



INSTITUTO SUPERIOR TÉCNICO  
Universidade Técnica de Lisboa

# **Interface and Web Server Implementation for an Industrial Automation Remote Laboratory**

**Miguel Francisco Amaral Ribeiro**

Dissertação para obtenção do Grau de Mestre em  
**Engenharia Mecânica**

## **Júri**

Presidente: Prof. Mário Manuel Gonçalves da Costa

Orientador: Prof. João Rogério Caldas Pinto

Vogais: Prof. João Carlos Prata dos Reis

Prof. José Luís Carrilho Sequeira

Prof. Paulo Alexandre Fernandes Ferreira

**May 2012**



## Resumo

O presente documento reflecte o trabalho efectuado no desenvolvimento de um novo ambiente gráfico para interação com o Laboratório Remoto de Automação Industrial do Departamento de Engenharia Mecânica do Instituto Superior Técnico. Foi implementada uma interface, acessível por *web browser*, para controlo e leitura de variáveis de controladores lógicos programáveis (PLCs) e teste de programas para automação desenvolvidos num programa que suporta os PLCs utilizados.

A motivação deste trabalho surge da necessidade de actualização do desempenho do laboratório remoto no contexto actual da experimentação à distância, pelo que se efectuou uma pesquisa das arquitecturas e software que mais são adoptados, bem como os campos de engenharia, e não só, em que este tipo de laboratórios são aplicáveis. Também foram investigados e testados os tipos de acesso de um conjunto de laboratórios remotos nacionais e internacionais.

Identificaram-se as principais lacunas da interface actual, que estiveram na base das principais características da nova interface implementada. Foram igualmente descritas as etapas de criação da interface, bem como o método de como incluí-la em qualquer projecto desenvolvido no SAIA PG5.

Finalmente, foi analisada a melhoria obtida com o novo ambiente gráfico, quer na representação de componentes reais, quer na significativa redução de passos para leitura e manipulação de variáveis.

*Palavras-chave: laboratório remoto, HMI, automação industrial.*



## **Abstract**

The present document reflects the work made on the development of a new graphical environment for the interaction with the Industrial Automation Remote Laboratory of the Mechanical Engineering Department of Instituto Superior Técnico. It was implemented an interface, accessible by web browser, to control and read programmable logic controllers (PLCs) variables and to test automation programs developed with a software that supports the used PLCs.

The motivation for this work appears from the necessity to interpret and update the remote laboratory performance in the present context of remote experimentation, whereupon a research on the most adopted architectures and software was made, as well as the engineering fields, and others, where this type of laboratories are applicable. It was also researched the access methods of a group of national and international remote laboratories.

The main flaws of the present interface were identified, and served as a base for the definition of the main characteristics of the new interface. The interface creation phases were also described, as well as the method of how to include the interface in any SAIA PG5 project.

Finally, it was analyzed the improvement obtained with the new graphical environment, not only in terms of real components representation, but also with the significant reduction on the number of steps to variable reading and manipulation.

*Keywords: remote laboratory, HMI, industrial automation.*



## **Acknowledgments**

I want to express my sincere acknowledgments:

To my thesis supervisor, Professor Caldas Pinto, for the availability, support and motivation provided which greatly contributed to the accomplishment of this work.

To Engineer Eric Serrano for the availability and supplied information on PLC's web server preparation and management.

To the DEM-IST systems laboratory technicians Camilo Christo and Luís Raposeiro for the support and availability on the remote laboratory management.

To all my friends, university colleagues, theater companions and girlfriend for their support, friendship and laughs through these academic years. Thank you so much.

To my parents, Ascensão Ribeiro and Francisco Ribeiro, for the support, comprehension and guidance through all this years of academic life, and to my sister, Inês Ribeiro, for her important friendship.

## **Agradecimentos**

Quero expressar os meus sinceros agradecimentos:

Ao meu orientador de tese, Professor Caldas Pinto, pela disponibilidade, apoio e motivação dispensadas que muito contribuíram para a realização deste trabalho.

Ao Engenheiro Eric Serrano, pela disponibilidade e fornecimento de informação sobre preparação e manutenção de web servers em PLCs.

Aos técnicos do laboratório de sistemas do DEM-IST, Sr. Camilo Christo e Sr. Luis Raposeiro, pelo apoio e disponibilidade na manutenção do laboratório remoto.

A todos os meus amigos, colegas de faculdade, companheiros de teatro e namorada pelo apoio, amizade e risos ao longo destes anos de universidade. Obrigado por tudo.

Aos meus pais, Ascensão Ribeiro e Francisco Ribeiro, pelo apoio, compreensão e orientação ao longo destes anos de vida académica, e à minha irmã, Inês Ribeiro, pela sua importante amizade.





# Table of Contents

<i>Resumo</i> .....	<i>iii</i>
<i>Abstract</i> .....	<i>v</i>
<i>Acknowledgements</i> .....	<i>vii</i>
<i>Table of Contents</i> .....	<i>ix</i>
<i>List of Figures</i> .....	<i>xiii</i>
<i>List of Tables</i> .....	<i>xv</i>
<i>Abbreviations</i> .....	<i>xvi</i>
 <b>CHAPTER 1</b> .....	 <b>1</b>
<b>INTRODUCTION</b>	
1.1 MOTIVATION .....	2
1.2 CONTRIBUTIONS .....	2
1.3 THESIS STRUCTURE.....	2
 <b>CHAPTER 2</b> .....	 <b>3</b>
<b>STATE OF THE ART</b>	
2.1 HANDS-ON, VIRTUAL OR REMOTE LABS? .....	3
2.2 AREAS OF APPLICABILITY .....	5
2.2.1 Industrial Automation .....	5
2.2.2 Systems Control.....	5
2.2.3 Others.....	7
2.3 NETWORK ARCHITECTURES AND SOFTWARE.....	8
 <b>CHAPTER 3</b> .....	 <b>15</b>
<b>LABORATORIES IN USE</b>	
3.1 INTRODUCTION .....	15
3.2 INTERNATIONAL EXAMPLES .....	15
3.2.1 MIT ‘iLabs’ .....	15
3.2.2 WebLab-Deusto .....	17
3.2.3 NetLab.....	19
3.3 NATIONAL EXAMPLES.....	21
3.3.1 FEUP Remote Experiments .....	21
3.3.2 FCT RemoteLab.....	23

3.3.3 ISR Remote FPGA Laboratory .....	24
<b>CHAPTER 4 .....</b>	<b>25</b>
<b>THE IST INDUSTRIAL AUTOMATION LABORATORY</b>	
4.1 LABORATORY SETUP .....	25
4.2 THE EXISTING INTERFACE.....	26
4.2.1 Menu Bar .....	27
4.2.2 Sending Section .....	28
4.2.3 Reading Section .....	28
4.2.4 Message Section .....	29
4.3 ISSUES AND IMPROVEMENTS, REASONS FOR A NEW INTERFACE .....	29
<b>CHAPTER 5 .....</b>	<b>33</b>
<b>WEBPAGES AS HMI INTERFACES</b>	
5.1 INTRODUCTION.....	33
5.2 FUNDAMENTS OF A WEB BASED HMI .....	34
5.3 DESIGNING A WEB PAGE USING THE SAIA S-WEB EDITOR.....	35
5.3.1 File Management Architecture .....	35
5.3.2 Designing a Web Page.....	37
5.3.2.1 Including a flag manipulator.....	39
5.3.2.2 Including a register manipulator.....	39
5.3.2.3 Including a variable reader-only and string box .....	39
5.3.2.4 Including an output interaction section.....	40
<b>CHAPTER 6 .....</b>	<b>41</b>
<b>DESIGNING THE NEW INTERFACE</b>	
6.1 INTRODUCTION.....	41
6.2 CREATING THE WEB PAGE.....	42
6.2.1 Flag Manipulation Section .....	44
6.2.2 Register Manipulation Section.....	47
6.2.3 Variable Reading Section .....	49
6.2.4 Output Section.....	49
6.3 INCLUDING THE INTERFACE ON A SAIA PROJECT.....	51
6.3.1 Including a new interface on a SAIA project, for a PLC with no Flash card.....	51
6.3.2 Including a new interface on a SAIA project, for a PLC with a Flash card.....	52
6.3.3 Including an existing interface on a SAIA project, for a PLC with a Flash card .....	53
6.4 RESULTS .....	54
<b>CHAPTER 7 .....</b>	<b>59</b>
<b>CONCLUSION</b>	
7.1 WORK FINAL NOTES.....	59

7.2 FUTURE WORK .....	59
-----------------------	----

<b>References</b> .....	<b>61</b>
-------------------------	-----------

<b>APPENDIX</b> .....	<b>65</b>
-----------------------	-----------



## List of Figures

FIGURE 1 – PNEUMATIC CYLINDERS FROM AN AUSTRALIAN AUTOMATION REMOTE LABORATORY. ....	5
FIGURE 2 – TO THE LEFT, INVERTED PENDULUM CONTROL PREPARATION, FROM THE UNIVERSITY OF QUEENSLAND, AUSTRALIA. ....	7
FIGURE 3 – GENERIC ARCHITECTURE OF A REMOTE LABORATORY NETWORK. ....	10
FIGURE 4 – ARCHITECTURE WITH MAIN AND SECONDARY SERVERS OF A REMOTE LABORATORY NETWORK. ....	11
FIGURE 5 – ONLINE FORM TO REGISTER IN MIT OPENILABS SERVICE BROKER. ....	16
FIGURE 6 – MIT MICROELECTRONICS DEVICE CHARACTERIZATION ILAB CLIENT LAB WEBCAM VIEW IN IDLE STATE (TO THE LEFT) AND RUNNING THE EXPERIMENT (TO THE RIGHT). ....	17
FIGURE 7 - MIT MICROELECTRONICS DEVICE CHARACTERIZATION ILAB CLIENT LAB INTERFACE. ....	17
FIGURE 8 – WEBLAB-DEUSTO LOG IN PAGE. ....	18
FIGURE 9 – WEBLAB-DEUSTO ROBOT EXPERIMENT: ....	18
FIGURE 10 – WEBLAB-DEUSTO AND BTH OPENLABS VISIR EXPERIMENT EXAMPLE. ....	19
FIGURE 11 – NETLAB REGISTRATION MENU. ....	20
FIGURE 12 – NETLAB BOOKING MENU. ....	20
FIGURE 13 – NETLAB LOGIN WINDOW. ....	21
FIGURE 14 – FEUP REMOTE EXPERIMENTS LOGIN PAGE. ....	22
FIGURE 15 – FEUP REMOTE EXPERIMENTS BOOKING SYSTEM DETAIL. ....	22
FIGURE 16 – FEUP STRAIGHTNESS EVALUATION REMOTE EXPERIMENT. ....	22
FIGURE 17 – FEUP MECHANICAL MATERIAL CHARACTERIZATION REMOTE EXPERIMENT. ....	23
FIGURE 18 – FCT REMOTELAB LOGIN PAGE. ....	23
FIGURE 19 – ISR REMOTE FPGA LABORATORY REGISTRATION REQUEST PAGE. ....	24
FIGURE 20 – GENERAL VIEW OF THE REMOTE LABORATORY PHYSICAL SETUP. ....	25
FIGURE 21 – CAMERA LOCATION IN RELATION TO THE REMOTE LABORATORY PHYSICAL SETUP. ....	26
FIGURE 22 - GENERAL VIEW OF “INTERFACE SAIA”. ....	26
FIGURE 23 - MENU “PCD”. ....	27
FIGURE 24 – FOUR STEPS PROCESS TO VARIABLE MANIPULATION: ....	28
FIGURE 25 – THREE STEPS PROCESS TO VARIABLE READING: ....	29
FIGURE 26 - EXAMPLE OF MESSAGE SECTION DISPLAY. ....	29
FIGURE 27 – SHIFTING THE EMPHASIS IN AUTOMATION PROJECTS FROM STRAIGHT MACHINE/PROCESS CONTROL TO HMI AND COMMUNICATIONS FUNCTIONALITY. (COURTESY OF INFOCONTROL) ....	34
FIGURE 28 – CLASSICAL HMI CONCEPT WITH SCREENS STORED IN THE CONTROL PANEL. (COURTESY OF INFOCONTROL) ....	34

FIGURE 29 – WEB BASED HMI CONCEPT WITH SCREENS STORED IN THE CONTROLLER'S WEB SERVER AND DISPLAY BY STANDARD BROWSER. (COURTESY OF INFOCONTROL).....	35
FIGURE 30 – SAIA FILE STRUCTURE WITH A PCD3.M3330. ....	36
FIGURE 31 – SIZE OF EACH GROUP OF FILES THAT COMPOSE THE WEB SERVER. THE TOTAL SIZE IS 446,37 Kb.....	37
FIGURE 32 - SAIA FILE STRUCTURE WITH A PCD3.M5540. ....	37
FIGURE 33 – VIEW/TEQ FILE WITH BACKGROUND TEQ. ON TOP, TO THE LEFT, IS A TEQ FILE WITH THE PRETENDED BACKGROUND IMAGE AND THE NAME OF MULTIPLE TABS; ON TOP, TO THE RIGHT, IS A TEQ FILE WITH NO BACKGROUND; ON THE BOTTOM IS THE RESULTING TEQ FILE WHEN SETTING THE FIRST TEQ AS A BACKGROUND TEQ TO THE SECOND ONE.....	38
FIGURE 34 – TWO STATES OF A FLAG MANIPULATOR. ....	39
FIGURE 35 – EXAMPLE OF A REGISTER MANIPULATOR. ....	39
FIGURE 36 – GROUP OF READ-ONLY BOXES AND THE RESPECTIVE NAMES IN STATIC PAINTERS. ....	39
FIGURE 37 – THE TESTING AREA (LEFT) AND VIEW AREA (RIGHT) OF THE OUTPUT INTERACTION SECTION. ....	40
FIGURE 38 – ADDING A NEW WEB EDITOR PROJECT FILE THROUGH THE SAIA PROJECT MANAGER.....	42
FIGURE 39 – S-WEB EDITOR MAIN WINDOW. ....	43
FIGURE 40 – A) HOW TO GO TO TEQ VIEW CONFIGURATION";.....	43
FIGURE 41 – PLACING A BUTTON PAINTER ON A TEQ VIEW. ....	44
FIGURE 42 – SETTINGS FOR IMPLEMENTING A PRESSURE BUTTON. ....	45
FIGURE 43 – SETTINGS FOR IMPLEMENTING A TOGGLE BUTTON. ....	45
FIGURE 44 – DISABLING BUTTON TEXT. ....	45
FIGURE 45 – REMOVING THE BUTTON INTERIOR AND OUTLINE COLORS. ....	46
FIGURE 46 – PLACING AN IMAGE PAINTER ON A TEQ VIEW.....	46
FIGURE 47 – SETTINGS FOR IMPLEMENTING THE BUTTON IMAGES.....	47
FIGURE 48 – ARRANGING THE BUTTON IN FRONT OF THE IMAGE AND GROUPING BOTH PAINTERS. ....	47
FIGURE 49 – PLACING AN EDIT BOX PAINTER ON A TEQ VIEW. ....	48
FIGURE 50 – SETTING A PPO FOR THE EDIT BOX. ....	48
FIGURE 51 – PLACING A STATIC TEXT BOX PAINTER ON A TEQ.....	48
FIGURE 52 – EDITING A STATIC TEXT FONT AND DISABLING THE GRID. ....	49
FIGURE 53 – PLACING THE BACKGROUND ALL-LIGHTS-OFF IMAGE.....	50
FIGURE 54 – A) SETTINGS FOR THE LIGHT ON IMAGE PAINTER; .....	50
FIGURE 55 – GENERATING A WEB SERVER FILE FOR THE PCD3M3330 CASE.....	52
FIGURE 56 – GENERATING A WEB SERVER FILE FOR THE PCD3M5540 CASE.....	52
FIGURE 57 – FILEZILLA MAIN WINDOW. ....	53
FIGURE 58 – INDUSTRIAL AUTOMATION COURSE FIRST LABORATORY WORK SYSTEM. ....	55
FIGURE 59 – GENERAL VIEW OF THE INDUSTRIAL AUTOMATION LABORATORY WORK INTERFACE. ON TOP, INITIAL STATE OF THE SYSTEM. ON THE BOTTOM, SYSTEM WITH FULLY ACTIVATED STATES....	56

## List of Tables

TABLE 1 - TYPES OF FILES INCLUDED IN A WEB EDITOR PROJECT. ....	36
TABLE 2 – COMPARISON BETWEEN THE NUMBER OF STEPS REQUIRED TO PERFORM EACH ACTION IN THE OLD AND NEW INTERFACES. E: ENABLE VARIABLE; A: WRITE ADDRESS; V: WRITE VALUE; S/R: SEND OR READ VARIABLE; T: TOTAL NUMBER OF STEPS. ....	54
TABLE 3 – LIST OF GLOBAL SYMBOLS USED FOR THE INTERFACE TESTING PROCESS. ....	57

## Abbreviations

<i>ACM</i>	Association for Computing Machinery
<i>CGI</i>	Common Gateway Interface
<i>CSS</i>	Cascading Style Sheets
<i>DAQ Card</i>	Data Acquisition Card
<i>DEM</i>	Mechanical Engineering Department
<i>FBD</i>	Function Block Diagram
<i>FDI</i>	Fault Detection and Isolation
<i>FEUP</i>	Faculdade de Engenharia da Universidade do Porto
<i>FPGA</i>	Field-Programmable Gate Array
<i>GUI</i>	Graphical User Interface
<i>HMI</i>	Human Machine Interface
<i>HTML</i>	HyperText Markup Language
<i>HTTP</i>	HyperText Transfer Protocol
<i>IARL</i>	Industrial Automation Remote Laboratory
<i>IEEE</i>	Institute of Electrical and Electronics Engineers
<i>IL</i>	Instruction List
<i>ISA</i>	iLab Shared Architecture
<i>IST</i>	Instituto Superior Técnico
<i>LD</i>	Ladder Diagram
<i>MSW</i>	Web Micro Server
<i>MVC</i>	Model-View-Controller
<i>OPC</i>	OLE (Object Linking and Embedding) for Process Control
<i>PCD</i>	Process Control Device
<i>PID</i>	Proportional-Integral-Derivative
<i>PLC</i>	Programmable Logic Controller
<i>PXI</i>	PCI (Peripheral component interconnect) eXtensions for Instrumentation
<i>RLC</i>	Resistor, Inductor and Capacitor circuit
<i>SCADA</i>	Supervisory Control And Data Acquisition
<i>SFC</i>	Sequential Function Chart
<i>SSI</i>	Server Side Includes
<i>ST</i>	Structured Text
<i>VISIR</i>	Virtual Instrument Systems in Reality
<i>VPN</i>	Virtual Private Network
<i>WUI</i>	Web User Interface



## Chapter 1

### INTRODUCTION

Our world is getting smaller. This is the metaphor that describes the global behavior of the present days. What was once unreachable, financially or not, is today feasible and not so complicated. The role of the Internet is immeasurable, offering access to, virtually, infinite knowledge, resources and effective support. But this reality isn't new, everyone is already aware of this. So we just have to take more steps towards that kind of globalization, and that is the path of remote experimentation.

Hands-on (physical) laboratories are usually expensive, with strict time tables and require additional operators and maintenance, but are essential in order to fully comprehend the real behavior and attributes of a system's components. Simulators (virtual laboratories) lack in physical perception quality, but don't require much maintenance or operators, and allow running multiple procedures at the same time. Remote experimentation is a step forward on delivering a good learning platform with less time and location constraints, without lacking in physical perception. This means more availability, more initiative and more flexibility, which, in other words, means a more student focused teaching approach.

Complementing and enhancing this previous characteristic, there should be a constant improvement in the accessibility to these laboratories, reducing the number of required software packages and scheduling access times.

The present work describes the research made on remote laboratories, national and international, and the implementation of a web based accessibility to the Institute of Mechanical Engineering DEM/IST Industrial Automation Remote Laboratory - IARL

Therewith, it was defined a group of objectives to achieve:

- Researching on remote laboratories, regarding the architecture and accessibilities.
- Running a survey of the national and international remote laboratories in use.
- Projecting and implementing a new way to access the remote laboratory based on a webpage.
- Analyzing and improving access strategy.

## **1.1 Motivation**

Since it has been installed, almost ten years ago, the industrial automation remote laboratory has not been updated in any way. The laboratory has been used as a learning and experimentation tool available to the students of the Industrial Automation course, as a way to test their logical code implementation, without having to be in the institute. It was proposed a verification of its condition, comparing it to other remote labs that are in use online, mainly developed by universities. Another motivation factor was to project a new way to access the laboratory, by designing a webpage with the SAIA 4 web editor, dedicated software for the type of PLC used in the laboratory. With the implementation of this webpage, stored in the PLC, there is no need of using the previous interface program, enabling the remote experimentation to be accessed with a common web browser.

## **1.2 Contributions**

The main contributions of this work are:

- Information gathering on several remote laboratories around the world, in terms of architecture, software, accessibility and overall management of the lab conditions.
- Implementation of an up-to-date web interface for the control of the Industrial Automation remote laboratory of DEM/IST/UTL.

## **1.3 Thesis Structure**

This work proceeds in chapter 2 where the importance and concept of remote experimentation is thoroughly analyzed. The architectures and softwares usually adopted will also be referred.

In chapter 3 are presented some of the most relevant remote laboratories and groups of laboratories available online, respective interfaces, and the description of the steps made while accessing them.

The system, architecture and other details of the IST industrial automation laboratory are described in chapter 4. Issues and possible improvements of the existing interface are also explained in this chapter.

In chapter 5 the concept of HMI interfaces is presented, and how it is related to the industrial automation remote laboratory and new interface. A small introduction to interface elements design is also presented in this chapter.

Chapter 6 consists in the detailed development of the new interface with a specific web editing software.

This work concludes in chapter 7 with some final notes and possibilities of future work.

## Chapter 2

### STATE OF THE ART

In order to better situate the current conditions of the IST/DEM Industrial Automation Remote Laboratory on the present educational reality, some research was done on a group of topics related to laboratories, not only remote, but also hands-on, virtual, or even hybrid laboratories.

The most predominant topics, and, at the same time, relevant to the objectives of this work, are:

- The impact of the different types of experimentations in terms of educational value, economic value and simplicity of use;
- Type of existing experiments (systems control, automation, electronics, etc);
- The various architectures adopted and implemented, which includes physical connections/components and specific softwares.

