, Yahoo!, Baidu, Bing, Ask, and others.

- *Provision of APIs for RL development.* Most of renowned RLMSs provide a set of programming tools and interfaces for development and implementation of laboratories. From this point of view, they act as modular environments for interconnection of client-side and experiment-side parts of RLs, which can be based on different technologies. This feature allows developers to incorporate they existing laboratories into one management system, even they have been designed to work as separate sources.

- *Typological grouping of laboratories.* In educational context, it is often very important to interconnect the remote laboratories with the specific subject. The RLMSs allows educators to group laboratories with similar topic (like electronics, physics, control systems, etc.) and provide them as a set of tools for laboratory exercises.

- *Allocation of users to groups and roles.* This feature is very common in all information systems where specific content is provided to user or a group of users, and it is based on permissions given by an administration authority (same principles as for LMSs).

- *Management of users' accesses and permissions.* From the nature of RL's operation it is obvious that users' accesses must be managed in whole different way than accesses to other kind of software-driven systems. It is caused by the presence of real laboratory equipment at the end of RL's architecture. The majority of remote experiments can be operated only by a one user at the time. This rule is often applied for security reasons, for example to avoid a contradictory effects caused by two or more users operating in RL at the same time, which causes at least corruption of experiment.

- *Sharing of resources.* Most of RLMSs are able to provide their resources to other management systems. This way, users of one RLMS can access laboratories published by other institution. Federation of RLs is nowadays one of the most frequented techniques of their expansion across academic area. The main benefits are

that even institutions with limited financial resources or developers base (which are essential for implementation of RLs to education) can make an agreements with other institutions and use their existing laboratories as their own.

6.3 RLMS WebLab-Deusto

WebLab-Deusto[4] (Fig. 6.1) is an open source[5] RLMS developed in the University of Deusto. The early versions of WebLab-Deusto were in the form of *ad hoc* RLs (García-Zubia et al., 2005). Since then, the whole system has undergone a significant development (Orduña et al., 2011) and became one of the most respected and used RLMSs in the academic area. A detailed description of WebLab-Deusto is provided in Orduña (2013).

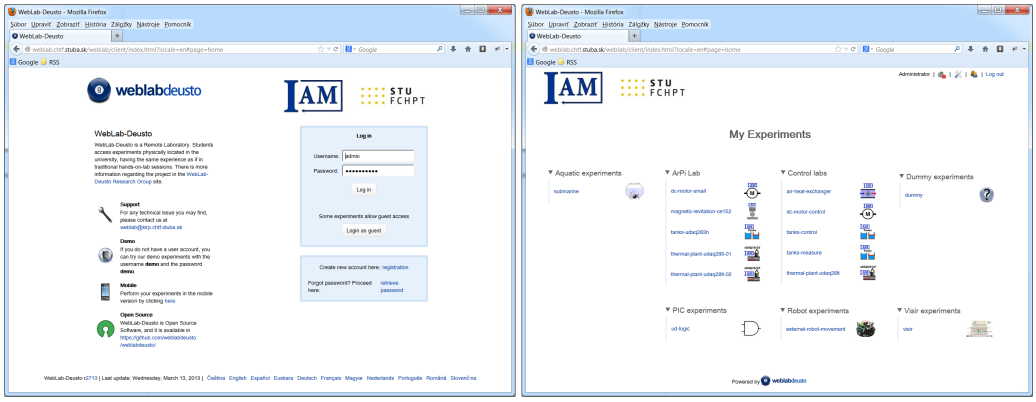


Figure 6.1: Login page of WebLab Deusto (left) and list of available laboratories (right)

In addition to the University of Deusto, several other educational institutions have adopted the WebLab-Deusto, mostly European universities, but also lower level institution such as high schools and even secondary schools. The Slovak University of Technology in Bratislava is the one of universities that participate on federation of remote laboratories using this system.

6.3.1 Laboratory Management and User Management

WebLab-Deusto allows administrator to manage instances of laboratories, in the meaning of visibility, accessibility, and level of privileges for particular user, group of users, or a role. Laboratories can be divided into groups by a specific type of experiment or by a application area. They can be e.g. robotic experiments, automatic control laboratories,

[4] http://www.weblab.deusto.es
[5] https://github.com/weblabdeusto/weblabdeusto

laboratories focused on electronics, etc. At the management level, administrator grants the permissions for users to access laboratories for a specific time and with a particular priority index. For example, students from home university have higher priority to access laboratory than other registered user, and they have higher priority than guests. The priority applies when several users with different privileges request the usage of the same laboratory. Since WebLab-Deusto uses queuing of users (alternative to calendar booking), they are additionally ordered by priority indexes (Fig. 6.2).

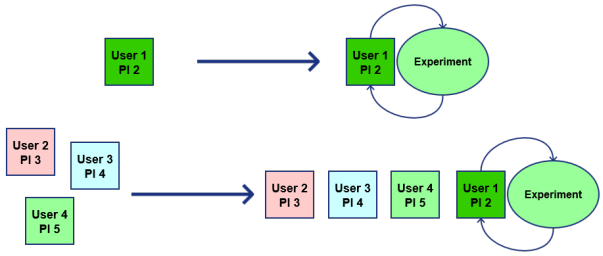


Figure 6.2: Queuing of users with different priority index

WebLab-Deusto also provides another feature of user access management, the load balance. It is applied in the situations when several users request to access the same experiment and system provides more than one copy of such experiment. Then accesses are distributed over the copies to serve users in less time. This principle is shown in Fig. 6.3.

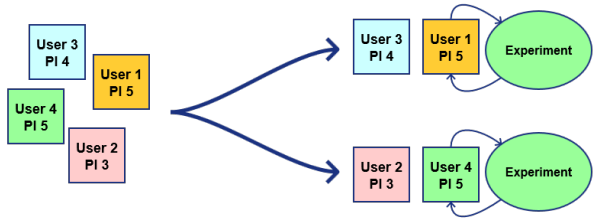


Figure 6.3: Distribution of users among more copies of the same laboratory

6.3.2 Remote Laboratory Development

Although RLMS WebLab-Deusto is primarily the management system, one of the main features is a provision of APIs for experiment development. It provides interfaces for both, connection of existing laboratories as well as development of new. In current ver-

sion, WebLab-Deusto allows developers to implement their own laboratories on different software technologies. They are JavaScript, Java and Adobe Flash for client side and C, C++, Java, LabVIEW, .NET, Node.js and Python for server side.

There are two types of principles how WebLab-Deusto can handle connected laboratories. Firstly, it can be used as pure management and scheduling system that handles only the access of users, while RL relies on different operational architecture. These laboratories are called unmanaged laboratories. Second type is the managed laboratory which is build on WebLab-Deusto APIs, while the whole communication is served through management system itself.

6.3.3 Inter-Institutional Usage

Inter-institutional collaboration is one of the most actual topics in the world of remote experimentation these days. Since the academic area provides a relatively high amount of e-learning tools and on-line experiments (virtual and remote laboratories) that are provided only for a certain groups of users, it is necessary to reduce their local isolation and expand them to the outside world.

> *"A common feature of most existing remote laboratories, whether for educational or industrial purposes, is that they offer stand-alone solutions, with limited or no capability to cooperate with other platforms. Most of these solutions are developed as special or ad hoc solutions relying on different types of technologies and both computer and human languages and often use heterogeneous and incompatible hardware and software tools. In that sense, the major challenge in current remote laboratories appears to be the lack of standardization, impeding the modularity, portability, and scalability of solutions, as well as interoperability between different solutions."*
>
> Gomes and Bogosyan (2009)

One of the most valuable features of WebLab-Deusto is capability to share remote laboratories between different instances of the same management system, while each can be deployed on different university and in the different country. This highly desired property is beneficial for any institution that wants to provide its remote laboratories in the world (advertise them), but even more for those institutions who cannot afford their own remote experiments. The principle of remote laboratory sharing through WebLab-Deusto is called federation. It is based on an agreement between two or more institutions, but it is fully managed by the system.

Each RL that has been developed in this work (Chapter 8) are provided to the University of Deusto. In return University of Deusto provides their laboratories (mostly focused on electronics and robotics) to the Slovak University of Technology in Bratislava (STU).

The WebLab-Deusto allows not only the sharing of resources, but also re-sharing. For explanation, if STU requests the University of Deusto (UD) for provision of new laboratory and UD will agree, the WebLab-Deusto (WD) instance of STU will have the permission to access the WD at UD and use that particular laboratory. In this mechanism the consumer's WD system acts as the regular user for provider's WD system. In addition, once the particular instance is allowed to consume laboratories of other system, it is also allowed to re-share them to third party, but only with as high level of permission as given by provider. This mechanism is shown in Fig.6.4.

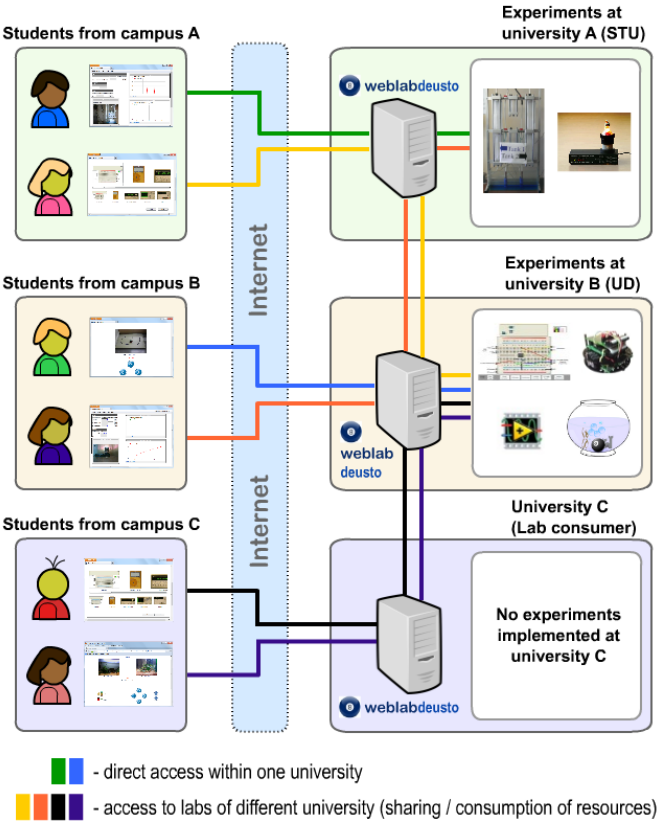


Figure 6.4: Federation of remote laboratories between three universities

Part III

Applications

Chapter 7

Implementation of Control Algorithms

This chapter deals with common problems of control implementation for real applications in remote laboratories. As it is well known, a significant part of control theory deals with mathematical expressions on theoretical level, using apparatus from differential calculus, which is very useful for control design of modern applications. However, a time when continuous physical controllers have been used in a large scale has passed, and now, most of applications rely on hardware that operates, based on its physical nature, in discrete time. Therefore the implementer of control has two options: to design controller directly with the respect to discrete operation; or to design a continuous controller and then to choose an appropriate method of implementation on discrete-working hardware.

The following sections 7.1–7.3 provide an overview of control scenarios used in developed remote laboratories. Each controller is described in general form and form suitable for implementation. Additionally, in section 7.4 a discussion of control loops execution with respect to time is provided.

7.1 PID Controllers

PID controllers belong to large class of controllers with three separate action components: proportional, integral and, derivative. The mathematical realization of PID controller can differ from type to type, but from all of them three forms are most commonly used in practice. They are the standard form, parallel form, and serial form. The theory of PID controllers described in this section is well elaborated in Åström and Hägglund (1995).

7.1.1 Standard Form

The standard form (often called the ideal form) is most commonly used form of PID controller realization in practice and theory. It is characterized by the interaction between

its action components. In this form, the proportional component multiplies both integral and derivative components. The schematic structure of standard from is show in Fig 7.1.

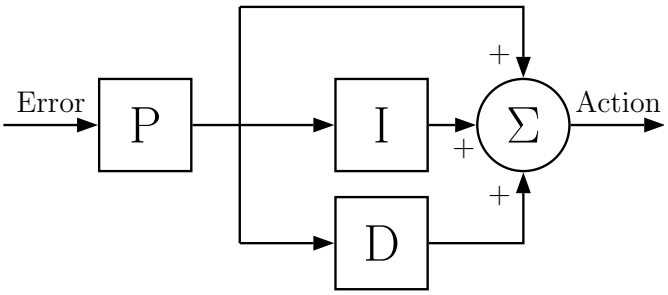


Figure 7.1: Standard form of PID controller

Assuming that input to controller structure is control error $e(t)$, proportional component is the gain $P = Ke(t)$, integral component I can be written as

$$I = \frac{1}{T_i} \int_0^t e(\tau) \mathrm{d}\tau, \tag{7.1}$$

and derivative component D as

$$D = T_d \frac{\mathrm{d}e(t)}{\mathrm{d}t}, \tag{7.2}$$

where K is the gain of controller, T_i is integral time constant, and T_d is derivative constant. Then the output of controller (action) $u(t)$ can be expressed as

$$u(t) = K \left(e(t) + \frac{1}{T_i} \int_0^t e(\tau) \mathrm{d}\tau + T_d \frac{\mathrm{d}e(t)}{\mathrm{d}t} \right). \tag{7.3}$$

7.1.2 Parallel Form

Parallel PID (Fig. 7.2) is the non-interacting form of realization. In this form the P, I, and D components are separated and final action is computed as sum of three independent actions. Parallel PID is often used for practical implementation due to its ease of realization and possibility to tune the parameters independently on each other.

