Faculdade de Engenharia da Universidade do Porto

FEUP

Desenvolvimento de uma Solução para Laboratórios Remotos Baseada em Serviços Web

Hugo Guimarães de Faria

Relatório de Projecto realizado no âmbito do
Mestrado Integrado em Engenharia Electrotécnica e de Computadores
Major Automação

Orientador: Prof. Dr. Joaquim Gabriel Magalhães Mendes
Co-orientador: Inv. Principal Maria Teresa Braga Valente de Almeida Restivo

Julho de 2008

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES 2007/2008

A Dissertação intitulada

"Desenvolvimento de uma Solução para Laboratórios Remotos Baseada em Serviços Web"

foi aprovada em provas realizadas em 17/Julho/2008

o júri

Presidente Professor Doutor Paulo José Lopes Machado Portugal
Professor Auxiliar da Faculdade de Engenharia da Universidade do Porto

Professor Doutor Manuel Gradim de Oliveira Gericota
Equiparado a Professor Adjunto do Instituto Superior de Engenharia do Porto

Professor Doutor Joaquim Gabriel Magalhães Mendes
Professor Auxiliar da Faculdade de Engenharia da Universidade do Porto

Professora Doutora Maria Teresa Braga Valente de Almeida Restivo
Investigadora Principal da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projecto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extractos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são correctamente citados.

Autor - Hugo Guimarães de Faria

Faculdade de Engenharia da Universidade do Porto

Este projecto foi desenvolvido no
Laboratório de Instrumentação para Medição

do Departamento de Engenharia Mecânica e Gestão Industrial.

Resumo

A evolução da tecnologia a nível dos equipamentos de aquisição de dados e das infra-estruturas de rede tem sido responsável pelo aparecimento de novos conceitos, entre eles o de *Laboratórios Remotos*, isto é, a possibilidade de aceder a um laboratório, remotamente, através da *Internet*.

Os *Laboratórios Remotos* contribuem para a partilha de experiências *on-line* destinadas ao ensino ou a actividades de investigação. As vantagens que daqui decorrem não se devem apenas à partilha do conhecimento inerente, mas também à optimização de recursos entre várias entidades, principalmente quando estão em causa equipamentos complexos e dispendiosos. Este conceito também pode ser muito útil nos países com falta de meios experimentais, onde a experimentação *on-line* pode contribuir para atenuar essas insuficiências. No entanto, a utilização dos *laboratórios remotos* ainda é limitada devido aos elevados custos de implementação associados ao *hardware* e ao *software* utilizado. Neste trabalho procurar-se-á dar uma contribuição no sentido de reduzir esses custos, utilizando software do tipo *freeware*.

Neste projecto é apresentado um caso de estudo baseado numa experiência laboratorial de *medição e controlo de nível de líquidos*, para a qual foram desenvolvido duas novas aplicações de monitorização e controlo, uma em Java® e outra em Lazarus®. Esta nova metodologia é uma alternativa à existente, que utiliza o software LabVIEW e o respectivo servidor Web para suporte à aplicação de monitorização e controlo remoto.

A metodologia adoptada mostrou ser bastante eficiente em termos de recursos e universal, funcionando em diferentes plataformas, podendo ser adoptada para outro tipo de aplicações, nomeadamente em ambientes industriais.

Abstract

The evolution of the data acquisition technology and network infrastructures have been responsible for the emergence of new concepts, among them the *Remote Laboratories*, that is, the ability to access to a laboratory, remotely, through the Internet.

Remote Laboratories contribute to online experiences' sharing aimed for learning and research activities. The advantages are not just due to the sharing of inherent knowledge, but also to the optimization of resources between the different entities, especially when one is concerned with complex and cost equipments.

This concept can also be very useful in countries with lack of facilities for education, where online testing can help to overcome these shortages.

However, the use of remote laboratories is still limited, due to high implementation costs associated with the hardware and software used. This work intends to give a contribution towards reducing these costs using freeware software.

This project presented a case study based on a laboratory experience of *measuring and liquid level control*. Two new applications were developed for this goal, one in Java ® and another in Lazarus®. This new methodology is an alternative to the existing one, which is based in LabVIEW software and its Web server for monitoring and remote control.

The methodology adopted has proved to be very effective and universal, working over different platforms, and can be easily implemented in other applications, namely on industrial environments.

Agradecimentos

Gostaria de agradecer às inúmeras pessoas que me acompanharam ao longo do curso e deste projecto, visto que, uma boa camaradagem permite não só ultrapassar os maus momentos que a vida nos reserva, como permite construir os bons momentos que esta nos oferece.

Gostaria também de agradecer aos amigos com os quais desenvolvi imensos trabalhos: *Vítor Silva, Rui Cachorreiro* e *Rui Pena*; aos amigos que me ajudaram nas dificuldades deste projecto: *Luís Rei*, *André Tavares;* e a todos os amigos que estiveram presentes ao longo da minha vida, com os quais partilhei bons momentos contíguos a este trabalho: *Natália Costa, António João Fernandes, André Moreira, Carlos Oliveira, Filipe Carvalho, Daniel Pina, Nuno Ildefonso, João Leal e Eduardo Sousa*. Pois, se estes não são os meus amigos, não sei o significado da palavra amigo.

Quero agradecer também a todas as pessoas do LIM, pelos bons momentos e todo o apoio prestado ao longo destes seis meses: *Cármen, Nuno, Isa, Helena e Tânia*. Agradeço em especial ao *Danilo,* pela compreensão e empenho demonstrados na correcção dos meus textos.

Não posso deixar de agradecer ao meu orientador, Professor Doutor *Joaquim Mendes* e à minha co-orientadora, Investigadora Principal *Teresa Restivo*, pela oportunidade e confiança que me deram.

Índice

Lista de figuras

Abreviaturas

DAQ	*Data acquisition* – Sistemas de aquisição de dados
PID	*Proportional Integral Derivative* – Proporcional, integral e derivativo
FCS	Frame Check Sequence
IE	Internet Explorer
LIM	Laboratório de Instrumentação para Medição

Capítulo 1

Introdução

A evolução tecnológica dos últimos tempos permitiu criar sistemas remotos capazes de executar tarefas complexas e perigosas. Esta evolução não se limitou aos equipamentos, mas reflectiu-se também a nível das infra-estruturas de rede; actualmente, as ligações são muito mais rápidas e fiáveis e o acesso é cada vez mais fácil em qualquer parte do mundo. Suportado pelas novas tecnologias, surgiu o conceito de *Laboratórios Remotos* que está na base deste trabalho.

1.1 - Contexto e motivação

Por *Laboratório Remoto* [1] entende-se um laboratório cujo acesso é possível a partir do exterior, remotamente, normalmente através da Internet. Este conceito, além de promover a interacção entre utilizadores e entidades, possibilita ainda a partilha dos equipamentos e dos custos associados a cada instalação. Uma vez que as experiências são acedidas através da internet, qualquer pessoa, em qualquer parte do mundo, a qualquer hora, pode usufruir da experiência.

Os *Laboratórios Remotos* também permitem o acesso e a utilização de equipamentos perigosos sem que isso represente um risco, nem para o utilizador, nem para os equipamentos. Tem ainda a vantagem de evitar deslocações e facilitar o acesso a utilizadores com limitações físicas.

Convém salientar aqui que o autor defende que a utilização de laboratórios remotos deve ser sempre vista como complemento à formação em ambiente laboratorial, e que não substitui o contacto directo com os equipamentos e materiais, desde que possível.

Dado que se espera que os laboratórios remotos estejam disponíveis 24 horas/dia, com o mínimo de manutenção, exige-se uma elevada fiabilidade e robustez do hardware e software.

Actualmente, a criação de *Laboratórios Remotos* tem sido dificultada pelos elevados custos de hardware e software necessários para a sua disponibilização *on-line*. No que respeita ao *hardware*, poderão surgir algumas dificuldades/restrições dependendo da especificidade do equipamento e dos sinais a processar. Assim, dadas as inúmeras soluções *freeware* para o desenvolvimento de interfaces, intervir a nível do *software* é mais simples e eficaz.

O principal objectivo deste projecto é o demonstrar a possibilidade de monitorizar e controlar um sistema constituído por dois tanques de água em circuito fechado usando software *freeware* e um autómato. O autor deste trabalho encarou este desafio como motivador para desenvolver a solução que será descrita neste relatório. Com ela procura-se aumentar a robustez do funcionamento do sistema, reduzindo o seu custo e disponibilizando uma interface com o utilizador mais leve, amigável e universal.

1.2 - Mais-valias do trabalho

Com esta dissertação espera-se contribuir, para:

- a promoção da interacção entre utilizadores e entidades;
- a redução de custos na implementação de experiências remotas;
- o desenvolvimento do controlo e da interface a partir de *freeware*;
- a definição de uma arquitectura de controlo robusta com capacidade de detecção/ recuperação de erros;
- a criação de alternativas para a arquitectura de comunicação entre o utilizador e a experiência.

1.3 - Estrutura do relatório

Este relatório de projecto é constituído por cinco capítulos.

No primeiro capítulo é feito uma introdução sucinta contextualizando o trabalho desenvolvido e identificando a sua estrutura.

No capítulo seguinte é realizada uma breve descrição das soluções de hardware e de software que podem estar na base de uma estrutura para comunicação de um utilizador com a experiência *on-line*.

No terceiro capítulo descreve-se o sistema para acesso remoto ao objecto deste trabalho, estrutura existente, a interface disponibilizada e os aspectos negativos a melhorar com a nova solução.

O quarto capítulo apresenta uma retrospectiva do trabalho realizado, começando por apresentar o novo *hardware* e seguidamente as aplicações para o controlo e monitorização do sistema de tanques em circuito fechado.

No quinto e último capítulo são apresentadas as conclusões finais baseadas também na avaliação de alguns utilizadores e são registadas várias propostas para trabalhos futuros.

Capítulo 2

Soluções de hardware e software para experimentação remota

No limite qualquer experiência que possa ser controlada por sinais eléctricos pode ser colocada *on-line*. No entanto, para além do grau de sofisticação que possa ter, e consequente custo, poderão também surgir dificuldades na transmissão dos sinais envolvidos. Para conceber um sistema lógico e funcional é essencial compreender muito bem o sistema, os seus requisitos de funcionamento e a definição da interface com o utilizador. Neste capítulo far-se-á uma descrição sintética de diferentes soluções de hardware e software que podem ser usadas neste âmbito.

2.1 - Sinais de entrada/saída dos DAQ

Para conceber um sistema lógico e funcional é necessário compreender bem o sistema a controlar, Figura 2.1, conhecer os seus parâmetros, isto é, o tipo, o número, as gamas de entradas/saídas, o tipo da variação temporal das grandezas e as acções de actuação e/ou controlo requeridos. Importa também referir que é essencial garantir a estabilidade do sistema, prever a detecção/reparação de erros, entre outros aspectos.

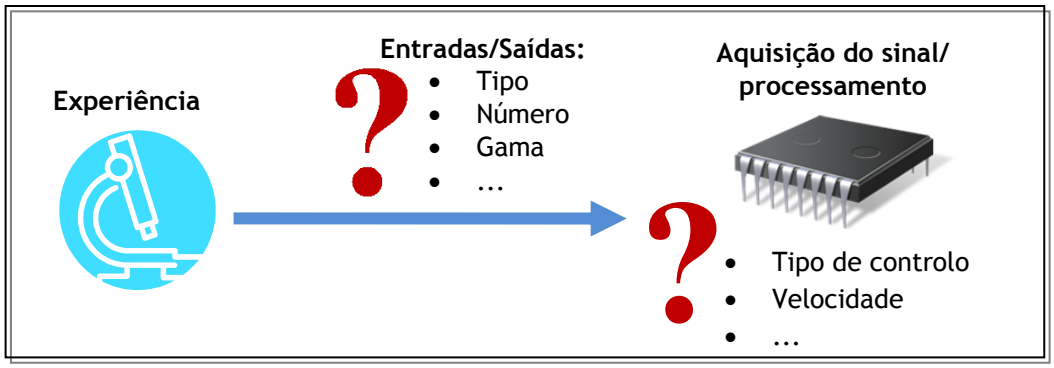


Figura 2.1 – Informações essenciais para projectar uma implementação.

Os resultados e as ordens para a experiência são expressos em tensão (U) e/ou corrente (I) e são este tipo de sinais que são transmitidos para o sistema de aquisição de dados na forma de um sinal analógico e/ou digital.

A enorme variedade de transdutores existentes traduz-se numa grande diversidade de sinais com características distintas. Assim, um condicionamento de sinal pode ser necessário para tornar o transdutor compatível com a gama e/ou tipo de entrada disponível no DAQ.