#### 2.1 Hands-On, Virtual or Remote Labs?

One of the most discussed subjects by researchers is the value of hands-on versus virtual laboratories. According to [MaNick06], who reviewed existing literature on three major electronic databases – ACM, IEEE, and ScienceDirect – in a total of 60 articles related to hands-on, simulated and remote labs, *“the debate over different technologies is confounded by the use of different educational objectives as criteria for judging the laboratories”*. This means that there is no standard criteria to evaluate the effectiveness of a given lab work, and that arguments in favor or against each lab type can be inconsistent and ambiguous, which does not invalidate those arguments, but difficulties a comparative evaluation. Nevertheless, it can be found a more or less linear way of thinking towards the qualities and downsides of each type of laboratory.

One of the strongest arguments presented in favor of hands-on labs is the physical presence of the students that are able to feel physical components and set up the experiment. Subsequently, real data and uncalculated behavior provides an important factor in order to apprehend the difference between theory and practical experimentation, factor that simulated

labs lack. As stated in [TMSTPC11], this physical presence serves primarily as to supplement classroom teaching and to improve practical competences that better suits the labour market.

However, hands-on experiments are somehow too costly, not only demanding expensive equipment, but also specific maintenance, space, time, appropriate infrastructure and a specific instructor/technician. For example, in [SGA11], students of networking subjects only have access to a dozen of experimentation sessions a semester, having to use simulators and remote labs as alternatives.

As for simulation or virtual laboratories, they can be defined as any computer environment that simulates physical behavior at some desired similarity. They are generally seen as being, at some extend, more cost-effective than traditional laboratories, without harming learning quality. The use of a simulator allows for a more controlled process and to pause, rewind or repeat steps for multiple cases of study and a deeper understand of things. This is synonym of an active, flexible and independent mode of learning, centered on the student. On the other hand, there are the ones that relate excessive exposure to simulation to a disconnection between real and virtual worlds, because there isn't that physical quality inherent to real systems, denying learning by trial-and-error. The cost of this type of laboratories is also not always seen as a positive characteristic, since it may surpass the ones from real labs. More realistic simulations require more time to develop and detailed software.

Remote labs are similar to hands-on labs, with the difference in the location of the experimenter that has not to be the same as the experiment. Such fact removes the time conditioning of traditional experimentation and flexibilizes the function of the instructor/technician, if one is absolutely needed, and the time or quantity of tests a student can do. Also, data is real data, which benefits learning by trial-and-error, and is easily shareable between learning communities, what translates to an upgraded availability. Arguments against remote labs are kind of subjective ones, for they are related to the focus and immersion sensation of such labs. Some argue that students may be more distracted and impatient with computers, or that some of them don't consider remote labs to be realistic.

As an overview to this arguments, [MaNick06] states that there is not a general agreement on types of experimentation effectiveness because those are evaluated regarding dissimilar educational objectives, and proposes four educational goals to classify laboratory learning: *conceptual understanding*, *design skills*, *social skills*, *professional skills*. Hands-on, simulated and remote laboratories are very similar in accomplish goals in conceptual understanding and professional skills, but hands-on excels in design skills. Other points stated are that today is very hard not to have hybrid experimentations, which are real experimentations with virtual/software complements incorporated, and that there are different kinds of perception that play an important part in the success of certain experiment.

## 2.2 Areas of Applicability

In this section are described different experimental apparatus that laboratories can incorporate in order to complement theory learning, and which areas of education can they involve. From these descriptions, a better view of the range of remote experiments will be achieved.

### 2.2.1 Industrial Automation

In this kind of laboratories it is very common to find setups that include some hydraulic or pneumatic cylinders (for example, the remote laboratory in *Figure 1*), conveyor belts, sensors and actuators, and so on, which states are defined by “on/off” evaluation.

In [PCBM11], is presented an online platform that includes an automation laboratory. Some of the available plants are DC and AC motors, pneumatic setups and water tanks. There are also available simulators related to traffic light control, washing machine, elevator and conveyor belt. Similar to the DEM IARL, that will be described in chapter 4, this lab allows students to program in all five IEC 61131-3 programming languages, which are LD (ladder), IL (instruction list), SFC/Grafcet, ST (structured text) and FBD (function block diagram).

In [GRR11], is presented a remote experimental setup based on a Porsche model car, intended to foment the interest of high school students in engineering. The experiment allows monitoring and controlling lights, solenoids, motors, doors, horn, etc.



*Figure 1 – Pneumatic cylinders from an Australian automation remote laboratory.*

### 2.2.2 Systems Control

This is one of the most explored areas in terms of remote control. There is a large variety of existing laboratories available online that offer the possibility to implement some sort of systems or process control, like PID control, non-linear control or other user defined control. For

example, the Automatic Control Telelab, developed at University of Siena, makes available many remote experiments, via web browser, only related to systems control [URLACT].

In [AmmSla06], it is described the structure of a remote linear control engineering laboratory, which replicates, on a smaller scale, PLC-controlled temperature and flow processes typical in biochemical industries. The physical system is constituted by containers, pumps, coolers, heaters and a mixer system. Students can then test and experiment with sensors and actuators, program the PLC using industrial tools or perform a PID control of temperature and flow processes.

In [MRTBM11], we have a description of the apparatus of a process control remote lab, consisting in two vessels, pressure transmitter, level transmitter with level controller, flow transmitter with flow controller, hydraulic pump and control valve. This setup was designed in order to formulate exercises consisting in developing a process control model and design liquid level controllers in simulations, which is followed by testing an appropriate controller in real-time experimentation.

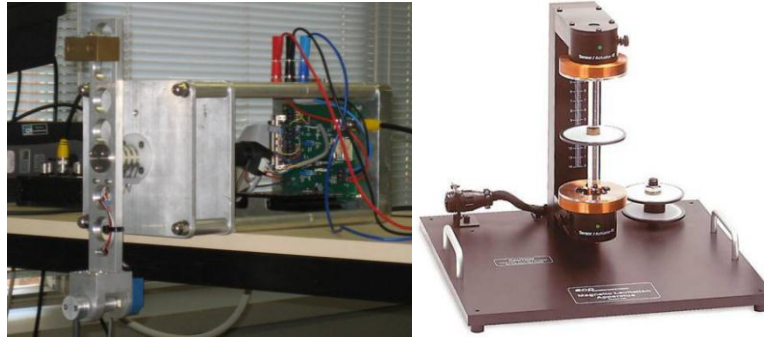
In [BMGT11], are described the initial steps towards the creation of a Fault Detection and Isolation remote laboratory, where students will be able to remotely develop and test model-based and data driven FDI schemes in Matlab/Simulink.

In [DFRPM08], is introduced the development of a remote laboratories network for the training in automatic control using the Internet. The physical system consists in an industrial scale model of four water tanks. The system has four inputs (2 variables of the pumps and the 2 positions of three-way valves) and four outputs (level of the tanks). The objective is to implement control loops of both SISO and MIMO systems with a PID control.

In [CVG11], are described the details related to an hybrid (remote and virtual) laboratory that provides simulation on signal processing, system control, system identification, and image processing on a Matlab/Simulink environment, and real experiments on networked control systems (such as the three-tank process) and distributed control systems.

Among the available experiments, there are, for example, the inverted pendulum control and power flow control generators [URLILABS], the couplet tank rig [URLUTS], and the following group composed by DC motor position/speed control, water tank level/flow control, magnetic levitation, 2 DOF helicopter control and team of mobile robots [URLACT].

Examples of remote laboratories on systems control are viewable in *Figure 2*.



**Figure 2** – To the left, inverted pendulum control preparation, from the University of Queensland, Australia.

To the right, magnetic levitation experiment, from the Automatic Control Telelab of the University of Siena.

### 2.2.3 Others

In the area of mechanical engineering are also available rigs for structural and thermodynamics analysis (for example in [RMLSC09]), or other form of material characterization, always keeping in mind that these experiments should be repeatable. For example, tests that involve material destruction are not possible. That would demand constantly setting up a new rig and someone to do that.

There can also be found many experiments in the electronics area, where it is possible to verify the impact of different values of frequency, current or voltage in a system, perform measurements on electronic circuits, learn the effect of OpAmps, etc. In general, it is a tool suitable for theoretical consolidation, allowing the user to “play” with multiple parameters and see the real response of each variation, and this motto is applied to the great majority of other remote laboratories. [MTTHM11] developed an online lab that addresses sequential logic circuits, and handles experiments in counters, shift registers and digital clocks. [TMSTPC11] implemented a remote laboratory to practice with FPGA (field-programmable gate array) devices. Integrated as a tool to the courses of quantum electronics and laser engineering from Bauman Moscow State Technical University, the Remote Laser Laboratory (RLL) grants safer lab sessions related to optical theory to their students, as seen in [TSGG11]. Another good examples, are the Virtual Instrument Systems in Reality (VISIR) – [GZHJGA11] - remote laboratory created at the Blekinge Institute of Technology, and the NetLab - [MaMe06] - implemented at the University of South Australia, that, in similar ways deal with implementing and analyzing circuits and the respective signals, with the help of realistic representations of function generators or oscilloscopes. In [Lobo11], is presented a remote digital design laboratory, in which students have access to a FPGA board where they can test combinational or sequential logic circuits, controllers, programmable processors, etc.

In the area of electricity, [CraMa11] took the first steps towards implementing a remote lab to perform three-phase induction motors experiments, and other electric machines. In [TkSch11]

are presented several experiments related to electricity and magnetism for high education purpose, such as experiments about: energy transfer in RLC circuits, phase in RLC circuits, Faraday's law of electromagnetic induction, transient phenomena in RLC circuits, source of DC voltage, transfer energy in DC circuits and emission of luminescent diodes.

Beyond mechanical, electric and electronic engineering, there are other areas of interest, such as physics and chemistry related laboratories ([URLELABIST]), computational intelligence ([BerBra11]), telecommunications ([SGA11]) and spectrometer experiments, or even meteorological stations ([URLTRUNI], [URLCUNI], [URLEMFEUP]) , these last ones being less interactive, but nonetheless very informative.

## 2.3 Network Architectures and Software

The connection between user and laboratory is accomplished with a chain of multiple elements with different and specific functions that, together, enable information to travel through all the system. In these networks we can have servers, dedicated PCs, the internet, controllers and other kinds of mechanisms that complement or improve the interaction with the object under control, for example web cameras or softwares. In this chapter, are illustrated several possible structures that are already implemented in known existing remote laboratories (which performance and accessibilities will be analyzed in the next chapter). Many variations can be found in the different networks studied, meaning that there is not a well-defined line between a type and another, but usually there are common points that can define a type of network. For this reason, they were assigned to the simplified architecture that better defines them, and any relevant variations to the architecture were described for each case.

In order to better understand the following architecture descriptions, a definition of the most common elements or concepts is presented next.

Web communication is commonly ensured by the TCP/IP protocol, which grants successful data exchange either on internet applications or local networks. Internet related services, such as e-mail, file transfer or world-wide web, in spite of communicating via their own protocols, are built based on TCP/IP, which demonstrates the importance of this protocol in any network.

In the same way as TCP/IP is to communication, HTML is to presentation. Web pages are edited in HTML and browsers interpret HTML files. This language is a powerful and flexible tool for design and structure documents, limited by some format properties, to be presented on the internet. Those format properties can be surpassed and further edited with an existing extension to the HTML tools, the more precise CSS style language. However, HTML, in its original form, is static, which means it does not include current data (i.e. current hours, day) on its pages. We then get to the concept of dynamic HTML. Dynamic data can be displayed on a web page with the help of a group of external technologies that enable web pages to be generated either on a server, or by the client of a network. SSI technology (Server Side Includes), for placing instructions related to current data on HTML pages, CGI (Common Gateway Interface), an



interface to start up script on the server, or PHP programming language, are all executed on the server. The advantage is that the browser only deals with pure HTML code and does not have to execute programs/scripts by itself, being a more secure option. The disadvantage is that continuous monitoring of a web page is more troublesome to achieve, since data is only sent to the client by refreshing the page, and even automatic refresh carries some issues such as unsteady/flickering image or modest rates of update. JavaScript programming language, noted directly onto the HTML page, and Java applets, applications sent from server to the client, are both executed on the client side in a browser, resulting in a steadier and smooth updating of values. Nowadays, in any browser, in addition to an HTML interpreter, is also implemented a JavaScript interpreter.

Dynamic pages are the base of any remote laboratory accessible by web browser and are the base of multiple software that serve the purpose of making the connection and data transfer between user and a system. One example of this software, widely used in industrial automation and other fields, is SCADA (Supervisory Control And Data Acquisition), an alternative to HTML pages that uses the concept of dynamic data - [URLSCADA I], [URLSCADA II]. A SCADA system allows to monitor and, at some extent, to control small or large infrastructures at a supervisory level. The software gets information on several points of the controlled site, presenting them to the operator through an HMI (human-machine interface), so he can monitor the system and, if necessary, send commands to available pumps, valves or actuators of the system, also represented in the HMI environment. Some important features usually integrated in SCADA software include alarm systems, process mimic, real time and historic trending, data acquisition and recording, data analysis and report generator. The connection and data transferring between SCADA and hardware devices can be done by several methods, depending on how they relate to each other. If SCADA and the devices use the same communication protocol then they communicate directly, for example by profibus or modbus; if they don't share the same standard communication protocol there are two solutions, one is using a native driver developed for a specific hardware with great execution performance, but bad compatible ability, and the other one is resorting to an OPC driver (Object Linking and Embedding for Process Control), which standardizes the communication protocol between SCADA and the hardware devices, without further incompatibility for future devices to be added on - [URLOPCI], [URLOPCI I], [URLOPCI II].

A generic architecture is presented in Figure 3. It is composed of one or multiple users connected to a server through the internet, which is responsible for accessing the process data and enable communication between user and process. The server can also connect to more than one process, and in some cases to a webcam for video streaming.



**Figure 3** – Generic architecture of a remote laboratory network.

In [TMSTPC11], a similar architecture was adopted, in which the user, through the internet, accesses a server responsible for a prototype board with a FPGA device and one webcam. The server connects to the prototype board through both a USB port and a RS-232 port, and to the webcam through an internal IP address. A database to save user data was also implemented in the server.

The following software was used for each section:

**User:** Web browser.

**Server:** Windows 2003 Server; MySQL for the database, supported by an Apache web server.

In [MRTBM11], the user, through a web browser, has access to a server with a SCADA system responsible for the connection between the client and the remote laboratory. This one is composed of an OMRON PLC (programmable logical controller), connected to the server from a RS232 port, and an experiment related to process control engineering.

The following software was used for each section:

**User:** Internet Explorer (web browser).

**Server:** LabVIEW with LabVIEW DSC Module, to function as a SCADA system; OMRON CX OPC 2.0 Server for I/O data transfer between the PLC and the LabVIEW SCADA.

In [TSGG11], the adopted architecture also falls in to this generic setup. In order to access the Remote Laser Laboratory, the user runs a Flash internet application, used to add animation video and interactivity to web pages, hosted by the server. The server manages communication with the application, executes control algorithms, provides administration level access and manages the webcam, among other minor related functions.

The following software was used for each section:

**User:** Web browser, to view the Flash internet application.

**Server:** LabVIEW.

In [SGA11], the user uses an application in Java to connect through TCP/IP to the server, which will authenticate the user and enable the desired actions. One of those actions is to gain access to a lab pc by remote desktop, which means that the user sees the remote desktop of the lab pc in his own monitor and can perform the same actions as if he were working physically on that pc. In this case the lab pc corresponds to the process of the generic network architecture.

The following software was used for each section:

**User:** Client application developed in Java; Client applet for remote desktop.

**Server:** Application developed in Java to authenticate the user and enable functions. It also “wakes up” the lab PCs and grants access to them.

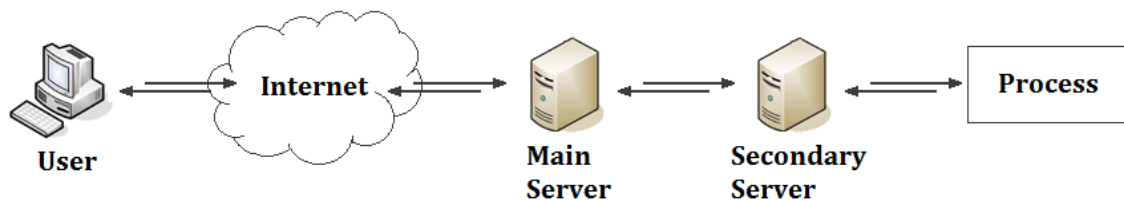
In [Lobo11], the user uses a web browser in which he has access to a webpage interface, plus specific software that virtualizes the Altera DE2 FPGA board switches and keys, that the user wants to control. The server has a database and is connected through USB to the board and to a webcam.

The following software was used in each section:

**User:** Web browser with interface; Altera Quartus II digital design IDE.

**Server:** Apache webserver; Altera Quartus II software; Yawcam applet for the webcam.

A modification of the previous generic architecture is the one presented in Figure 4. The difference is the addition of one other server, generally dedicated to just one experiment. In other words, the main server connects to any number of secondary servers, and each secondary server connects to a process.



**Figure 4** – Architecture with Main and Secondary Servers of a remote laboratory network.

In [GZHHGA11], it is described the data flow of VISIR remote laboratory for analog electronics experimentation. The user creates a circuit, in a virtual way, with the help of a web interface that will send commands and components values to a measurement server, which contains the maximum components values, instruments adjustments and allowed circuits of a certain experiment. From this server the commands will be passed to an equipment server over TCP/IP. This server controls a relay switching matrix coupled to a PXI chassis that contains the instruments. Both of them will implement the circuit created by the user. Results are then sent to the measurement server and presented to the user in the web interface.

The following software was used for each section:

**User:** Web browser; Interface developed in Adobe Flash.

**Main Server:** (Measurement Server) Not mentioned.

**Secondary Server:** (Equipment Server) LabVIEW.

In [HDFPFP11], it is shown the structure and connections of the Instituto Superior Técnico remotely controlled laboratory “e-lab”. The user accesses, through the internet, to the e-lab master server that reroutes the user to the desired experiment. Each one of the experiments is

controlled by a driver running on a single computer, and each of these computers are part of a local network connected to the master server, where also exists a video streaming server. The end point process is composed of the experimental apparatus and a microprocessor.

The following software was used for each section:

**User:** Web browser.

**Main Server:** Not mentioned.

**Secondary Server:** Driver, programmed in Java, and based in the ReC (Remote experienced Control).

In [CVG11], the user, running a client application installed on his computer, connects to a server via an Ethernet local area network or the internet. In this server runs an application that establishes the interaction between any real process and the client application. A PC with I/O interface boards is connected to each one of the experimental apparatus, and each one of the PCs is connected to the server.

The following software was used for each section:

**User:** Web browser; client application.

**Main Server:** Server application written in Java, and making use of Matlab functions for I/O communication with DAQ cards.

**Secondary Server:** (Experiment PC) Software based on Matlab/Simulink with the Humusoft Extended Real-Time Toolbox.

In [GRR11], the user makes use of a web browser to connect to a main server, with a database, responsible for authentication, video streaming, scheduling and informative purposes. The main server then connects to a lab server that manages the data transferring between the end point system (through a Data Acquisition Card) and camera, and the main server.

The following software was used for each section:

**User:** Web browser and LabVIEW runtime.

**Main Server:** Windows Server 2003; IIS web server; Macromedia Flash Server Communication, for webcam video streaming; PHP application integrated in Moodle, for scheduling.

**Secondary Server:** (Lab server) LabVIEW and LabVIEW web server.

In [MSFSA11], there is a slight difference to the described architecture, but it can still be included in this type of structure. The user has access to a 3D Virtual World that, in theory, has similar functions to a server. In this 3D Virtual World, which contains a database, the user can view files, interact with other users and choose a real experiment to perform. When activating this last option, the 3D Virtual World changes data between a MSW (Micro Servidor Web) that in turn connects to the real apparatus of the experiment and a webcam.

The following software was used for each section:

**User:** Web browser; Hippo Viewer client software, for viewing the 3D Virtual World; Quick Time image viewer.

**Main Server:** (3D Virtual World) Moodle; Sloodle, to present Powerpoint slides within the virtual environment; OpenSim, to create the virtual environment.

**Secondary Server:** MSW.

In [CNM11], is described the RemoteLabs platform architecture, for electric machines remote experimentation. The architecture is structured around two applications and a database. A web application, hosted by an Apache application server, provides the user interface that he accesses through a web browser. This web application can schedule the experiment and save data in the database. There is also a Motor Interface application, composed of two modules, a client module and a server module. The first one is launched by the web application in order to submit the necessary experiment parameters to the motor interface server module, which in turn will interact with the motor drive to make it run the experiment. When the experiment ends the results will travel backwards until the web application stores them in the database and generates XML files. The user has the option to get the experiment results in PDF or CSV files.

The following software was used for each section:

**User:** Web browser; web application, written with HTML and CSS code.

**Main Server:** Apache application server, to host the web application; MySQL database and phpMyAdmin, a graphical user interface to interact with MySQL; client module of the motor interface application, written in C#.

**Secondary Server:** (Experiment Server) Server module of the motor interface application, written in C#.



## Chapter 3

# REMOTE LABORATORIES IN USE

### 3.1 Introduction

As a way to research on different accessibility methods, some of the most known international and national remote laboratories were accessed, and the experience described in the following sections. In most of the cases there was a heavy limitation to guest users, to users external to the respective university or institution, granting only demonstration experiments. The booking system, when present, was also addressed, and how available is the information related to any experiment. Unfortunately, for the chosen national remote laboratories, only access to a small group of experiments was possible, due to maintenance or registration limitation.

### 3.2 International Examples

#### 3.2.1 MIT 'iLabs'

The MIT iLabs Project is intended to gather a large and rich set of online labs at a global scale, aiming to share “as broadly as possible” the remote lab experience within higher education and beyond. For this purpose it was created the ISA – iLab Shared Architecture, a “robust, scalable, open-source infrastructure built on web service that has been developed to provide a unifying software framework that can support access to a wide variety of online laboratories”, as stated in *[URLILABS]*.

The main page of MIT iLab Project offers a lot of information on the project concept and its background, the implemented architecture and information on software requirements for a potential remote laboratory inclusion on iLabs.