Unlike the standard form, in parallel PID the control error $e(t)$ is directly propagated through all three components separately. Then the P component is $P = K_p e(t)$, where K_p is the gain of proportional action. Other two components use the same expressions as the standard form (equations (7.1) and (7.2)). The mathematical expression for controller's output $u(t)$ will be

$$u(t) = K_p e(t) + \frac{1}{T_i} \int_0^t e(\tau) \mathrm{d}\tau + T_d \frac{\mathrm{d}e(t)}{\mathrm{d}t}. \tag{7.4}$$

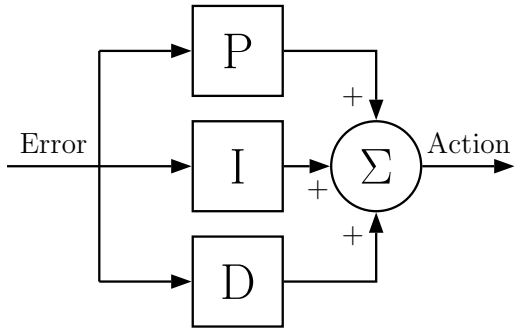


Figure 7.2: Parallel form of PID controller

7.1.3 Serial Form

Serial form of PID controller (Fig. 7.3) is used mostly in applications of process industry. This form can be described as the serial connection of PD and PI controller.

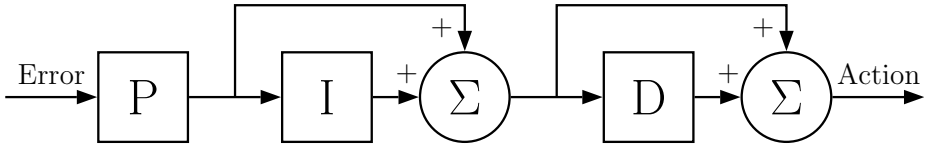


Figure 7.3: Serial form of PID controller

The output of serial PID controller is defined as

$$u(t) = K\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)\mathrm{d}\tau\right)\left(1 + T_d\frac{\mathrm{d}e(t)}{\mathrm{d}t}\right). \tag{7.5}$$

7.1.4 Setpoint Weighting

PID controllers are often implemented in two-degree-of-freedom control scheme using the setpoint weighting. For standard form (7.3), the weighting factors $\beta \in \langle 0; 1\rangle$ and $\gamma \in \langle 0; 1\rangle$ are defined to distribute the propagation of control error (equations (7.6) and (7.7)) through P and D components.

$$e_P(t) = \beta w(t) - y(t) \tag{7.6}$$

$$e_D(t) = \gamma w(t) - y(t) \tag{7.7}$$

Then, (7.3) can be rewritten as

$$u(t) = K\left(e_P(t) + \frac{1}{T_i}\int_0^t e(\tau)\mathrm{d}\tau + T_d\frac{\mathrm{d}e_D(t)}{\mathrm{d}t}\right). \tag{7.8}$$

This form can be presented as two controllers and parameters β and γ allow to handle reaction to setpoint and disturbance changes separately.

7.1.5 Implementation of PID Controller

Since all three mentioned forms of PID controllers (equations (7.3), (7.4), and (7.5)) are continuous time control laws, they cannot be directly implemented as a set of algorithmic expressions. They must be transferred to discrete time before their implementation into micro-controller's logic. By discretization, a continuous time domain became a set of points with a fixed step h that represents absolute distance between two neighboring points k and $k+1$.

Then, the control error function can be rewritten as

$$e(t) = e(k). \tag{7.9}$$

The integral term can be approximated by a summation of partial areas computed by trapezoidal rule (other possibilities include forward, backward, or more advanced forms, see for example Bobál et al. (2005))

$$\int_0^t e(\tau)d\tau \approx h\sum_{i=1}^{k}\frac{e(i)+e(i-1)}{2}, \tag{7.10}$$

and derivative term can be approximated by a backward difference

$$\frac{de(t)}{dt} \approx \frac{e(k)-e(k-1)}{h}. \tag{7.11}$$

Then the standard form of PID controller (7.3) can be rewritten as

$$u(k) = K\left(e(k) + \frac{h}{T_i}\sum_{i=1}^{k}\frac{e(i)+e(i-1)}{2} + T_d\frac{e(k)-e(k-1)}{h}\right). \tag{7.12}$$

The equation (7.12) is called position form of digital PID control algorithm. However this form is unsuitable for direct implementation in micro-controllers, because it requires all previous values of control error in each step. This would have an undesirable effect on micro-controller's memory. More effective form of this controller can be achieved by its transcription into incremental form (often called velocity form of digital PID). Incremental

form is derived from (7.12) defining the difference of controller's output between two steps $\Delta u = u(k) - u(k-1)$.

$$u(k) = u(k-1) + K\left[(e(k) - e(k-1)) + \frac{h}{T_i}\left(\frac{e(k) - e(k-1)}{2}\right) + \right. \\ \left. + \frac{T_d}{h}(e(k) - 2e(k-1) + e(k-2))\right] \tag{7.13}$$

The velocity form of digital PID algorithm (7.13) has several beneficial properties contrary to position form (equation (7.12)). It requires only one previous value of control action $u(k-1)$ and three values of control error, the actual $e(k)$ and two previous $e(k-1)$ and $e(k-2)$. Another important property is the bumpless transfer, which avoids bumps of control signal in situations such as online changes of parameters, switching of controller's mode (on/off), or changing between different controllers. This form can be directly implemented into computer or micro-controller as a set of algorithmic expressions.

7.1.6 Algorithmization of PID Controller

The algorithm for computation of control law can be split into two parts. First is the initialization, where all controller parameters and initial values of control actions and errors have to be set. The second part is periodical execution of control task, which involves measurement of system's output (controlled variable), computation of control error, computation of control action, and update of system's input (manipulated variable). This second part of control algorithm can be handled in two different ways with respect to timing: with variable step size; and as real time control. Each algorithm implemented in ArPi Lab is applied in real time (Section 7.4). Since the control law requires values of errors and actions in previous steps, they must be defined in first part of algorithm. Initial states of control actions can be set to zero values, and for control errors to either zero values or values of actual error between setpoint and measured output of system (lessen the bump effect). The algorithmization of PID controller in velocity form (Equation (7.13)) can be then written as follows.

```
Initialization:
  define parameters K, Ti, Td, h
  define setpoint w
  read system's output y
  define e(k-1) = e(k-2) = w - y
  define u(k-1) = 0

Loop:
```

```
read system's output y(k)
compute control error e(k) = w − y(k)
compute control law:
   P(k) = K(e(k) − e(k − 1))
   I(k) = Kh/Ti((e(k) − e(k − 1)/2))
   D(k) = KTd/h(e(k) − 2e(k − 1) + e(k − 2))
   u(k) = u(k − 1) + P(k) + I(k) + D(k)
apply saturation for u(k) (if needed)
update system (apply u(k) on input)
shift k → k − 1
Go to Loop
```

PID controllers can be tuned using various techniques. Traditional tuning methods, which are still very popular and often used in practice are for example Ziegler-Nichols (Ziegler and Nichols, 1942) and Cohen-Coon (Cohen and Coon, 1953). Some modern methods are based on iterative and learning approaches (Abedini and Zarabadipour, 2011; Jian-Xin and Deqing, 2007) or autotuning approaches using relay-based feedback (Prokop et al., 2005) and swarm algorithms (Zhu et al., 2013). An exhaustive survey of PID tuning methods can be found in O'Dwyer (2009).

7.2 Transfer Function and State-Space Representation of Systems

Transfer function is the frequency domain expression of Linear Time Invariant (LTI) system. In control theory, a wide spectrum of controller design methods are heading to this form. The relation between frequency domain and state space representations of LTI systems is described in detail in Mikleš and Fikar (2007). A general form of transfer function is

$$G(s) = \frac{Y(s)}{U(s)} = \frac{b_m s^m + b_{m-1} s^{m-1} + \cdots + b_1 s + b_0}{a_n s^n + a_{n-1} s^{n-1} + \cdots + a_1 s + a_0} = \frac{B(s)}{A(s)}, \tag{7.14}$$

where $Y(s)$ is the Laplace transform of system's output and $U(s)$ is the Laplace transform of system's input in time domain $y(t)$ and $u(t)$, for zero initial conditions. Physical properness requirement dictates that $n \geq m$.

In order to obtain a form suitable for algorithmic implementation it is necessary to transfer equation (7.14) into time domain by the inverse Laplace transform

$$\mathcal{L}^{-1}\{Y(s)A(s)\} = \mathcal{L}^{-1}\{U(s)B(s)\}, \tag{7.15}$$

which leads to a differential equation

$$a_n y^{(n)}(t) + a_{n-1} y^{(n-1)}(t) + \cdots + a_1 y'(t) + a_0 y(t) = \\ b_m u^{(m)}(t) + b_{m-1} u^{(m-1)}(t) + \cdots + b_1 u'(t) + b_0 u(t). \tag{7.16}$$

Since (7.16) requires computation of n-th derivative of $y(t)$ and m-th derivative of $u(t)$, it is transformed to a state-space form, which is much easier to solve by numerical algorithms. Using the substitution

$$x_1(t) = y(t),\ x_2(t) = y'(t),\ \ldots,\ x_n(t) = y^{(n-1)}(t), \tag{7.17}$$

the equation (7.16) can be rewritten as a set of ordinary differential equations in state-space form (Equations (7.18)–(7.22)).

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}u \tag{7.18}$$

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_{n-1} \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} -\frac{a_{n-1}}{a_n} & -\frac{a_{n-2}}{a_n} & \cdots & -\frac{a_1}{a_n} & -\frac{a_0}{a_n} \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} u \tag{7.19}$$

$$y = \boldsymbol{C}\boldsymbol{x} + Du \tag{7.20}$$

$$C = \begin{bmatrix} \frac{b_{m-1}}{a_n} - \frac{a_{n-1}b_m}{a_n^2} & \frac{b_{m-2}}{a_n} - \frac{a_{n-2}b_m}{a_n^2} & \cdots & \frac{b_1}{a_n} - \frac{a_1 b_m}{a_n^2} & \frac{b_0}{a_n} - \frac{a_0 b_m}{a_n^2} \end{bmatrix} \tag{7.21}$$

$$D = \frac{b_n}{a_n} \tag{7.22}$$

7.2.1 Implementation of State-Space

Previous section has shown that each controller defined as the ratio of two polynomials can be transferred into a set of Ordinary Differential Equations (ODEs) (equations (7.23)–(7.25)).

$$\dot{x}_1 = -\frac{a_{n-1}}{a_n} x_1 - \frac{a_{n-2}}{a_n} x_2 - \cdots - \frac{a_1}{a_n} x_{n-1} - \frac{a_0}{a_n} x_n + u \tag{7.23}$$

$$\dot{x}_2 = x_1 \tag{7.24}$$

$$\vdots$$

$$\dot{x}_n = x_{n-1} \tag{7.25}$$

Since the computer-based implementation of problem defined in the form of ODEs must be solved numerically, an appropriate solver method must be chosen. Because state space representation is a set of ODEs of 1st order, a non-stiff method Runge-Kutta 4 (RK4) with fixed step (Butcher, 2008) has been selected as numerical solver. Using the numerical solver RK4 is simplified for time invariant systems

$$x'(t) = f(x(t)),\ x(t_0) = x_0, \tag{7.26}$$

for step size $h = t(k) - t(k-1)$, the numerical solution of ODE (7.26) will be

$$x(k) = x(k-1) + \frac{1}{6}h(p_1 + 2p_2 + 2p_3 + p_4), \tag{7.27}$$

where

$$p_1 = f(x(k-1)) \tag{7.28}$$

$$p_2 = f(x(k-1) + \frac{h}{2}p_1) \tag{7.29}$$

$$p_3 = f(x(k-1) + \frac{h}{2}p_2) \tag{7.30}$$

$$p_4 = f(x(k-1) + hp_3). \tag{7.31}$$

The original problem of differential calculus is reduced to algebraic problem and can be directly implemented as a set of algorithmic expressions into micro-controller.