O condicionamento de sinal é um conceito que não pode ser ignorado, pois não só é importante para projectar um sistema viável, como também poderá envolver custos significativos.

2.2 - Soluções para aquisição e processamento de sinal

2.2.1 – Hardware de aquisição de dados

A expressão "aquisição de dados" (DAQ - *Data Acquisition*) é utilizada para definir um processo automatizado de recolha de informação relativa a um determinado processo (leitura das variáveis do processo: temperatura, pressão, etc.). No entanto, muitas vezes é associado a um processo de controlo, ou seja, um sistema que faz não só monitorização, como também a actuação de forma a controlar o valor de uma ou mais variáveis. O termo "aquisição de dados" acaba por englobar ambas as vertentes, se bem que a designação mais correcta fosse "aquisição de dados e controlo".

Um sistema de aquisição de dados permite a leitura dos sensores, detectores e/ou controlar actuadores através de um elemento de interface de condicionamento de sinal, que processa e armazena os dados recolhidos num computador. Por outras palavras, é o processo pelo qual um sinal eléctrico é convertido num formato digital para posterior visualização, armazenamento, processamento e análise.

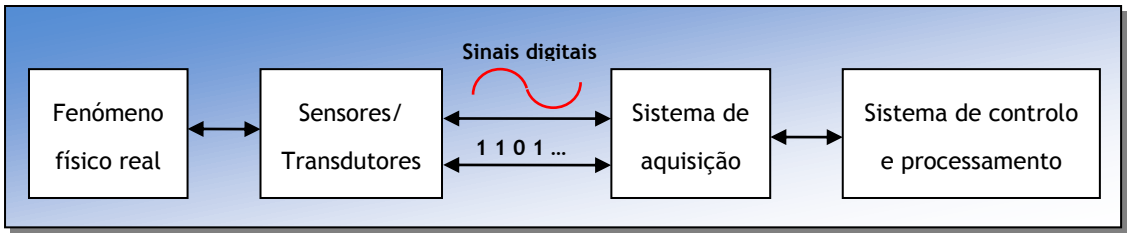


Figura 2.2 – Diagrama de blocos de um sistema de aquisição.

Os sistemas de aquisição de sinal dividem-se essencialmente em dois grandes grupos: os que fazem apenas a aquisição e os que também processam as acções de controlo. Os primeiros necessitam de um computador externo (ou algo equivalente) enquanto os segundos já incluem um processador.

Dos inúmeros tipos de sistemas de aquisição, sem capacidade de controlo, destacam-se:

- Placas de aquisição simples;
- Placas de aquisição com servidor Web;
- Sistemas modulares.

As placas de aquisição simples são as mais económicas, embora o preço aumente proporcionalmente ao número de entradas / saídas e à qualidade dos conversores de sinais analógicos.

A placa da ***LabJack UE9*** [2], Figura 2.3, é um exemplo de uma placa de aquisição simples que apresenta uma boa relação qualidade/preço.

Características:

- Comunicação USB ou Ethernet
- 23 Saídas/entradas digitais, podendo ter 2 *counters* e 6 *timers*
- 14 Entradas analógicas (12 até 16 bit)
- 2 Saídas analógicas (12 bit)
- Preço ≈ **360 € + IVA**

Figura 2.3 – Placa de aquisição **LabJack UE9**

A principal aplicação das placas de aquisição com a funcionalidade de servidor Web é a monitorização, por estarem associadas a uma página Web (HTML dinâmico), onde se apresentam os valores das entradas/saídas.

A placa de aquisição com servidor Web ***WS10*** da ***Novus*** [3], Figura 2.4, é um bom exemplo de relação qualidade / preço.

Características:

- 2 Saídas em relé;
- 4 Entradas analógicas ou digitais;
- Comunicação ETHERNET, Modbus sobre TCP.
- Preço ≈ **540 € + IVA**

Figura 2.4 – Placa de aquisição **Novus WS10.**

A figura 2.5 mostra um tipo de interface para visualização de parâmetros de uma experiência remota com o sistema ***Novus WS10***.

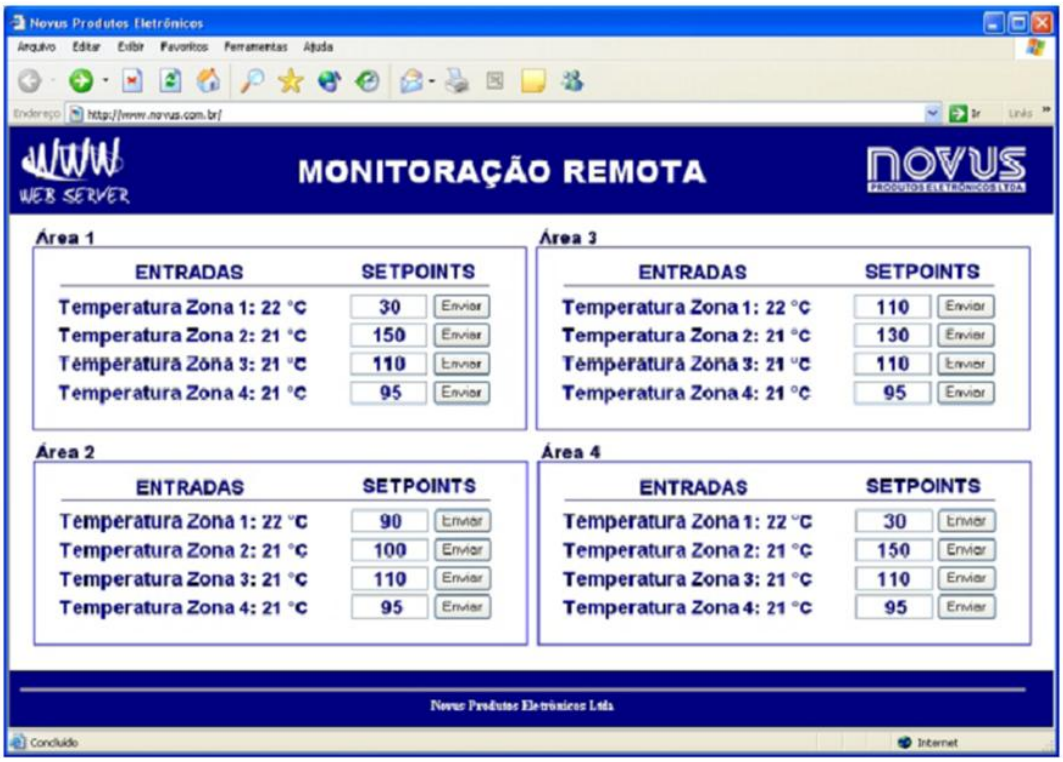


Figura 2.5 – Página Web da **Novus WS10.**

Num outro nível surgem os sistemas modulares. Estes são compostos por um conjunto de cartas (compradas individualmente) que permitem personalizar o tipo/número de entradas e saídas. Porém, as vantagens não se ficam por aqui. Estes sistemas permitem igualmente futuras expansões e, em caso de avaria, a substituição de uma carta com muita facilidade, reduzindo, deste modo, os gastos associados à reparação. Os sistemas modulares são fabricados por várias marcas, sendo as mais conhecidas a ***Omron*** e a ***Schneider Electric***. O sistema modelar da ***Schneider Electric*** [4], figura 2.6, é um bom exemplo para este tipo de placas de aquisição.

Características

- 4 Saídas digitais (com resistências de *pull-up*);
- 2 Entradas analógicas, *resolução de* 11bits + *signo;*
- Comunicação ETHERNET, *Modbus* TCP;
- Preço ≈ **500 € com IVA**

Figura 2.6 – Sistemas de aquisição modelares.

Os sistemas de aquisição com capacidade de controlo têm uma vantagem face aos sistemas supracitados, pois o recurso a um computador é opcional. Estes sistemas são mais caros do que os de aquisição simples, mas, por vezes, esse custo é compensado, visto não necessitarem de um computador para o controlo.

Dos sistemas de aquisição dotados de capacidade de controlo, destacam-se os autómatos, conhecidos pela sua robustez, fiabilidade e volume reduzido. Existe inúmeros fabricantes de autómatos, sendo a ***Mitsubishi***, ***Siemens***, ***Schneider Electric*** e a ***Omron***, Figura 2.7, os mais utilizados.

Esses sistemas são muito usados no mundo industrial, tendo capacidade de executar algoritmos de controlo extremamente complexos. Em caso de avaria, é fácil substituir o dispositivo ou parte dele.

Figura 2.7 – Omron CJ1M.

Podemos ainda usar microcontroladores na aquisição e processamento, mas estes são dispositivos que geralmente tem uma memória limitada e não estão protegidos electricamente, Figura 2.8. Os microcontroladores, por exemplo ***Atmel*** e ***Microchip***, são económicos e consomem uma corrente muito reduzida, razão porque são muito usados em sistemas sem fios.

Figura 2.8 – Micro controladores.

2.2.2 – Comunicações mais comuns dos sistemas de aquisição

Os sistemas de aquisição comunicam com o computador por um determinado meio físico (cabo ou rádio frequência) e por um protocolo que determinará a forma e a taxa a que se poderão trocar os dados.

O meio físico e o protocolo utilizado na comunicação entre a placa de aquisição e o computador variam consoante o tipo de equipamento e de fabricante. Por vezes, os fabricantes usam protocolos próprios que podem trazer dificuldades no desenvolvimento de aplicações de monitorização e controlo.

A comunicação dos sistemas de aquisição e o computador pode ser do tipo:

- **Placas internas (PCI) -** Essas placas são inseridas num *slot* da placa-mãe do computador (sistema interno); um cabo efectua a ligação entre a placa e um bloco externo (placa de terminais) onde se encontram ligados os sensores e actuadores, assim como o condicionamento desses. Este tipo de comunicação obriga ao uso de um computador que, geralmente, se encontra perto do sistema onde se está a fazer a monitorização e/ou o controlo.
- **Comunicação USB -** Estes sistemas permitem uma elevada taxa de aquisição e geralmente a alimentação é feita a partir do próprio cabo USB.

Figura 2.9 – USB.

- **Comunicação série -** Sistema externo cuja ligação ao computador é feita a partir da porta série. É uma comunicação lenta permitindo geralmente algumas dezenas de aquisição por segundo. Tem uma vantagem em relação aos sistemas anteriormente descritos, que reside no facto da dimensão do cabo que liga o computador ao sistema de aquisição poder atingir um comprimento de 15 metros, quando usado o protocolo RS-232. Se se recorrer ao protocolo RS-485, o comprimento do cabo pode atingir, aproximadamente, 1200 metros.

Figura 2.10 – Série.

- **Sistema Ethernet -** O sistema comunica usando a infra-estrutura de Ethernet, quer interna à empresa – Intranet -, quer externa através de um modem, ou router - Internet. Estes sistemas têm uma velocidade de comunicação elevada, embora varie consoante a distância entre o sistema de aquisição e o computador. Neste tipo de sistemas de aquisição é muito frequente o uso do protocolo *Modbus*/TCP, protocolo que é também partilhado

Figura 2.11 – Ethernet.

por vários autómatos (como por exemplo os da ***Schneider Electric***).

2.3 - Software para desenvolvimento de interfaces

Uma experiência online não se resume ao hardware de aquisição e controlo. É necessária uma interface que permita o "diálogo" entre o utilizador e a experiência.

A interface monitoriza as variáveis do sistema, assim como permite ao utilizador actuar no respectivo sistema. Para o seu desenvolvimento, existem alguns conceitos que devem ser respeitados, entre os quais:

- Não apresentar demasiada informação;
- Organizar a informação de forma lógica;
- Usar um grafismo simples para interface;
- Evitar que o utilizador tenha de instalar alguma aplicação (tipo *plug-in*) para permitir o acesso à interface;
- Garantir a comunicação entre a interface e a experiência, implementando algoritmos de correcção de erros.

Existem inúmeras ferramentas para o desenvolvimento de interfaces que poderão ser do tipo *freeware* e/ou *software* pago.

Nos *Laboratórios Remotos* utilizam-se em geral os programas LabVIEW e o MATLAB que permitem o desenvolvimento de aplicações para a monitorização, actuação e/ou controlo de uma experiência a partir da sua própria interface. São ferramentas muito completas e com capacidade de execução de cálculos sofisticados. A figura 2.12 exemplifica uma aplicação LabVIEW lançada numa página Web, experiência disponibilizada pela FEUP [5].