Currently there are available nineteen experiments, divided in various fields such as: electronics, control, physics, spectrometer, telecommunications, RF and microwave communications. Clicking in one of the experiments opens a page where the user can read

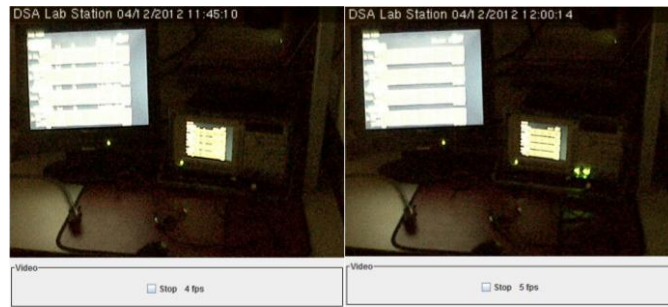
tutorials, manuals, assignments, papers and theses related to the lab, system requirements and device specifications. In the case of being an experiment developed by a foreign university, it redirects the user to the correspondent page. To gain access to each experiment the user has to register in a database, different from university to university.

Registering in MIT OpeniLabs service broker grants access to three experiments: MIT Microelectronics Device Characterization, MIT NI-ELVIS and MIT Dynamic Signal Analyzer. In order to register in MIT OpeniLabs service broker, the user has to fill out and submit an online form - *Figure 5*, at the end of which an email will be send to confirm the creation of the user's account. The user can then access his account and see what experiments he is allowed to run, he can view the date and time of past performed experiments, and make annotations on each of them.

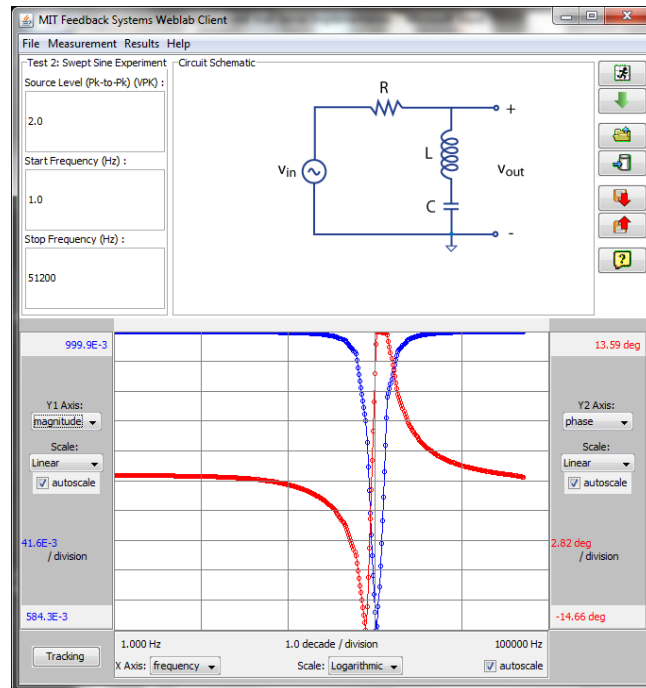
**Figure 5** – Online form to register in MIT OpeniLabs Service Broker.

Selecting the Dynamic Signal Analyzer the user can activate a Lab Webcam, as seen in *Figure 6*, and launch the lab interface, presented in *Figure 7*. A Java-enabled web browser is required in order to run both the experiment and the webcam. The experiment is about frequency domain measurements on electronic circuits and control systems, done by an Agilent 35670A Dynamic Signal Analyzer. The interface, in the form of a Java applet, has a virtual representation of the circuit and allows the user to run the measurement, save or load a setup, retrieve the graph data after running the experiment and retrieve offline data based on theoretical models.





**Figure 6** – MIT Microelectronics Device Characterization iLab Client Lab Webcam view in idle state (to the left) and running the experiment (to the right).



**Figure 7** - MIT Microelectronics Device Characterization iLab Client Lab Interface.

### 3.2.2 WebLab-Deusto

WebLab-Deusto, as stated at its webpage [URLWLD], is “an open-source distributed Remote Lab used with students at the University of Deusto since February 2005 as an essential tool for their practice works in different engineering-related subjects.”. WebLab-Deusto allows for easy integration of new experiments from outside developers, resulting in a progressive enlargement of its experiments database. For these experiments WebLab-Deusto offers several important features such as, for example, authentication, queue management, user tracking, administration, linking accounts with facebook or adapting interfaces to mobile devices.

In order to have access the experiments, the user has to log in with an username and password - Figure 8. Students from Deusto University are already present in its database, and can link an account with their facebook. Exterior users have to log in as guest, and have limited access to a group of experiments.

### WebLab-Deusto

WebLab-Deusto is a Remote Laboratory. Students access experiments physically located in the university, having the same experience as if in traditional hands-on-lab sessions. There is more information regarding the project in the [WebLab-Deusto Research Group](#) site.



#### Support

For any technical issue you may find, please contact us at [weblab@deusto.es](mailto:weblab@deusto.es)



#### Demo

If you do not have a user account, you can try our demo experiments with the username **demo** and the password **demo**.



#### Mobile

Perform your experiments in the mobile version by clicking [here](#)



#### Open Source

WebLab-Deusto is Open Source Software, and it is available in <http://code.google.com/p/weblabdeusto/>

Log in

Username:

Password:

Log in

Some experiments allow guest access

Login as guest

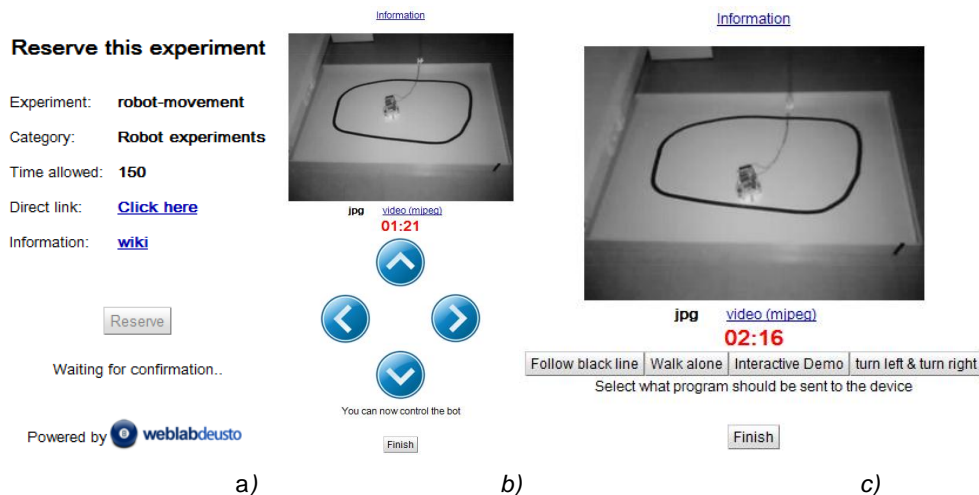
Don't have an account? Create one through Facebook

Create an account

**Figure 8 – WebLab-Deusto log in page.**

The available experiments for guests include a dummy experiment (just to test Java or Flash usability in potential experiments), a xilinx experiment (to test queue management) and demo versions of real remote experiments, such as robot movement, FPGA control or electronic circuits implementation through VISIR.

The robot experiment - *Figure 9* - lets the user control a bot remotely, ordering it to turn left, turn right, move forward and move backward. A video streaming gives the visual feedback to the user, which has a 2:30 minutes window to interact with the bot. Optionally, the user can choose a program to be sent to the bot, like “*Follow black line*” or “*Walk alone*”.

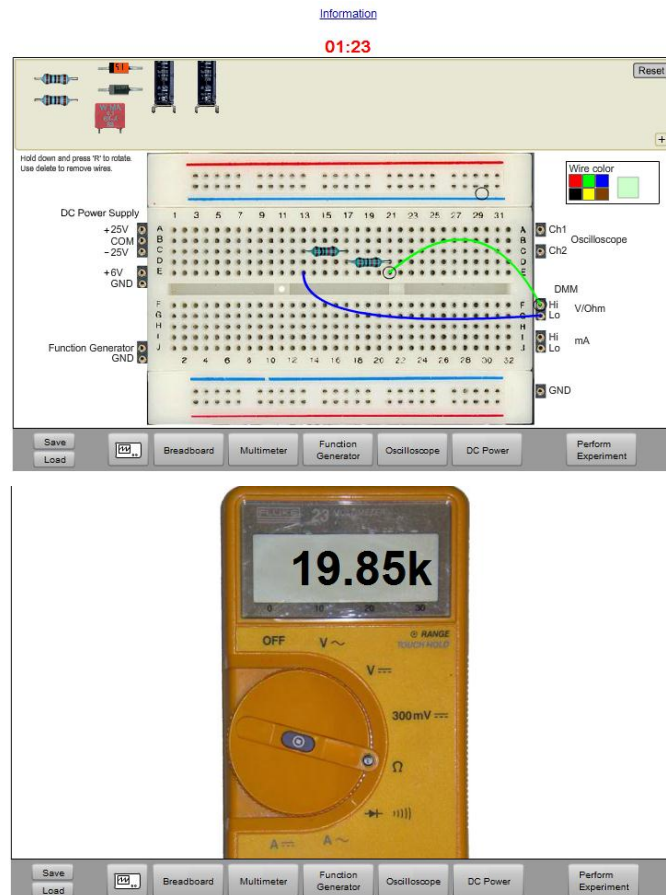


**Figure 9 – WebLab-Deusto robot experiment:**

- Reserving the experiment;
- Controlling the bot movement;
- Sending a specific program to be ran.

The VISIR (Virtual Instrument Systems In Reality) experiment lets the user access the Blekinge Tekniska Högskola OpenLabs VISIR through WebLab-Deusto, created for electronic

circuits remote experimentation. It is shown a first page where the user can choose which elements he wants to use in the experiment, such as breadboard, multimeter, function generator, dc power or oscilloscope. Then, the breadboard is presented, where the user can create circuits with resistors, capacitors or inductors, and perform measurements with a multimeter or analyze the signal with the oscilloscope. A maximum of 5 minutes is allowed to the guest user to run the experiment.

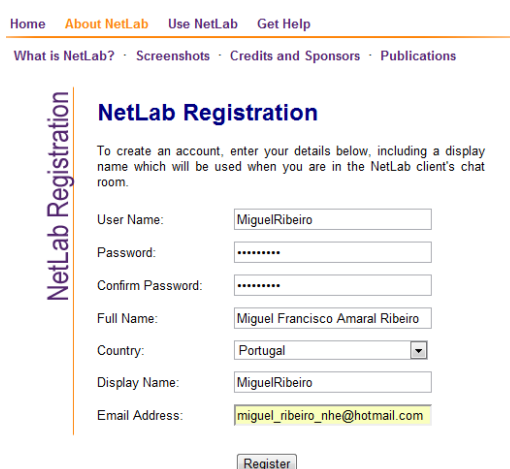


**Figure 10** – WebLab-Deusto and BTH OpenLabs VISIR experiment example.

### 3.2.3 NetLab

Netlab [URLNETLAB] is an online remote laboratory developed by the University of South Australia for the teaching of electrical engineering. It provides a realistic GUI (Graphical User Interface) that the user can interact with as if it was a real instrument, and analyze the measurements done in real apparatus accessed remotely. The NetLab also has available a circuit builder feature, which lets the user build his own circuit with a set of components such as resistors, capacitors and inductors. A switching matrix will then implement that circuit and share the measurements with the user. A high quality web camera is also available for video feedback of the physical instruments, enhancing the feeling of the student being on control of a real experiment.

In order to test NetLab interface, an account has to be created - *Figure 11*, and an hour has to be booked - *Figure 12*. In the booking menu, the user can see two tables with time slots. At the left a table with the available time slots is shown, and to the right is the table where the user selects the desired time slots. The user is allowed to register up to three hours per week, and only three booking spaces are available for each hour. After booking a time slot, the user can download a NetLab Introduction PDF file for a brief explanation on how to use the interface, and students can download practical works to implement with the remote laboratory help. To launch NetLab the user goes to the respective menu, where a notification on the suitability of the user's Java version is shown, and also links to steps not taken by the user in order to run NetLab. When clicking *Launch Netlab* a JNLP file is downloaded, responsible for the Java applet parameters and on how to implement the launching mechanism [URLJWS]. Executing the file will allow to launch NetLab.



Home [About NetLab](#) [Use NetLab](#) [Get Help](#)

[What is NetLab?](#) · [Screenshots](#) · [Credits and Sponsors](#) · [Publications](#)

### NetLab Registration

To create an account, enter your details below, including a display name which will be used when you are in the NetLab client's chat room.

User Name:

Password:

Confirm Password:

Full Name:

Country:

Display Name:

Email Address:

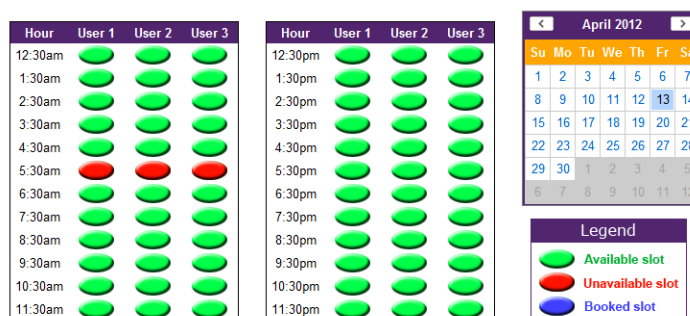
**Figure 11 – NetLab registration menu.**

## Booking

Please select the time you wish to book for: **Friday, 13 April 2012.**

All times are shown in your local timezone, according to your system clock.

**You can book up to 3 hours this week.**



Hour	User 1	User 2	User 3
12:30am	Available	Available	Available
1:30am	Available	Available	Available
2:30am	Available	Available	Available
3:30am	Available	Available	Available
4:30am	Available	Available	Available
5:30am	Unavailable	Unavailable	Unavailable
6:30am	Available	Available	Available
7:30am	Available	Available	Available
8:30am	Available	Available	Available
9:30am	Available	Available	Available
10:30am	Available	Available	Available
11:30am	Available	Available	Available

Hour	User 1	User 2	User 3
12:30pm	Available	Available	Available
1:30pm	Available	Available	Available
2:30pm	Available	Available	Available
3:30pm	Available	Available	Available
4:30pm	Available	Available	Available
5:30pm	Available	Available	Available
6:30pm	Available	Available	Available
7:30pm	Available	Available	Available
8:30pm	Available	Available	Available
9:30pm	Available	Available	Available
10:30pm	Available	Available	Available
11:30pm	Available	Available	Available

April 2012

Su	Mo	Tu	We	Th	Fr	Sa
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	1	2	3	4	5
6	7	8	9	10	11	12

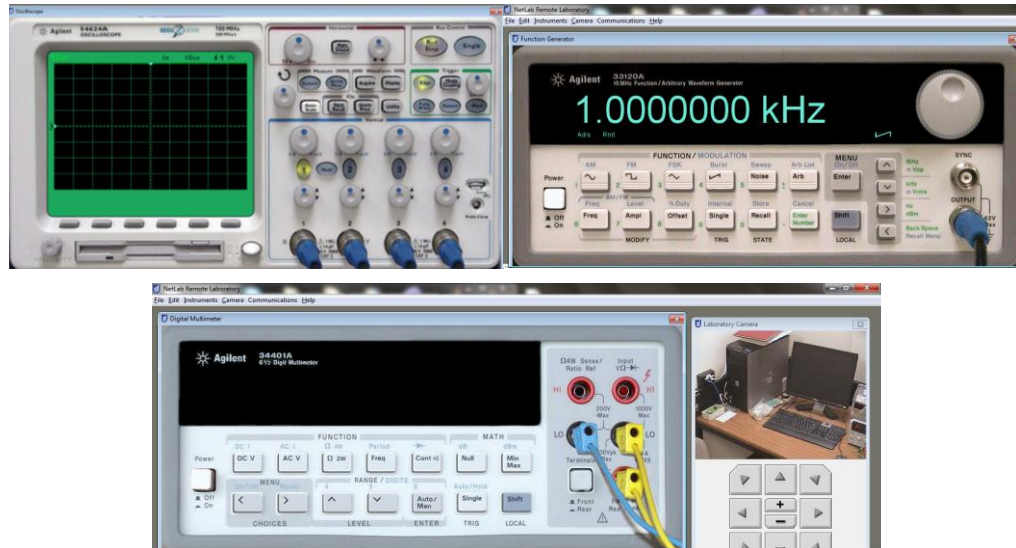
Legend

- Available slot
- Unavailable slot
- Booked slot

**Figure 12 – Netlab booking menu.**

Initializing NetLab opens a login window requesting a registered username and password. A connecting will be made and the interface main window opens. The user can select from three virtual instruments - *Figure 13*, an oscilloscope, a function generator and a digital multimeter. There is also a menu for camera settings, including predefined positions, for example, on the real oscillator, instruments, the server, switching matrix and even the door of

the lab. The camera also zooms in or out and one can control its position manually. The remaining time left for the user NetLab session is presented in the bottom of the main window, and a warning message appears when only 10 minutes are left. The data obtained while measuring the resulting values of a circuit can then be saved in a \*.txt file.



**Figure 13** – NetLab virtual instruments.

### 3.3 National Examples

#### 3.3.1 FEUP Remote Experiments

Currently, FEUP has available a site [URLFEUP], based on Moodle, to manage access to a group of 13 remote experiments. Each of them has a corresponding informative *pdf* file on how to use and run the experiments. Most of them require LabVIEW 7.1 or LabVIEW 7.1 Run-Time to work properly in a web browser, the second one being easily downloaded on a given link. Due to the maintenance state of some of the experiments, and to a guest type login, most of them were not available.

Full registration is done by request; however there isn't a direct link to perform it. Optionally, the user can login as guest – *Figure 14*, granting a type of access that is not too limited, just a couple of unavailable experiments.

Login here using your username and password:  
(Cookies must be enabled in your browser) ?

Username:

Password:

---

Some courses may allow guest access:

---

Forgotten your username or password?

**Figure 14 – FEUP Remote Experiments login page.**

The existing remote experiments range from mechanical material characterization to PID control, remote control of a Porsche model, meteorological station monitorization or temperature calibration, among others.

The user can perform an experiment by booking a one hour session from an online table that shows the availability/occupancy of the lab for any day - *Figure 15*. From there the user can call the remote experiment that will run in the web browser. An attempt at opening the booking page in Google Chrome resulted in a flawed presentation, meaning that is incompatible with that type of web browser. Opening with Internet Explorer solved this problem.

Server clock reported to local time zone: quarta-feira, 2 de Maio de 2012 17:44:43					Server time: 17:44:43
Local time: quarta-feira, 2 de Maio de 2012 17:44:32					Discrepancy: 00:00:11
	Wed May 2	Thu May 3	Fri May 4	Sat May 5	
0:00					
1:00					
2:00					
3:00					
4:00					
5:00					
6:00					
7:00					

**Figure 15 – FEUP Remote Experiments booking system detail.**

Selecting (by booking a session) the Straightness Evaluation remote experiment - *Figure 16*, the user interface is loaded, with a real-time video section to the right. The user can edit some experiment parameters such as the starting and end position of the measurement, the step size of the digital dial indicator, the number of consecutive test to be performed and the email where the results should be sent to.

Edit Operate

**FEUP**  
Faculdade de Engenharia da Universidade do Porto

**LIM**  
Laboratory of Instrumentation and Measurement

Virtual Left Limit: 0.0 mm

Virtual Right Limit: 180.0 mm

Step size: 10.0 mm

Number of Tests: 1

Email:

Sponsored by:

Laboratory of Instrumentation for Measurements (lim@fe.up.pt)

**Straightness Evaluation**

TIME OUT: 13:45

Left Switch: ☐ Right Switch: ☐

Dial Test Indicator (mm)

Position (mm)

Set Point: 160 Actual Position: 158.3 Test Number: 1

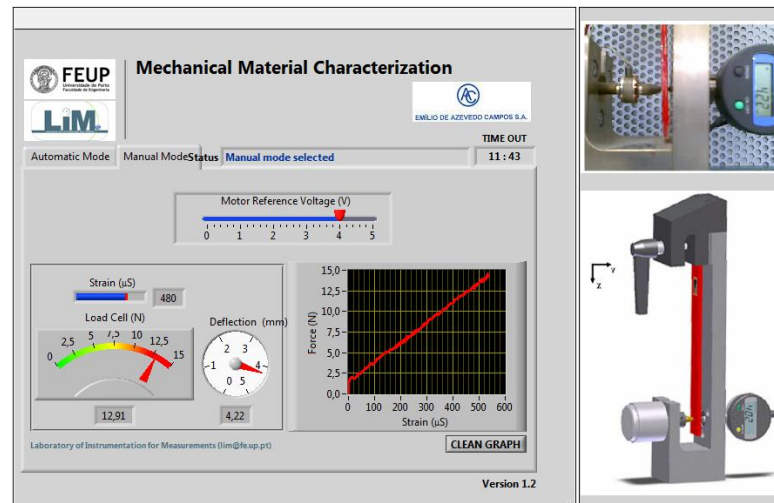
Status:

Version 1.1

LIM – Laboratory of Instrumentation and Measurement

**Figure 16 – FEUP Straightness Evaluation remote experiment.**

Selecting the Mechanical Material Characterization remote experiment - *Figure 17*, the user interface is loaded, with a real-time video section and an image of the system to the right. The user can choose to run the experiment in manual or automatic mode. In manual mode the user controls the motor reference voltage which will impose a load to the cantilever beam, for strain and deflection analysis. In automatic mode a predefined procedure is executed. The results can then be sent to the user email.



**Figure 17** – FEUP Mechanical Material Characterization remote experiment.

### 3.3.2 FCT RemoteLab

The FCT remote automation and control laboratory [URLFCT], described in chapter 2, is only accessible by users registered in the university MySQL database. It is only requested a valid login username and password - *Figure 18*, and there exists a separate login for administration purpose.

**Figure 18** – FCT RemoteLab Login page.



### 3.3.3 ISR Remote FPGA Laboratory

The ISR Remote FPGA Laboratory is online at *[URLISR]*, where are also available some digital systems games/quizzes, chat and downloadable content. In order to access the FPGA board the user has to request registration, by sending his name, email and institution, plus a brief note about the motives for the registration - *Figure 19*. Unfortunately, no feedback came back from the request, so no further analysis was possible.

**Registo: não tem convite?**

Para criar uma conta no servidor necessita de um convite. Pode solicitar um convite, preenchendo para o efeito o seu nome, e-mail, o nome da instituição a que pertence e, caso seja aluno do DEEC, o seu número de aluno, no formulário que apresentamos.

No caso de ser externo ao DEEC ou ao ISR-UC, deverá acrescentar ainda uma nota de apresentação, em que indique a sua área científica e os motivos pelos quais tiraria partido do nosso sistema.

O registo do servidor permitirá não só obter o acesso à placa remota, como também acesso à área de jogos e à área de OpenCores@ISR-UC. No entanto, se não estiver interessado em obter acesso à placa, desmarque a caixa de verificação correspondente.