7.2.2 Algorithmization of Transfer Function in State-Space Form

Numerical solution of above mentioned mathematical problem will give the control action in k-th step of sampling. To perform this task, control system requires its previous states $\boldsymbol{x}(k-1) = \{x_1(k-1), x_2(k-1), \cdots, x_n(k-1)\}$ and actual control error $e(k)$. If transfer function is used as control system, the control error e substitutes the input u in equation (7.23) and the output of control system y (equation (7.20)) represents the control action u (input to controlled system). Then the algorithmization of transfer function controller computed in state space is following.

```
Initialization:
  define TF numerator b̄ = (b_0, b_1, ··· , b_m)
  define TF denominator ā = (a_0, a_1, ··· , a_n)
  define state vector x̄ = (0, 0, ··· , 0)
  for i = 1 to n:
    calculate C_i (from (7.21))
  define D (from (7.22))
  define setpoint w
```

```
  define step size h

Loop:
  read system's output y(k)
  compute control error e(k) = w − y(k)
  compute control law:
    define X = 0
    for i = 1 to n:
      p̄ = (p1, p2, p3, p4) for ẋi (from 7.28 − 7.31)
      xi(k) = xi(k − 1) + h/6(p1 + 2p2 + 2p3 + p4)
      X = X + Ci xi(k)
    u(k) = X + D e(k)
  apply saturation for u(k) (if needed)
  update system (apply u(k) on input)
  Go to Loop
```

7.3 Discrete Transfer Function

The most suitable forms of controllers for implementation on micro-controllers are those defined in discrete time. The discrete transfer function is suitable form because it can be designed both, directly with the respect to control sampling or in continuous time and then discretized. Discrete transfer function can be written in general form as

$$G(z) = \frac{Y(z)}{U(z)} = \frac{b_0 + b_1 z^{-1} + \cdots + b_{m-1} z^{-m+1} + b_m z^{-m}}{a_0 + a_1 z^{-1} + \cdots + a_{n-1} z^{-n+1} + a_n z^{-n}} = \frac{B(z)}{A(z)}. \tag{7.32}$$

Using the inverse Z-transform

$$\mathcal{Z}^{-1}\{Y(z)A(z)\} = \mathcal{Z}^{-1}\{U(z)B(z)\} \tag{7.33}$$

for the initial conditions $\forall\, k < 0 : y(k) = 0 \wedge u(k) = 0$, we obtain the difference equation

$$\begin{aligned} &a_0 y(k) + a_1 y(k-1) + \cdots + a_{n-1} y(k-n+1) + a_n y(k-n) = \\ &b_0 u(k) + b_1 u(k-1) + \cdots + b_{m-1} u(k-m+1) + b_m u(k-m) \end{aligned} \tag{7.34}$$

This form can be rewritten into (7.35), which is suitable for direct implementation as the algorithmic expression.

$$y(k) = \sum_{i=1}^{n} -\frac{a_i}{a_0} y(k-i) + \sum_{j=0}^{m} \frac{b_j}{a_0} u(k-j) \tag{7.35}$$

7.3.1 Algorithmization of Discrete Transfer Function

Unlike for the continuous problem (section 7.2.2), the algorithmization of direct discrete form is less computationally complex and therefore suitable for control of fast systems or for implementation on less powerful hardware. The algorithm of discrete transfer function of n-th order is shown below. In this control system, the control error notation e substitutes the notation of inputs u in equation (7.35) and the notation for output of control system y represents the control action u (input to controlled system).

```
Initialization:
  define TF numerator  b̄ = (b_0, b_1, ··· , b_m)
  define TF denominator  ā = (a_0, a_1, ··· , a_n)
  define setpoint  w
  read system's output  y
  define  e(k − 1) = e(k − 2) = ··· = e(k − n) = w − y(k)
  define  u(k − 1) = u(k − 2) = ··· = u(k − m) = 0

Loop:
  read system's output  y(k)
  compute control error  e(k) = w − y(k)
  compute control law:
    u(k) = 0
    for i = 1 to n:
      u(k) = u(k) + u(n − i) a(i)/a(0)
    for j = 1 to m:
      u(k) = u(k) + e(m − j) b(j)/a(0)
  apply saturation for u(k) (if needed)
  update system (apply u(k) on input)
  shift  k → k − 1
  Go to Loop
```

7.4 Real-Time Control

All control scenarios in ArPi Lab are implemented as Real-Time Control Systems (RCSs) with the use of embedded controllers (AVR micro-controllers). RCS is a closed-loop control system which is executed in precise fixed time steps, in which following tasks have to be done:

- gathering of process data,

- computation of control law,
- process update.

Literature provides some interesting approaches of real-time control using the low-cost micro-controllers (Marau et al., 2008) and also those equipped on Arduino development boards (Shajahan and Anand, 2013). Approaches that apply the real-time control for remote laboratories can be found in Janík and Žáková (2012). Theory and design of RCSs are well described in Laplante and Ovaska (2011).

7.4.1 Implementation of RCS

In the world of micro-controllers, real-time computations are performed via Interrupt Service Routines (ISRs), invoked by events of internal timers. ISR is a set of operations with higher execution priority than other operations programmed in micro-controller (commonly in loop section). During such event, a standard program flow is interrupted for necessary time to perform operations defined in ISR. As mentioned in Section 5.1.2, Arduino UNO contains three timers (`Timer0`, `Timer1`, and `Timer2`) from which the `Timer1` is a 16-bit timer and other two are 8-bit. Each timer has its own counter register that stores an integer value representing the amount of timer ticks from last counter overflow/reset. A timer with 8-bit resolution overflows each 256 ticks and timer with 16-bit resolution each 65536 ticks. Timer-based interrupts on AVR micro-controllers provide two ways how ISRs can be handled. First is based on event of counter overflow and second on event of compare match. Counter overflow is limited to 7 different values of clock divisor. A timer speed (frequency of counter register increment) can be written as follows:

$$f_T = \frac{f_C}{d}, \quad d = \{1, 8, 32, 64, 128, 256, 1024\}, \tag{7.36}$$

where f_T is timer frequency, f_C is a base clock frequency of micro-controller and d is divisor. Assume the use of `Timer1`, with divisor $d = 1024$. The frequency of timer will be $f_T = \frac{16 \cdot 10^6}{1024} = 15625\text{Hz}$, which is the number of counted register increments per 1 second. As mentioned above, `Timer1` will overflow after 65536 increments that is approximately every 4.2 seconds. This value is also the highest possible step size that can be achieved on interrupts of ATMega328p. However, this is not a limitation, because higher sizes of steps can be achieved using program flags based on count of overflows.

Scaling the timer frequency is only one of two available mechanism how to set up timing for ISR execution. Second approach is based on comparison of current counter value with predefined value in Compare Match Register (CMR). This register is of same resolution as counter associated to it. Implementer of micro-controller logic can choose the value of CMR which will be compared with counter value in each tick of timer. If

values in both registers are equal, the ISR is executed. The frequency of ISR execution f_{ISR} can be computed as

$$f_{ISR} = \frac{1}{h} = \frac{f_C}{d(V_{CMR}+1)}, \tag{7.37}$$

where V_{CMR} is the current value of match register of counter and h is time step between executions. Since the timing of ISRs is set up programmatically by choice of V_{CMR}, the equation (7.37) can be rewritten as

$$V_{CMR} = \frac{f_C}{d\, f_{ISR}} - 1, \tag{7.38}$$

where -1 term ($+1$ in equation (7.37)) is the shift of integer value, because micro-controllers' programs use the indexing from 0.

7.4.2 Example: Real-Time Control of Magnetic Levitation

In this section we show a demonstration of real time control implementation, which is used for all controllers applied in ArPi Lab. In order to demonstrate the possibilities of Arduino micro-controllers and their suitability for control of various systems, an unstable system with very fast dynamic CE152[1] (Fig. 7.4) was selected for this example.

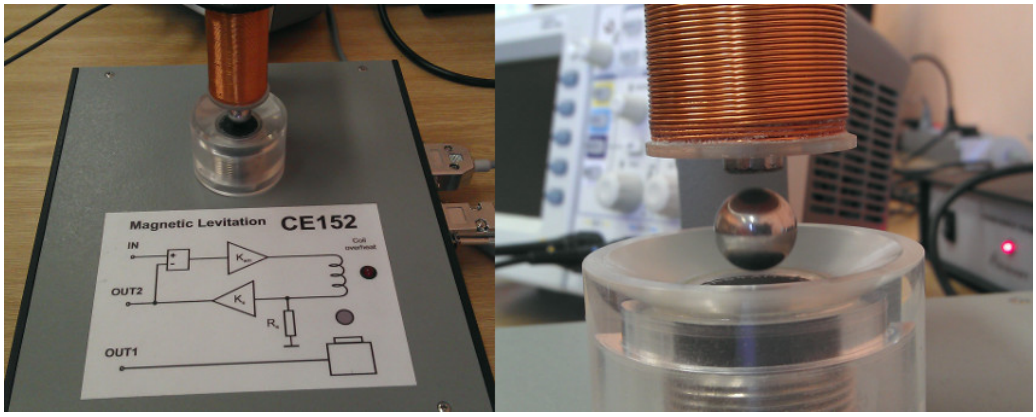


Figure 7.4: Magnetic levitation CE152

Manipulated variable is voltage applied on coil and controlled variable is a position of the ferromagnetic ball inside magnetic field. In signal interface, the position of ball is represented by a voltage taken from the proximity sensor. According to model based simulations, this process requires control loop execution frequency no less than 500Hz ($h = 2 \times 10^{-3}$s), to ensure stability. The sufficient control sampling for CE152 proved

[1] http://www.humusoft.com/produkty/models/ce152/

to be $h = 10^{-3}$s. As mentioned above, RCSs are operated with precise frequency of execution. ISR-driven control system creates a time windows from one interrupt to the following one (Fig. 7.5).

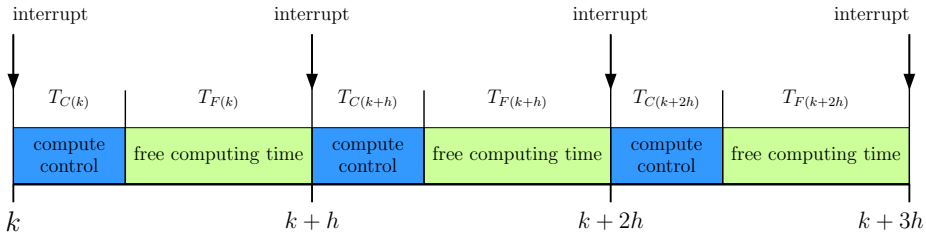


Figure 7.5: Execution of control in real time using ISR

The choice of proper step size h is essential for real time control of micro-controllers. Assuming that implementation of control scenario must satisfy the real time criterion $T_C(k) \leq h$, which says that time required for computation of control law T_C in step k (including all operations of measurement and update) must be less or equal to step size h, we must determine the $T_{C\ max}$ in order to choose an appropriate step size. The maximum time of control law computation depends on its complexity and micro-controller's performance. The $T_{C\ max}$ can be determined experimentally as a maximum time measured for the execution of control law in its most complex form. Due to the requirements for fast control loops we have selected the form of controller as the discrete transfer function (Section 7.3). This structure of controller does not contain any logical nodes in algorithm flow, therefore the computation time is approximately the same for executions even with different values of variables and parameters (e.g. one operation on floating point number takes always the same time, regardless of its value). Then the only factor with strong impact on computation time is the complexity of controller itself, which for discrete transfer function is represented by its order.

First phase of control system design with the respect to time window constraint is determination of $T_{C\ max}$ values for different orders of discrete controllers. Measurement of execution time was performed for orders from 0 to 8, while duration of ISR was measured by two independent devices. First is the internal Timer of Arduino and second is digital oscilloscope. Results are show in Fig. 7.6.

Figure shows the linear grow of $T_{C\ max}$ with the increasing order of transfer function. Another important conclusion of measurement is that the micro-controller ATMega328p (Arduino UNO) is capable to fulfill the requirement for real time control of magnetic levitation CE152 with $h = 10^{-3}s$ for controller complexity up to 6^{th} order.