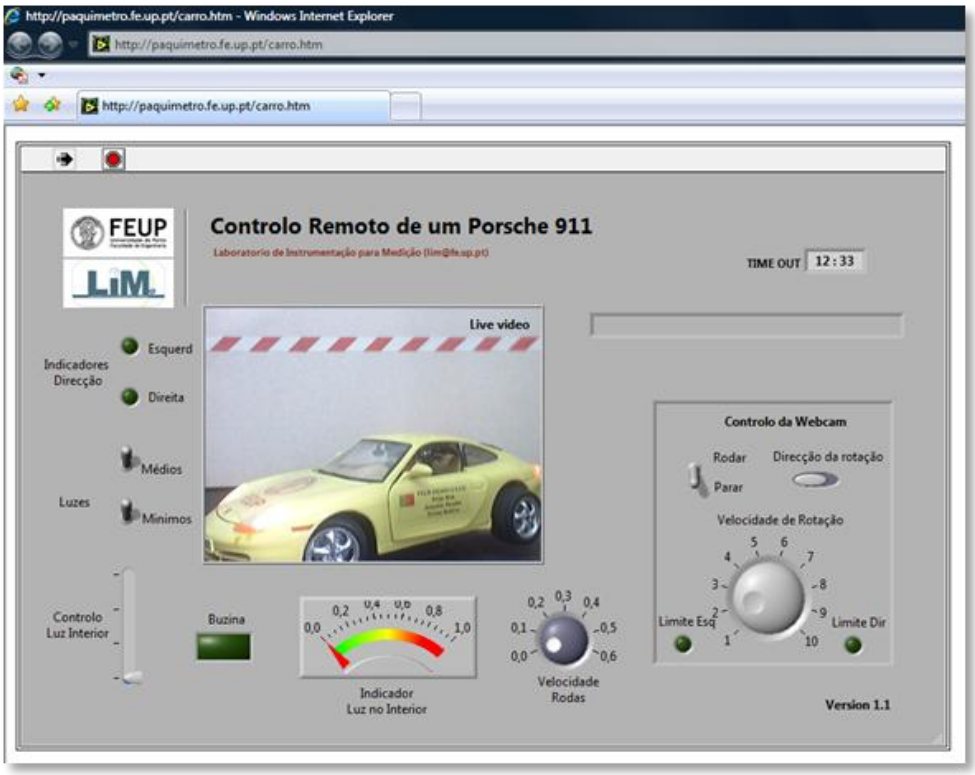


Figura 2.12 – Interface HTML/LabVIEW.

O LabVIEW e o MATLAB são ferramentas computacionais sobejamente conhecidas na comunidade de automação e controlo, e usadas no mundo industrial, no entanto, existem outros tipos de *software* para este efeito. Nos últimos anos, têm sido desenvolvidas interfaces em HTML/PHP/Java, sendo o controlo geralmente feito a partir de CGI (*Common Gateway Interface*), Figura 2.13.

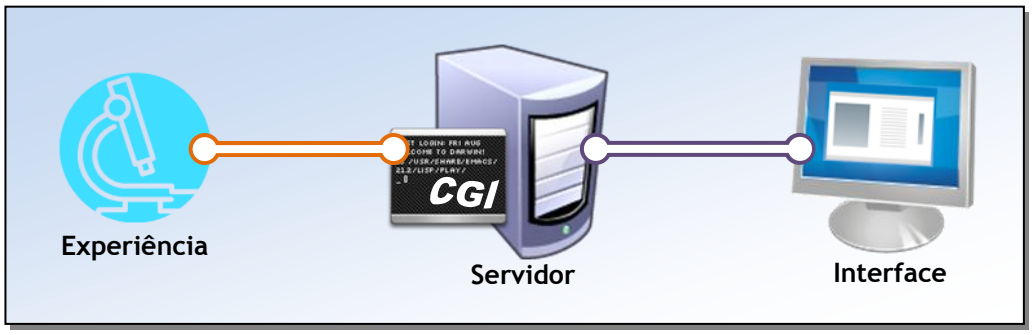


Figura 2.13 – Controlo com recurso a CGI.

CGI é uma importante tecnologia que permite gerar páginas dinâmicas, possibilitando a um utilizador Web transferir parâmetros para uma aplicação alojada num servidor Web (programas que fazem controlo da experiência), podendo recorrer a uma base de dados. A partir do HTML é feita a leitura/escrita na base de dados, sendo essa interpretada por uma

aplicação que se encontra alojada no servidor e que, por sua vez, actua no sistema. Essa técnica tem um inconveniente, face ao uso de CGI, a sua baixa velocidade.

As aplicações que residem no servidor podem ser desenvolvidas a partir de diversas linguagens como C/C++, Visual Basic, Python, Free Pascal (Delphi, Lazarus, …), Java, entre outras.

Resumo desde capítulo

No desenvolvimento de um sistema de monitorização e controlo remoto é essencial ter uma definição clara do número e do tipo de entradas/saídas necessárias, bem como das acções pretendidas. A definição da solução passa ainda pelo software a ser utilizado na realização da interface com o utilizador, do seu grau de interactividade e do nível de grafismo desejado.

Capítulo 3

Monitorização e controlo de nível

No Laboratório de Instrumentação para Medição (LIM) do Departamento de Engenharia Mecânica e Gestão Industrial da FEUP, local onde o presente trabalho de projecto foi realizado, existe um conjunto de experiências *on-line*, cada uma com a sua especificidade, mas partilhando quase todas a mesma estrutura baseada em LabVIEW, no seu servidor Web e em placas de aquisição *plug-in* da National Instruments, [6,7].

3.1 - Sistema experimental

Para desenvolver este projecto, foi utilizada a experiência de "Medição e Controlo de Nível" que tem por base um sistema constituído por dois tanques de água ligados em circuito fechado, figura 3.1.

Figura 3.1 – Tanque superior e inferior e armário eléctrico.

O tanque superior assume a função principal, enquanto o outro, situado na parte inferior, desempenha a função de depósito do sistema. Para permitir a circulação da água do tanque

inferior para o tanque principal existe uma bomba DC cujo débito é regulado por variação da sua tensão de alimentação. A circulação de água é realizada pelo circuito hidráulico com a ajuda de um tubo para enchimento do tanque principal, estando o de descarga para o tanque inferior munido de uma válvula.

A válvula de descarga e a bomba são os elementos actuadores deste sistema, em que a monitorização do nível é realizada a partir de vários transdutores, transmissores e detectores, presentes no tanque principal, figura 3.2.

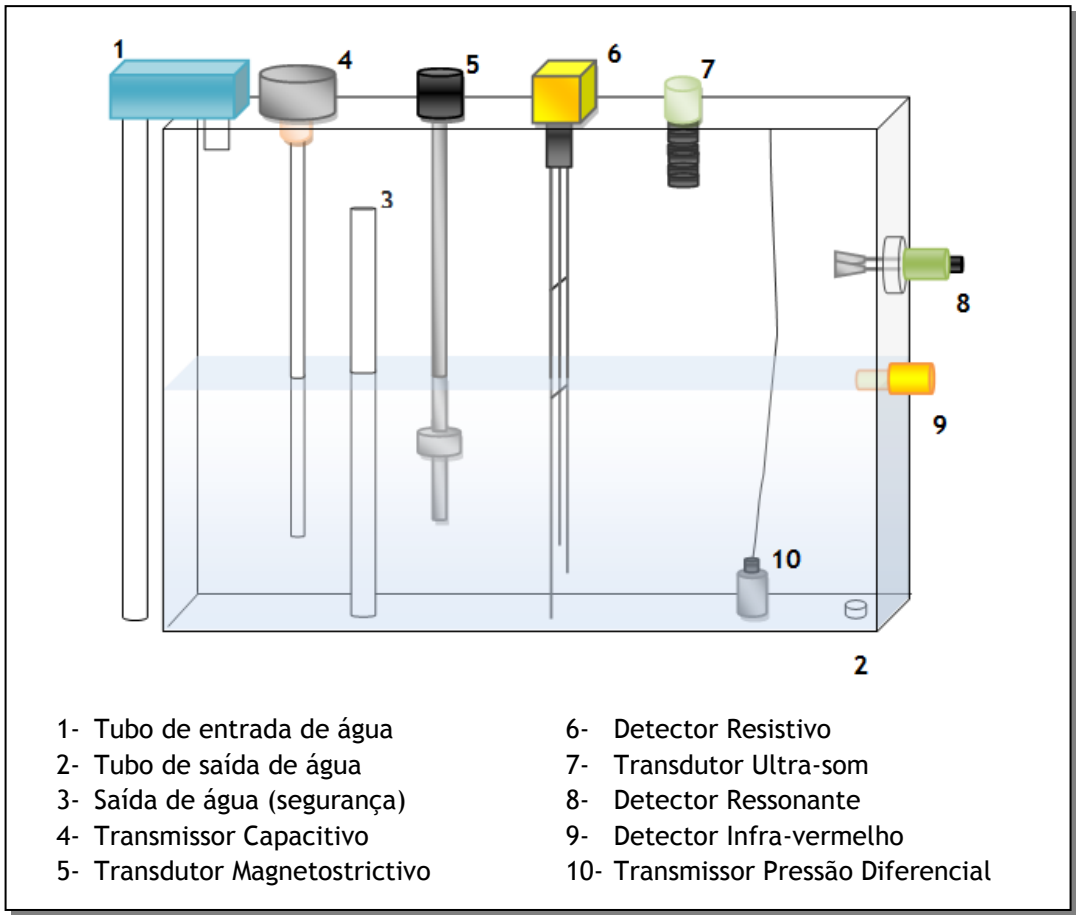


Figura 3.2 – Distribuição física dos componentes.

Na parte inferior ao lado do reservatório em aço inox, existe um quadro eléctrico onde estão reunidos todos os circuitos de condicionamento dos vários transdutores, detectores e actuadores, assim como as fontes de alimentação e o sistema de aquisição (um autómato).

3.2 - Hardware de aquisição

Para além do equipamento descrito acima, é utilizado um PC que comunica com o autómato por porta série, que contém a aplicação de monitorização e controlo desenvolvida em LabVIEW e um servidor Web (partilhado por todas as experiências).

Neste projecto é dado maior destaque ao *hardware* de aquisição e processamento de sinal, assim como à solução que permite disponibilizar a experiência *on-line*.

O sistema de aquisição desta experiência é um autómato **Omron CJ1M-312** [8]. O controlo é partilhado pelo autómato e pela aplicação desenvolvida em LabVIEW. O autómato contém os seguintes módulos:

- Módulo de entradas digitais (ID211);
- Módulo de saídas digitais (OD211);
- Módulo de entradas analógicas (AD081-V1);
- Módulo de entradas/saídas analógicas (MAD42).

Figura 3.3 – Autómato Omron.

O autómato comunica com a aplicação LabVIEW através da porta série (protocolo *Hostlink*), que possibilita a monitorização e o controlo da experiência e que implementa o servidor Web, permitindo disponibilizar a experiência *on-line*. O servidor externo (foton) mantém um directório das experiências, a base de dados dos utilizadores e a agenda para a marcação da experiência.

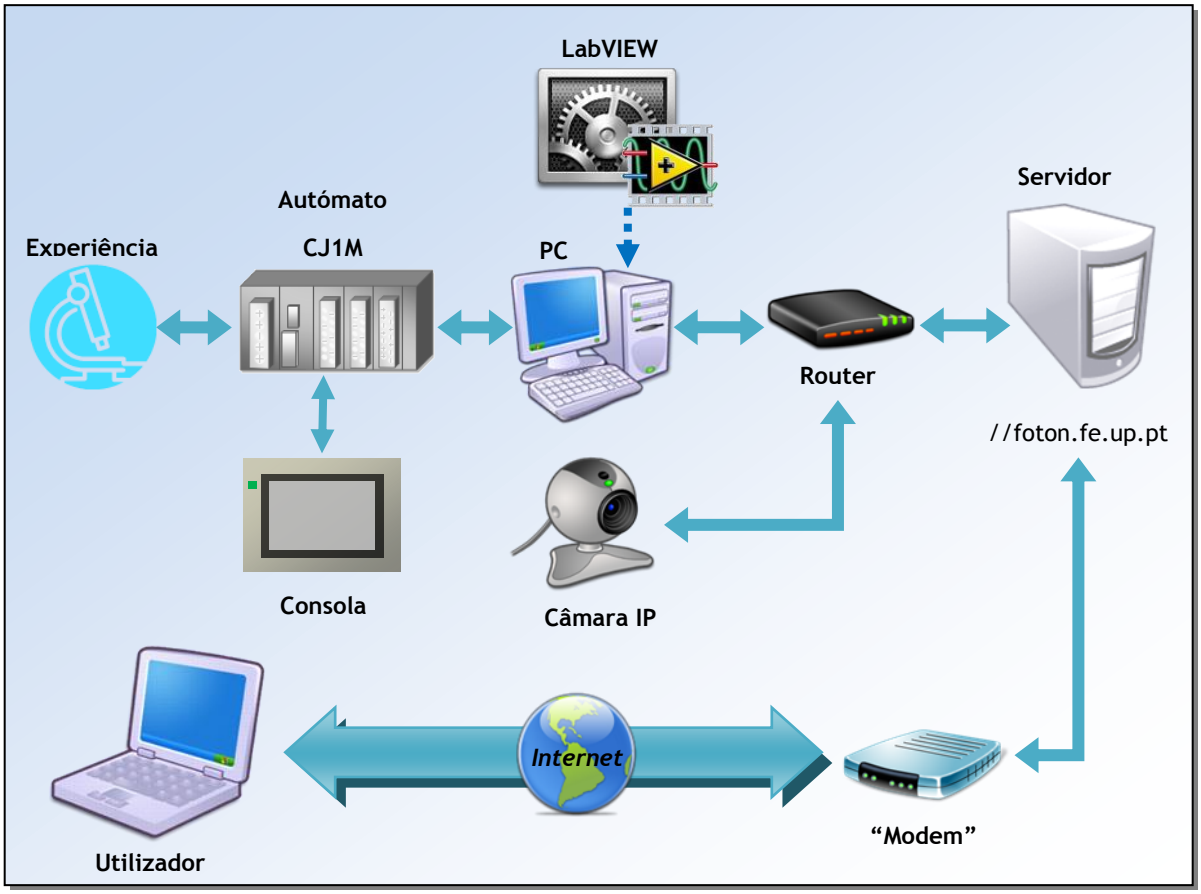


Figura 3.4 – Arquitectura do sistema.