Nome:  Instituição:

E-mail:  N.º de aluno:

Quero obter acesso à placa remota: ☒

Nota de apresentação:

Aluno de Mestrado em Engenharia Mecânica do ISI, ramo de Sistemas. Gostaria de ter acesso ao vosso laboratório remoto para pesquisa de tese sobre experimentação remota. Obrigado.

**Figure 19** – ISR Remote FPGA Laboratory registration request page.

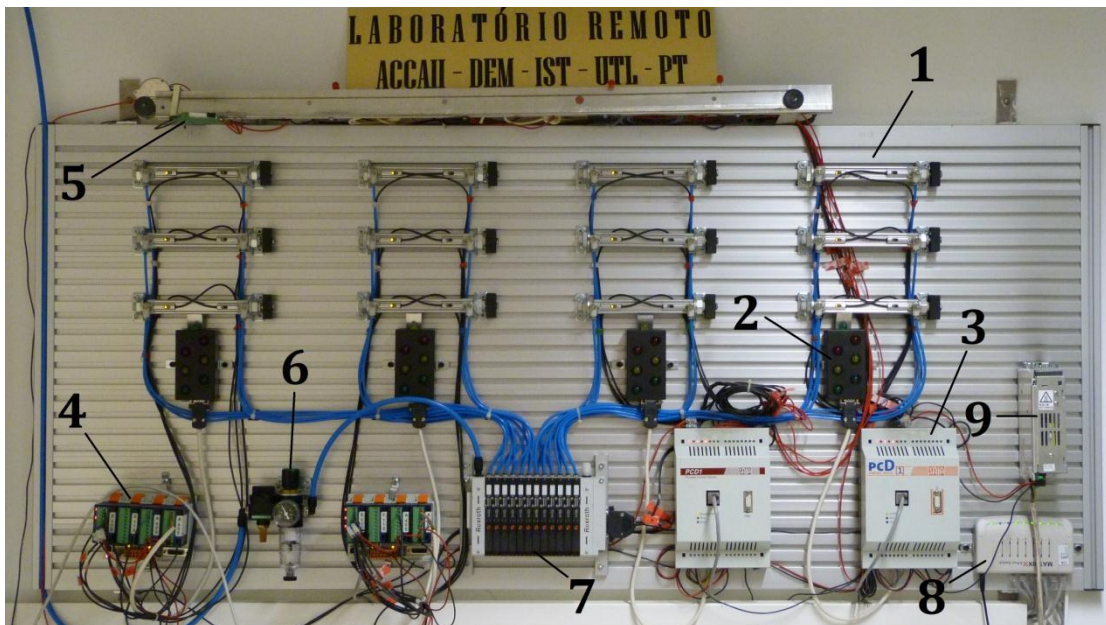


## Chapter 4

# THE IST INDUSTRIAL AUTOMATION LABORATORY

### 4.1 Laboratory Setup

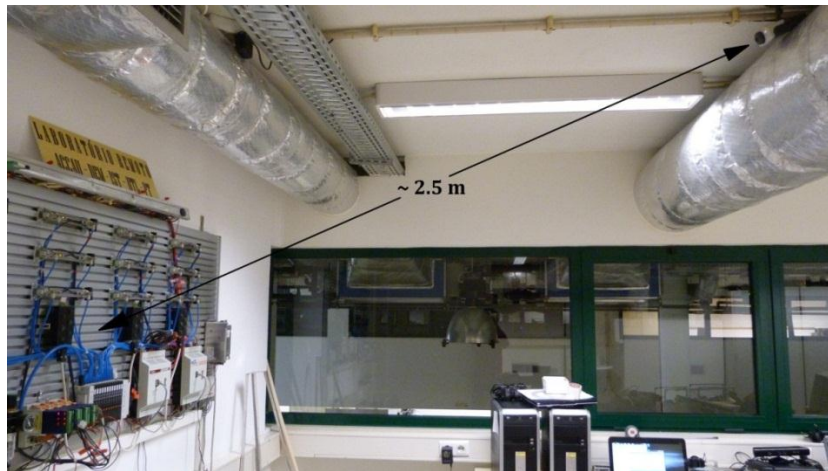
A general overview of the remote laboratory composition is showed in *Figure 20*.



**Figure 20** – General view of the remote laboratory physical setup.

The laboratory is composed of four stations, each one of them consisting of three pneumatic cylinders – **1** – and seven LED's - **2**. For each station there is a PLC responsible for data management and control. Two of them are older PCD1 – **3** – models, with no web server feature included, and the other two are PCD3M3330 – **4** – models, compatible with web page storage within the PLC. There is also installed an horizontal elevator – **5**. For pressure regulation there is a manometer – **6** – installed where the air accesses the laboratory. Then the air passes through a pneumatic valve system – **7** – that redirects the air to actuate the cylinders when demanded. An eight port switch – **8** – connects every PLC and the webcam to the VPN, so they can be accessed by the user of the remote laboratory. A power supply – **9** – distributes electricity to all components of the lab.

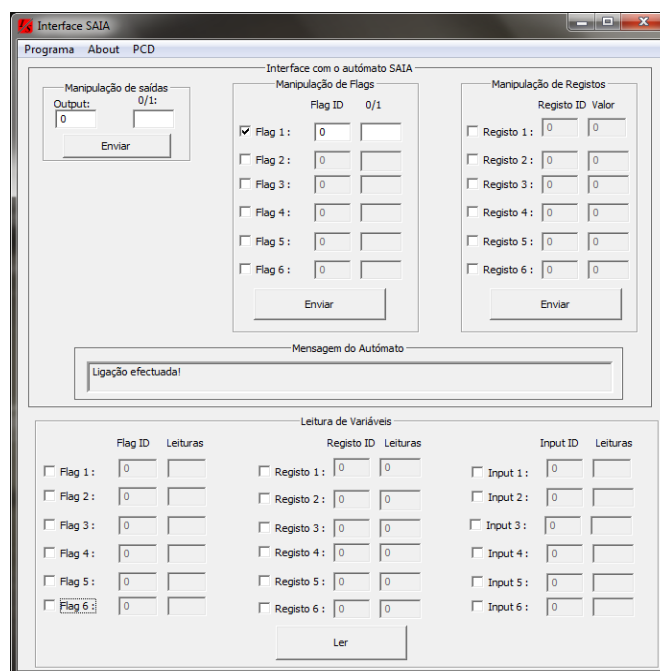
For video streaming there exists a webcam located, approximately, at 2.5 meters of the stations, as seen in Figure 21. The camera is static and it captures the entire group of stations.



**Figure 21** – Camera location in relation to the remote laboratory physical setup.

## 4.2 The Existing Interface

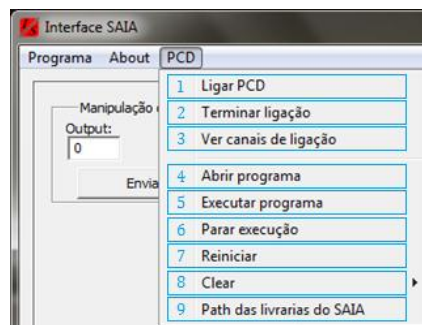
The present interface is a program independent from SAIA Project Manager, for inputs, outputs, flags and registers reading/manipulation. The simpler version is as seen in *Figure 22*, which is similar to the normal version, but without a section that takes advantage of the user's webcam as a motion sensor, that alternates output values depending on if it detects movement or not. To the purpose of this work, an analysis of the simpler version is sufficient.



**Figure 22** - General view of "Interface SAIA".

### 4.2.1 Menu Bar

The menu bar shows three menus – *Programa*, *About* and *PCD*. In menu *Programa* we can exit/close the interface, same as the “x” button on the top right of the window. In menu *About* we can see information on the makers of the Interface. The *PCD* menu is the more complete one. In *Figure 23* it is shown the *PCD* menu and the available commands.



**Figure 23 - Menu “PCD”.**

Ordered by the numeration above, it is described the function of each command:

**1 Ligar PCD** – It opens a window where the user can choose the type of channel he wants to connect to a PCD. The range of options goes from PGU, S-Bus, PC104, S-Bus Modem, SOCKET (which is the one currently used in the lab), S-Bus USB to Profi-S-Bus. For each type of channel there is a section to edit the connection specifications, such as CPU Number or IP Address. Then, pressing “Ok” it establishes the connection to one of the PCDs.

**2 Terminar ligação** – It cuts the connection to the PCD, making the interface go offline.

**3 Ver canais de ligação** – After connecting to the PCD, this option becomes available, opening a window that shows information about every type of channel already enumerated above.

**4 Abrir programa** – It opens an explorer window, where we can search a \*.pcd file where the SAIA project is contained. Then the interface downloads it to the PCD, which will allow running the program.

**5 Executar programa** – It runs the downloaded program, if there is one.

**6 Parar execução** – It immobilizes the PLC, stopping the program.

**7 Reiniciar** – It resets the PLC. All outputs, flags and registers go to zero.

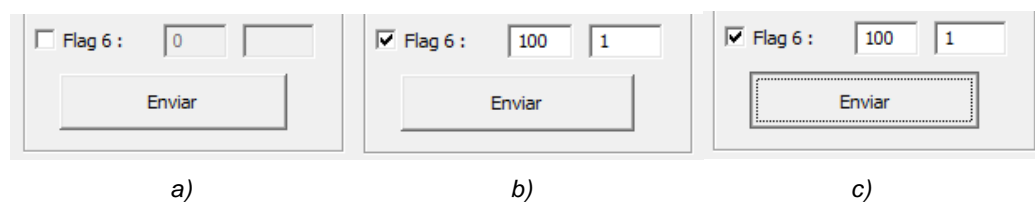
**8 Clear** – It shows a sub-menu with the options *Outputs*, *Flags*, *Registers*, allowing to reset any group of variables.

**9 Path das livrarias do SAIA** – It opens a window where the user can set a new path for the SAIA libraries, where the interface gets all the information that enables it to interact with each and any of the PCDs.

## 4.2.2 Sending Section

The main window is composed of, essentially three sections, one to send values, one to read values and one to display specific information.

The sending section lets the user manipulate output, flags and registers values. To modify a value of a flag or register, the user has to check any number of boxes from the six available lines that each type of variable has, insert the respective address, insert the desired value, and click *Enviar*. It is important to state that only the checked lines will be taken in account at that moment, and that each type of variable has its own sending button. In the case of output manipulation, the steps are identical, except that only one line is available and that there is no box to check. This process is exemplified in *Figure 24*.

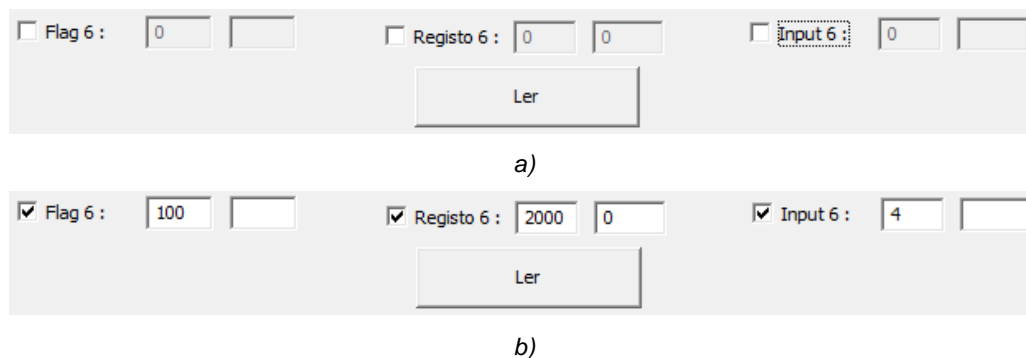


**Figure 24** – Four steps process to variable manipulation:

- a) Initial state of the interface sending section, only line six is visible;
- b) Steps one, two and three – checking, inserting address, inserting value;
- c) Step four – clicking *Enviar* to send information to the PLC.

## 4.2.3 Reading Section

The reading section lets the user read flags, registers and input values. Similar to the sending process, to read a value of a variable, the user has to check any number of boxes from the six available lines, insert the respective address and click *Ler*. In this case, only one reading button exists for all checked lines of all types of variables. This process is exemplified in *Figure 25*.



☒ Flag 6 : 100 1  
☒ Registo 6 : 2000 20  
☒ Input 6 : 4 1  
 Ler

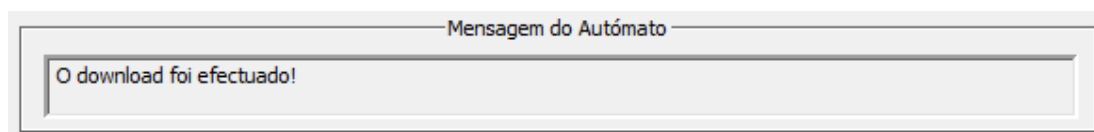
c)

**Figure 25** – Three steps process to variable reading:

- a) Initial state of the interface reading section, only line six is visible;
- b) Steps one and two – checking, inserting address;
- c) Step three – clicking Ler to read information from the PLC.

#### 4.2.4 Message Section

The message section displays important information on the most recent action taken by the interface. Messages are shown after connecting successfully or not to the PCD, when downloading a program or resetting variables, etc. This means that the user can always confirm if his last action resulted in the desired outcome. One of these messages is exemplified in Figure 26.



**Figure 26** - Example of message section display.

#### 4.3 Issues and Improvements, reasons for a new interface

From previous experience with the existing interface, and after studying the state of the art, there can be found some points where this program may be improved.

The purpose of the interface is to simplify the interaction with the laboratory, which the present one accomplishes in comparison with the *Online Debug* function of the SAIA Project Manager. However, it lacks in terms of feeling of immersion, to better represent the components of IST pneumatic industrial automation stations, and, at the same time, it could be simpler.

The first and most immediate improvement to be done is changing the platform of the interface, as proposed in the objectives of this dissertation. Aiming at a more flexible tool is essential to reduce the software needed to interact with the laboratory. By using a standard web browser as a new way to connect with the remote laboratory, the requirements list is downsized to only SAIA software (and ipView for web cam viewing, but even that turns out to be optional, as it will be explained further ahead).

It is important, as a complement to the hands-on laboratories, to have procedures that strongly resemble traditional experimentation so that users are prepared to work with physical

equipment, giving, at the same time a more immersive and intuitive interaction. The present interface lacks in physical representation, denying a good mechanism to absorb concepts of real relation amongst elements. If I send a flag with address 101 and value 1, what does it mean? How does it translate to a physical industrial automation process? Is it a button? Toggle or pressure/release? This kind of questions could be answered with the help of a work environment (in this case an interface) more realistic.

As a matter of fact, in the course of Industrial Automation, are requested specific works on LD, IL, SFC/Grafset, ST and FBD programming, that require first steps to be taken with the help of the remote laboratory. The student develops his programming project, downloads it to the PLC and tests it through the interface. In this phase, and with this interface, students don't deal with the symbol names that they have associated to their variables. Variables are only manipulated by reference to their address. This is quite an important fact that makes impact between this phase and the next one where students experiment in the hands-on laboratory. There, in spite of having the possibility to continue using the interface as register manipulation, students are required to make use of the apparatus at their disposal, which include not only the pneumatic cylinders and lights, but also several on/off buttons, pressure buttons, conveyor belts or elevators. So, symbol names have to be readjusted to these new components and flags are no longer manipulated through the interface. The first assign, or even second, proves to be more time consuming, due to this small difference in procedure that has to be assimilated, and that could be shortened with an interface improvement.

In chapter 4.3 were described the Four Steps Process and Three Step Process of, respectively, sending and reading sections. These processes could and should be simplified. While using the interface, disconnecting with a remote station clears every box of the main window, this means rewriting every address of every variable that we want to manipulate or read if we want to connect again to the same or other station. This variable setup is exhausting, time consuming and not practical. The read button also turns out to be dispensable since an apparent continuous reading is more preferable.

Independent to the interface, the accessibility to the remote laboratory is an unresolved issue. There is no way to block the access of a student to a station that is already occupied, originating conflicts in variable manipulation. As a way to bypass this problem, it was proposed to students to set one of the outputs of the remote laboratory, which in this case is a presence light, which signals to other potential users that one certain station is in use. And there is no time limits also, only common sense.

Related to the previous issue, students have access to a remote camera that provides an image of the four existing stations, including the presence light of each one. This is the only way, besides Saia Project Manager, to view the state of outputs. The current interface does not inform on output values, especially in the case of light outputs, since pneumatic cylinders activate sensors that can be read. In the case of remote cam malfunctioning or not having the ipView software on the computer, it will be harder to follow up the running of a program, or even

notice that certain station is currently occupied. A representation of these outputs on the interface will be an important improvement

A final, but not less important reason is the building of these web pages as part of a course. Indeed, currently any PLC is web connected and to build web interfaces as HMI is now part of any PLC command project. Then, get students aware of this reality, and even teaching students to build these interfaces is an important issue that this new interface will allow.





## Chapter 5

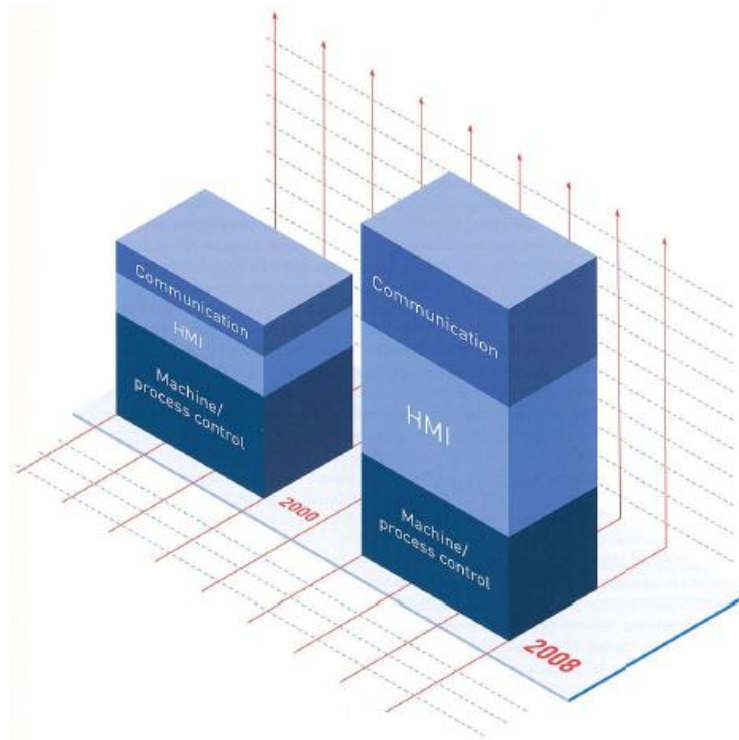
### Web pages as HMI interfaces

#### 5.1 Introduction

A Human-Machine Interface (HMI), in the industrial field, is commonly associated to a physical system (it may also have logical components) that works as an interface to a given industrial installation. These HMIs grant some level of control and access to the system, sending control actions to it and receiving information about the system state. An example of a generic HMI interface can be met in the driving process. A car is composed of a set of pedals (clutch, accelerator, brake), gearstick and other elements responsible to control the car speed, direction, lights, etc, and by a panel that feedbacks information of the car state as speed, velocity and gas levels. The overall system is constituted by a user (driver), an HMI (pedals, speedometer, lights buttons and panels) and a process (car) with several inputs and outputs.

Currently, an HMI is associated to a Graphical User Interface (GUI). Then, instead of a physical interface, there are images, text or audio representing the same components and the same connections to the system under control, which are activated through the keyboard, mouse or touchscreen. With the computerization of the industrial area these HMI became dominant - *Figure 27*. When these HMI are written as web pages and accessed by an internet or any other industrial protocol they are called “Web User Interfaces” (WUI). The interface we intend to write to the Industrial Automation Remote Laboratory belongs to this class of interfaces. The WUIs generate web pages via internet to deal with the inputs and outputs of the end point system. They can be viewed by the user via a web browser, either from a personal computer, a cell phone or a tablet, offering them greater flexibility.

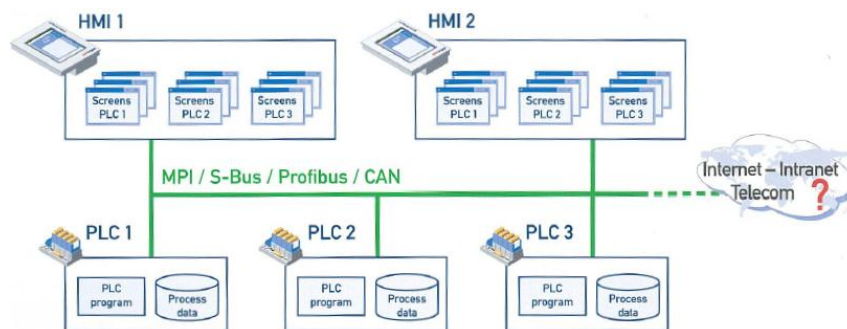
The main purpose of any kind of interface between human and machine is to guarantee a simple, easy and intuitive way to manage a system, maximizing the user's productivity and satisfaction. For that reason, the design phase takes in consideration ergonomics, linguistics or cognitive psychology, leading to a more user-centered design and, at the same time, being more efficient.



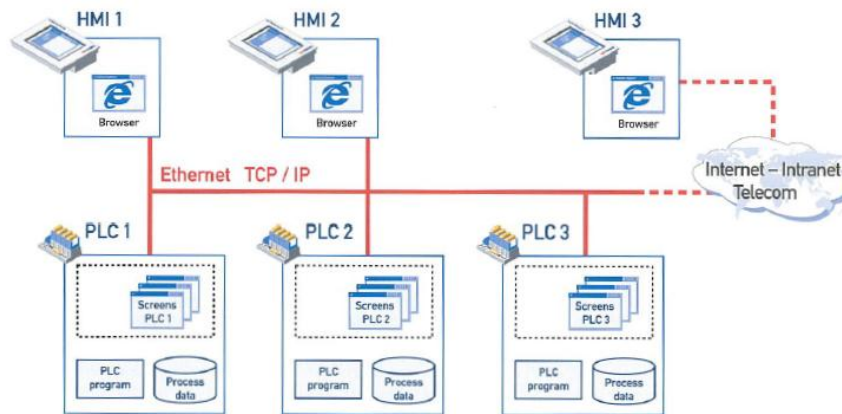
**Figure 27** – Shifting the emphasis in automation projects from straight machine/process control to HMI and communications functionality. (courtesy of Infocontrol - [Steib08])

## 5.2 Fundamentals of a web based HMI

Web based HMIs can be seen as a more flexible and simplified alternative to proprietary control panels or special PC software. Easily accessed from anywhere, they replaced rigid point-to-point connections and the resulting limited controller communication, through a client/server topology. This concept consists of storing the visual display system on the controllers (server), and having a browser to view the interface on the other end point, like the user PC (client, or also called thin client, due to its view-only purpose). This results in a simplified maintenance and updating process, since modifications only have to be applied at the controller, making all control units that access it immediately updated too.