To achieve $h = 10^{-3}$s using the micro-controller ATMega328p (ArPi Lab node), which

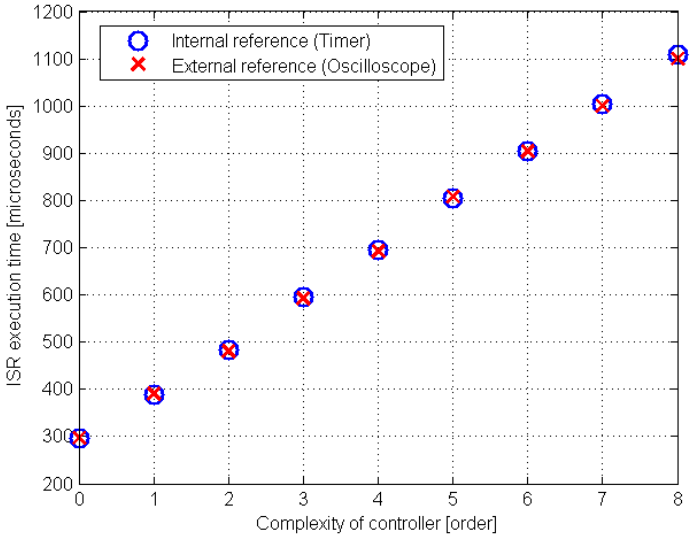


Figure 7.6: Execution time of discrete controllers of different complexity

base clock works on frequency $f_C = 16.10^6$Hz, we must select clock divisor value $d = 1$ (for the best precision) and compute the value for CMR from equation (7.38):

$$V_{CMR} = \frac{f_C}{df_{ISR}} - 1 = \frac{16 \times 10^6}{1 \times 10^3} - 1 = 15999. \tag{7.39}$$

Using these settings, the CMR will reach value 15999 exactly 1 millisecond after last reset. Another important setting is the frequency of PWM signal used as control variable. Because the CE152 is characterized by its very fast dynamics, it is appropriate to select the highest possible PWM frequency provided by micro-controller. According to section 5.1.2, this frequency is 62.5kHz and it can be achieved on Arduino UNO's pin 3 or 11, using divisor $d = 1$.

The next step is design of controller. Using the mathematical model provided by the manufacturer of CE152 and simulating the PID control in MATLAB Simulink with manual tuning of parameters, the following discrete controller was designed.

$$G(z) = \frac{105.5 - 206.2338z^{-1} + 100.7429z^{-2}}{1 - 1.1353z^{-1} + 0.1353z^{-2}} \tag{7.40}$$

The controller (7.40) is of 2$^{\text{nd}}$ order and according to data in Fig. 7.6 its execution time should not exceed 500 μs. This statement has been proved by an external measurement on oscilloscope (Fig. 7.7), using two reference digital signals. First signal (blue line) switches between states every time the ISR is executed, while width of green pulses represents the

duration of controller computation. As can be seen in figure, this control scenario still leaves more than half of computing time free for other tasks like communication and processing.

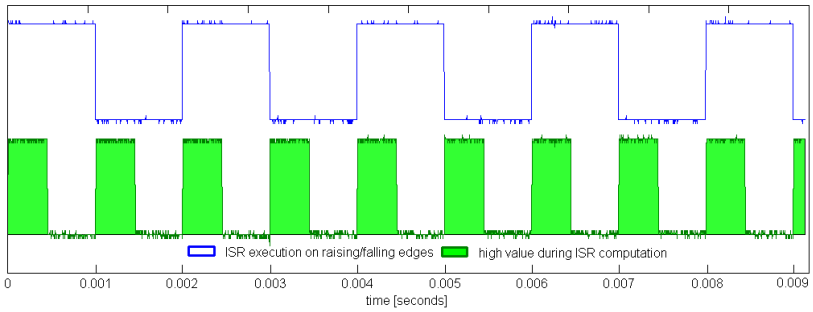


Figure 7.7: Execution of control in real time using ISR

An application of real time control scenario, using discrete transfer function (7.40) is shown in Fig. 7.8. In this case, the magnetic coil voltage has been manipulated in order to control the position of ball over the desired setpoint. It is important to mention that upper graph does not show real values of control variable, but mean values of PWM. Since the control signal uses high frequency, it is impossible to visualize it in its raw from.

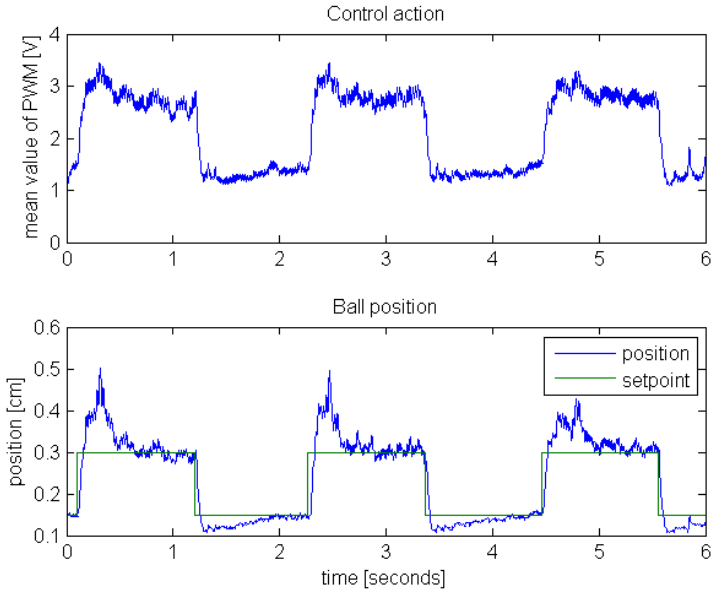


Figure 7.8: Real time control of magnetic levitation CE152

Chapter 8

Implemented Laboratories

As the practical result of this thesis, several different applications of remote laboratories have been implemented on both mentioned architectures. These laboratories provide remote control of following systems:

- thermo-optical device uDAQ28/LT,
- system of coupled tanks for liquid storage,
- three tank system uDAQ28/3H,
- air flow heat exchanger,
- DC motors.

8.1 Thermo-Optical System Laboratory

The uDAQ28/LT (Huba et al., 2006) is a laboratory training system designed for control education. It is suitable for signal processing, experimental system identification, data acquisition, and various types of control tasks. The plant provides three inputs (voltage for bulb, fan, and light diode) and eight outputs which can be measured (actual and filtered temperature inside tube, reference environment temperature, actual and filtered light intensity inside tube, fan RPM, and current taken by fan). Therefore, several subsystems of the plant can be measured. The most commonly used are the optical channel where bulb voltage acts as input, light intensity as output and LED voltage as optional disturbance signal, and thermo-optical channel with bulb voltage as input, temperature as output and fan voltage as optional disturbance signal. The physical description and detailed mathematical model of plant is provided in Jelenčiak et al. (2009).

Currently three units of this system are implemented as RL on both architectures. One device is connected to MHSA through PLC Siemens S7-300 and two are provided through ABLA served by Arduino UNO as experiment servers.

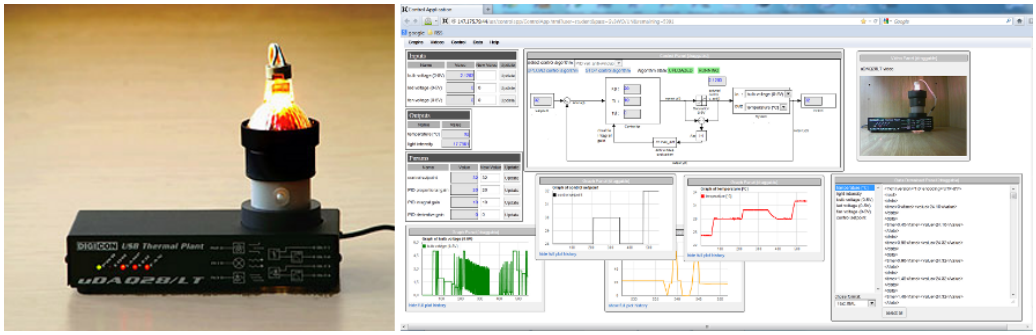


Figure 8.1: Thermo-optical device uDAQ28/LT (left) and corresponding remote laboratory session (right)

The MHSA-based laboratory with uDAQ28/LT (Fig. 8.1) was actually the first laboratory implemented in this work and it is also the laboratory with the longest uninterrupted period of operation (15 months). Since the deployment of RLMS WebLab-Deusto for management of RLs (May 2013), 214 experimental sessions have been performed in this particular laboratory.

8.2 Hydraulic System Laboratory

A system of coupled tanks is the educational process representing a very common problem in industry, the control of fluid level in storage reservoirs. This system consists of two series of tanks separated by manually operated valves and buffer tanks. In the actual setup, this system is split in two independent subsystems and implemented as two separately operated RLs. Both instances are connected to MHSA and served by the PLC VIPA 300S.

A couple of tanks represents a second order system because the level measurement is performed only in the bottom tanks. From the control point of view, the experiment acts as single-input single-output system, where input is the power of pump in percent of maximum performance (operated by voltage signal) and output is the level in second tank (measured in voltage).

Another similar system uDAQ28/3H has been implemented on ABLA (Fig. 8.3). It provides three tanks interconnected with hoses, operated by two-state valves. The liquid is pumped into tanks from the main reservoir by two pumps (operated by analog

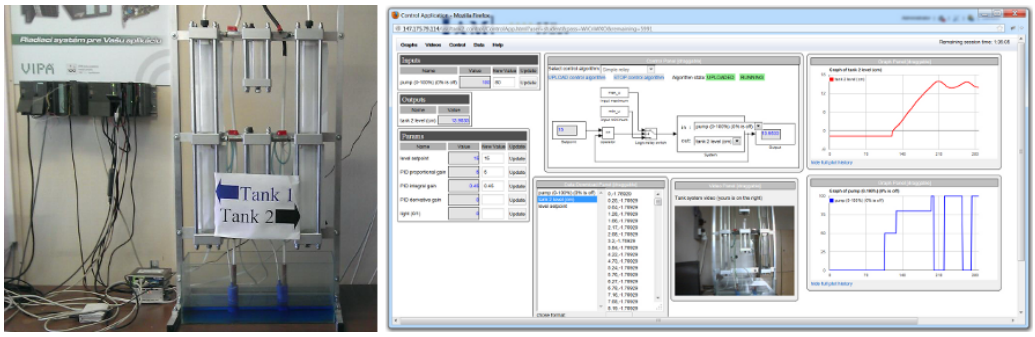


Figure 8.2: System of coupled tanks (left) and corresponding remote laboratory session (right)

signals) which can be controlled in manual or automatic way. Two of valves separate tanks from each other and another three are used to control the outlet from tanks to the main reservoir. These valves are used to reconfigure the experimental system to the form required for the measured scenario. The manipulated variables in process are voltages given on pumps and controlled variables are levels in individual tanks, measured by a pressure difference sensors.

Figure 8.3: Hydraulic system uDAQ28/3H

8.3 DC Motor Laboratory

The direct current motor can be used as very clear demonstration of electro-mechanical system. Implemented RLs provide two such experiments. First is re-configurable DC motor shown in Fig. 8.4. This device provide several different physical configuration by varying the inertia (different sizes of flywheel) and friction (break position). The

manipulated variable of this system is the motor voltage and measured variable is speed of rotor (rotations per minute). This experiment is provided as MHSA-based laboratory and it is connected to architecture through older type of PLC Siemens S7-200.

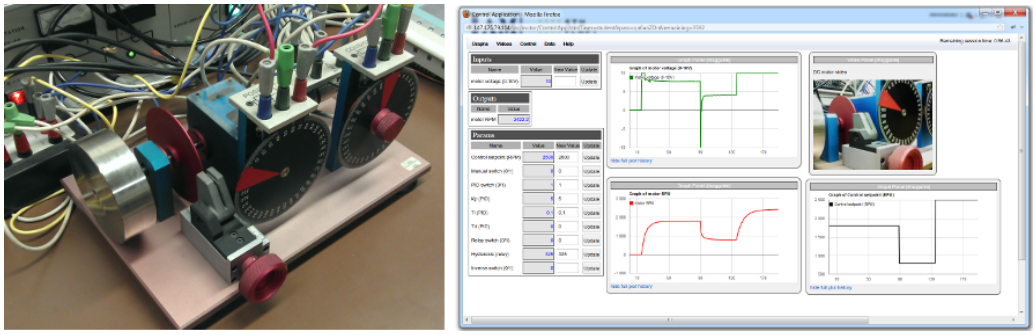


Figure 8.4: DC motor (left) and corresponding remote laboratory session (right)

The second direct current motor provided in RL is a small low-cost device show in Fig. 8.5. This experiment is connected to ArPi Lab. System provides one input (motor voltage) and one output (speed of rotor). The measurement of motor speed is implemented by counting pulses on the optical sensor, while one full rotation of motor is represented by four pulses. This device is directly connected to the signal interface of Arduino board and does not require any additional electronic components.