Neste sistema existe ainda uma consola táctil (Omron NS5-MQ00-V2) que comunica directamente com o autómato, esta é responsável pela monitorização e controlo local da experiência.

Uma câmara IP (**Axis 210**) permite visualizar a experiência, factor importante quando se está a interagir remotamente, reforçando assim a sensação de realidade.

3.3 - Software de monitorização e controlo

A aplicação de supervisão e controlo foi desenvolvida em software LabVIEW e encontra-se alojada no computador local. Esta aplicação é composta por um programa principal e dois subprogramas. O principal comunica directamente com o autómato através da porta série do computador e é a interface do utilizador para acesso *online* à experiência.

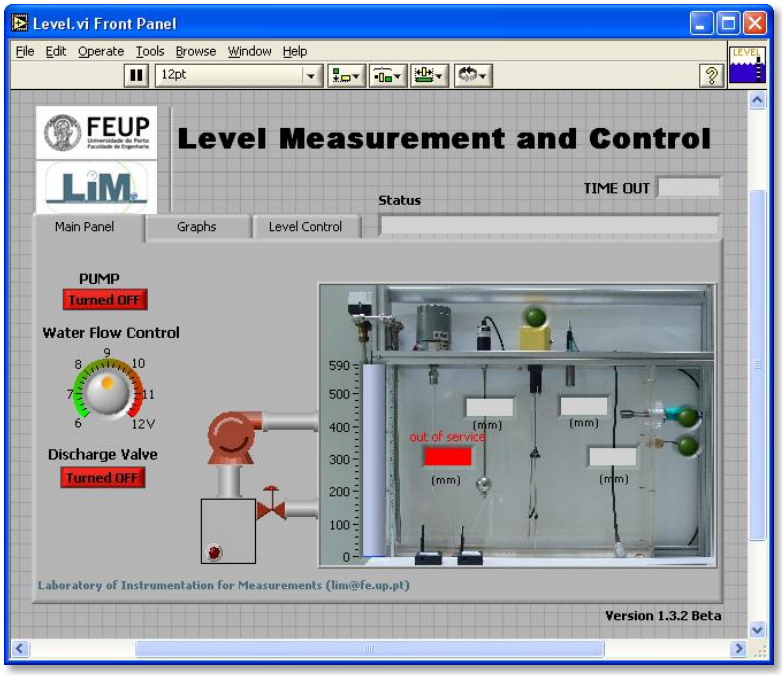


Figura 3.5 – Interface/programa principal.

Os dois subprogramas que suportam a aplicação principal são igualmente desenvolvidos em LabVIEW, e têm as seguintes funcionalidades:

- Verifica_estado_booking.vi

Este subprograma verifica se o utilizador está a utilizar a interface, e em caso contrário reinicializa a experiência de modo a prepará-la para o próximo utilizador.

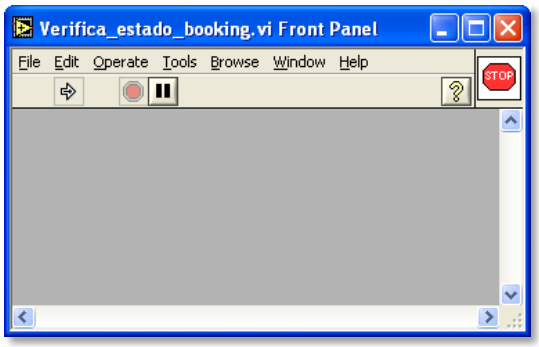


Figura 3.6 – Verifica_estado_booking.vi

- MoodleCommandInterpreterLevel.vi

Este subprograma verifica o "booking", ou seja, se um utilizador fez a reserva da experiência. Desta forma só utilizador registado poderá aceder e manipular remotamente a experiência.

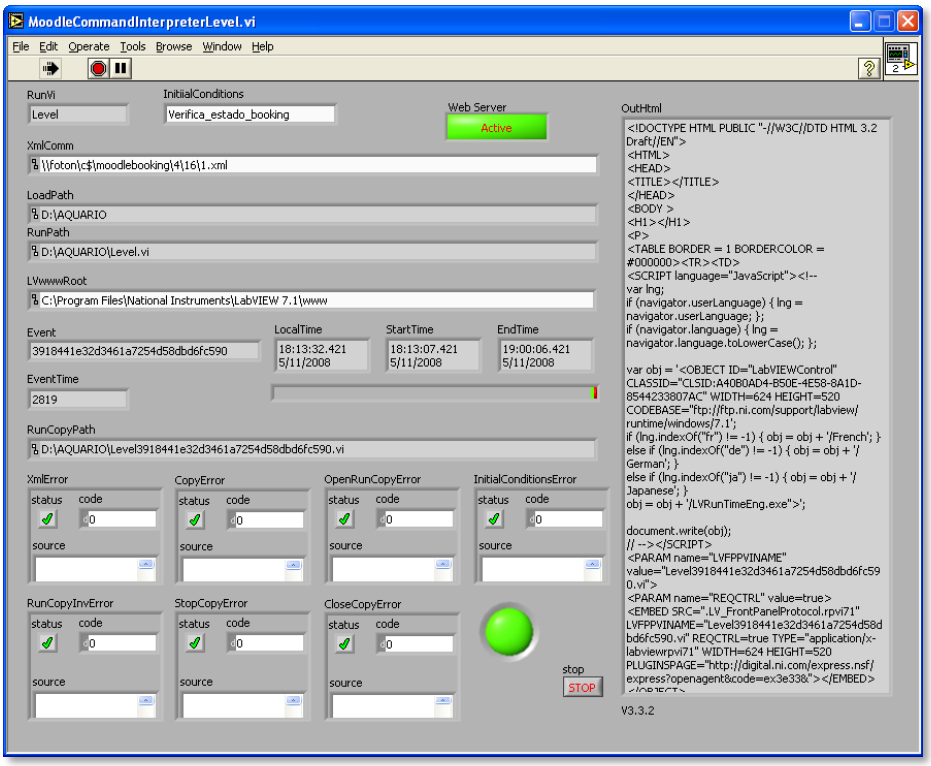


Figura 3.7 – MoodleCommandInterpreterLeve.vi

3.4 - Acesso à experiência

Para o utilizador usufruir de uma experiência on-line deve aceder ao link http://elabs.fe.up.pt, e depois *remotelab*, figura 3.8, onde é apresentada a lista das várias experiências disponíveis. Caso o utilizador queira interagir com a experiência para **medição e controlo de nível**, deverá seleccionar o link *"Level measurement and control"* e efectuar o *login*, figura 3.9.

Figura 3.8 – Página principal.

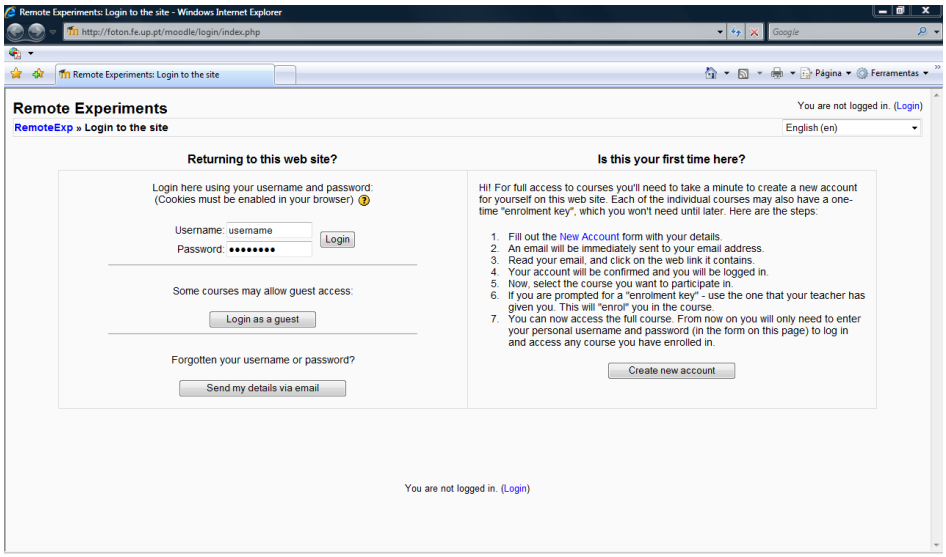


Figura 3.9 – Página de *login*.

Após concluído o processo do *login* surge o acesso à página da experiência, figura 3.10.

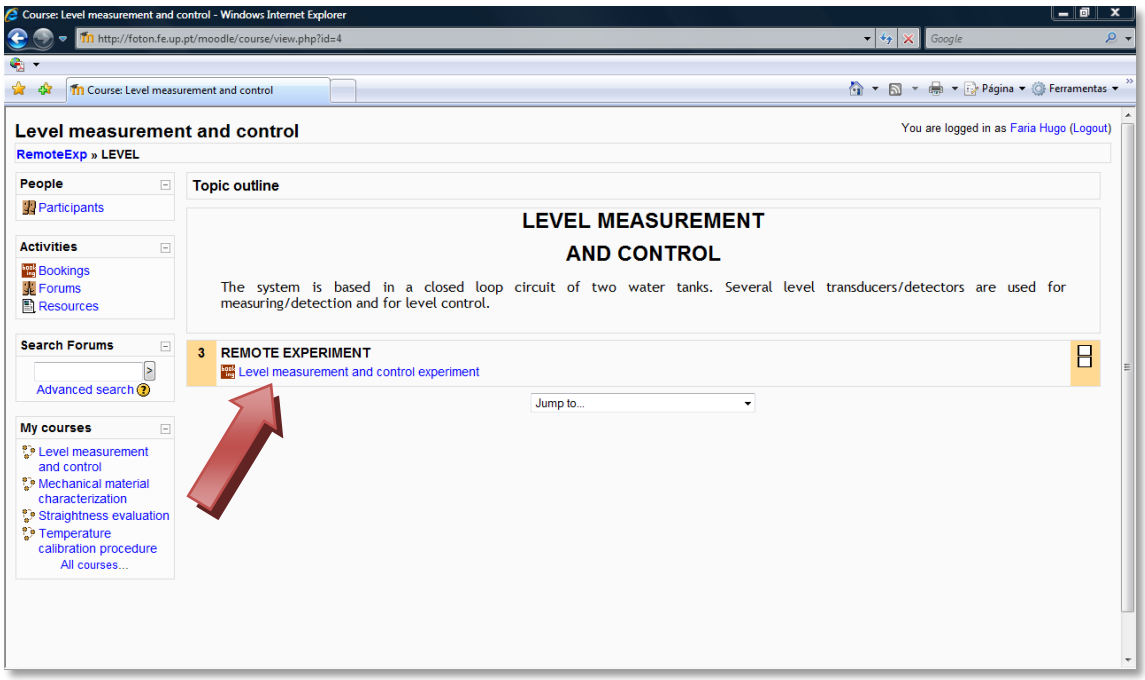


Figura 3.10 – Acesso à experiência.

Ao seleccionar a experiência "medição e controlo de nível", o próximo passo será fazer a reserva da desta, através do sistema de booking, figura 3.11.