**Figure 28** – Classical HMI concept with screens stored in the control panel. (courtesy of Infocontrol)



**Figure 29** – Web based HMI concept with screens stored in the controller's web server and display by standard browser. (courtesy of Infocontrol - [Steib08])

### 5.3 Designing a web page using the SAIA S-WEB Editor

The design of web pages to run on PLC servers follows similar principles in all PLC trademarks. For example, Panasonic has available a FP Web-Server [URLPANA] that connects to a Panasonic PLC, making it accessible via Internet. The FP Web-Server grants PLC data integration in HTML pages and Java Applets, as some email related features, and some other communication and security options. The SC143-C1 PLC [URLOVER] is denominated as a Web PLC, and allows for a large set of features related to remote controllability. It allows designing a web site that interacts with the PLC variables, sending emails with attachments, handling MySQL databases and it supports two auxiliary drives in addition to its internal hard-disk, an SD card and a USB memory stick. The SAIA PG5 software has an add-on for the designing of web pages to interact with its type of controllers – S-Web editor - with sufficient options for a correct use with the industrial automation remote laboratory. The Saia S-Web editor was the one used in this work, then, without loss of generality, this is the program that is going to be used to support the remaining sections of this chapter.

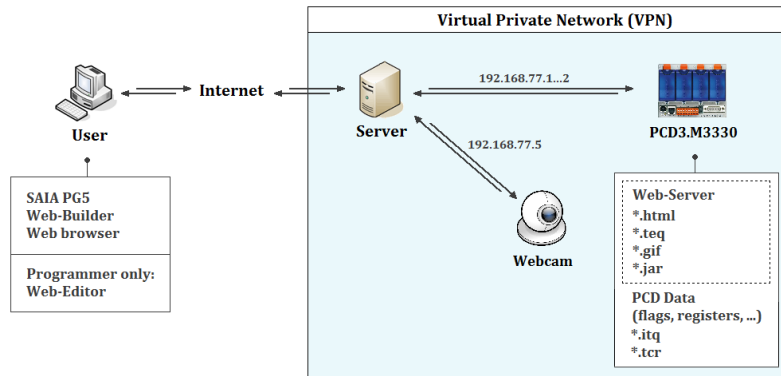
#### 5.3.1 File Management Architecture

In order to successfully enable the interaction between the user PC and the end point PLC, SAIA divides its functions through different parts of the network. When a connection to a PLC is done, a network composed of a user, a server and a lab station is made, as seen in Figure 30. Naturally the PLC should belong to a generation that supports web pages. For SAIA products this happens from the PCD3.M3330 series. The user's pc has installed a web browser, for accessing the interface, and the SAIA PG5 programming tools. These include:

- the Saia Project Manager where projects and files are created, organized and built into a downloadable program;

- the Online Debugger where the user can manipulate the PLC variables and test programs (in a command like procedure), and some other add-ons;
- the S-Web-Editor, for webpage creation.

The user accesses the VPN through the Internet, and a server, reroutes the user to any desired station.



**Figure 30** – SAIA File structure with a PCD3.M3330.

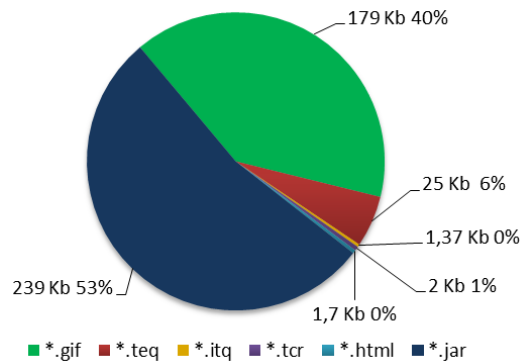
The PCD has Web-Server integrated on it, which manages information transferring related to the interface, such as html pages, java applets and images, to the user pc web browser. The PCD also stores the user programs created with Saia Project Manager, the PCD variables, such as flags, registers, inputs or outputs, and the webpage files. A concise definition of each of the file's function is presented in *Table 1*:

File Extension	Function
*.teq	Web-Server view file
*.html	Web page file
*.gif	Only Image file extension allowed to use in the web editor.
*.jar	Contains all required java classes used in the applet.
*.tcr	Contains information on PPO variable initialization.
*.itq	Contains information on Container variable initialization.

**Table 1** - Types of files included in a web editor project.

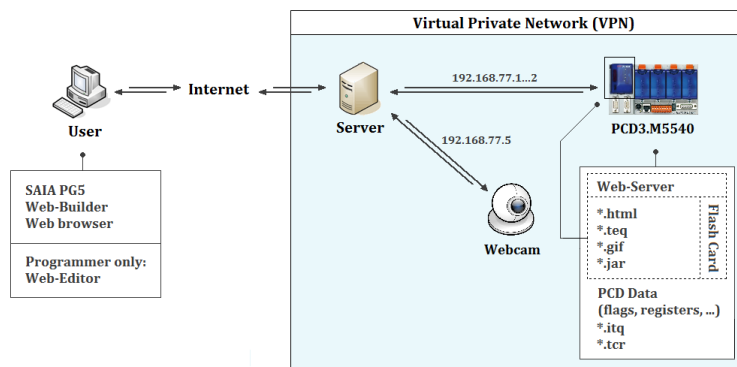
Usually, the webpage files can be included in the program created by the user, however, while attempting that action with this project a problem occurred, In fact, the PCD3M330 capacity is of only 256 Kb of RAM. The \*.jar file of the web server project is almost the same size of the PCD3M330 RAM, as seen in Figure 31, and that is a file with constant size, independently of the web page complexity. This means that, in theory, a project has to have less than 17 Kb size for images, variable lists, view files, etc. This is not feasible. A tested solution to work around this issue was to not include the \*.jar file in the PCD program download. The \*.jar file allows to access the web page via HTTP Direct, which means, one can simply connect to the VPN and write the respective address in the web browser. Without it, the user, in addition to connecting to

the VPN, has to enable SAIA.net Web-Connect on his own computer, a program dedicated to manage and connect with laboratory stations. This solution worked partially, since the web page was accessed, but it didn't recognize the laboratory variables, invalidating any kind of interaction, manipulation or reading, with the laboratory, and no further advances were achieved in this way.



**Figure 31** – Size of each group of files that compose the web server. The total size is 446,37 Kb.

A posterior solution was to replace the existing PCD3.M330 for a PCD3.M5540 - Figure 32, which has a RAM of 1024 Kb plus a SD Flash card of 512 Kb. This proved to be the best solution, since the existence of a separate storage space (the Flash card) will retain the necessary interface files independently of the programs that are downloaded by the user.



**Figure 32** - SAIA File structure with a PCD3.M5540.

### 5.3.2 Designing a Web Page

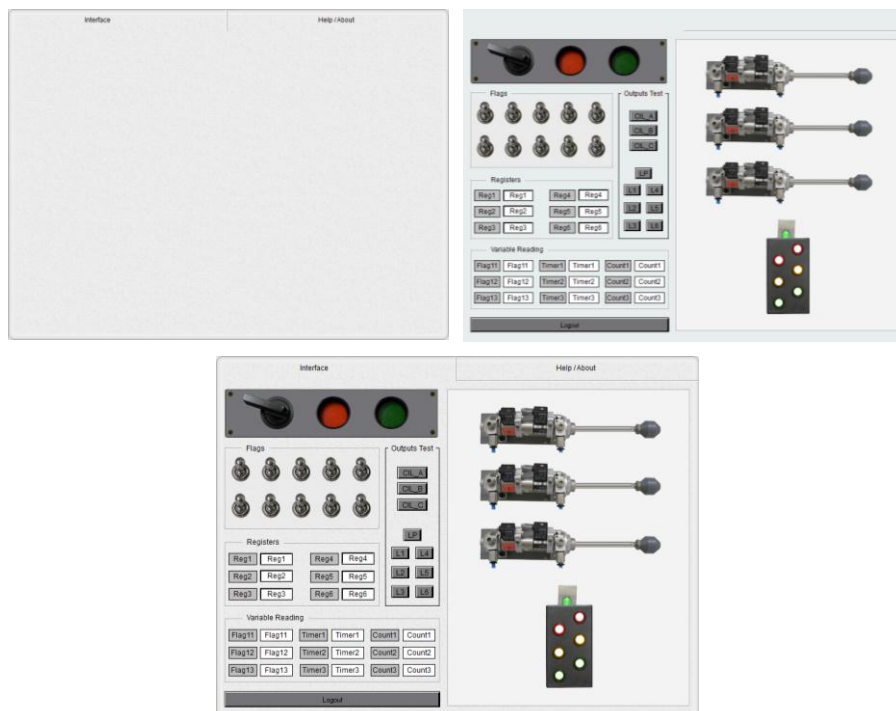
To create the web page it was used the S-Web-Editor, an add-on to the SAIA PG5 software, which allows constructing an html page without the language programming knowledge. The program can be called directly through its executable file or within Saia Project Manager after creating a new project and selecting a new file to that project.

When browsing through the internet and accessing one site, the user view new pages by clicking in a certain menu and, generally, some word is added to the end of the current address, meaning that a new page is presented. In the case of the web editor new pages can be accessed this way, by changing to another html, or by changing to other view within the same

html. One can create several html files with the editor, but we can add different views for each one html. These are called teq files, and, in S-Web-Editor, is synonym of view files. At the beginning of the web page project, one can set the dimensions and colors of the teq files to be created after. Adding a new teq file will automatically open that view file with the predefined specifications. The editor can also set a teq to be a background or foreground. In a project with a great number of views that share the same elements or background image it is usually better to create a background teq composed of all those elements. The same happens to a foreground teq, for example, if a warning message is to appear occasionally in several views.

To complete a view the web editor uses graphic objects called painters. Every image, text, button, variable caller or other functions of the web page are created by inserting a painter on the teq file.

To refer to a PCD variable the web editor has process points, PPOs. In fact, during the design of the web page, there is no distinction between flags, registers or other types of variables. The painters read or manipulate a certain variable of type PPO with a certain name, and only after finalizing the web page, in the Saia Project Manager, are the names associated to a type of PCD variable and to an address. There also exists in the web editor another type of variable, a local variable called Container. Containers are unrelated to the PCD and serve to exchange values between different painters or different views. It is possible, for that reason to make a limited simulator to the PCD interface without having to go online, by creating an exact replica of a teq file, but exchanging the PPOs with Containers. This simulator is limited because it only works for direct manipulation of variables. It is not possible to read programs in fupla, IL or other languages in this way.



**Figure 33** – View/Teq file with background teq. On top, to the left, is a teq file with the pretended background image and the name of multiple tabs; on top, to the right, is a teq file with no background; on the bottom is the resulting teq file when setting the first teq as a background teq to the second one.

### 5.3.2.1 Including a flag manipulator

Through the process of creating the interface it was necessary to implement a set of buttons, on/off and pressure/release, to manipulate flags. This was done by including, for a single flag manipulator, two types of graphic objects: a *Button* painter, and an *Image* painter. The *Button* painter would be “invisible” to the user and perform the PPO change of values from zero to one, and vice-versa, when clicking on it, and the *Image* painter would show the state of the button, in this case representing two states. The *Button* painter lets the programmer choose the action that triggers a variable value change, so different settings have to be defined for on/off and press/release buttons. This painter should be colorless and hidden from the user (there are settings that allow this) and be positioned above the image painter. The *Image* painter would be defined to show two images on two conditions. One of the images shows up when variable X is equal to 0, the other image shows up when variable is equal to 1. Grouping both painters will complete a flag manipulator.



**Figure 34** – Two states of a flag manipulator.

### 5.3.2.2 Including a register manipulator

In order to create a register manipulator an *Edit Box* painter was used. This painter lets the user write and read a variable, in other words, one is always getting feedback of a variable value, can send a value in the same box, and continue to receive information on the variable afterwards. It's a 2-in-1 painter that occupies much less space than two separate boxes.



**Figure 35** – Example of a register manipulator.

### 5.3.2.3 Including a variable reader-only and string box

The interface was designed with a section dedicated to variable reading, namely flags, timers and counters, not counting with the registers manipulators/readers. To implement a strictly reading box one has to use a *Static Text* painter. This painter is set in the same way as an *Edit Box* painter, and is quite similar to it. However, this painter prevents the user from changing directly the values of target variables.

This painter can also be used to include a string on a view, for informative purpose, or to identify a section or object, by only changing the type of source in the painter to string.



**Figure 36** – Group of read-only boxes and the respective names in static painters.

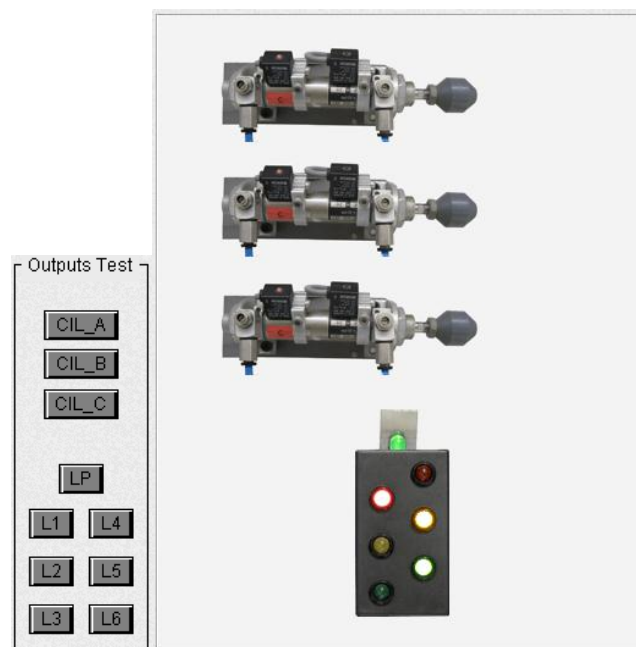


#### 5.3.2.4 Including an output interaction section

A section responsible for the outputs of the remote laboratory was created in this new interface, being the most important change implemented. This section was divided into testing area and viewing area.

The testing area existed before as a single output, selectable address and value, box, and was now improved to a set of buttons, each one corresponding to a cylinder or light, where changes are made with only one click. These buttons were implemented with *Button* painters, in a process similar to the on/off flag manipulators. However, these painters weren't set to be hidden, are colored and have the respective variable name above them.

The viewing area replicates the laboratory apparatus, with three pneumatic cylinders and seven lights. To simulate a single cylinder it was necessary to represent three states: no length (initial sensor on), middle length (no sensor on), and full length (end sensor on). For that reason, were used two *Image* painters. One *Image* painter was defined to represent two states, no length and full length, by associating the respective images when PPO of the initial sensor equals 1 and when PPO of the end sensor equals 1. The second *Image* painter, for the middle length state, was defined to be always present, but in a position below the previous painter. Both painters were grouped and a cylinder representation was correctly implemented. To simulate the seven lights another process was followed, with the help of eight *Image* painters. One painter was set as a background all-lights-off image, always present, and the rest seven painters were set as images of each of light-on states above the background painter, making themselves visible only when the respective variable is equal to 1.



**Figure 37** – The testing area (left) and view area (right) of the output interaction section.



## **Chapter 6**

### **Designing the New Interface**

#### **6.1 Introduction**

The proposed web page will be used for the industrial automation discipline to support the hands-on laboratories on PLC programming. Then, this interface should be a user friendly environment, easy to use and intuitive. It also should be the most simple possible, without compromising its pretended functionality.

Another requirement to the web page is to be more realistic and, therefore, more immersive. It should include images of physical elements of the existing laboratory, recognizable by the students, including a representation of its outputs, cylinders and lights. It should also include real images of different types of switches.

Since a continuous reading of variables is intended, it would be interesting to have a section with information on Timers and Counters values, where students would have feedback on how are both types of variables evolving in time.

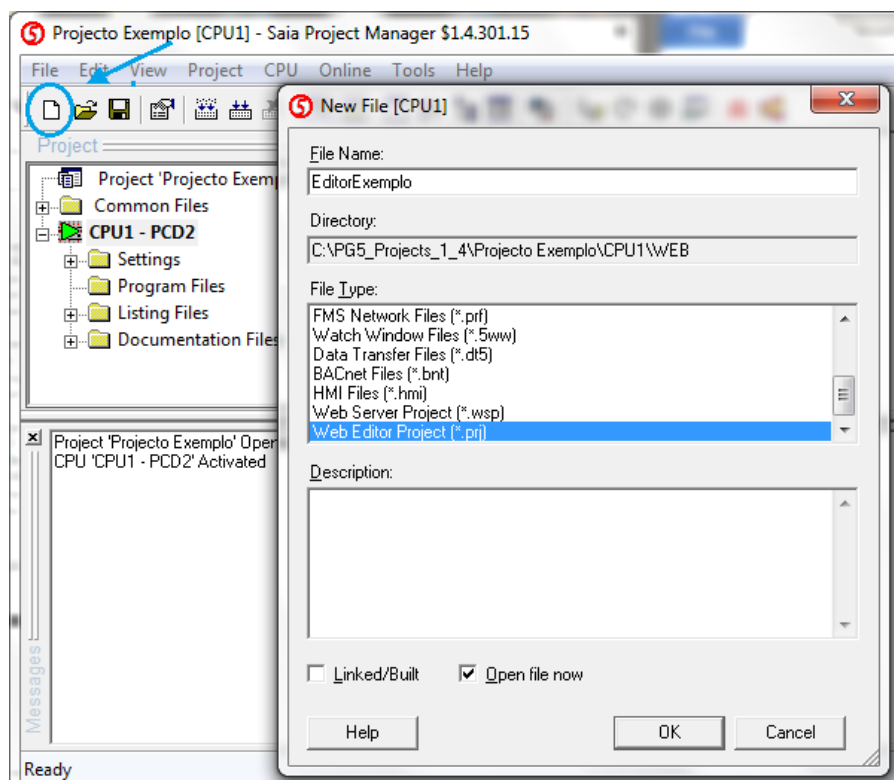
The old interface connected itself to the PLC, performed the download of a program to the PLC, and had some other functions related to the connection to the PLC, but that is not needed, since the Saia Project Manager can perform those tasks. The interface should be the most simple possible, so those options are not included in the objectives to the construction of the web page.

Regarding the accessibility, the goal is to search if there is a way to prevent multiple accesses at the same time and to set a couple of levels of restriction. This last option, if possible, should be accomplished by setting different passwords, each of them granting control to more sections of the interface. For example, a level 1 password would allow view-only interaction where the user only sees the output values, a level 2 could grant access to output control and viewing, and a level 3 would grant access to registers, flags and output control and viewing.

## 6.2 Creating the web page

The SAIA add-on, S-Web-Editor, provides the tools to create dynamic java based web pages, without deep knowledge of HTML or Java programming. The following description focuses on the steps taken in order to create the web page, and how to correctly include it on the PCD.

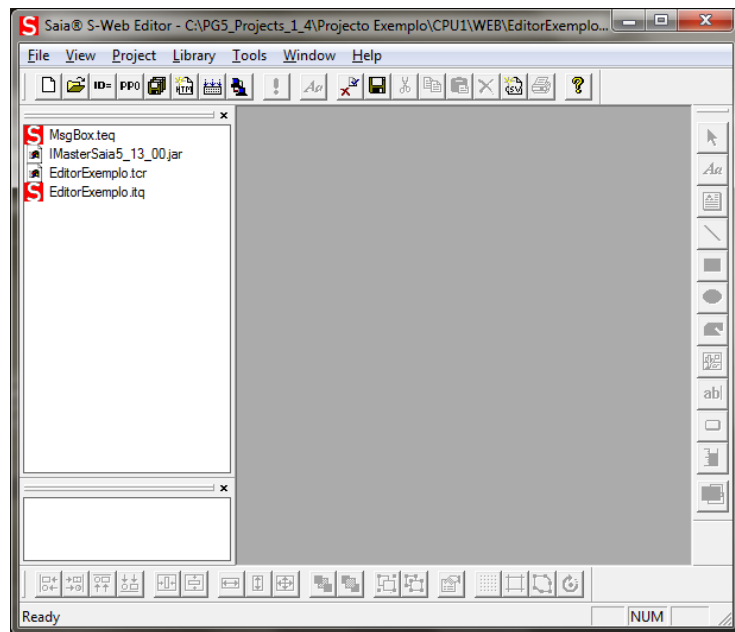
S-Web-Editor can be accessed directly by clicking the *Web Editor* executable, or by clicking *New file* in the Saia Project Manager window, and choosing a Web Editor Project (\*.prj), as seen in *Figure 38*.



**Figure 38** – Adding a new web editor project file through the Saia Project Manager.

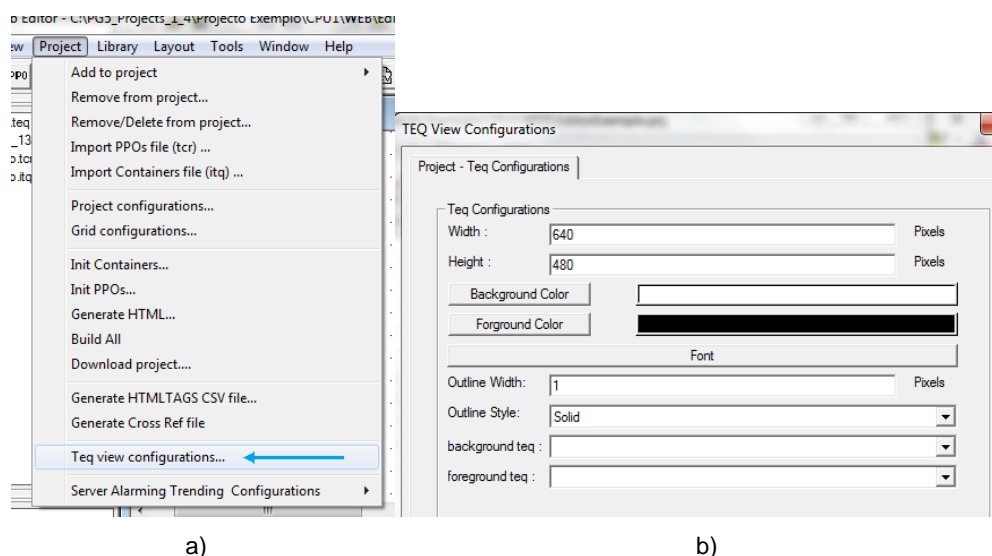
An S-Web Editor window will automatically open. On the left area there is a list of existing project files. It is important to understand with what types of files we are working.