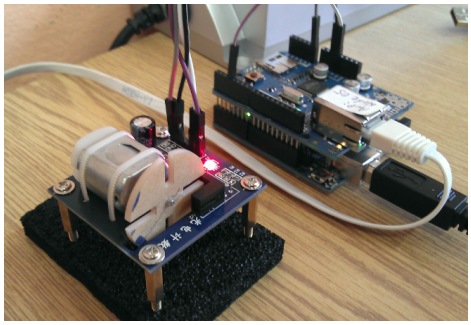


Figure 8.5: Low-cost DC motor connected to Arduino UNO (experiment server)

8.4 Heat Exchanger Laboratory

The air-flow heat exchanger is a process that demonstrates the mechanism of heat transfer from solid heating element to the air. This heat exchanger consists of tube inside which

the actuators (fan and resistive coil) and sensors (air flow and temperature) are located. The fan is used to control the air flow inside the tube and electrical resistive coil to heat it up. Students can measure the transient processes on three subsystems. First uses the coil voltage as input, temperature as output, and changes of air flow as optional disturbance. The second is similar heat transfer subsystem, but uses air flow as manipulated variable and coil voltage as disturbance. The third subsystem is a pure mass-transfer system, where the manipulated variable is voltage given on fan and controlled variable is air flow rate.

This RL is implemented on MHSA and the experiment serving device used is the PLC VIPA 300S, which is the same device that controls the hydraulic system of coupled tanks (section 8.2). This is the proof of architecture's branching capability.

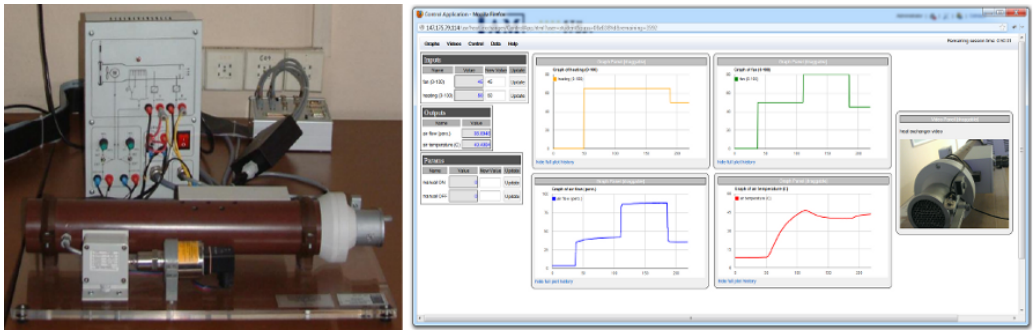


Figure 8.6: Air flow heat exchanger (left) and corresponding remote laboratory session (right)

8.5 Usage

This section provides the information about RL usage. Since the laboratories based on low-cost architecture (ArPi Lab) have been implemented recently and they are still in the pre-final development phase, we will provide only the evaluation of usage for MHSA-based laboratories.

The administration tools of RLMS WebLab Deusto allow to track users' activities, as well as the usage of laboratories in general. We have collected the usage data for all laboratories provided to students. From overall 9 laboratories, 5 have been developed at home institute and 4 are provided as federated from University of Deusto. Table 8.1 shows a list of laboratories and their rough usage information.

According to the user tracking there were overall 714 laboratory sessions since the March 20, 2013. From these, 288 accesses were from the university campus (computer

Table 8.1: List of laboratories and their usage information

System	RL identifier	Provider	Category	Available since	In use for	Sessions	Avg. time/session
uDAQ28/LT	therma-plant-udaq28tl	STU	Control labs	March 20, 2013	14 months	214	6min 23,8s
Coupled tanks	tanks-measure	STU	Control labs	September 24, 2012	8 months	61	12min 40,7s
Coupled tanks	tanks-control	STU	Control labs	September 24, 2013	8 months	74	4min 5,5s
DC motor	dc-motor-control	STU	Control labs	December 10, 2013	5 months	133	6min 25,9s
Heat exchanger	air-heat-exchanger	STU	Control labs	January 22, 2014	4 months	47	3min 2,9s
Robot	robot_movement	UD	Robot exp.	March 20, 2013	14 months	56	1min 36,2s
Aquarium /w submarine	submarine	UD	Aquatic exp.	March 20, 2013	14 months	94	1min 21,8s
PIC	ud-logic	UD	PIC exp.	March 20, 2014	14 months	21	42,2s
VISIR	visir	UD	VISIR exp.	March 20, 2015	14 months	14	52,2s

laboratories and access points of local network), and 426 accesses were from outside the campus. From the all evaluated sessions, more than one third (282) were performed through the demo user account, which allows to access laboratories in the same way as regular users, but with less available session time. The number of sessions performed by regular registered users is 432.

Since the RLs are particularly bounded to the education process, the schedule of academic year is also reflected in the time-line of laboratory usage. Most of experimental sessions were performed during the running semester, while during the holidays the usage of laboratories dropped down rapidly. The time-line of usage for above mentioned laboratories is shown in Fig. 8.7.

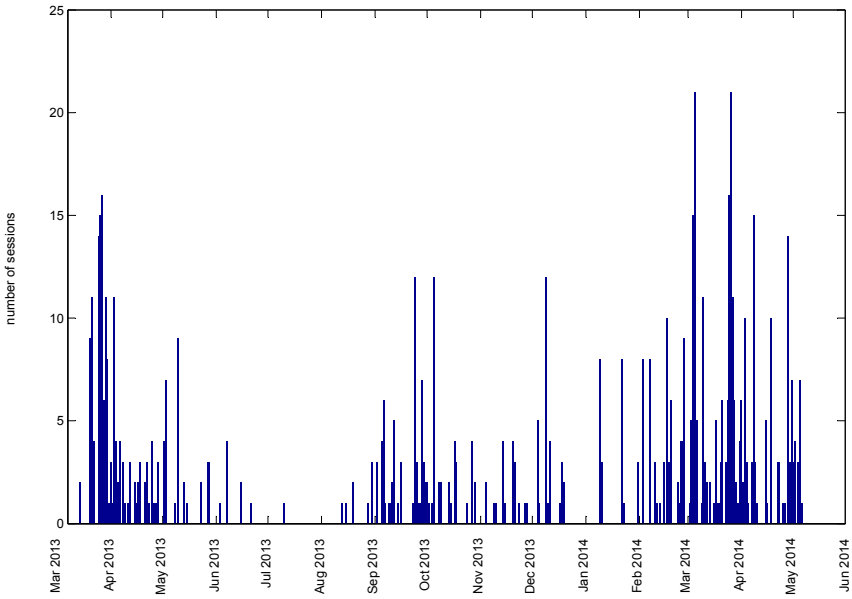


Figure 8.7: Time-line of laboratory usage from March 2013 to May 2014

In the past two semesters, RLs have been also used as the exercising supplement for the standard education process. The laboratories with the thermo-optical system have been used in the laboratory exercises of the course "Theory of Automatic Control II". In this course, students performed their tasks in RL, such as measurement of transient characteristics of system and PID control.

Other students were encouraged to use the RLs to perform measurement for their term projects, as well as the final projects of bachelor study. Several other students have participated on the RL implementation, mostly of MHSA-based laboratories. They performed the connection of controlled processes to PLCs, implementation of control algorithms, and the configuration of INR variables to allow remote control. Students have participated on implementation of coupled tanks, DC motor, and heat exchanger.

Currently, the MHSA-based RLs are located in two laboratory rooms. Each laboratory room is connected to outside world by its own INR. The local infrastructure is shown in Fig. 8.8. In this figure a laboratory B is a typical case of branched architecture, where one INR is handling two PLCs, as well as one PLC is controlling two different experiments and three RLs (coupled tanks provide two systems).

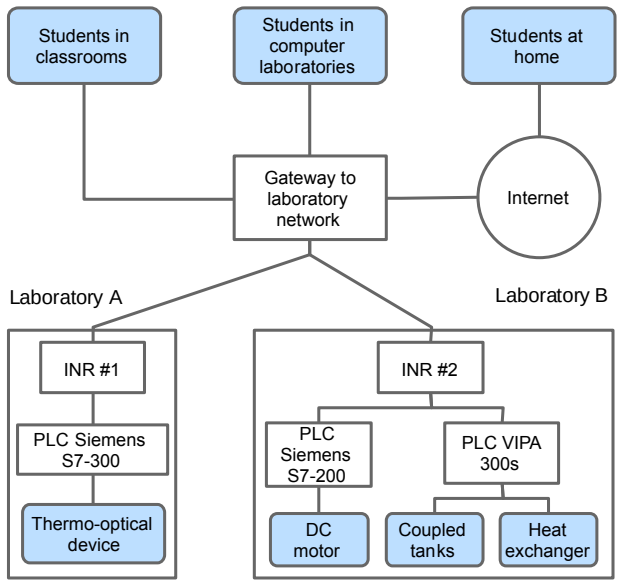


Figure 8.8: Local laboratory infrastructure

All implemented remote laboratories are available through RLMS WebLab-Deusto, published at `http://weblab.chtf.stuba.sk`.

Chapter 9

Conclusions and Future Work

This work offers two concepts of multipurpose architectures for remote laboratories. These approaches bring a new way of development, which significantly reduces the time and effort requirements needed for implementation of new laboratory instances. The possibilities of both architectures were investigated in practice on the specific cases of implemented laboratories. The evaluation has been performed mostly for laboratories based on Multipurpose Hardware and Software Architecture (MHSA), since the ArPi Branched Low-Cost Architecture (ABLA) has been developed only recently. The MHSA-based laboratories provide several experiments, namely thermo-optical device, hydraulic system with coupled tanks, DC motor, and air-flow heat exchanger. These experiments have been in usage for a longer period of time. The growing usage statistics and positive users' feedback shown the potential of remote laboratories as the full-featured supplementary tools for education and exercising in automation related areas. Created remote laboratories have been used as the educational tools in regular course with laboratory exercises and several students also participated on their development and the implementation of control scenarios.

The MHSA proved to be very suitable for a wide scope of applications on different laboratory processes. Additionally it provided very robust industrial solution with lesser requirement on maintenance than other types of architectures. The capability of branching on two levels (INR and PLC) is also very beneficial, mostly from the perspective of price of future extensions. From the control point of view, MHSA provides two processing layers that can be used for wide scale of automatic control schemes.

The ABLA has shown a potential of low-cost hardware for the development of remote laboratories. Contrary to common solutions based on relatively expensive computers or industrial devices, which price can grow up to thousands of euros, the ABLA uses the single-board computers Raspberry Pi and Arduino development boards in overall cost of several dozens of euros for one experimental node. Even this architecture shows some

drawbacks from development view such as requirement of building from scratch, and limitations in connectivity and performance, this solution is still beyond the early expectations and proved to be an adequate alternative to commonly used types of architectures with high benefit on the price and technological openness.

However the main contribution of both types of developed architectures is their universality of usage and way how new experiments can be incorporated in the remote laboratories. The concept of implementation is designed as close as possible to *plug and play* idea so implementer does not require deeper knowledge of technologies used in architecture or writing the source code for new laboratories. The process of implementation is reduced just to the connection of experiment to architecture and its appropriate configuration.

During the last years at the Institute of Information Engineering, Automation and Mathematics, we realized that remote laboratory development is never-ending challenge with the increasing requirements and evolving technologies. There is no remote laboratory solution that can be considered as perfect. There are always possible improvements and extensions that can be applied from different points of view such as didactics, usage, interoperabillity, technological enhancements, and many other. Our future work will focus mostly on those aspects, which we consider beneficial for the fluent usage in the education process. The main idea is to provide complex solution, a set of tools, that will allow students to perform wide scale tasks and study assignments remotely through the Internet. Since remote laboratories are just the part of this wider context, we plan to develop an additional extensions for them, which will provide the following features:

- *Online data storage.* Current remote laboratories allow user to download data measured during session in the batch form. The must do that before they leave the laboratory, otherwise the data are lost. Therefore the future vision is to develop such kind of acquisition system that will automatically store all data measured during the remote session and assign them to a specific user. User will be able access and visualize data any time without need to access the specific remote laboratory again. This system is considered to be designed as PHP or Python Web-based module for MySQL database with a set of service-oriented APIs for the interconnection with remote laboratories and other systems that will be able to process measured data.

- *Processing of data.* In the control education and especially in laboratory practices, students often require more than just possibility to measure process data, but also supplementary processing tools. Once the student acquire process data, there are several necessary tasks before the controller for process can be designed and tested in remote laboratory. In our future work we plant to design modular Web tools

that will allow students to filter measured data, determine the mathematical model of process by identification, and design and tune the controller by a set of standard methods.

- *Design of custom control schemes.* Allowing users to design their own automatic control scenarios is one of the most desired challenges in the current state-of-the-art of remote laboratories. This problem is still opened with no general solution available in literature. Even some of published laboratories, mostly those focused on electronics, provide several features that allows users to upload simple control programs, the full-featured solution that would allow users to generate custom control programs for automatic control experiments, with the respect to target device restrictions, still does not exists. There are two main problems that must be solved in order to bring such feature into practice. First is the security concern of laboratory. This can solved by an appropriate choice of technologies, concept of control design system and selection of restrictions for users. The second and most challenging problem is the question how to design and compose such software that will transfer human-comprehensible form of control scheme into source dedicated to target device with limited memory, performance and specific modes of operation. We consider that visual environment for control design in the form of graphical block diagramming tool would be the best solution even for users with no programming skills. Moreover, this kind of software must perform several mandatory tasks such as transcription of visual representation of control into algorithmic form, its validation and security check, addition of operational source code, pre-upload testing, and compilation into machine code (required for most of control devices such as PLCs and micro-controllers).