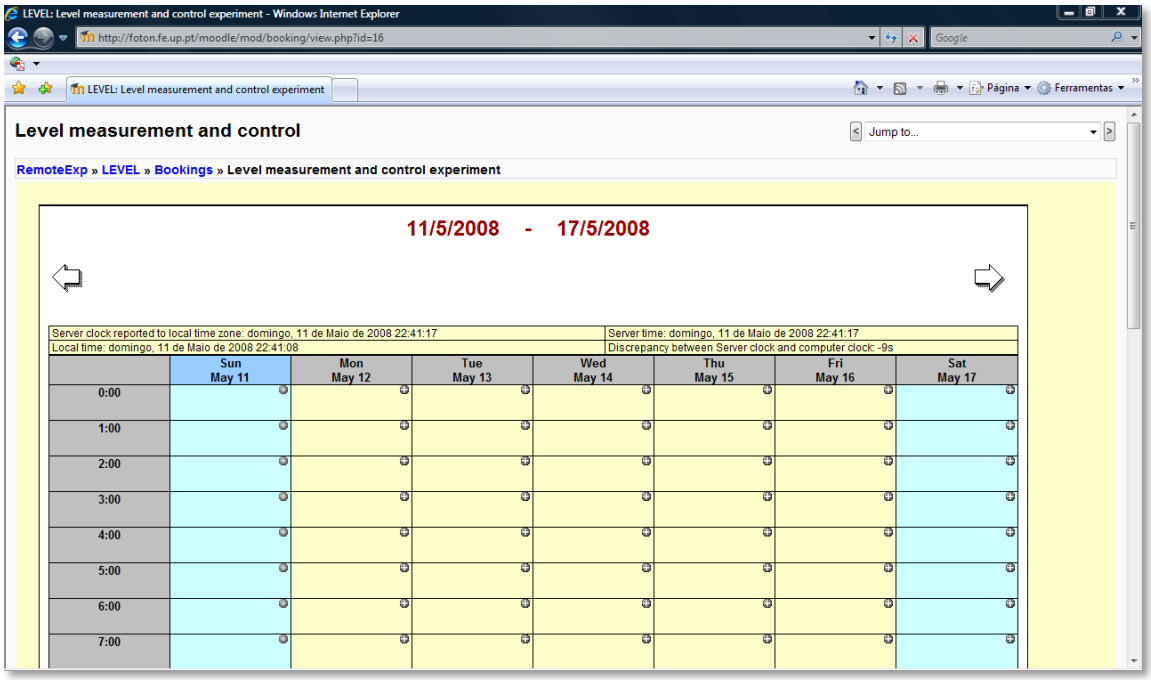


Figura 3.11 – Página de reserva ("booking").

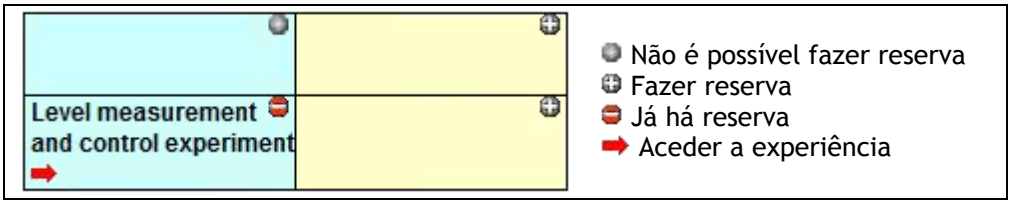


Figura 3.12 – Reservar experiência.

Após a marcação da reserva, o utilizador pode aceder à página da interface (página de monitorização e controlo), Figura 3.13, bastando-lhe pressionar o ícone ➡.

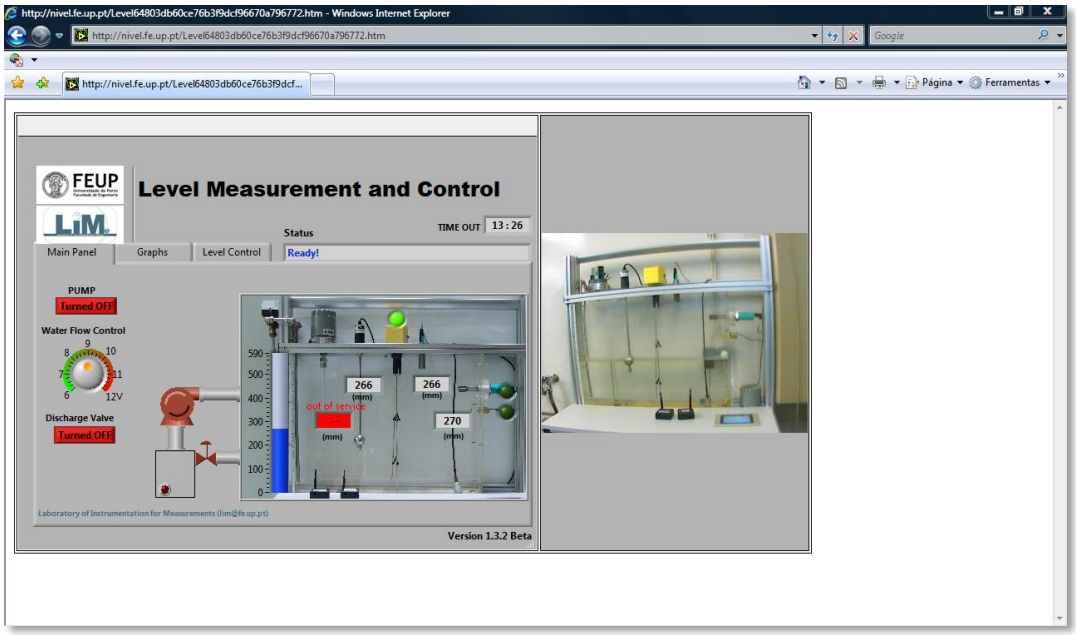


Figura 3.13 – Página de monitorização e controlo da experiência (LabVIEW).

O utilizador remoto interage com a experiência através do menu principal da interface que permite a monitorização e a actuação da experiência. Essa interface é composta por três painéis:

- Painel principal (*Main Panel)*

  Neste painel, figura 3.14, o utilizador pode actuar de forma directa (malha aberta) sobre a bomba da água (1) (ligar /desligar) e a válvula de saída (2) (ligar/desligar). Pode ainda variar o caudal da água (3), variando a tensão de alimentação da bomba. Neste painel, é igualmente possível a monitorização de três transdutores e dois detectores: transdutor Magnetostrictivo (4), transdutor Ultra-som (5), transmissor de Pressão Diferencial (6), Detector Ressonante (7) e Detector Infravermelho (8).

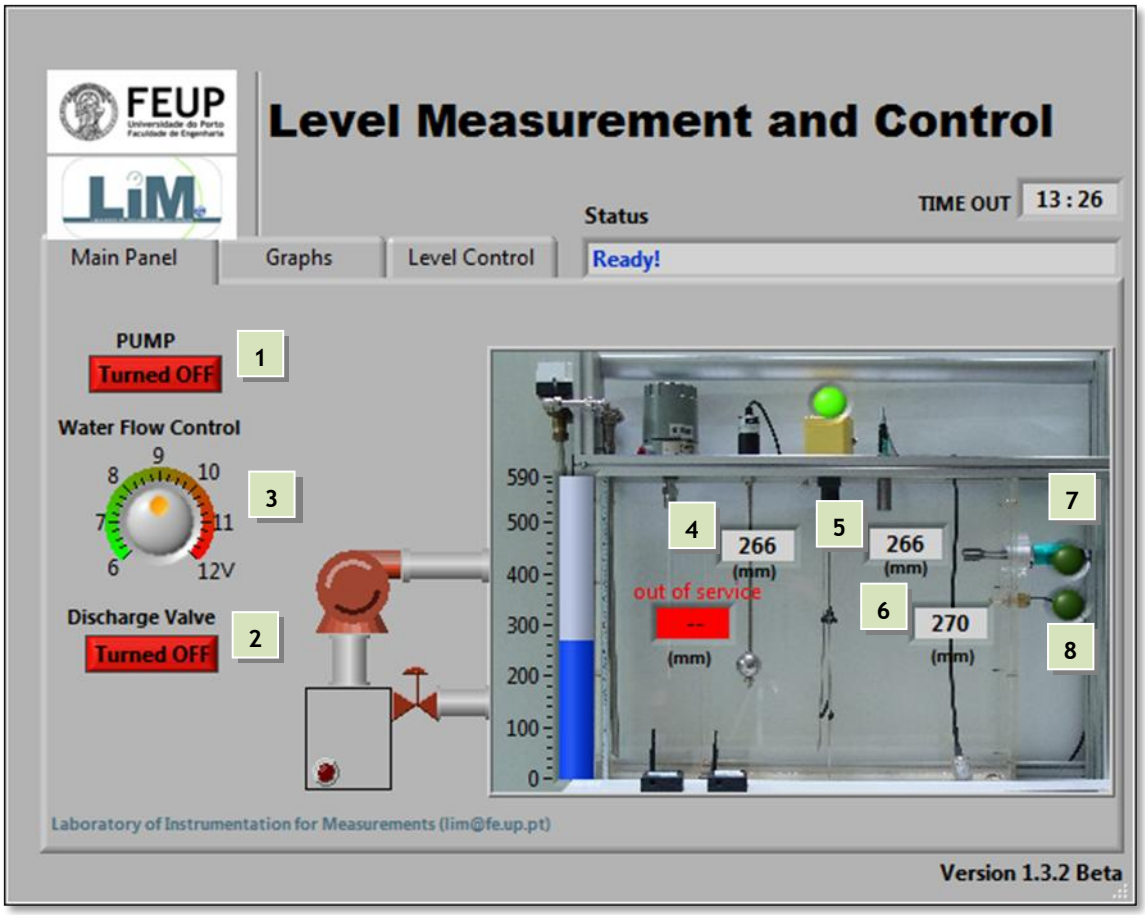


Figura 3.14 – Painel principal da interface do utilizador.

- Evolução temporal do nível *(Graphs)*

Neste painel é possível verificar a evolução temporal do nível da água registada pelos três transdutores (Ultra-som, Magnetostrictivo e Pressão Diferencial). Através da selecção das respectivas *Check box*, o utilizador pode tornar visível o respectivo gráfico, figura 3.15.

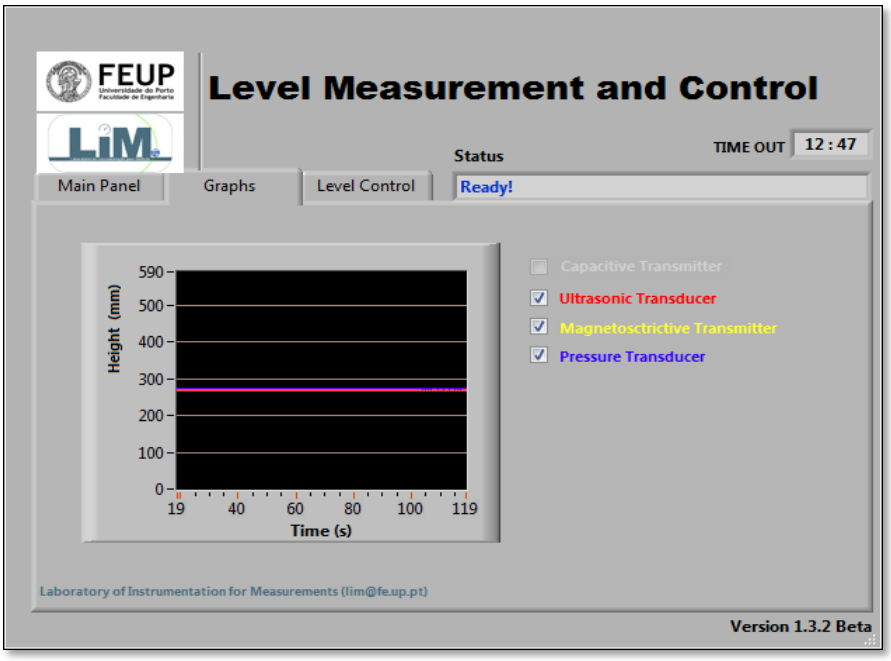


Figura 3.15 – Histórico dos sensores.

- Painel do controlo de nível *(Level Control)*

  Na figura 3.16 está representado o painel de controlo de nível, onde é possível definir o "set-point" bem como os diferentes parâmetros do algoritmo PID de controlo (desenhado na aplicação LabVIEW). Neste painel é também possível ao utilizador visualizar a resposta do sistema aos parâmetros do algoritmo de controlo introduzidos (evolução do nível da água, valor desejado).