From the list of project files we can see that there already exists a teq file. It should be deleted since it will not be used in the project. On the left side of the main window there is the painter toolbar. Painters are the graphical objects created by the web editor, such as buttons, images, text, graphics, lines, etc. A global view of the editor is shown in *Figure 39*.



**Figure 39** – S-Web Editor main window.

The next step is to go to *Project* menu, from the top menu bar, and select *Project configurations...*. This will open a window with the predefined characteristics of each and all of the teq files that will be created further on, meaning that this should always be one of the first steps to be taken. The available configurations include the width and height of the view files, their background and foreground colors, font and line styles. The interface project was built with teqs of 800x600 pixels. There is also an option to choose a background or foreground teq. In a project with multiple teqs that share almost the same layout, is better to define a background teq that will be used by any number of other teqs. Same logic goes to the foreground teq, in the case of wanting a message to pop up in various teqs. Figure 40 shows the “Teq view configurations...” window.



**Figure 40** – a) How to go to Teq view configuration”;

b) Teq view configurations window.

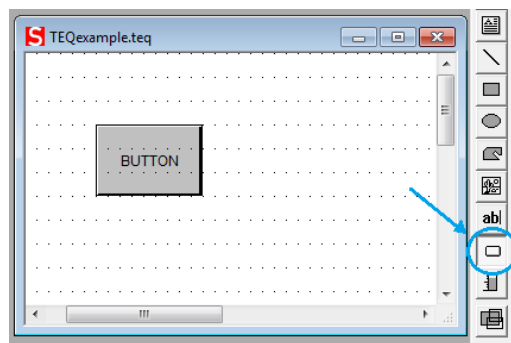
After defining the project, we create a new teq file by clicking the *New file* button of the web editor, and selecting the proper extension. A blank teq file will appear, with the size and color previously defined.

## 6.2.1 Flag Manipulation Section

In order to create a flag manipulator, the following steps were taken.

First, we want to create a button, so we click the *Button* icon located in the Painter Toolbar, then left click the teq area and, keeping the mouse down, drag it to another point, to form the button area. This area can be changed later, if needed. A squared button will be positioned in the teq. Double clicking it opens the painter properties. The purpose of the button is to change a flag value from 0 to 1, or vice-versa, and this can be accomplished in two different ways, either by

- i) pressure (when pressed down the flag takes the value 1, when released the flag takes the value 0)
- ii) toggle (on/off value changing).



**Figure 41** – Placing a button painter on a teq view.

To implement i), we have to click the *Actions Set Variables* tab in the painter properties. There, we have two sections, the *Set Var on Mouse DOWN* section and the *Set Var on Mouse UP* section. Selecting the box corresponding to *Set a Variable* on both of them will activate a new area where a type of variable can be chosen between PPO or Container. In this case it will be a PPO variable. The variable should be the same on both sections, the only difference being the value taken in that action. The flag should take the value 1 when Mouse is DOWN and value 0 when Mouse is UP. *Figure 42* shows the settings for this implementation.

Hide and Disable Painter		Border Advanced	Text Positions Advanced	Function Keys
General	Repaints	Actions Set Variables	Actions Toggle Increment Variables	Actions Jump

Set Var on Mouse DOWN  
☒ Set a Variable    Type: PPO  
Name: ButtonPressure    Select  
Value: 1  
☐ On Condition

Set Var on Mouse UP  
☒ Set a Variable    Type: PPO  
Name: ButtonPressure    Select  
Value: 0  
☐ On Condition

**Figure 42** – Settings for implementing a pressure button.

To implement *ii*), we have to click the *Actions Toggle Increment Variables* in the painter properties. There, we have the *Toggle Button* section. Selecting the box corresponding to *Toggle* will activate a new area where a type of variable can be chosen (in this case it will be a PPO variable). The values are automatically defined. *Figure 43* shows the settings for this implementation.

Hide and Disable Painter		Border Advanced	Text Positions Advanced	Function Keys
General	Repaints	Actions Set Variables	Actions Toggle Increment Variables	Actions Jump

Toggle Button  
☒ Toggle    Type: PPO  
Name: ButtonToggle    Select  
Toggle String 0: 0  
Toggle String 1: 1

**Figure 43** – Settings for implementing a toggle button.

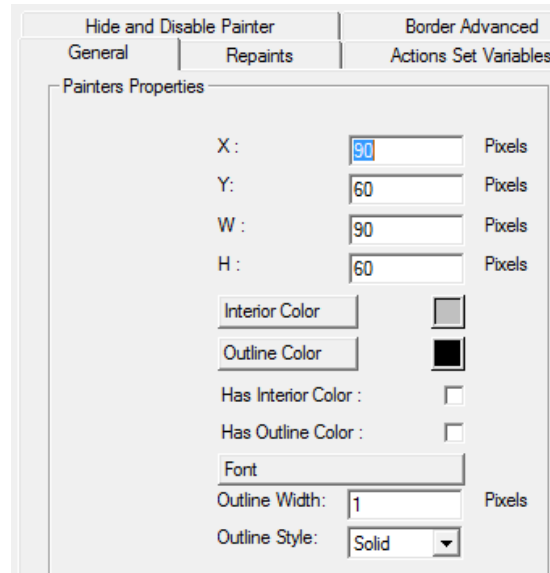
We now have the button functionality defined. However, further changes have to be done to better implement the flag manipulator. The goal is to associate an image to each state of the flag to represent a real button. For that goal, we want an “invisible” button, hidden from the user, only visible through the corresponding image. Clicking the *Repaints* tab in the painter properties, there is an *Edit a Source* section with a selected STRING type source. This should be disabled, since no text should appear on the button.

Hide and Disable Painter	
General	Repaints

Edit a Source  
☐ Edit a Source

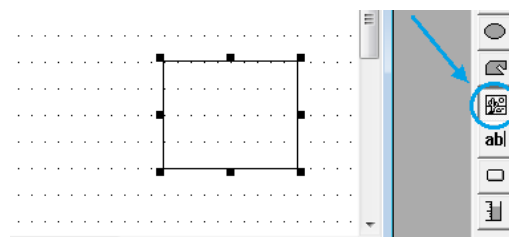
**Figure 44** – Disabling button text.

Then, clicking the *General* tab, the interior and outline colors should also be disabled. This will turn the button colorless and invisible. This step could also be taken after associating the image if there is the risk of losing track of the button on the teq. Clicking *OK* will confirm all the changes made and exits the painter properties.



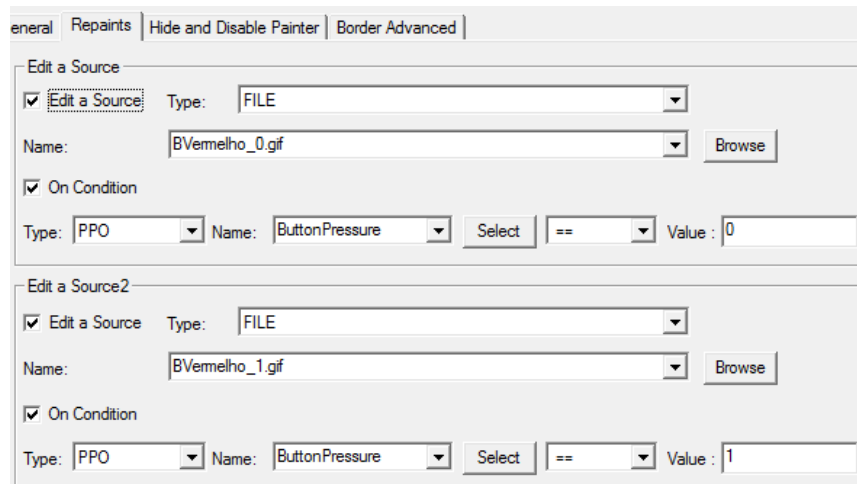
**Figure 45** – Removing the button interior and outline colors.

Now, an image should be placed in the teq. This is done by clicking the *Image* painter in the painter toolbar, and clicking and dragging the same way the button painter was included. The dimension of the box is not important, since it will adapt to the image file selected. Also important to note is that the image should already have the proper size before included, because web editor do not allow image resizing.



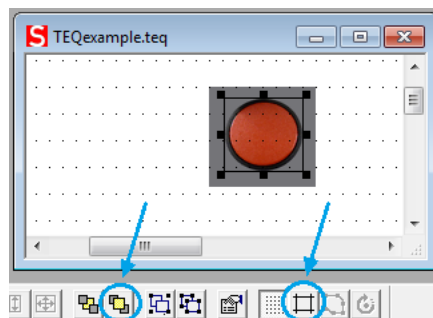
**Figure 46** – Placing an image painter on a teq view.

Double clicking the painter opens its properties. In the *Repaints* tab, there are two *Edit a Source* sections. Both *Edit a Source* boxes must be checked, which will activate a new area where an image file should be selected for each section. The web editor only accepts gif files, due to being lighter than other image files. Each image corresponds to a button state, either pressed/toggled or released/untoggled. The respective flag variable name is selected, with different values for each image. Then, we click *OK* to save the painter properties.



**Figure 47** – Settings for implementing the button images.

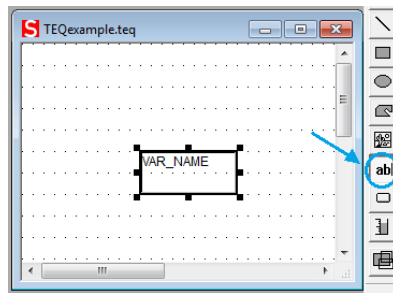
Finally, with both painters defined, they must be repositioned one on top of the other. For better maneuverability, the button should be positioned on top of the image and grid snap should be disabled. To guarantee this, we select the button painter and then click the *Front* icon on the bottom of the window, and we also click the *Grid Snap* icon. Now, it should be repositioned on top of the image and his size readjusted. After properly arrange the two painters, we can now group them, helping to maintain an organized project. This is done by selecting both painters and clicking *Group* in the bottom of the main window.



**Figure 48** – Arranging the button in Front of the image and grouping both painters.

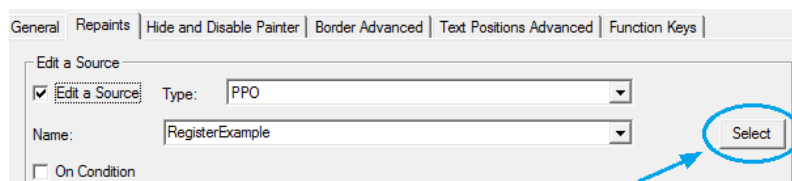
## 6.2.2 Register Manipulation Section

To implement a register manipulator we have to use the *Edit Box* icon on the painter toolbar, which allows to read and write a PPO value. The box is set in the same way as the boxes before.



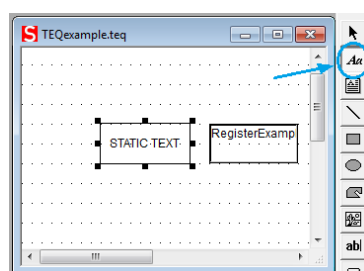
**Figure 49** – Placing an Edit Box painter on a teq view.

Double clicking the painter opens its properties. Selecting *Repaints* tab, the type of the variable can be set to PPO, and a name of a variable can be written. One important note is that if there already is a list of variables that were defined as global symbols in the Saia Project Manager, one of them can be selected by clicking the *Select* button. If no variables have been defined, then any name can be set. In that case, and also in the previous *Flag Manipulation Section* chapter, there are no flags or registers yet defined, only PPOs and their names. These names have to be posteriorly associated to PCD variables in the Saia Project Manager as global symbols.



**Figure 50** – Setting a PPO for the Edit Box.

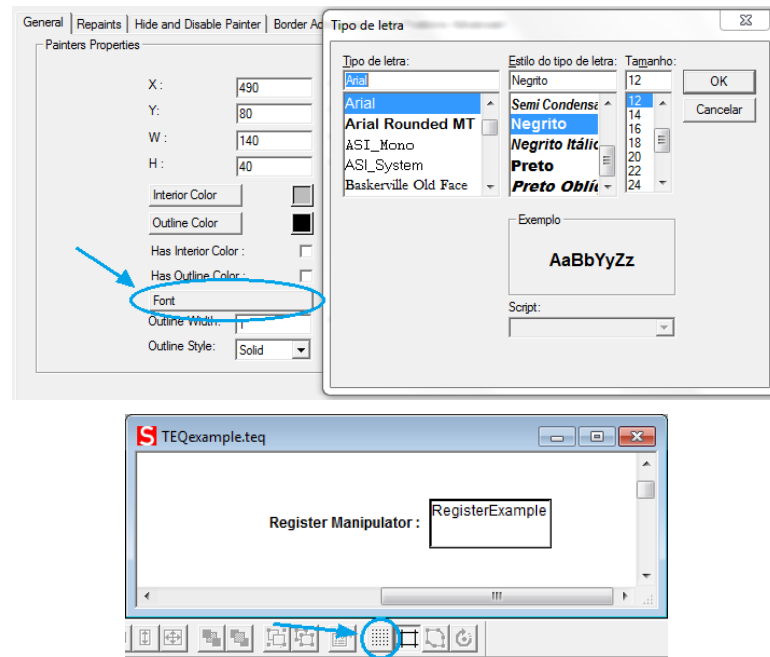
It is wise, after defining the Edit Box, to add a text or small tag referring to which PPO is this box dealing with. This is done by adding a *Static Text* box, located on the painter toolbar to the right, as seen in *Figure 51*.



**Figure 51** – Placing a Static Text box painter on a teq.

Double clicking the box, and selecting the *Repaints* tab, there is already defined a String with the name STATIC TEXT. This field can be edited to any string we want to place in the box, and the font can also be edited. So, for example, we write “*Register Manipulator:*” in the name field. Then we click the *General* tab and click *Font*, making a new window pop up. In that window the letter type, style and size can be chosen. Setting the letter type to Arial, the style to Negrito and the size to 12, we will get something like shown in *Figure 52*. The grid was disabled for better viewing of how it would appear in a webpage, by untoggle the *Grid visible* button.





**Figure 52** – Editing a Static Text font and disabling the grid.

### 6.2.3 Variable Reading Section

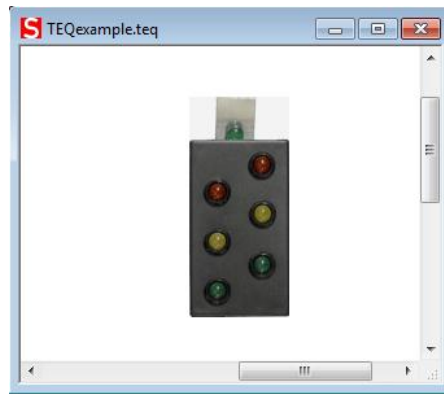
To implement a read-only box we use the *Static Text* painter used in the example before. But, instead of choosing a string type in the *Edit a source* of the *Repaints* tab, a PPO type must be set, and then the name of the PPO must be selected or written. This process is identical to an *Edit Box* PPO manipulator.

### 6.2.4 Output Section

The output section is composed of a view area, and a test area.

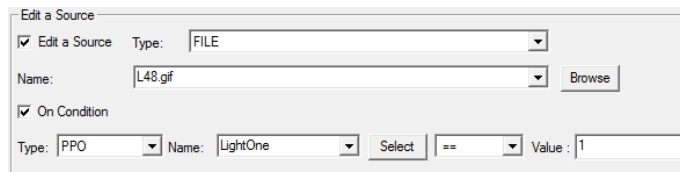
The view area is made by a group of images simulating the laboratory elements behavior. If a representation of a two state output is wanted, like a light blinking, then a single *Image* painter is sufficient, with the light turned off and turned on. The process was described in the *Flag Manipulation Section* chapter. If a group of more than one light is to be implemented, that share or compose a background image is preferable and easier to have an image of all lights turned off, and then just create an *Image* painter to represent each light turned on. An example would be the following:

i) Create an *Image* painter with the all-lights-off image. In the *Repaints* tab of the painter properties, choose the image file without any condition added on. Click *Ok*.

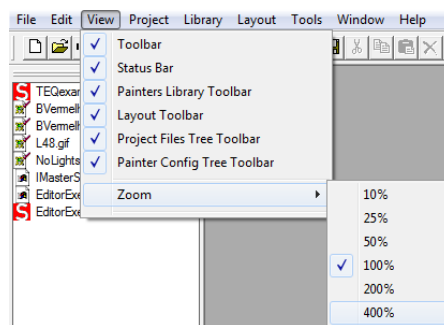


**Figure 53** – Placing the background all-lights-off image.

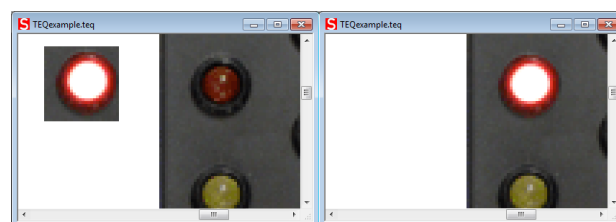
ii) Create an *Image* painter for a single light in ON state. In the *Repaints* tab of the painter properties, choose the image file and check the *On Condition* box. Choose the respective PPO, select “==” and value 1. This settings means that this image will only appear when the selected PPO (in this case it will be an output) takes the value 1. The image must then be repositioned above the respective OFF state area of the all-lights-off image. This last step is easier if the working area is zoomed. To zoom in, we go to the top menu bar and select *View*, and then *Zoom*, which will show the available percentages, being 400% the more zoomed in working area and 10% the least zoomed in.



a)



b)



c)

**Figure 54** – a) Settings for the light ON image painter;  
b) Choosing zoom quantity;  
c) Placing the light ON image painter.

- iii) Repeat step ii) until all ON states are represented.
- iv) *Group* all painters created.

### 6.3 Including the interface on a SAIA project

With the web editor interface fully designed, it can now be added to a SAIA project and tested in real conditions. There are three different situations that the user can encounter while trying to complement a project with an interface:

- The user developed his own interface within a project, and the PLC has no Flash card;
- The user developed his own interface within a project, and the PLC has a Flash card;
- The user wants to integrate an existing interface in his project, and the PLC has a Flash card;

Independently of the method used, the webpage can be accessed by writing in a web browser, the IP of a respective station, followed by the name of the *html* file.

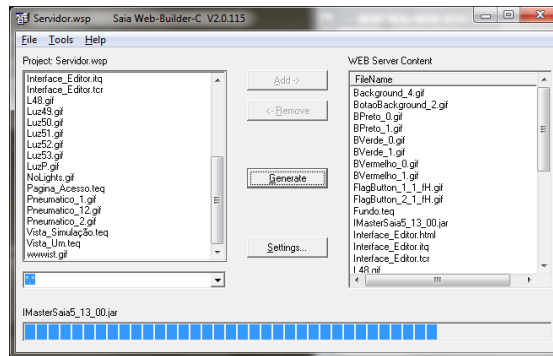
Webpage Address: **192.168.77.<station\_number>/<html file name>**

Examples: 192.168.77.2/Interface.html

192.168.77.4/InitialView.html

#### 6.3.1 Including a new interface on a SAIA project, for a PLC with no Flash card

In Saia Project Manager window, the user has to add a Web Server file (\*.wsp) to the project, by clicking the *New File* button. A small window, like the one seen in *Figure 55*, will pop up. In the left side are shown all the files created with S-Web Editor, and they should be added to the right side of the window, constituting the list which the builder will include in the generation of the web server. In this case it would be any \*.gif, \*.html, \*.jar, \*.teq, \*.itq and \*.tcr file.

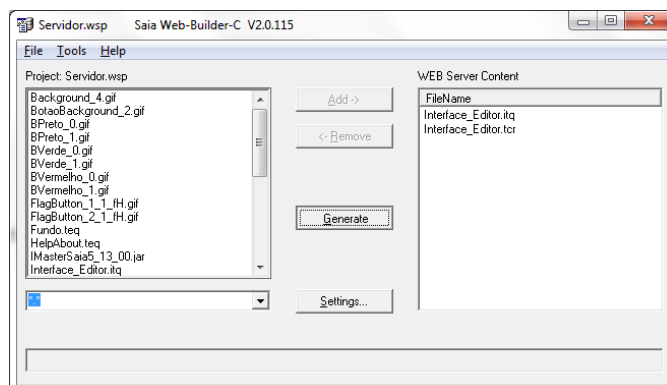


**Figure 55** – Generating a Web Server file for the PCD3M3330 case.

In this case, all the files needed are completely represented by the Web Server file, and would now be downloaded to the PCD as part of a normal project built in Saia Project Manager, together with any other files developed in, for example, fupla or instruction list. This solution is not adequate to the remote laboratory, since several users would have to systematically download heavy programs to the PLC with every reuse of the laboratory.

### 6.3.2 Including a new interface on a SAIA project, for a PLC with a Flash card

Having an additional storing place will sharply lighten the project built in Saia Project Manager. During the process of generating the web server file, only the \*.itq and \*.tcr files, responsible for PPOs and Containers variable names, are added to the content - *Figure 56*.