Another goals of future work are related to expansion of remote experimentation across the domestic and foreign education systems, and the extension of current inter-institutional collaboration.

Bibliography

S. Abedini and H. Zarabadipour. Tuning of an optimal PID controller with iterative feedback tuning method for DC motor. In *2nd International Conference on Control, Instrumentation and Automation (ICCIA), 2011*, pages 611–615, Dec. 2011. doi: 10.1109/ICCIAutom.2011.6356728. 102

M. F. Aburdene, E. Mastascusa, and R. Massengale. A proposal for a remotely shared control systems laboratory. In *Frontiers in Education Conference, 1991. Twenty-First Annual Conference. 'Engineering Education in a New World Order.' Proceedings.*, pages 589–592, 1991. doi: 10.1109/FIE.1991.187556. 21

B. Aktan, C. A. Bohus, L. A. Crowl, and M. H. Shor. Distance learning applied to control engineering laboratories. *IEEE Transactions on Education*, 39(3):320–326, 1996. ISSN 0018-9359. doi: 10.1109/13.538754. 21

A. M. Al-Busaidi. Development of an educational environment for online control of a biped robot using MATLAB and Arduino. In *13th Int'l Workshop on Mechatronics (MECATRONICS), 9th France-Japan 7th Europe-Asia Congress on and Research and Education in Mechatronics (REM)*, pages 337–344, Paris, France, 2012. doi: 10.1109/MECATRONICS.2012.6451030. 87

K. J. Åström and T. Hägglund. *PID Controllers - Theory, Design, and Tuning (2nd Edition)*. Instrument Society of America, Research Triangle Park, North Carolina, 1995. ISBN 978-1-55617-516-9. 97

Atmel Corp. Atmega48pa/88pa/168pa/328p data sheet. `http://www.atmel.com/Images/doc8161.pdf`, 2009. 71

Atmel Corp. ATmega16/32U4 Data Sheet. `http://www.atmel.com/Images/doc7766.pdf`, 2010. 71

Z. Aydogmus and O. Aydogmus. A Web-Based Remote Access Laboratory Using SCADA.

IEEE Transactions on Education, 52(1):126–132, 2009. ISSN 0018-9359. doi: 10.1109/TE.2008.921445. 21

M. Barber, R. nad Horra and J. Crespo. Control practices using simulink with Arduino as low cost hardware. In *10th IFAC Symposium Advances in Control Education*, pages 250–255, Sheffield, UK, 2013. University of Sheffield. doi: 10.3182/20130828-3-UK-2039.00061. 87

C. Barros, C. P. Leao, F. Soares, G. Minas, and J. Machado. Students' perspectives on remote physiological signals acquisition experiments. In *1st International Conference of the Portuguese Society for Engineering Education (CISPEE)*, pages 1–8, 2013. doi: 10.1109/CISPEE.2013.6701975. 21

E. Besada-Portas, J. A. Lopez-Orozco, L. de la Torre, and J. M. de la Cruz. Remote Control Laboratory Using EJS Applets and TwinCAT Programmable Logic Controllers. *IEEE Transactions on Education*, PP(99):1, 2012. ISSN 0018-9359. doi: 10.1109/TE.2012.2204754. 46

P. Bisták. Remote laboratory server based on Java Matlab interface. In *14th International Conference on Interactive Collaborative Learning (ICL)*, pages 344–347, 2011. doi: 10.1109/ICL.2011.6059601. 45

V. Bobál, J. Böhm, and J. Fessl. *Digital Self-tuning Controllers*. Springer Verlag, London, 2005. 100

C. Bohus, B. Atkan, M. H. Shor, and L. A. Crowl. 95-60-07. Technical report, Department of Computer Science, Oregon State University, Corvallis, Oregon, August 1995. 21

J. C. Butcher. *Numerical Methods for Ordinary Differential Equations, 2nd Edition*. John Wiley & Sons, 2008. ISBN 978-0-470-72335-7. 104

A. P. J. Chandra and C. R. Venugopal. Novel Design Solutions for Remote Access, Acquire and Control of Laboratory Experiments on DC Machines. *IEEE Transactions on Instrumentation and Measurement*, 61(2):349–357, 2012. ISSN 0018-9456. doi: 10.1109/TIM.2011.2164291. 46

A. Choudhary, S. A. Singh, M. F. Malik, A. Kumar, M. K. Pathak, and V. Kumar. Virtual lab: Remote access and speed control of DC motor using Ward-Leonard system. In *IEEE International Conference on Technology Enhanced Education (ICTEE)*, pages 1–7, 2012. doi: 10.1109/ICTEE.2012.6208666. 46

G. H. Cohen and G. A. Coon. Theoretical consideration of retarded control. *Transactions of the ASME*, 75:827–834, 1953. 102

E. R. Collins. An Energy Conversion Laboratory Using Industrial-Grade Equipment. *IEEE Transactions on Power Systems*, 24(1):3–11, 2009. ISSN 0885-8950. doi: 10.1109/TPWRS.2008.2007006. 21

R. J. Costa, G. R. Alves, and M. Zenha-Rela. Reconfigurable IEEE1451-FPGA based weblab infrastructure. In *9th International Conference on Remote Engineering and Virtual Instrumentation (REV), 2012*, pages 1–9, July 2012. doi: 10.1109/REV.2012.6293107. 47

Lin Cui, Fung Po Tso, Di Yao, and Weijia Jia. WeFiLab: A Web-Based WiFi Laboratory Platform for Wireless Networking Education. *IEEE Transactions on Learning Technologies*, 5(4):291–303, 2012. ISSN 1939-1382. doi: 10.1109/TLT.2012.6. 46

S. Dormido. Control learning: present and future. *Annual Reviews in Control*, 28(1): 115–136, 2004. 30

O. Dziabenko and J. García-Zubía, editors. *IT Innovative Practices in Secondary Schools: Remote Experiments*. University of Deusto, Bilbao, Spain, 2013. ISBN 978-84-15759-16-4. 36, 43

W. M. El Medany. FPGA remote laboratory for hardware e-learning courses. In *IEEE Region 8 International Conference on Computational Technologies in Electrical and Electronics Engineering, SIBIRCON 2008*, pages 106–109, July 2008. doi: 10.1109/SIBIRCON.2008.4602596. 47

eWON Inc. *Industrial VPN LAN Router eWON 2005CD/4005CD Product data sheet*, Nov. 2011. 49

G. C. Fernandez, M. C. Gil, and F. M. Perez. Remote robotic laboratory as nexus between students and real engineering. In *15th International Conference on Interactive Collaborative Learning (ICL)*, pages 1–4, 2012a. doi: 10.1109/ICL.2012.6402136. 21, 89

J. Fernandez, R. Bragos, M. Cabrera, A. Abello, N. Arroyo, D. Gonzalez, F. Garofano, A. Cortes, and A. Fabra. Interoperability platform for virtual and remote laboratories. In *9th International Conference on Remote Engineering and Virtual Instrumentation (REV)*, pages 1–7, July 2012b. doi: 10.1109/REV.2012.6293163. 36

J. E. Froyd, P. C. Wankat, and K. A. Smith. Five Major Shifts in 100 Years of Engineering Education. *Proceedings of the IEEE*, 100(Special Centennial Issue):1344–1360, 2012. ISSN 0018-9219. doi: 10.1109/JPROC.2012.2190167. 22

A. Gampe, A. Melkonyan, M. Pontual, and D. Akopian. An Assessment of Remote Laboratory Experiments in Radio Communication. *IEEE Transactions on Education*, 57(1):12–19, Feb. 2014. ISSN 0018-9359. doi: 10.1109/TE.2013.2262685. 21

H. Gao. Development of a remote laboratory for process control experiments. In *International Conference on E-Health Networking, Digital Ecosystems and Technologies (EDT)*, volume 2, pages 87 –90, April 2010. 46

J. García-Zubía and G. R. Alves, editors. *Using Remote Labs in Education: Two Little Ducks in Remote Experimentation*. University of Deusto, Bilbao, Spain, 2011. ISBN 978-84-9830-335-3. 36, 43

J. García-Zubia, D. López-de Ipiña, and P. Orduña. Towards a canonical software architecture for multi-device WebLabs. In *31st Annual Conference of IEEE Industrial Electronics Society, IECON 2005*, pages 6–pp. IEEE, 2005. 91

J. Garcia-Zubia, P. Orduna, D. Lopez-de Ipina, and G. R. Alves. Addressing Software Impact in the Design of Remote Laboratories. *IEEE Transactions on Industrial Electronics*, 56(12):4757–4767, Dec. 2009. ISSN 0278-0046. doi: 10.1109/TIE.2009.2026368. 43, 44

K. Goldberg, M. Mascha, S. Gentner, N. Rothenberg, C. Sutter, and J. Wiegley. Desktop teleoperation via the World Wide Web. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 654–659, 1995. doi: 10.1109/ROBOT.1995.525358. 21

L. Gomes and S. Bogosyan. Current Trends in Remote Laboratories. *IEEE Transactions on Industrial Electronics*, 56(12):4744–4756, 2009. ISSN 0278-0046. doi: 10.1109/TIE.2009.2033293. 30, 43, 93

L. Gomes and J. García-Zubía, editors. *Advances on Remote Laboratories and E-learning Experiences*. University of Deusto, Bilbao, Spain, 2007. ISBN 978-84-9830-662-0. 36, 43

L. Gomes, F. Coito, A. Costa, and L. B. Palma. Teaching, learning, and remote laboratories. *Advances on remote laboratories and e-learning experiences*, 2007. 35

J. L. Hardison, K. DeLong, P. H. Bailey, and V. J. Harward. Deploying interactive remote labs using the iLab Shared Architecture. In *38th Annual Frontiers in Education Conference, 2008*, pages S2A–1 –S2A–6, Oct. 2008. doi: 10.1109/FIE.2008.4720536. 34, 44

R. Hashemian and T. R. Pearson. A low-cost server-client methodology for remote laboratory access for hardware design. In *39th IEEE Frontiers in Education Conference. FIE '09*, pages 1–5, 2009. doi: 10.1109/FIE.2009.5350632. 47

S. K. Herman and M. F. Aburdene. Design of a Graphical User Interface for Laboratory Instruments. *Journal of Engineering Design*, 2(3):197–218, 1991. doi: 10.1080/09544829108901681. 21

W. Hu, G. Liu, and H. Zhou. Web-Based 3-D Control Laboratory for Remote Real-Time Experimentation. *IEEE Transactions on Industrial Electronics*, 60(10):4673–4682, 2013. ISSN 0278-0046. doi: 10.1109/TIE.2012.2208440. 46

M. Huba, P. Kurčík, and M. Kamenský. *Thermo-optical device uDAQ28/LT*. STU Bratislava, Illkovičova 3, Bratislava, 2006. 113

C. M. Ionescu, E. Fabregas, S. M. Cristescu, S. Dormido, and R. De Keyser. A Remote Laboratory as an Innovative Educational Tool for Practicing Control Engineering Concepts. *IEEE Transactions on Education*, 56(4):436–442, 2013. ISSN 0018-9359. doi: 10.1109/TE.2013.2249516. 21

Z. Janík and K. Žáková. Real-time experiments in remote laboratories based on RTAI. In *15th International Conference on Interactive Collaborative Learning (ICL)*, pages 1–5, Sept. 2012. doi: 10.1109/ICL.2012.6402075. 107

F. Jelenčiak, P. Ťapák, and M. Huba. Mathematical Modelling and Identification of Thermal Plant. In M. Fikar and M. Kvasnica, editors, *Proceedings of the 17th International Conference on Process Control '09*, pages 219–225, Štrbské Pleso, Slovakia, 2009. Slovak University of Technology in Bratislava. 113

X. Jian-Xin and H. Deqing. Optimal Tuning of PID Parameters Using Iterative Learning Approach. In *IEEE 22nd International Symposium on Intelligent Control, 2007*, pages 226–231, Oct. 2007. doi: 10.1109/ISIC.2007.4450889. 102

M. Kolenčík and K. Žáková. A contribution to remote control of inverted pendulum. In *17th Mediterranean Conference on Control and Automation*, pages 1433–1438, June 2009. doi: 10.1109/MED.2009.5164748. 45

P. A. Laplante and S. J. Ovaska. *Real-Time Systems Design and Analysis, 4th Edition*. Wiley, 2011. ISBN 978-0-470-76864-8. 107