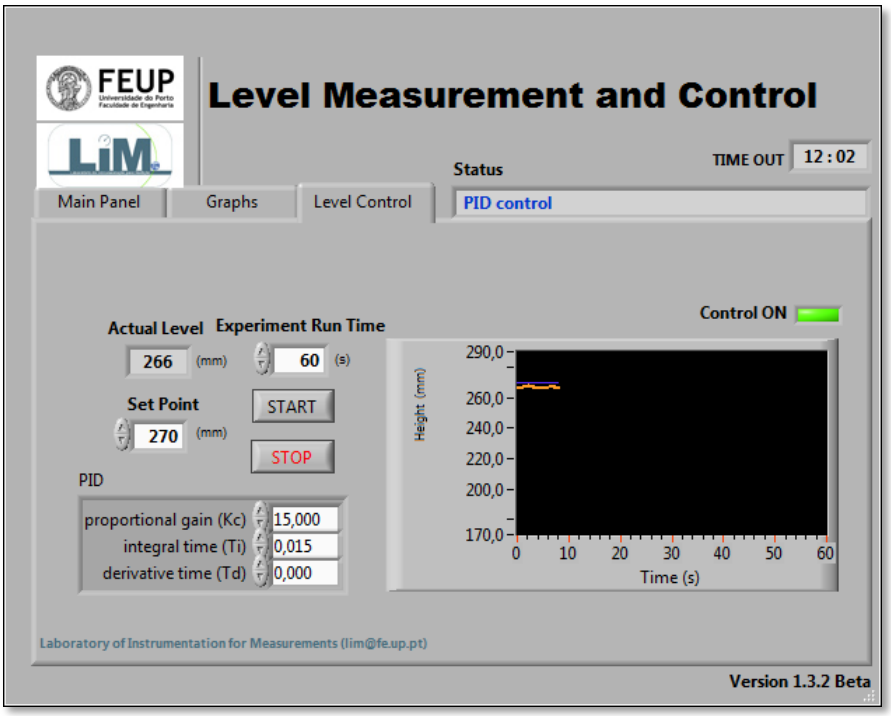


Figura 3.16 – Painel do controlo do nível.

3.5 - Vantagens e desvantagens

A metodologia usada para colocar a experiência "medição e controlo de nível" on-line apresenta inúmeras vantagens, entre as quais a sua robustez de funcionamento, a interface amigável e a velocidade de comunicação da interface com a experiência. No entanto, apresenta algumas desvantagens:

- a ferramenta usada para a monitorização e controlo não é *freeware*;
- necessidade de um computador local alojado junto da experiência;
- necessidade de um *plug-in* específico para o utilizador poder visualizar a interface;
  De facto, o utilizador deverá instalar no seu computador o programa LabVIEW 7.1 ou então o LabVIEW 7.1 Run-Time (~32MBytes).
- tempo de transferência da interface para o computador do utilizador; quando o utilizador acede à experiência, é-lhe transferido a interface, figura 3.17, cujo tempo de transferência é elevado, variando com a capacidade da rede.

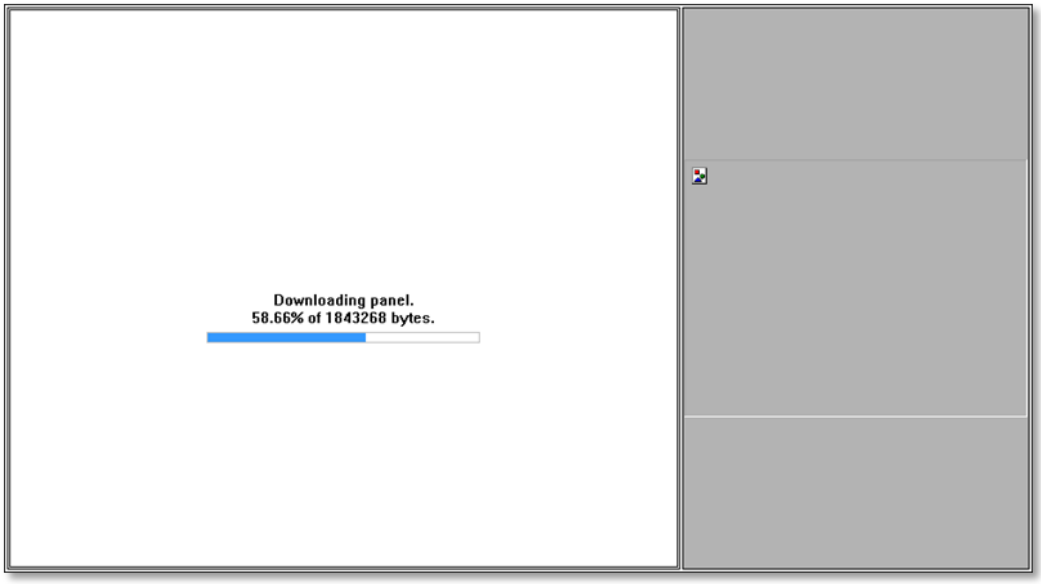


Figura 3.17 – Transferência da interface.

- Incompatibilidade com os browsers; verifica-se também que existe uma incompatibilidade com alguns browsers, por ex. a página de reserva ("booking") não é totalmente visível no ***mozilla***, figura 3.18.

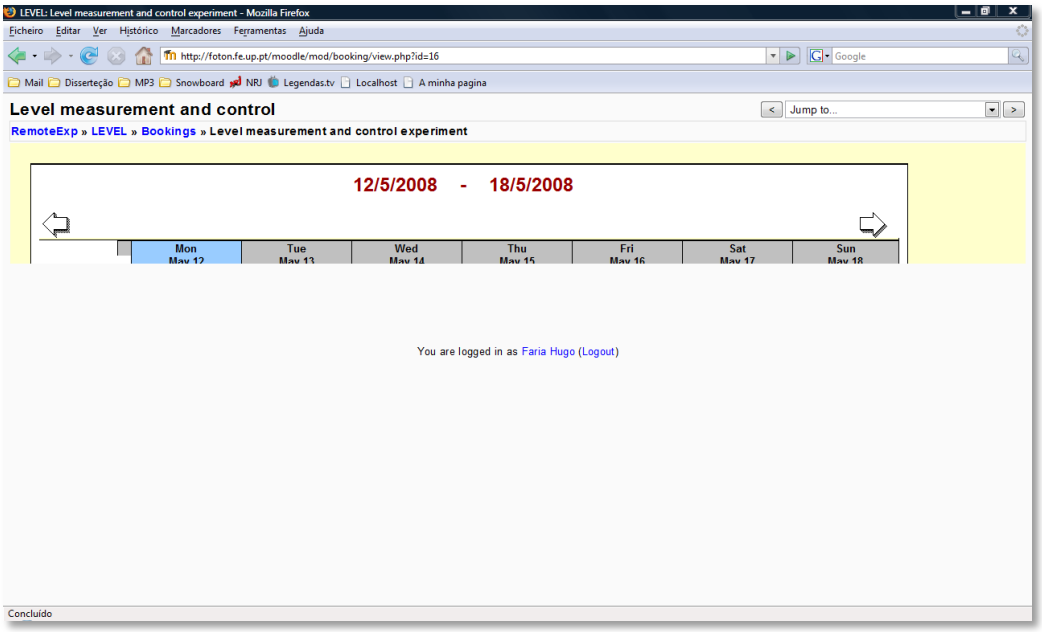


Figura 3.18 – Incompatibilidade do *browser*.

Verificou-se ainda problemas na ligação à câmara IP através do Internet Explorer 7. De facto as restrições de segurança impedem por "default" a instalação do respectivo driver, impossibilitando a transmissão de vídeo.

Esses são os aspectos negativos na metodologia utilizada nas experiências do *Laboratório de Instrumentação para Medição* e que actualmente se pretende melhorar.

Capítulo 4

Trabalho desenvolvido

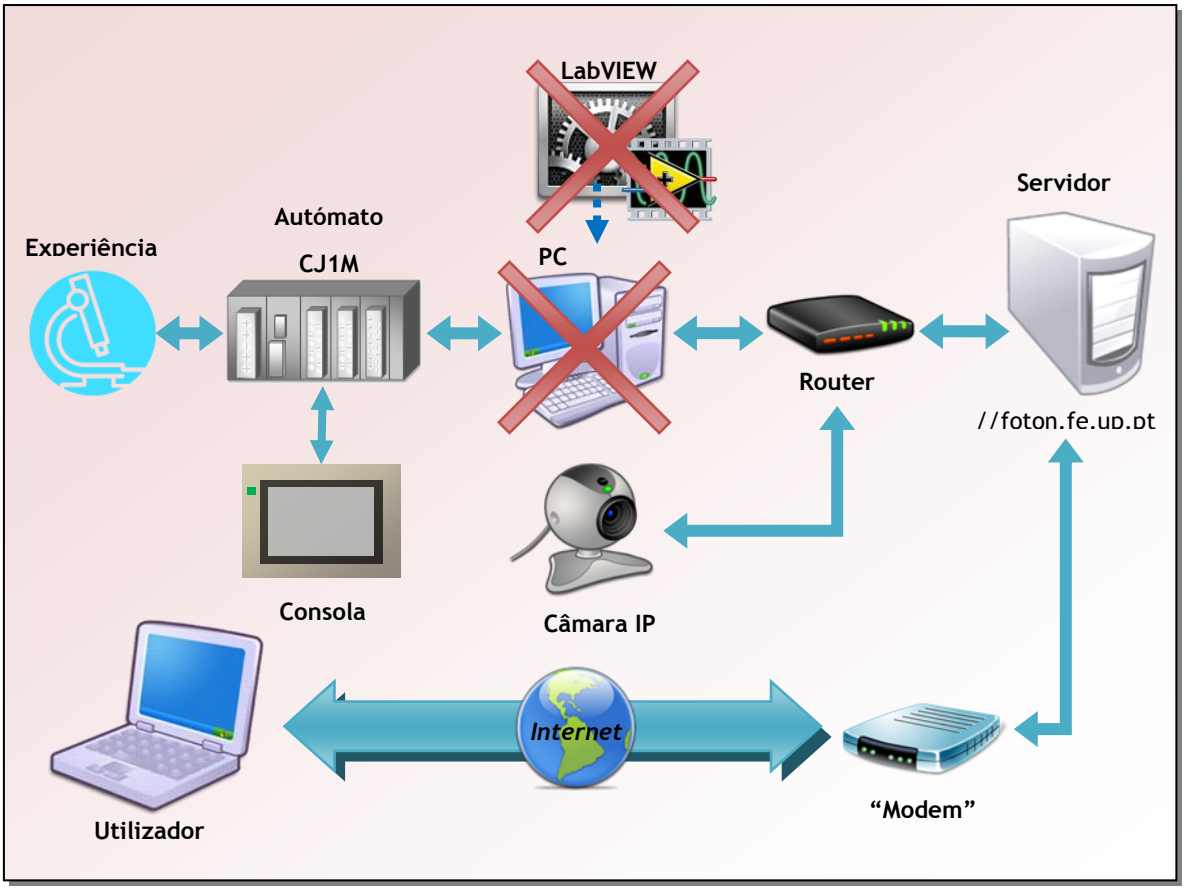
No presente projecto desenvolveu-se uma metodologia para monitorização e controlo desta experiência e de laboratórios remotos com filosofia semelhante, que permite baixar os custos e disponibilizar uma solução leve, estável e robusta.

4.1 - Proposta de arquitectura

Quase todas as experiências *on-line* do **LIM** partilham a utilização de software de LabVIEW, a utilização de um sistema de booking e o uso de um servidor Web comum, requerendo ainda um PC por experiência. Assim as desvantagens são em geral, as mesmas. Apesar desta nova solução ser desenvolvida para a experiência de ***medição e controlo de nível***, poderá ser aplicada futuramente a todas as restantes experiências do laboratório, eliminado os problemas referidos no capítulo anterior, a saber:

- Eliminar o computador que se encontra dedicado à experiência, reduzindo simultaneamente:
  - ✓ o custo de *hardware*;
  - ✓ o custo de *software* (Windows, LabVIEW, etc.);
  - ✓ o consumo energético;
  - ✓ uma fonte de calor e ruído;
  - ✓ o espaço ocupado.
- Remover as aplicações de controlo e monitorização feitas em LabVIEW, porque, apesar de poderosa, esta ferramenta tem desvantagens como:
  - ✓ custo de software;
  - ✓ obrigatoriedade de utilização de um *plug-in* específico.
- Reduzir o tempo de transferência da interface para o computador do utilizador;

- Facilitar o uso das experiências *on-line* do laboratório de instrumentação para medição;
- Alargar o leque dos browsers que podem ser usados.