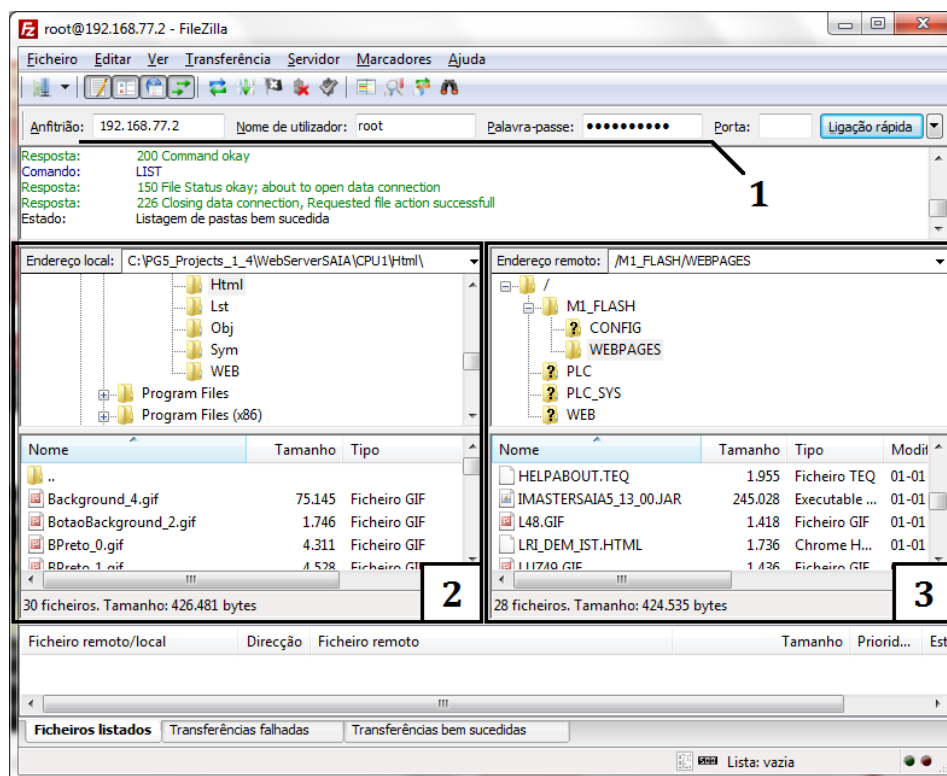


**Figure 56** – Generating a Web Server file for the PCD3M5540 case.

The rest of the files have to be stored in the Flash Card by using Saia Project Manager command *Flash Explorer* in the *Online* menu, or by using a proper software that follows FTP (File Transfer Protocol), like Filezilla [URLFZ], an open-source software. This last alternative has proven to be more efficient and fast, but using Saia Project Manager is also possible. Filezilla allows to access the controller folders, including the Flash card folder, where the \*.jar, \*.html, \*.gif and \*.teq files are going to be stored, and allows to transfer those files to that folder. Further modifications to these files have to be updated in this folder, except for the \*.jar file that

stays unchanged for any web page project. This turns out to be an advantage since the largest file doesn't need to be removed or added again to the folder.

Working with Filezilla is similar to a standard explorer, but requests an initial connection to be made with the destination server. This is done by completing the blank fields on top of the main window - *Figure 57*. There, it is written the IP relative to the destination server, one username and one password – **1** – and then establishing the connection. The main window will show two different locations, the local folder where the files we want to transfer are located – **2** – and the remote folder where we want to transfer those files to – **3**. The web page related files have to be uploaded to the */M1\_FLASH/WEBPAGES* directory in order to be available for the user.



*Figure 57 – Filezilla main window.*

### 6.3.3 Including an existing interface on a SAIA project, for a PLC with a Flash card

If the user already has a developed project in Saia Project Manager, or wants to start developing one, and wants to add an interface made by somebody else, he has to perform some additional steps. This will mainly happen in the case of a PLC with a Flash card, because it is the only feasible way. If there wasn't a Flash card, the user would have to be provided with all of the web page files, and he would have to include all of them in his project and download them every time he wanted to test his program. It would be the same problem as if he had made

the web page himself. So, for a PLC with a Flash card, the needed files will already be located in the correct Flash card folder, stored by someone else by following the process described before, with Filezilla. This solution separates the user tasks from the programmer tasks, since the user does not get access to the core files that define the web page.

In order to enable the interface to work for a certain project the following steps have to be taken:

- Add a new folder with the name “*html*” to the project folder.
- Add the \*.itq and \*.tcr files, which should be provided by the programmer to the user, to the *html* folder.
- In Saia Project Manager, add a new Web Server file to the project, generated with the \*.itq and \*.tcr files.
- Import list of Global Symbols to Saia Project Manager (these should be used by any other project files, to be recognized by the interface).

## 6.4 Results

The previous interface demanded that students wrote their variable addresses, adding a step to variable manipulation that is not desirable. Enabling the reading of variables by a specific click was also a step that slowed the interaction. Then a desirable objective for the new interface was reducing the steps needed for variable manipulation and reading, for a limited, but sufficient, number of variables. The following table, *Table 2*, shows the improvement obtained with the interface. It is visible the enhancement of the interface performance, which no more requests to write addresses every time that is initialized, and reducing actions number of steps to one and, in reading cases, to zero, meaning that the user has instant feedback of target variable values.

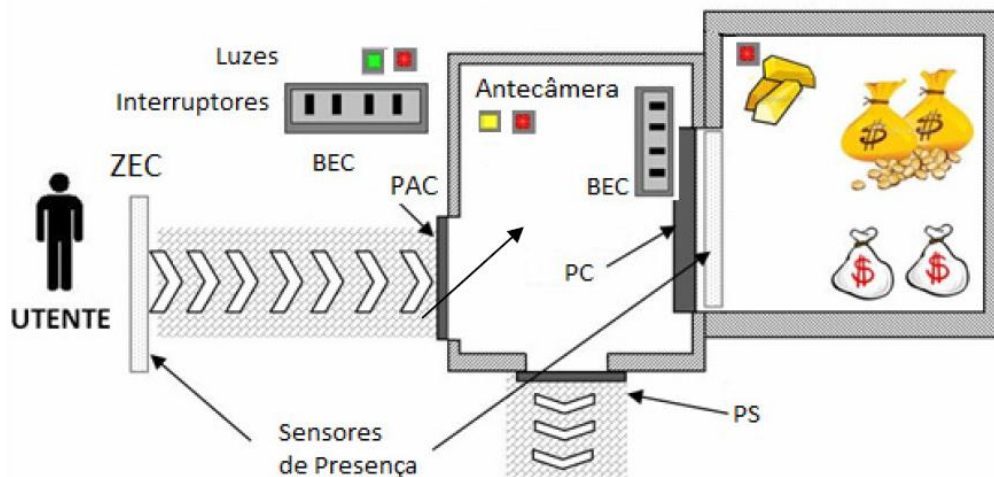
	Old Interface					New Interface				
	E	A	V	S/R	T	E	A	V	S/R	T
<b>Output Sending Steps</b>	0	1	1	1	3	0	0	0	1	1
<b>Register Sending Steps</b>	1	1	1	1	4	0	0	1	0	1
<b>Flag Sending Steps</b>	1	1	1	1	4	0	0	0	1	1
<b>Register Reading Steps</b>	1	1	0	1	3	0	0	0	0	0
<b>Flag Reading Steps</b>	1	1	0	1	3	0	0	0	0	0
<b>Input Reading Steps</b>	1	1	0	1	3	0	0	0	0	0
<b>Timer Reading Steps</b>	-	-	-	-	-	0	0	0	0	0
<b>Counter Reading Steps</b>	-	-	-	-	-	0	0	0	0	0

**Table 2** – Comparison between the number of steps required to perform each action in the old and new interfaces. *E*: enable variable; *A*: write address; *V*: write value; *S/R*: send or read variable; *T*: total number of steps.

The new interface also features a counter and timer reading section, inexistent before, that informs on the respective values of up to three counters and up to three timers. The register sending and reading sections are now a single section that performs both actions at the same time.

To prove the flexibility and adaptation capacity of a web interface of this kind, it was developed a different layout with S-Web Editor, involving the necessary PLC variables, to test the first laboratory work of the Industrial Automation course of Mechanical Engineering students in 2011/2012. An interface, following the already described procedure on implementing flag manipulators, output viewers, and so on, was designed for the respective work, which consists on implementing an automatic control system for accessing a safe.

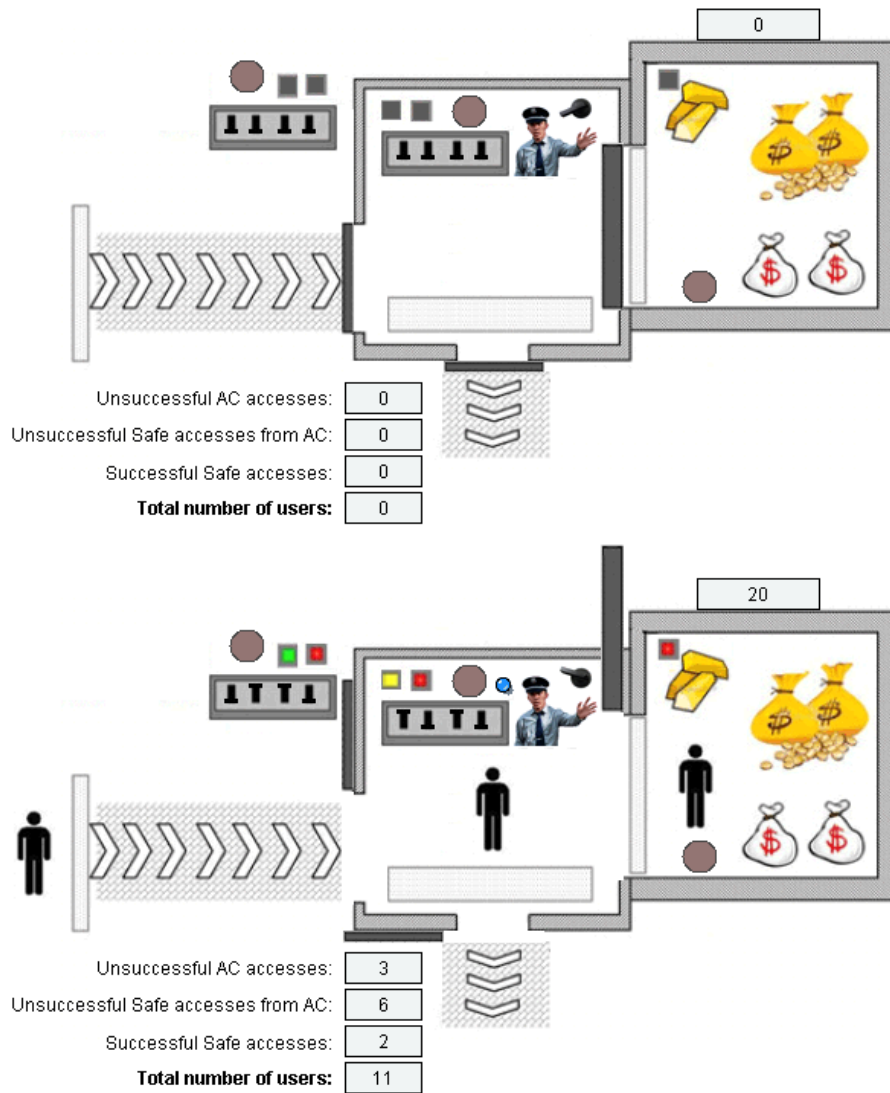
The *pdf* file given to students with the process description contains an image of the system to be programmed – *Figure 58* – which was used to create several *gif* images to be included in the interface.



**Figure 58** – Industrial Automation course First Laboratory Work system.

In resume, it consists in a safe which can be accessed through a secured chamber. There exists a first area, where the subject is detected, and he has to insert a certain code based on four switches, and confirm by pressing a button BEC. Two lights exist in this area to represent specific states (e.g. subject detected, wrong code). The door PAC will open in certain conditions and the subject will access the antechamber. In this room there exists four more switches, a confirmation button BEC, lights to represent specific states, a security guard, an alarm and a motion sensor (not all elements are represented in *Figure 58*, but are referred in the process description). Finally, in the safe there exists a motion sensor, a light, a timer and an exit button BSC. The complete description of the process and system is available in Appendix C.

Students simulated these elements by using the available remote laboratory components. The three doors would be represented by the three pneumatic cylinders, the switches, sensors, alarm and other buttons would be flags changing from value 0 to 1, and students wouldn't have feedback on timer values with the old interface. So the desired environment had to include these elements, directly related with this specific work. The following image shows the general view of the designed interface – *Figure 59*.



**Figure 59** – General view of the industrial automation laboratory work interface. On top, initial state of the system. On the bottom, system with fully activated states.

The sensors are activated by clicking the correspondent rectangle, making a symbol of a person appear in the room. The doors have three states: closed, semi-opened and opened. These are associated to the PLC pneumatic cylinders inputs. The user can also set the codes by clicking directly in the switches of the entrance and of the antechamber. Round buttons correspond to pressure/release buttons (identical to the green and red buttons of the automation laboratory), and the black button next to the security guard corresponds to an on/off button (identical to the start/stop button of the automation laboratory). The SOUND alarm is represented by a small blinking blue light, next to the image of the security guard. The lights can be of any desired color, and, even representing the remote laboratory LEDs outputs, they are not limited to those colors or number of colors. For example, the process requested three red lights, as seen in the previous image, which would be represented by two LEDs without this interface, since only two red colored LEDs exist in the lab. Above the safe room was included a timer viewer, that returns the timer count down values. Finally, it was implemented a group of counters that return the number of accesses to the facility. Working with this simple



implemented interface proved to be better representative of the system process and specifications, incorporating the described elements. At the same time demonstrates the potential of the work done in improving situational adaptation, and how simpler and faster is to emulate any given system.

. The following global symbols list, presented in *Table 3*, was used to define the variables of the project.

SYMBOLNAME	TYPE	ADDRESS	SYMBOLNAME	TYPE	ADDRESS
BEC1	F	100	Sensor2	F	113
BEC2	F	101	Sensor3	F	114
BSC1	F	102	Alarm	F	115
BSC2	F	103	Luz1	Output	48
Interr1	F	104	Luz2	Output	49
Interr2	F	105	Luz3	Output	50
Interr3	F	106	Luz4	Output	51
Interr4	F	107	Luz5	Output	52
Interr5	F	108	Porta1	Output	32
Interr6	F	109	Porta2	Output	33
Interr7	F	110	Porta3	Output	34
Interr8	F	111	Timer	Timer	11
Sensor1	F	112			

**Table 3** – List of Global Symbols used for the interface testing process.



## **Chapter 7**

### **Conclusion**

#### **7.1 Work Final Notes**

A research on existing remote laboratories architectures, softwares and accessibilities was the base to the creation of a centralized, user friendly and simplified interface. It also showed the necessity of having a realistic interface accessible by web browser, as done by most of the national and international remote laboratories. An evaluation of the old interface showed relevant flaws related to some unpractical functionality while manipulating PLC variables, unnecessary PLC initializing and clearing functions, and decentralized execution, since every user had to have the application file stored in his computer.

The following are improvements accomplished with the web page implementation:

- The new interface is centralized, stored in the PLC flash card, existing one single version to all users. Updates done to the interface can now be performed in only one place, simplifying future modifications.
- The visualization of realistic components shortens the difference between remote experimentation and hands-on experimentation, offering a more immersive and appellative working environment to students.
- Variable manipulation and reading is simplified, requiring fewer steps to perform. Also, counters and timers reading feature was added to this interface.
- Implementation of an output reading section makes the use of video streaming software optional.

#### **7.2 Future Work**

Now that improvements were done towards a more flexibilized, simplified and realistic interface environment, it is essential to regulate accessibility to the remote laboratory. First, it would be useful to implement multiple levels of access to the interface, separating student focused functionality from guest user functionality (e.g. view only access). This would allow

expanding the range of potential users, without damaging the time needed for students to complete their automation programs. Related to this issue, it would also be interesting to implement a solid booking system, a reservation method to help users to manage their available time, and granting a way for the administrator (e.g. teacher) to get a notion on laboratory occupation percentage per day.

The graphical component of the interface can now be modified to a desired case of study. Variables are associated to images, and these can be changed to represent automation components other than pneumatic cylinders. Situational interfaces can be designed and redesigned, and can be easily placed in the PLC web server. Even students could be introduced to the S-Web editor and create their own interfaces, enlarging their learning sphere.

It would be practical to place video streaming in an html page so that it could be called from the designed web page. This would centralize all functions related to the PLC control/viewing to SAIA and a web browser, instead of having to execute ipView for video feedback. Other add-ons can also be added, if programmed in the form of an html page, and associated to the interface, for example, informative pages, simple simulators, or the booking system.

## References

- [AmmSla06] Ammari, A. C., Slama, J. B. H., "The Development of a Remote Laboratory for Internet-Based Engineering Education", 2006.
- [BerBra11] Bernal, M. P., Bravo, E. C., "Remote Experimentation of Complex Systems and Computational Intelligence Algorithms on Graphics Processing Units" in 1<sup>st</sup> Experiment@ International Conference – Remote & Virtual Labs – exp.at'11, November 2011.
- [BMGT11] Bratina, B., Muskinja, N., Golob, M., Tovornik, B., "Fault Detection and Isolation Remote Laboratory Concept and Experiments", in 1<sup>st</sup> Experiment@ International Conference – Remote & Virtual Labs - exp.at'11, November 2011.
- [CNM11] Crabeel, N., Neves, B., Malheiro, B., "RemoteLabs Platform", in 1<sup>st</sup> Experiment@ International Conference – Remote & Virtual Labs - exp.at'11, November 2011.
- [CVG11] Cardoso, A., Vieira, M., Gil, P., "A Remote and Virtual Lab with Experiments for Secondary Education, Engineering and Lifelong Learning Courses", International Journal of Online Engineering (iJOE), vol.8, issue S2, pp. 49-54, 2012.
- [DFRPM08] Domínguez, M., Fuertes, J. J., Reguera, P., Prada, M. A., Morán, A., "Inter-University Network of Remote Laboratories", 17th IFAC World Congress (IFAC'08), pp. 13623-13628, Seoul, Korea, July 6-11, 2008.
- [GRR11] Gabriel, J., Restivo, T., Reis, J., "Remote Control of a Model Car", in 1<sup>st</sup> Experiment@ International Conference – Remote & Virtual Labs - exp.at'11, November 2011.
- [HDFPFP11] *Henriques, R. B., Duarte, A. S., Fernandes, H., Pereira, T., Fortunato, J., Pereira, J.*, "Generic protocol for hardware control @ e-lab", in 1<sup>st</sup> Experiment@ International Conference – Remote & Virtual Labs - exp.at'11, November 2011.
- [Lobo11] Lobo, J., "A Remote Reconfigurable Logic Laboratory for Basic Digital Design", in 1<sup>st</sup> Experiment@ International Conference – Remote & Virtual Labs – exp.at'11, November 2011.

- [MaNick06] Ma, J., Nickerson, J. V., "Hands-On, Simulated, and Remote Laboratories: A Comparative Literature Review", ACM Computing Surveys, (38:3) Article 7, pp 1-24, 2006.
- [MaMe06] Machotka, J., Medic, Z., "The Remote Laboratory *NetLab* for Teaching Engineering Courses", Global J. of Engng. Educ., vol.10, no.2, pp 205-212, Australia, 2006.
- [MRTBM11] Muškinja, N., Rižnar, M., Tovornik, B., Bolf, N., Mohler, I., "Remote Lab for Process Control Education Using OPC and Web based Technology", in 1<sup>st</sup> Experiment@ International Conference – Remote & Virtual Labs – exp.at'11, November 2011.
- [MSFSA11] Marcelino, R., Silva, J., Fidalgo, A., Schaeffer, L., Alves, J., "Virtual 3D Worlds and Remote Experimentation - A Methodology Proposal", in 1<sup>st</sup> Experiment@ International Conference – Remote & Virtual Labs – exp.at'11, November 2011.
- [MTTHM11] Mwikirize, C., Tickodri-Togboa, S. S., Harward, J., Tumusiime, A. A., Musaizi, P. I., "A Sequential Logic iLab Utilizing NI ELVIS II+ and the Interactive iLab Architecture", in 1<sup>st</sup> Experiment@ International Conference – Remote & Virtual Labs – exp.at'11, November 2011.
- [PCBM11] Palma, L. B., Coito, F. V., Borracha, A. G., Martins, J. F., "A Platform to Support Remote Automation and Control Laboratories" in 1<sup>st</sup> Experiment@ International Conference – Remote & Virtual Labs – exp.at'11, November 2011.
- [RMLSC09] Restivo, M. T., Mendes, J., Lopes, A. M., Silva, C. M., Chouzal, F., "A Remote Laboratory in Engineering Measurement", IEEE Trans. On Industr. Electr., vol.56, no.12, pp. 4836-4842, December 2009.
- [SGA11] Sanguino, T., Garcia, D., Ancos, E., "Remote Power Control: a Low-Cost Solution for Enabling Telematic Practices in a Network Lab", in 1<sup>st</sup> Experiment@ International Conference – Remote & Virtual Labs - exp.at'11, November 2011.
- [Steib08] Steib, Peter M., "Web technology in automation", Saia-Burgess Controls, 2008.
- [TMSTPC11] Tauste, M., Martin, S., Sancristobal, E., Tawfik, M., Peire, J., Castro, M., "Implementation of a remote laboratory for practices in FPGAs Programmable Logic Devices", in 1<sup>st</sup> Experiment@ International Conference – Remote & Virtual Labs - exp.at'11, November 2011.