C. Lazar and S. Carari. A Remote-Control Engineering Laboratory. *IEEE Transactions on Industrial Electronics*, 55(6):2368–2375, June 2008. ISSN 0278-0046. doi: 10.1109/TIE.2008.920650. 46

S. Leal and J. P. Leal. A new chemistry e-lab experiment chemical equilibirum reaction. In *2nd Experiment@ International Conference (exp.at'13)*, pages 154–155, 2013. doi: 10.1109/ExpAt.2013.6703050. 21

D. Lowe, S. Murray, L. Weber, and M. de la Villefromoy. LabShare: Towards a National Approach to Laboratory Sharing. In *20th Australasian Association for Engineering Education Conference*, pages 458–463, University of Adelaide, Dec. 2009. 34, 44, 46

R. Marau, P. Leite, M. Velasco, P. Marti, L. Almeida, P. Pedreiras, and J. M. Fuertes. Performing Flexible Control on Low-Cost Microcontrollers Using a Minimal Real-Time Kernel. *IEEE Transactions on Industrial Informatics*, 4(2):125–133, May 2008. ISSN 1551-3203. doi: 10.1109/TII.2008.923787. 107

R. Marques, J. Rocha, S. Rafael, and J. F. Martins. Design and Implementation of a Reconfigurable Remote Laboratory, Using Oscilloscope/PLC Network for WWW Access. *IEEE Transactions on Industrial Electronics*, 55(6):2425–2432, June 2008. ISSN 0278-0046. doi: 10.1109/TIE.2008.922400. 46

A. Mesbah and M. R. Prasad. Automated cross-browser compatibility testing. In *33rd International Conference on Software Engineering (ICSE)*, pages 561–570, May 2011. doi: 10.1145/1985793.1985870. 44

J. Mikleš and M. Fikar. *Process Modelling, Identification, and Control*. Springer Verlag, Berlin Heidelberg, 2007. 102

A. Mohtar, Z. Nedic, and J. Machotka. A remote laboratory for microelectronics fabrication. In *38th Annual Frontiers in Education Conference, FIE 2008*, pages S2F–7 –S2F–12, 2008. doi: 10.1109/FIE.2008.4720477. 45

J.M. Neto, S. Paladini, C.E. Pereira, and R. Marcelino. Remote educational experiment applied to electrical engineering. In *9th International Conference on Remote Engineering and Virtual Instrumentation (REV)*, pages 1–5, July 2012. doi: 10.1109/REV.2012.6293164. 47, 87

A. O'Dwyer. *Handbook of PI and PID Controller Tuning Rules*. Imperial College Press, 3rd edition, 2009. 102

P. Orduña. *Transitive and Scalable Federation Model for Remote Laboratories*. PhD thesis, University of Deusto, Bilbao, Spain, 2013. 34, 37, 91

P. Orduña, J. Irurzun, L. Rodriguez-Gil, J. García-Zubia, F. Gazzola, and D. Lopez-de Ipina. Adding new features to new and existing remote experiments through their

integration in weblab-deusto. *International Journal of Online Engineering (iJOE), Special issue 2 (REV2011)*, 7:33–39, Oct. 2011. 44, 91

R. Prokop, J. Korbel, and R. Matušů. Relay-based autotuning: a second order algebraic design. In *IEEE International Workshop on Intelligent Signal Processing, 2005*, pages 98–103, Sept. 2005. doi: 10.1109/WISP.2005.1531640. 102

I. Ruano-Ruano, J. Gomez-Ortega, J. Gamez-Garcia, and E. Estevez-Estevez. Integration of Online Laboratories - LMS via SCORM. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3163–3167, Oct. 2013. doi: 10.1109/SMC.2013.539. 36

I. Santana, M. Ferre, E. Izaguirre, R. Aracil, and L. Hernandez. Remote Laboratories for Education and Research Purposes in Automatic Control Systems. *IEEE Transactions on Industrial Informatics*, 9(1):547–556, 2013. ISSN 1551-3203. doi: 10.1109/TII.2011.2182518. 21, 46

A. H. Shajahan and A. Anand. Data acquisition and control using Arduino-Android platform: Smart plug. In *International Conference on Energy Efficient Technologies for Sustainability (ICEETS), 2013*, pages 241–244, April 2013. doi: 10.1109/ICEETS.2013.6533389. 107

C. E. Shannon. Communication In The Presence Of Noise. *Proceedings of the IEEE*, 86 (2):447–457, Feb. 1998. ISSN 0018-9219. doi: 10.1109/JPROC.1998.659497. 70

F. Sievers, J. S. E. Germano, and F. de Almeida. The use inform environment WebLab in high school with learning objects using learning interactive. In *International Conference on Information Society (i-Society)*, pages 572–577, 2010. 21

F. Smrcka, F. Zezulka, and M. Bilek. Teaching by means of remote access to models. In *15th International Symposium MECHATRONIKA, 2012*, pages 1–5, Dec. 2012. 34

S. Stankovski, L. Tarjan, D. Skrinjar, G. Ostojic, and I. Senk. Using a Didactic Manipulator in Mechatronics and Industrial Engineering Courses. *IEEE Transactions on Education*, 53(4):572–579, 2010. ISSN 0018-9359. doi: 10.1109/TE.2009.2036002. 63

M. Tawfik, E. Sancristobal, S. Martin, G. Diaz, J. Peire, and M. Castro. Expanding the Boundaries of the Classroom: Implementation of Remote Laboratories for Industrial Electronics Disciplines. *IEEE Industrial Electronics Magazine*, 7(1):41–49, 2013a. ISSN 1932-4529. doi: 10.1109/MIE.2012.2206872. 21

M. Tawfik, E. Sancristobal, S. Martin, R. Gil, G. Diaz, A. Colmenar, J. Peire, M. Castro, K. Nilsson, J. Zackrisson, L. Håkansson, and I. Gustavsson. Virtual Instrument Systems

in Reality (VISIR) for Remote Wiring and Measurement of Electronic Circuits on Breadboard. *IEEE Transactions on Learning Technologies*, 6(1):60–72, 2013b. ISSN 1939-1382. doi: 10.1109/TLT.2012.20. 46

S. Uran, D. Hercog, and K. Jezernik. Remote Control Laboratory with Moodle Booking System. In *IEEE International Symposium on Industrial Electronics, ISIE 2007*, pages 2978–2983, June 2007. doi: 10.1109/ISIE.2007.4375089. 34

H. Vargas, J. Sanchez Moreno, C. A. Jara, F. A. Candelas, F. Torres, and S. Dormido. A Network of Automatic Control Web-Based Laboratories. *IEEE Transactions on Learning Technologies*, 4(3):197–208, 2011. ISSN 1939-1382. doi: 10.1109/TLT.2010.35. 21, 34

A. Yazidi, H. Henao, G. A. Capolino, F. Betin, and F. Filippetti. A Web-Based Remote Laboratory for Monitoring and Diagnosis of AC Electrical Machines. *Industrial Electronics, IEEE Transactions on*, 58(10):4950–4959, Oct. 2011. ISSN 0278-0046. doi: 10.1109/TIE.2011.2109331. 37

Q. Zhong. Design of manipulator based on PLC. In *IEEE Fifth International Conference on Advanced Computational Intelligence (ICACI), 2012*, pages 988–992, 2012. doi: 10.1109/ICACI.2012.6463319. 63

M. Zhu, Ch. Yang, and W. Li. Autotuning algorithm of particle swarm PID parameter based on D-Tent chaotic model. *Journal of Systems Engineering and Electronics*, 24 (5):828–837, Oct. 2013. doi: 10.1109/JSEE.2013.00096. 102

J. G. Ziegler and N. B. Nichols. Optimum settings for automatic controllers. *Transactions of the ASME*, 64:759–768, 1942. 102

Author's Publications

Article in journal:

1. M. Kalúz, L. Čirka, and M. Fikar: Virtual and remote laboratories in process of control education. *International Journal of Online Engineering*, 8(1): 8–13, 2012. doi: 10.3991/ijoe.v8i1.1830.

2. M. Kalúz, L. Čirka, and M. Fikar: Remote experiment in control education. *AT&P Journal Plus*,(2): 50–54, 2011.

Article in conference proceedings:

1. L. Čirka, M. Kalúz, M. Kvasnica, and M. Fikar: Virtual laboratory. In *Proceedings of the 9th International Scientific - Technical Conference Process Control 2010*, pages C029a – 1–C029a – 8, Kouty nad Desnou, Czech Republic, June 7-10, 2010 2010. University of Pardubice.

2. L. Čirka, M. Kalúz, and M. Fikar: New features in random assignment - module for lms moodle. In *Zborník príspevkov z medzinárodnej vedeckej konferencie: Inovačný proces v e-learningu*, pages 1–6. Ekonóm, 2013.

3. L. Čirka, M. Kalúz, and M. Fikar: Virtual laboratories for control education. In *Zborník príspevkov z medzinárodnej vedeckej konferencie: Inovačný proces v e-learningu*, pages 1–5. Ekonóm, 2013.

4. L. Čirka, M. Kalúz, and M. Fikar: On-line remote control of matlab simulations based on asynchronous communication model. In Petr Byron, editor, *21th Annual Conference Proceedings: Technical Computing Prague 2013*, pages 1–6. Institute of Chemical Technology, Prague, ICT Prague Press, 2013.

5. M. Kalúz, Ľ. Čirka, and M. Fikar: Matlab builder ja in control engineering education at FCFT STU. In *Technical Computing Bratislava 2010*, volume 18, pages 053_1–053_5. HUMUSOFT s.r.o., RT Systems, s.r.o., 2010.

6. M. Kalúz, Ľ. Čirka, and M. Fikar: Virtual laboratory of process control. In Kvasnica. M. Fikar, M., editor, *Proceedings of the 18th International Conference on Process Control*, pages 348–351, Tatranská Lomnica, Slovakia, June 14-17, Slovak University of Technology in Bratislava, 2011.

7. M. Kalúz, Ľ. Čirka, and M. Fikar: Remote control software for thermo-optical plant. In Kvasnica. M. Fikar, M., editor, *Proceedings of the 18th International Conference on Process Control*, pages 587–592, Tatranská Lomnica, Slovakia, June 14-17, Slovak University of Technology in Bratislava, 2011.

8. M. Kalúz, Ľ. Čirka, and M. Fikar: Virtual and remote laboratories in education process at FCFT STU. Mikuláš Huba and Michael E. Auer, editors, In *Proceedings of the 14th International Conference on Interactive Collaborative Learning*, pages 134–139, Piešťany, Slovakia, International Association of Online Engineering, Kirchengasse 10/200, A-1070, Wien, Austria, September 21 - 23 2011. doi: 10.1109/ICL.2011.6059562.

9. M. Kalúz, Ľ. Čirka, and M. Fikar: Matlab tool for identification of nonlinear systems. Petr Byron, editor, In *19th Annual Conference Proceedings: Technical Computing Prague 2011*, pages 62–62. Humusoft s.r.o., Humusoft s.r.o., 2011.

10. M. Kalúz, Ľ. Čirka, and M. Fikar: Advances in online courses on process control. Pavel Pakshin, editor, In *Proceedings of 9th IFAC Symposium Advances in Control Education*, volume 9, pages 235–240, Resort Automobilist, Russia, 2012-06-23 2012. IFAC - PapersOnline. doi: 10.3182/20120619-3-RU-2024.00029.

11. M. Kalúz, Ľ. Čirka, and M. Fikar: On-line matlab-based educational tools for process control related courses. In *20th Annual Conference Proceedings: Technical Computing Bratislava 2012*, pages 35–35. Humusoft s.r.o., RT Systems, s.r.o., 2012.

12. M. Kalúz, R. Halás, P. Ďurina, R. Valo, and Ľ. Čirka: Remote laboratory based on industrial hardware. Milan Javurek, Ivan Taufer and Daniel Honc, editors, In *Proceedings of the 10th International Scientific - Technical Conference Process Control 2012*, Kouty nad Desnou, Czech Republic, 11.-14. 06. 2012.

13. M. Kalúz, Ľ. Čirka, and M. Fikar: Simplifying the implementation of remote laboratories in educational environments using industrial hardware. Kvasnica M. and

Fikar, M., editors, In *Proceedings of the 19th International Conference on Process Control*, pages 522–527, Štrbské Pleso, Slovakia, June 18-21, 2013.

14. M. Kalúz, J. García-Zubía, P. Orduña, M. Fikar, and Ľ. Čirka: Sharing control laboratories by remote laboratory management system WebLab-Deusto. Sebastián Dormido, editor, In *Proceedings of 10th IFAC Symposium on Advances in Control Education*, volume 10 of *Advances in Control Education*, pages 345–350, Sheffield, UK, 2013. University of Sheffield, International Federation of Automatic Control. doi: 10.3182/20130828-3-UK-2039.00048.