4.2 - Hardware

Com a metodologia que se pretende implementar, Figura 4.2, o sistema de aquisição deve ser dotado de uma porta *Ethernet*, para assim comunicar directamente com o servidor Web, contribuindo, dessa forma, para a eliminação do computador local. A experiência de ***medição e controlo de nível*** tem um autómato Omron-CJ1M (módulo de comunicação e processamento) que funciona como sistema de aquisição de dados. O equipamento de origem não estava dotado de porta *Ethernet*, o que obrigaria à utilização de um conversor RS-232/Ethernet, Figura 4.1. Contudo optou-se antes pela substituição do CPU pelo modelo Omron-CJ1M CPU12, semelhante ao anterior, mas dotado de uma porta *Ethernet*. Esta alteração possibilitou

Figura 4.1 – Conv. RS-232/Ethernet

ao autómato comunicar directamente com o servidor Web. Esta mudança fez com que os programas de monitorização e controlo passassem a ser alojados no servidor Web.

4.3 - Software

A nível de software foram realizadas várias alterações, entre as quais a alteração do programa residente no autómato por dois motivos. Primeiro, porque como foi dito a carta de CPU foi alterada o que implicou o ajuste da configuração. Segundo, uma vez que se pretendia comunicar com o autómato a partir da nova solução, deveria assegurar-se que esta não iria interferir no controlo já existente, se ambas estivessem activas. Na fase de teste mantiveram-se as várias soluções para comparar desempenhos, estabilidades, funcionamentos, etc. Outra alteração, foi a substituição do LabVIEW por ferramentas *freeware*, tais como:

- **Lazarus** [9] **-** usa como linguagem de programação o *free pascal* e é uma linguagem orientada a eventos.
- **HTML *(Hypertext Markup Language)*** [10] – é uma linguagem de marcação, utilizada para construir páginas na Web.
- ***PHP*** - Inicialmente conhecido por "Personal Home Pages" hoje em dia por "PHP: Hypertext Preprocessor", é uma linguagem de programação de computadores interpretada (linguagem *script*) e muito utilizada para gerar conteúdo dinâmico nas páginas Web.
- ***Java Applet*** [11] **-** uma linguagem orientada a objectos, interpretada, dinâmica, *multithreaded*. Estas aplicações desenvolvidas em Java e que podem ser chamadas dentro de páginas HTML.

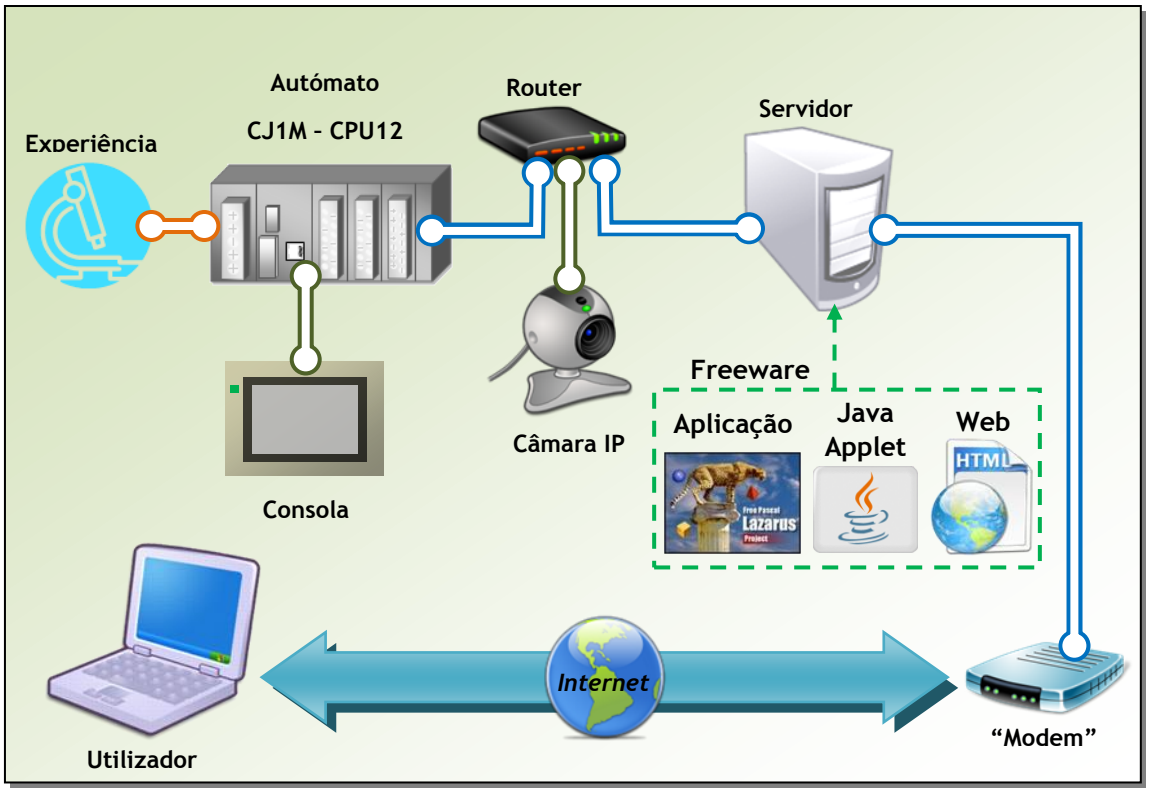


Figura 4.2 – Metodologia implementada.

Neste projecto foi desenvolvida uma aplicação em *Freepascal* que estabelece a ligação entre a aplicação de controlo/monitorização e o sistema de aquisição (autómato), figura 4.3, ("*driver*")

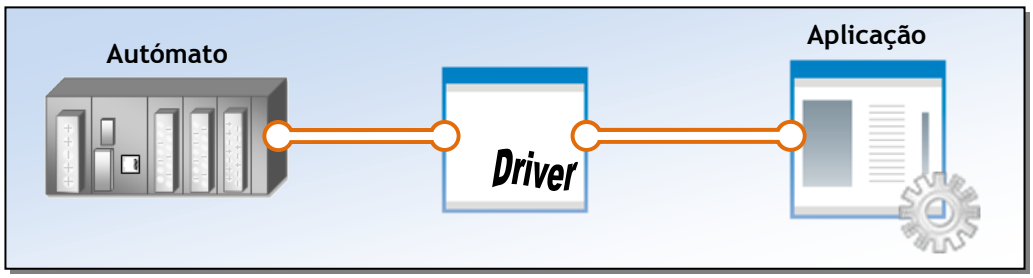


Figura 4.3 – *Driver*.

4.4 - Driver

Este *driver* tem como objectivo transformar os comandos enviados pela aplicação no formato usado no autómato, dependente do tipo de comunicação usado.

O *driver* simplifica o desenvolvimento de aplicações e a comunicação de e para o autómato. O investigador pode desenvolver uma aplicação sem se preocupar com o protocolo usado no autómato, tendo apenas que respeitar a semântica usada pelo *driver*. Este *driver*

permite fazer leituras e escritas no autómato. Para tal, deverá receber da aplicação o respectivo pedido (uma determinada trama TCP/IP). Os pedidos feitos ao *driver* têm que respeitar uma determinada metodologia. Caso se pretenda um pedido de:

- **Leitura**

  Quando a aplicação necessita de fazer uma leitura de uma variável do autómato deverá enviar uma trama TCP/IP ao *driver*. A trama é do tipo "*string*", composta por dois parâmetros (separados por um espaço), figura 4.4. No primeiro é indicado o tipo de pedido, seguido do tipo/número da variável (zona de memória a aceder e endereço).

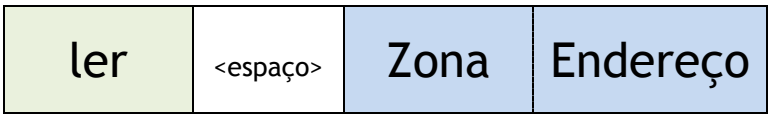


Figura 4.4 – Envio da trama de leitura.

Após enviar o pedido de leitura, a aplicação recebe do *driver* a trama de resposta, que é do mesmo tipo da enviada, mas que inclui um terceiro parâmetro, figura 4.5, onde é indicado o valor da variável (em decimal).

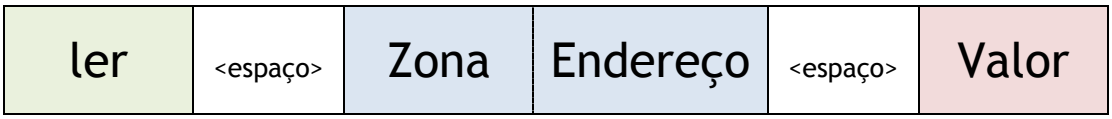


Figura 4.5 – Recepção da trama de leitura.

- **Escrita**

  Caso a aplicação pretenda escrever numa variável do autómato, deverá enviar para o *driver* uma trama idêntica a anterior, composta igualmente por três parâmetros.

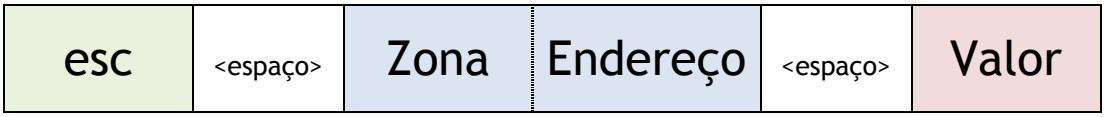


Figura 4.6 – Envio/recepção da trama de escrita.

Após o envio da trama, a aplicação recebe uma trama de resposta igual à enviada, que confirma o êxito do pedido.

Para todos os pedidos, sejam de leitura ou de escrita, o segundo parâmetro (tipo/número) deve respeitar a metodologia. Ele é composto por uma letra que antecede um número de cinco dígitos. Isso deve-se ao facto do autómato dispor de cinco áreas áreas de memória onde são alojadas as variáveis, figura 4.7:

Área	AR	CIO	DM	EM	HR
Nº de posições	959	6143	32767	32767	511
Letra do 2º parâmetro	A	C	D	E	H

Figura 4.7 – Área de memória.

A figura 4.8 mostra um exemplo de um pedido de escrita enviado ao *driver*.

esc	___	D	01265	___	120

Figura 4.8 – Exemplo da trama de escrita.

Para este projecto, foram desenvolvidos dois *drivers*: um "Driver RS-232" e um "Driver Ethernet", ambos usam as mesmas tramas de pedido de leitura e de escrita, embora para situações bem distintas.

4.4.1 – Driver RS-232

O "Driver RS-232" foi desenvolvido para ser usado com a porta série do autómato. Essa aplicação deve ser alojada no computador onde se encontra ligado o equipamento. O *driver RS-232* permite que a aplicação de monitorização/controlo ser alojada num local remoto, dado que "abre uma porta" TCP no computador local, ficando esse à escuta dos pedidos de leitura e escrita.

Quando se inicia o *driver*, é-nos mostrada uma janela onde é possível configurar os vários pârametros da porta série, figura 4.9, como "Baud Rate", "Databits", "Device", "Parity" e "Stop Bits".

Figura 4.9 – Configuração da porta série.

Após a configuração da porta serie, a aplicação verifica os parâmetros, caso esses estejam correctos, a janela é fechada dando lugar a uma nova janela, figura 4.10. Essa serve para configurar a "porta" TCP de recepção do *driver*, por onde se irá estabelecer a comunicação com a aplicação de monitorização e controlo.

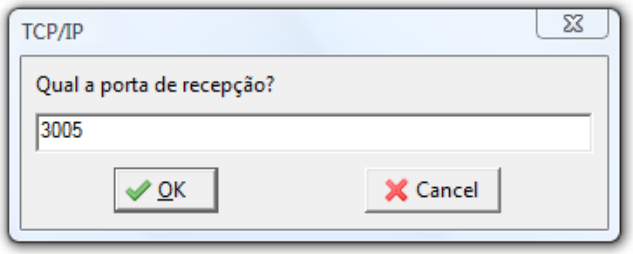


Figura 4.10 – Configuração da "porta" de recepção.

Após a configuração do *driver*, é aberta a janela de comunicação, figura 4.11, onde se dá o processamento das tramas de chegada e partida de ambas as comunicações.

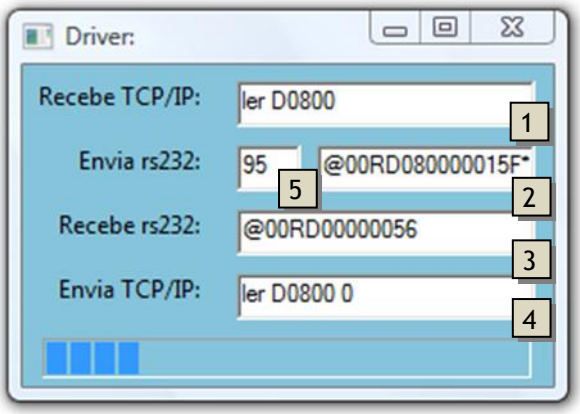


Figura 4.11 – Janela de comunicação do *driver RS-232*.