- [TkSch11] Tkáč, L., Schauer, F., “Remote experiments for basic course Electricity and Magnetism implemented into education by Integrated e-learning”, in 1<sup>st</sup> Experiment@ International Conference – Remote & Virtual Labs - exp.at’11, November 2011.
- [TSGG11] Titov, I., Smirnova, O., Glotov, A., Golovin, A., “Remote Laser Laboratory First Demonstration”, in 1<sup>st</sup> Experiment@ International Conference – Remote & Virtual Labs - exp.at’11, November 2011.
- [GZHHGA11] Garcia-Zubía, J., Hernández-Jayo, U., Gustavsson, I., Alves, G., “Academic effectiveness of VISIR remote lab in analog electronics”, 1<sup>st</sup> Experiment@ International Conference – Remote & Virtual Labs - exp.at’11, November 2011.
- [URLACT] [act.dii.unisi.it/experiments.php](http://act.dii.unisi.it/experiments.php) (accessed at the 17th October of 2011)
- [URLAPACHE] [http://httpd.apache.org/ABOUT\\_APACHE.html](http://httpd.apache.org/ABOUT_APACHE.html) (accessed at the 27<sup>th</sup> January of 2012)
- [URLCLP] [pt.wikipedia.org/wiki/Controlador\\_lógico\\_programável](http://pt.wikipedia.org/wiki/Controlador_lógico_programável) (accessed at the 15th October of 2011)
- [URLCOR] [www.corel.com/corel/allProducts.jsp](http://www.corel.com/corel/allProducts.jsp) (accessed at the 20<sup>th</sup> April of 2012)
- [URLCUNI] <http://kdt16.karlov.mff.cuni.cz/en/mereni.html> (accessed at the 17th October of 2011)
- [URLELABIST] <http://elab.ist.utl.pt> (accessed at the 17th October of 2012)
- [URLEMFEUP] <http://experimenta.fe.up.pt/estacaometeorologica/index.php> (accessed at the 17th October of 2011)
- [URLFCT] <http://193.136.127.204> (accessed at the 12th April of 2012)
- [URLFEUP] <http://foton.fe.up.pt/moodle/> (accessed at the 12th April of 2012)
- [URLFZ] <http://filezilla-project.org/> (accessed at the 6th March of 2011)

[URLILABS] <https://wikis.mit.edu/confluence/display/ILAB2/Home> (accessed at the 10th April of 2011)

[URLISR] <http://lsd.deec.uc.pt> (accessed at the 10th April of 2011)

[URLJAVA] [http://www.java.com/pt\\_BR/download/faq/java\\_webstart.xml](http://www.java.com/pt_BR/download/faq/java_webstart.xml) (accessed at the 19<sup>th</sup> March of 2012)

[URLNETLAB] <http://netlab.unisa.edu.au/index.xhtml> (accessed at the 20th October of 2011)

[URLOPCI] [http://en.wikipedia.org/wiki/OLE\\_for\\_process\\_control](http://en.wikipedia.org/wiki/OLE_for_process_control) (accessed at the 2<sup>nd</sup> February of 2012)

[URLOPCI] <http://www.opcdatahub.com/WhatIsOPC.html> (accessed at the 2<sup>nd</sup> February of 2012)

[URLOPCI] [http://www.opcfoundation.org/Default.aspx/01\\_about/01\\_what\\_is.asp](http://www.opcfoundation.org/Default.aspx/01_about/01_what_is.asp) (accessed at the 2<sup>nd</sup> February of 2012)

[URLOVER] [http://www.overdigit.com/data/Products/SC143-C1\\_brochure\\_EN.pdf](http://www.overdigit.com/data/Products/SC143-C1_brochure_EN.pdf) (accessed at the 20<sup>th</sup> April of 2012)

[URLPANA] <http://www.powelectrics.co.uk/content/pdf/fpwebserver.pdf> (accessed at the 20<sup>th</sup> April of 2012)

[URLRS232I] <http://controls.ame.nd.edu/microcontroller/main/node24.html> (accessed at the 19<sup>th</sup> January of 2012) 19-01-2012

[URLRS232II] <http://www.arcelect.com/rs232.htm> (accessed at the 27<sup>th</sup> January of 2012)

[URLSCADA] <http://en.wikipedia.org/wiki/SCADA> (accessed at the 2<sup>nd</sup> February of 2012)

[URLSCADAII] [http://www.icpdas.com/products/PAC/i-7188\\_7186/whatisscada.htm](http://www.icpdas.com/products/PAC/i-7188_7186/whatisscada.htm) (accessed at the 2<sup>nd</sup> February of 2012)

[URLTRUNI] <http://remotelab1.truni.sk> (accessed at the 17th October of 2011)

[URLUTS] <http://remotelabs.eng.uts.edu.au/labinfo> (accessed at the 17th October of 2011)



# **APPENDIX**



## Appendix A – Editing realistic laboratory components

To include realistic elements that the students would recognize from the physical laboratory, specific photographs were taken and then edited with Corel PHOTO-PAINT X5, from CorelDRAW Graphics Suite X5 software package [URLCOR]. Not only a detailed cleaning editing is accomplished with this software, but is also possible to create good quality and low size *gif* images. It was also possible to use Windows Paint, however the produced gif images are of lower quality.

To create the up/down flag buttons - *Figure A 1*, two photos were taken for each state. Then, the background was removed with the help of a special tool from Corel PHOTO-PAINT, and replaced by the same background of the interface.



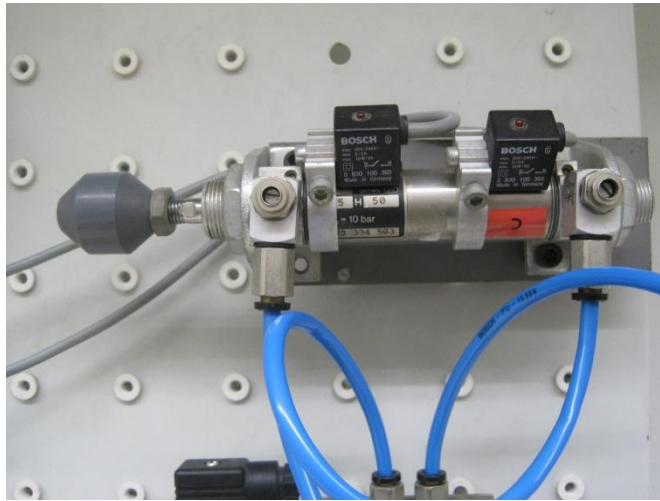
**Figure A 1** – Flag button (up/down) image editing process for both states.

To create the pressure/release flag buttons *Figure A 2*, a similar process was executed. Two photos were taken for each state of the red button, with the help of a small object that could be easily erased with the editing software. Then the background and the object were removed and replaced with the proper background. With the red button images completed, the color was replaced to green, completing the required images of the green button also.



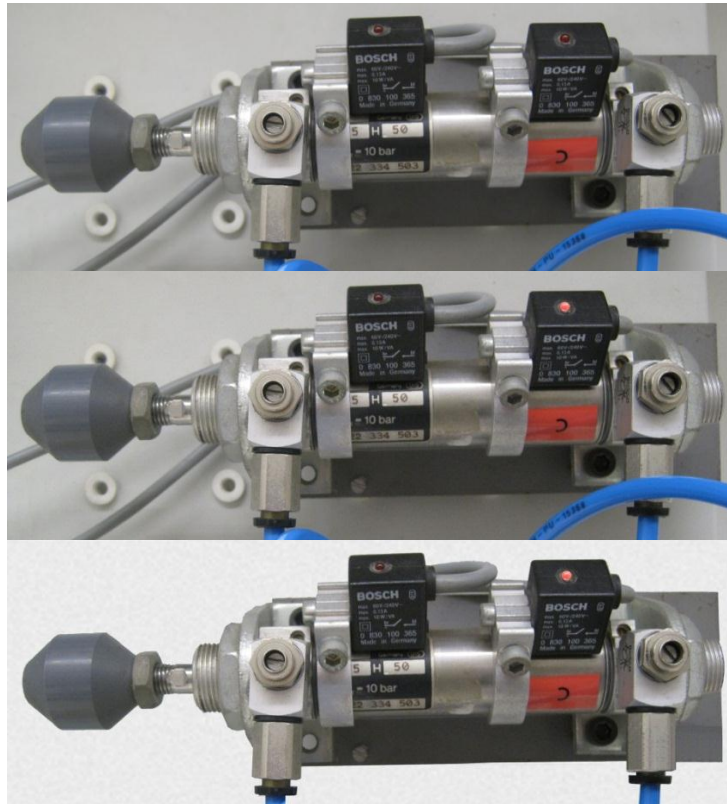
**Figure A 2** – Flag buttons (pressure/release) image editing process for both states.

To create the moving cylinders one photograph was taken with the pneumatic cylinder withdrawn - *Figure A 3*. The sensors' LEDs should be visible for posterior editing.



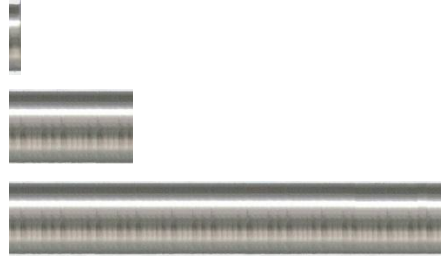
**Figure A 3** – Initial image of a pneumatic cylinder for posterior editing.

To represent each state of the cylinder (initial sensor activated and cylinder withdrawn, no sensors activated and cylinder at middle position, final sensor activated and cylinder at full length), the metal “neck” was manually created after the creation of the initial state image - *Figure A 4*. In this initial image the sensor light was created by increasing that zone brightness.



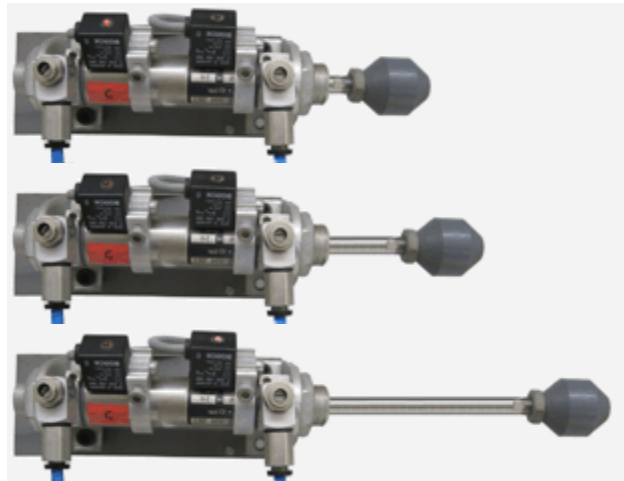
**Figure A 4** – Pneumatic cylinder withdrawn state, image editing process.

With the initial state image completed, the neck was edited. This was done by progressively “copy+pasting” larger areas of the visible neck - *Figure A 5*. Two photos could have been taken of each end point state, but that would bring bigger problems while attempting to center both images equally. In this case, adding the neck granted a correctly placed body for the three images.



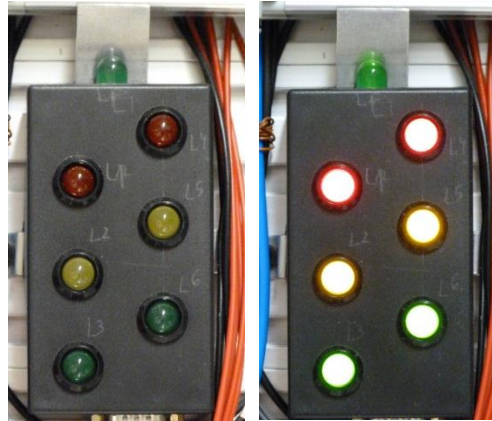
**Figure A 5** – Different states of the cylinder metal “neck” enlargement.

The images were reverted - *Figure A 6*, to better represent the real positions of the remote laboratory cylinders, and the three states were then completed:



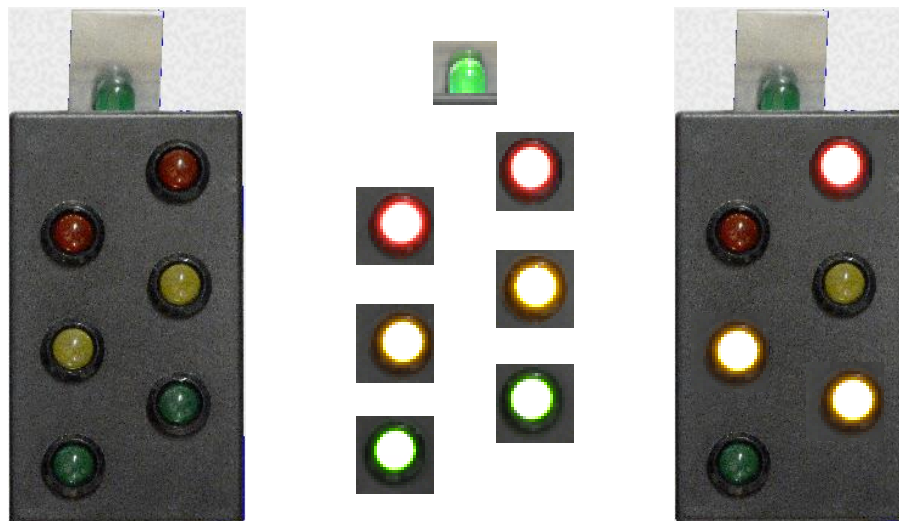
**Figure A 6** – Final images of the pneumatic cylinder three states.

To create the laboratory group of seven LEDs, two photographs were taken, one with all lights on, and another with all lights off, as seen in *Figure A 7*.



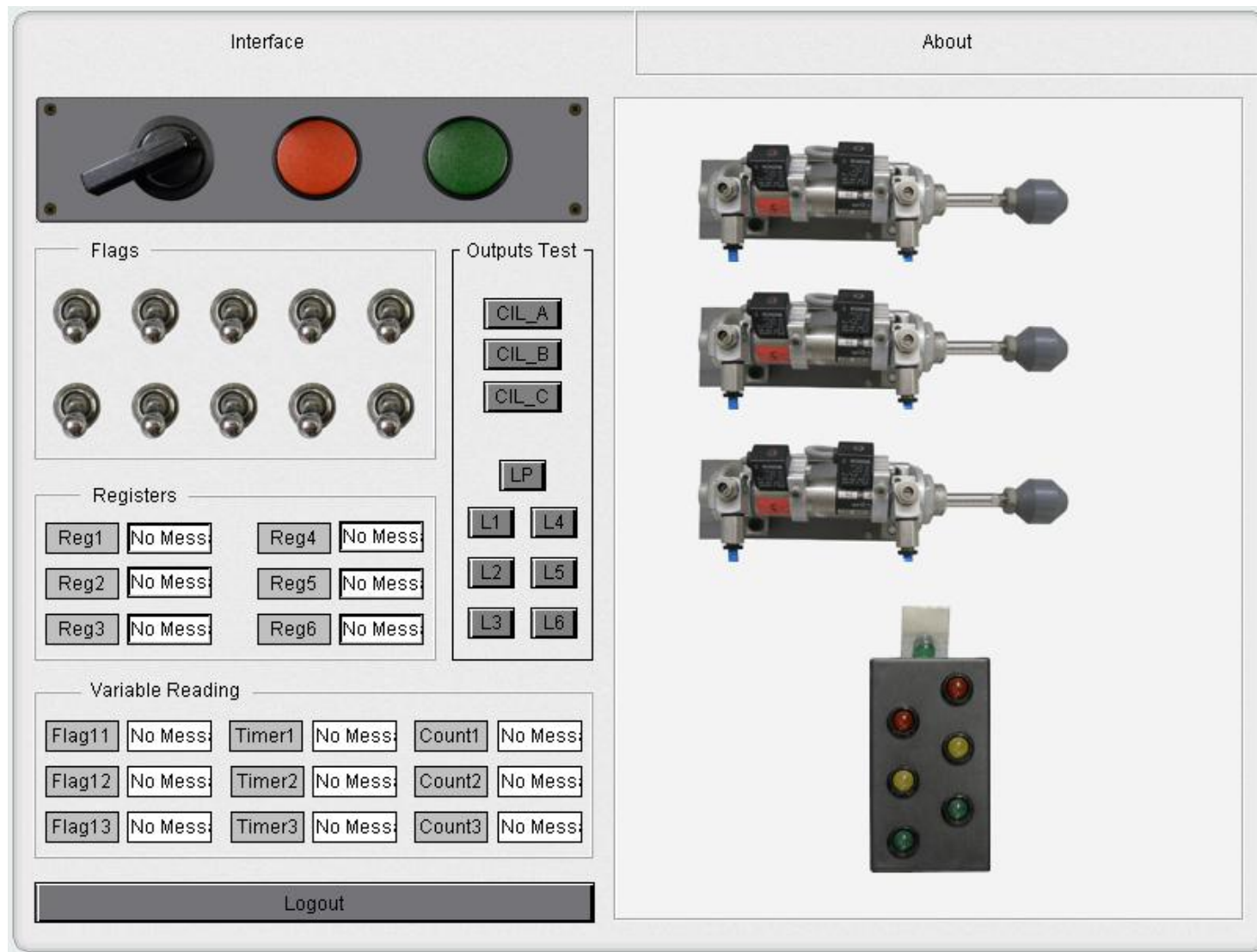
**Figure A 7** – Two initial photographs of the laboratory group of seven LEDs; to the left, all lights off state; to the right, all lights on state.

In this case the objective was to have a background image that represented the “all lights off” state, so the first image was cleaned and saved as a *gif* file, ready to be placed in the interface. The next step was to make separated *gif* images of each of the lights, which would appear on top of the “all lights off” state image, when needed - *Figure A 8*.



**Figure A 8** – LEDs images editing. To the left, “all lights off” state image; to the middle, the several lights on images; to the right, a composition of the background image with some lights on images.

The final layout of the new interface - *Figure A 9* – consisted of having two clearly divided sections, one for output viewing, and one to read or manipulate variables. This arrangement allows the user to neglect the output viewing section, if he is not interested in the feedback or if he gets it from elsewhere (e.g. ipView). The several sub-sections are also well identified, and the overall work environment is not too heavy. It may seem that the colors were chosen randomly, but it is not the case. Having a soft/light work environment is extremely beneficial for users, so light colors and some minor textures were tested and selected as background for the interface.



**Figure A 9 – General View of the implemented new interface.**

## Appendix B – Interface Global Symbols List

To include the designed interface in a SAIA PG5 project, this list of global symbols - *Table A 1*, should be imported and used in any other program of the project. It will make the connection between the names used in the interface design and the symbols used in the project. The inputs and outputs addresses should remain unmodified, but any other address can be changed if needed.

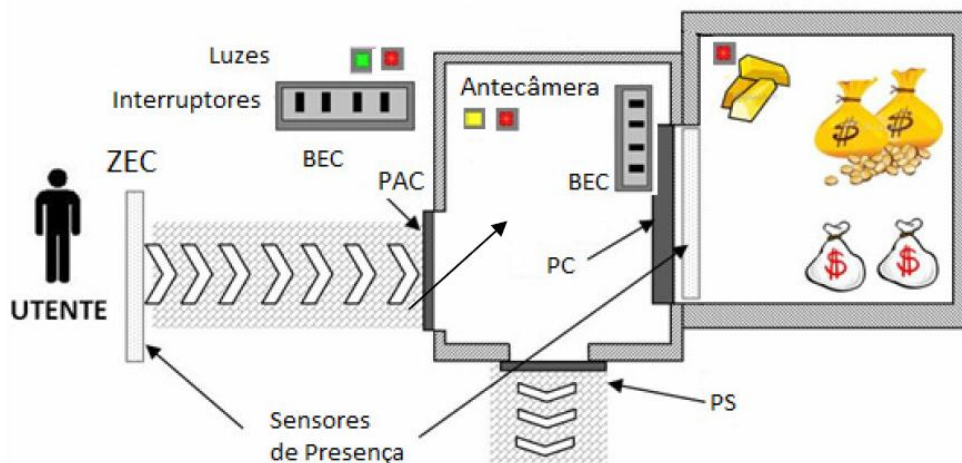
SYMBOLNAME	TYPE	ADDRESS	SYMBOLNAME	TYPE	ADDRESS
a0	Input	0	BOnOff4	F	104
a1	Input	1	BOnOff5	F	105
b0	Input	2	BOnOff6	F	106
b1	Input	3	BOnOff7	F	107
c0	Input	4	BOnOff8	F	108
c1	Input	5	BOnOff9	F	109
Timer1	Timer	11	BOnOff10	F	110
Timer2	Timer	12	Flag11	F	111
Timer3	Timer	13	Flag12	F	112
Cil_A	Output	32	Flag13	F	113
Cil_B	Output	33	BStartStop	F	114
Cil_C	Output	34	BRed	F	115
LP	Output	54	BGreen	F	116
L1	Output	48	Reg1	R	1001
L2	Output	49	Reg2	R	1002
L3	Output	50	Reg3	R	1003
L4	Output	51	Reg4	R	1004
L5	Output	52	Reg5	R	1005
L6	Output	53	Reg6	R	1006
BOnOff1	F	101	Count1	Counter	1401
BOnOff2	F	102	Count2	Counter	1402
BOnOff3	F	103	Count3	Counter	1403

**Table A 1** - List of Global Symbols used for the new interface.



## Appendix C – Industrial Automation course laboratory work, process description <sup>1</sup>

Pretende-se controlar a acessibilidade a um cofre instalado numa sala de segurança, precedida de uma antecâmara (AC), onde tem acesso livremente um segurança, e para onde inicialmente se entra para se ganhar acesso à sala do cofre através de uma porta (PC). É também a AC que tem a porta de saída (PS). Um esquema simplificado do processo de acessibilidade está ilustrado na figura abaixo.



**Figure A 10 - Industrial Automation course First Laboratory Work system.**

O cenário correspondente a este processo é o seguinte:

As instalações estão acessíveis 24h por dia e, não havendo ninguém na zona de entrada do código (ZEC) estará uma luz verde (LV) acesa.

A chegada de um utente à ZEC é detectado pela actuação de um sensor de presença que permanece atuado enquanto este estiver nessa zona, apagando-se então a LV. O utente prepara-se para introduzir o código que lhe é solicitado através de um pequeno sinal sonoro .

A introdução do código corresponde em fixar a posição de quatro interruptores, após o qual deve acionar o botão de envio de código (BEC) tipo pulso. O código da fechadura corresponde a uma dada configuração de quatro interruptores, o que por comodidade se expressa por uma expressão lógica mais à frente apresentada para os diferentes turnos.

É admitida a entrada na AC desde que (1) o utente introduza o código certo ou (2) se engane **apenas** numa das posições dos interruptores. Caso se engane em mais de uma posição terá de abandonar o local para início do processo por outro utente, sinalizado através de uma luz encarnada.

No caso (1) e desde que seja detetada a presença do utente em AC junto à porta do cofre (PC) esta abre. O utente entra e a porta fecha automaticamente quando este é detectado dentro do cofre. O tempo previsto para a permanência dentro do cofre é de 20s, tempo

<sup>1</sup> I keep the original Portuguese version.

considerado suficiente. Para sair o utente deve atuar num botão (BSC). Podem ocorrer duas situações:

(a) Ao fim desse tempo o utente não atua o botão para sair. Então uma luz vermelha na antecâmara deve piscar e se ao fim de 10s ainda não tiver saído, a luz vermelha passa a permanentemente ligada. O utente deixa de poder sair pelos seus meios e será o segurança que abrirá a porta ligando e desligando um interruptor. O utente sairá com o segurança pela PS, que abre após o fecho da porta do cofre e a deteção do utente e segurança em AC. A PS fecha após a deteção da passagem do utente e o processo de entrada de novo utente pode recomeçar acendendo a LV da ZEC.

(b) O utente sai pelos seus meios. Quando a porta do cofre se fecha, com a consequente deteção do utente em AC, a porta de saída da antecâmara abre-se o utente sai, a porta fecha após a passagem do utente como atrás, e o processo de entrada de novo utente pode recomeçar acendendo a LV da ZEC.

No caso (2), fechada a porta de entrada da AC é acesa uma luz amarela e o utente é novamente solicitado, através de dois pequenos sinais sonoros, a introduzir o código num conjunto idêntico de interruptores junto à porta do cofre. Se o código estiver correto a porta do cofre abre seguindo-se a sequência atrás definida. A luz amarela apaga logo a seguir à introdução do código.

No caso do código estar de novo incorrecto, acende uma luz encarnada e toca um alarme junto do segurança que acompanhará o utente à porta de saída da AC, abrindo esta manualmente (seja por um botão start/stop). Depois desta fechada, quando o utente passa, o sistema fica pronto para novo utente.