15. M. Kalúz, Ľ. Čirka, R. Valo, and M. Fikar: ArPi Lab: A Low-cost Remote Laboratory for Control Education. In *19th IFAC World Congress*, Cape Town, South Africa, August 24-29, 2014, *accepted*.

Curriculum Vitae

Martin Kalúz

Date of Birth: April 21, 1985
Citizenship: Slovakia
Email: martin.kaluz@gmail.com
Homepage: http://www.kirp.chtf.stuba.sk/∼kaluz

Education

- B.S. Food Technology and Biotechnology, Slovak University of Technology in Bratislava, 2008.
- M.A. Automation and Information Engineering in Chemistry and Food Industry, Slovak University of Technology in Bratislava, 2010
- Ph.D. Process Control, Slovak University of Technology in Bratislava, *expected* 2014

Research Stays

- WebLab-Deusto research group, Univesrity of Deusto, Bilbao, Spain, September 2012 – February 2013

Research Fields

- Information Technologies, Communication Systems, Remote Sensing and Control, Virtual and Remote Laboratories, Embedded Systems, Control Education

Computer skills

- C/C++, Java, PHP, MySQL, Python, ECMA-based languages (JavaScript, ActionScript), W3C Standards, Google Web Toolkit, Adobe Flash, MATLAB, Simulink, LaTeX, MS Windows, MS Office, Unix/Linux

Language Skills

- fluent English

Resumé

Predkladaná dizertačná práca pojednáva o vývoji viacúčelových a nízkonákladových architektúr vzdialených laboratórií. Vzdialené laboratórium je špeciálny prípad experimentálneho laboratória s priestorovo rozloženými časťami. Pozostáva z pevných komponentov (hardvéru) a programových komponentov (softvéru), ktoré sú medzi sebou špecificky previazané. Neoddeliteľnou súčasťou vzdialených laboratórií je experimentálne vybavenie, ktoré je obsluhované na diaľku prostredníctvom počítačových sietí. Vo väčšine prípadov sú takéto laboratóriá pripojené na Internet a sú ovládané webovými aplikáciami cez internetový prehliadač alebo dedikovanými aplikáciami umiestnenými v počítači, resp. mobilnom zariadení užívateľa. Takýto softvér, tiež nazývaný klientská aplikácia, komunikuje prostredníctvom siete s koncovými službami architektúry vzdialeného laboratória a poskytuje užívateľovi sadu nástrojov pre pozorovanie reálneho laboratórneho vybavenia a jeho ovládanie na diaľku. Koncová architektúra vzdialených laboratórií je sústava zariadení, ktoré obsluhujú komunikáciu medzi užívateľom a experimentom, pričom pre jednotlivé aplikačné uplatnenia môže byť rôzneho typu a nadobúdať rôzne rozmiestnenie z pohľadu infraštruktúry a prenosu informácií. Typ architektúry je zväčša určený na základe požadovaného spôsobu využitia a v nemalej miere aj povahou experimentálneho zariadenia, ktoré obsluhuje. V práci uvádzame základné typy architektúr, ktoré sa najčastejšie vyskytujú v praxi, pričom ku každej z nich je uvedená charakteristika a príklady využitia. Jednotlivé prípady sú hodnotené na základe ukazovateľov, akými sú cena, požiadavky na hardvérový a softvérový vývoj, výpočtový výkon, flexibilita využitia, a technologická rozšíriteľnosť. Hardvérové usporiadanie jednotlivých typov architektúr je zobrazené na obrázku 4.1 a ich vyhodnotenie na základe vyššie spomenutých ukazovateľov je uvedené v tabuľke 4.1.

Vzdialené laboratóriá majú využitie najmä v oblasti vzdelávania, kde tvoria plnohodnotný doplnok pre laboratórnu prax, pričom prinášajú výhody, akými sú možnosť vykonávať experimenty z ľubovoľného miesta s pripojením na Internet a v ľubovoľnom čase, bez nutnosti fyzickej prítomnosti v laboratóriu. Aplikácie vzdialených laboratórií sa vyskytujú takmer vo všetkých oblastiach vzdelávania a aplikovanej vedy. Sem patrí fyzika,

chémia, medicína a v podstate celá oblasť inžinierskych disciplín, napr. energetika, priemyselné systémy, počítačová komunikácia, elektronika, robotika, systémy automatického riadenia a iné.

Hlavnou motiváciou pre vznik tejto práce je fakt, že väčšina dostupných hardvérovo-softvérových architektúr sú v podstate *ad hoc* riešenia, alebo riešenia založené na poskytovaní vývojových aplikačných rozhraní a neponúkajú možnosť univerzálneho využitia a budovania vzdialených laboratórií nenáročnou cestou. Architektúry takýchto laboratórií sú navrhnuté pre použitie s konkrétnym typom experimentálneho zariadenia, a to nielen po hardvérovej, ale hlavne po softvérovej stránke. Jednotlivé súčasti, akými sú operačný a komunikačný softvér, klientske aplikácie a softvér pre ovládanie experimentu, sú navrhnuté tak, aby fungovali iba pre experiment, pre ktorý boli vytvorené. Aj keď takýto prístup prináša funkčné vzdialené laboratória, je sám o sebe veľmi neefektívny, a to hlavne vtedy, ak je vyžadované pripojenie iného typu experimentálneho zariadenia do architektúry, resp. reprodukovanie riešenia pre viacero rôznych laboratórií. V takom prípade musia vývojári často pristúpiť k tvorbe architektúry od začiatku a navrhnúť/upraviť ju pre toľko prípadov, koľko rôznych typov zariadení zamýšľajú použiť. Vzhľadom k tomu, že samotný vývoj vzdialených laboratórií je časovo náročná úloha, sú nové prístupy budovania viacúčelových architektúr veľmi žiadané.

Predkladaná dizertačná práca ponúka nový koncept tvorby takýchto experimentálnych prostriedkov, ktorého úlohou je oddelenie fázy implementácie vzdialených laboratórií od samotnej fázy vývoja architektonických častí. Hlavnou myšlienkou je poskytnúť taký technologický aparát, ktorý umožní implementovať širokú triedu procesných laboratórií časovo nenáročným spôsobom a bez nutnosti toho, aby realizátor do hĺbky ovládal technické prostriedky, ktoré laboratórium využíva (hardvér, programovacie jazyky a pod.). Takýto cieľ bolo možné dosiahnuť s využitím koncových obslužných zariadení, ktoré umožnia pripojiť do architektúry rôzne typy technologických procesov a súčasne operačný a klientský softvér, ktorý je navrhnutý univerzálne a je možné ho jednoducho nakonfigurovať pre využitie s akýmkoľvek typom ovládaného procesu.

Teoretickým prínosom práce sú dva koncepty viac-účelovej hardvérovo-softvérovej architektúry. Prvá je založená na priemyselných zariadeniach, akými sú programovateľné logické automaty (PLC) a priemyselné sieťové smerovače. Táto architektúra je špecifická tým, že je postavená na robustných priemyselných komponentoch, priamo navrhnutých na nepretržitú prevádzku. Hlavnou výhodou tejto architektúry je, že poskytuje hotové hardvérové a čiastočne aj softvérové riešenie a dokáže obsluhovať veľmi širokú triedu procesných zariadení. Architektúra pozostáva z dvoch operačných vrstiev (obrázok 4.2), pričom na oboch je možné ju rozvetviť (obrázok 4.3). Spodná vrstva (tzv. riadiaca vrstva) obsahuje ovládaný/riadený proces (experiment) pripojený prostredníctvom elektrických

signálov k PLC, ktoré slúži na priamu interakciu s akčnými a meracími členmi procesu. Na vyššej vrstve (tzv. správcovskej vrstve) sa nachádza priemyselný sieťový smerovač, ktorý umožňuje obsluhovať jednotlivé dvojice PLC-proces a taktiež sprostredkovať komunikáciu medzi procesom a užívateľom. Z pohľadu automatického riadenia poskytuje táto architektúra dve riadiace vrstvy, keďže okrem PLC aj priemyselný smerovač obsahuje prostredie pre vykonávanie užívateľských algoritmov. Klientská časť je poskytovaná ako webová aplikácia, ktorá sa spúšťa v prostredí internetového prehliadača. Je navrhnutá tak, aby bola konfigurovateľná pre akýkoľvek experiment, ktorý je možné pripojiť k architektúre.

Druhá architektúra vychádza z rovnakého konceptu univerzálnosti, avšak jej hardvérové časti pozostávajú z nízkonákladových komponentov (obrázok 5.2). Fyzikálna konštrukcia obsahuje laboratórny server, založený na otvorených a voľne dostupných technológiách a implementovaný na jednodoskovom počítači Raspberry Pi. Na nižšej úrovni z pohľadu architektúry sa nachádzajú tzv. experimentálne uzly, ktoré pozostávajú z ovládaných/riadených zariadení a obslužného hardvéru, ktorým sú vývojové elektronické platformy na báze mikroovládačov Arduino UNO. Mikroovládače prostredníctvom elektrického signálového rozhrania zabezpečujú priamu interakciu s akčnými a meracími členmi laboratórneho procesu. Hlavnou výhodou tejto architektúry je jej cena, ktorá je rádovo nižšia ako u bežných typov vzdialených laboratórií, ktoré používajú štandardné počítače a priemyselné moduly/karty na ovládanie procesov a zber dát. Použitie laboratórneho servera ako komunikačnej brány do privátnej siete ponúka možnosť vetvenia architektúry, pričom jeden takýto server môže (v závislosti od typu počítača a jeho výkonu) obsluhovať aj desiatky vzdialených laboratórií. Klientská aplikácia pracuje na podobnom princípe ako u predchádzajúcej architektúry, teda je plne konfigurovateľná pre akýkoľvek laboratórny proces, ktorý je možné pripojiť k signálovému rozhraniu mikroovládača.

Práca sa taktiež zaoberá implementáciou riadiacich algoritmov pre mikroovládače. V praktickej časti uvádzame postup vhodnej úpravy a algoritmizácie niektorých štandardných typov regulátorov pre ich v diskrétne vykonávanie a taktiež princíp realizácie riadenia v reálnom čase. Riadiace algoritmy implementované pre vzdialené laboratóriá sú napr. štandardné, sériové a paralelné PID regulátory, spojité prenosové funkcie (a vo všeobecnosti regulátory vo forme podielu dvoch polynómov) a diskrétne prenosové funkcie.

Na vyššie spomínaných architektúrach bolo v rámci praktického prínosu práce implementovaných niekoľko vzdialených laboratórií, poskytujúcich rôzne typy experimentov. Pripojené boli procesy ako tepelno-optická sústava, dva typy hydraulických systémov so zásobníkmi kvapaliny, dva typy elektrických pohonov a vzduchový výmenník tepla. Všetky laboratória založené na prvom type architektúry boli v prevádzke dostatočne dlho, aby bolo možné vyhodnotiť ich využívanie. Od 20. marca 2013 bolo evidovaných celkovo 714 prípadov užívania vzdialených laboratórií, z čoho 288 prístupov bolo evidovaných z uni-

verzitnej siete a 426 z vonkajších sietí. Zoznam laboratórií a základné informácie o ich užívaní sú uvedené v tabuľke 8.1 a časová os užívania laboratórií počas jedného roka je zobrazená na obrázku 8.7. Použitie laboratórií v praxi ukázalo, že navrhnuté architektúry ponúkajú veľmi robustné riešenie, čo sa týka prevádzkyschopnosti a spôsob implementácie nových laboratórií prináša výrazné časové úspory v porovnaní s existujúcimi prístupmi.

V predkladanej práci taktiež uvádzame spôsob publikovania experimentov na Internete prostredníctvom systému pre správu vzdialených laboratórií RLMS WebLab-Deusto. Tento systém bol zavedený na Ústave informatizácie, automatizácie a matematiky, STU v Bratislave, ako súčasť spolupráce zdieľania vzdialených laboratórií s Univerzitou Deusto v Bilbau. Prostredníctvom WebLab-Deusto dochádza k poskytovaniu vytvorených procesných laboratórií univerzite v Španielsku. Opačným smerom, Univerzita Deusto poskytuje svoje vzdialené laboratóriá STU v Bratislave.

Pre ďalšie pokračovanie tejto práce uvažujeme s vývojom komplexného softvérového riešenia pre zber a spracovanie dát nameraných vo vzdialených laboratóriách, ako aj s vývojom systému, ktorý by umožnil užívateľom navrhovať a aplikovať vlastné riadiace algoritmy pre procesné laboratóriá. Taktiež uvažujeme s rozširovaním experimentálnej základne o nové vzdialené laboratórne procesy, či už ich pripájaním k vyvinutým architektúram, alebo prostredníctvom zdieľania existujúcich laboratórií s inými akademickými inštitúciami.