Na janela de comunicação é mostrada a transformação sucessiva das tramas, na qual, no primeiro campo (1), é-nos mostrada a trama de recepção vinda da aplicação de monitorização/controlo. Após a sua transformação, ela é enviada para o autómato (2). O autómato responde com uma trama (3), que por sua vez é transformada (4) e enviada à aplicação. Por vezes, as tramas que preenchem os vários campos não são perceptíveis. Isso acontece quando se está perante uma troca de dados a alta velocidade, entre a aplicação de monitorização/controlo e o autómato.

As transformações feitas às tramas de recepção variam consoante o tipo de pedido:

- **Pedido de leitura ao autómato (aplicação → autómato):**

  Quando o *driver* recebe um pedido de leitura constrói uma determinada trama que será enviada pela porta série (para o autómato). Esse tipo de trama é composta por 7 campos, tendo no total 17 caracteres, figura 4.12.

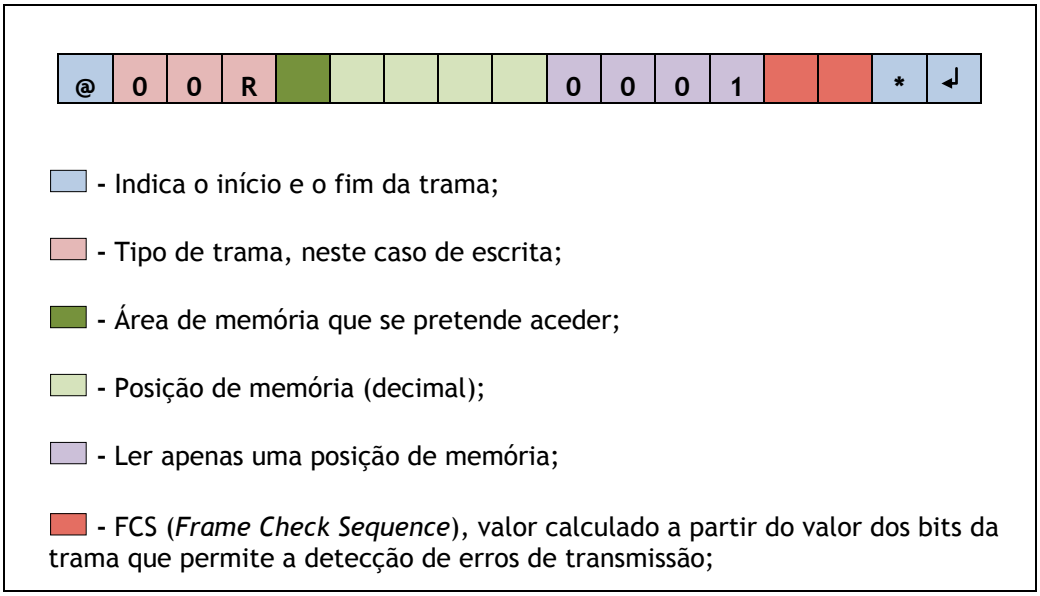


Figura 4.12 – Trama de pedido de leitura ao autómato.

Na recepção de uma trama, o autómato calcula o FCS, comparando-o com o FCS recebido na trama. No caso de serem diferentes, significa que houve um erro na transmissão (sendo por isso solicitado um novo pedido). O FCS (5) é calculado sempre da mesma forma e consiste numa soma aritmética (*XOR*) dos cincos primeiros campos da trama (13 primeiros caracteres). Após o cálculo do FCS, é colocado na trama o respectivo valor, construindo assim o 6º campo.

O autómato, após a recepção de uma trama deste tipo, responde com uma trama composta por 7 campos (15 caracteres), figura 4.14.

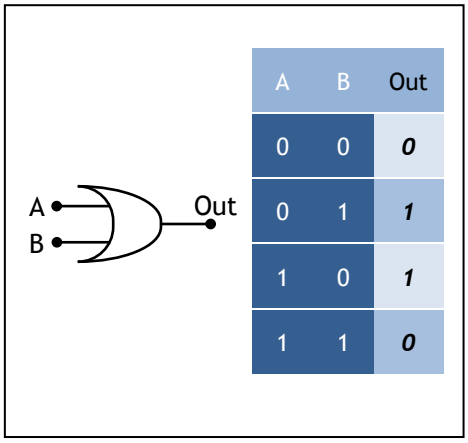


Figura 4.13 – Função lógica *XOR*

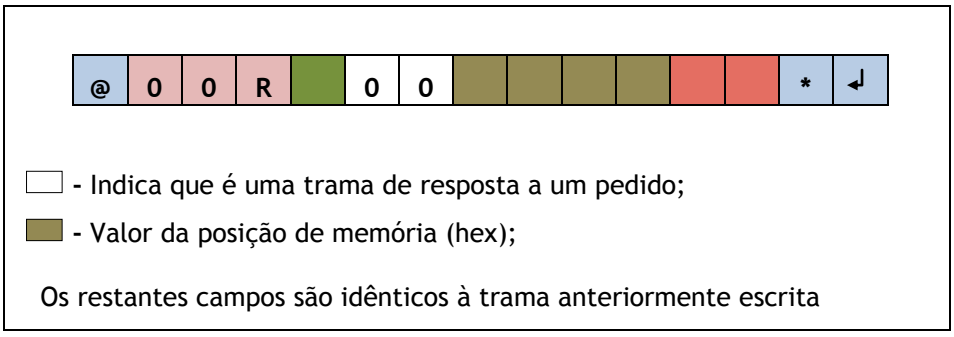


Figura 4.14 – Trama de resposta ao pedido de leitura.

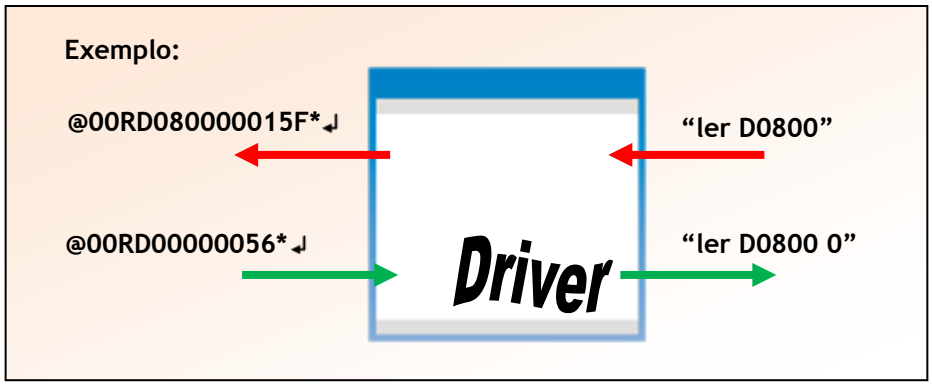


Figura 4.15 – Exemplo de um pedido de leitura ao driver RS-232.

- **Pedido de escrita ao autómato (aplicação → autómato):**

Para este tipo de pedido, o algoritmo implementado no *driver* é muito similar ao anterior, diferenciando-se apenas na construção das tramas de envio e recepção do autómato.

Quando o driver recebe este tipo de pedido, constrói uma trama de 8 campos, com um total de 21 caracteres, figura 4.16.

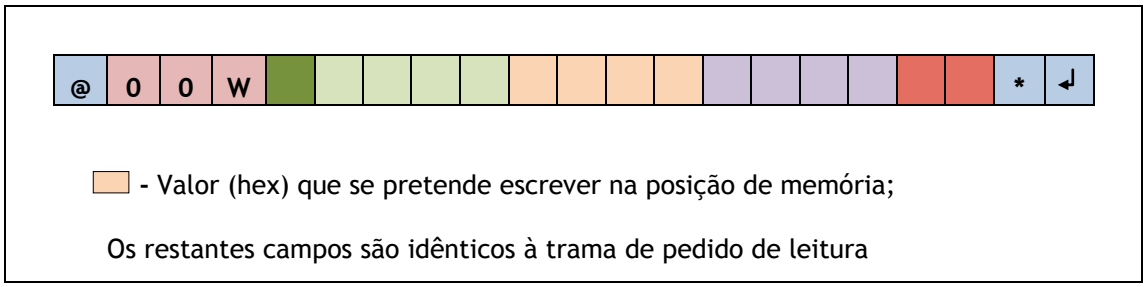


Figura 4.16 – Trama de pedido de escrita ao autómato.

O FCS é calculado da mesma forma, mas, neste tipo de trama, é calculado para os primeiros seis campos da trama.

Quando o autómato recebe este tipo de pedido, responde com uma trama semelhante à do pedido de leitura, sendo essa composta por apenas seis campos, figura 4.17.

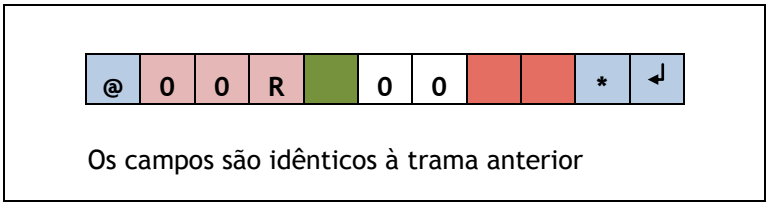


Figura 4.17 – Trama de resposta ao pedido de escrita.

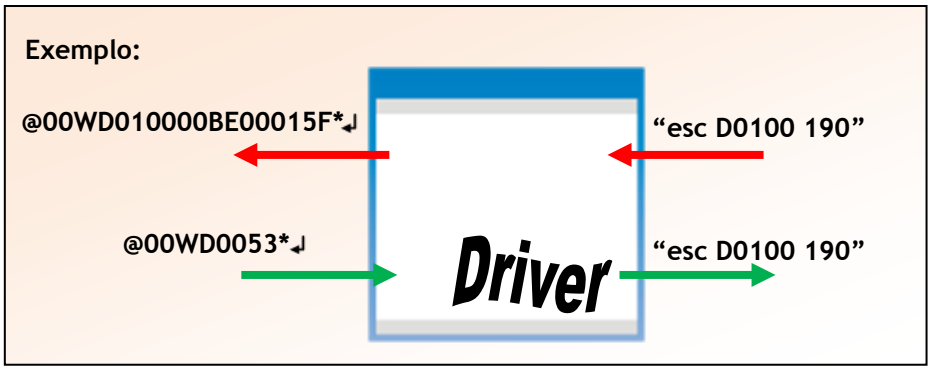


Figura 4.18 – Exemplo de um pedido de escrita ao driver RS-232.

4.4.2 – Driver Ethernet

O "Driver Ethernet" tem a mesma filosofia que o "Driver RS-232"; a comunicação entre o *driver* e o autómato é feita a partir da porta *Ethernet* e do protocolo ***FINS***. Este tipo de comunicação é vantajoso face ao anterior, pois a velocidade de comunicação é muito superior.

Quando o *driver* é inicializado, este abre uma janela de configuração, figura 4.19, onde é possível configurar os campos da comunicação *Ethernet*, como o IP (1) e a "porta" (2), processo através do qual se pode comunicar com o autómato, assim como a "porta" TCP de recepção do *driver* (3).

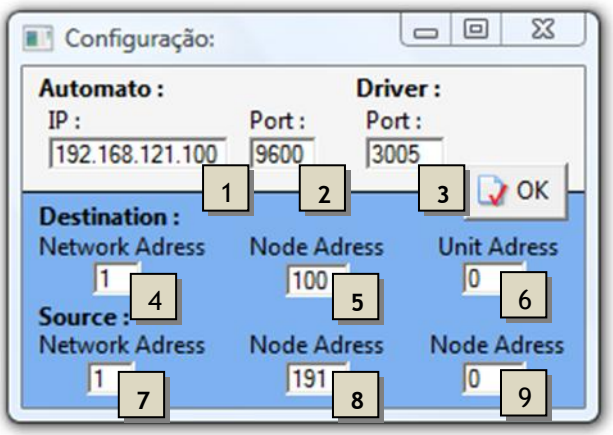


Figura 4.19 – Janela de configuração do *Driver Ethernet*.

Quando se usa a *Ethernet* para comunicar com o autómato, existem parâmetros que necessitam de configuração e que não estão presentes na comunicação RS-232, como a identificação de um autómato nessa rede.

Na janela de configuração, é possível configurar os parâmetros, necessários para o protocolo *FINS*.

- Destino (Autómato):

  Destination Network Address (**DNA**) (4) **-** Indica o endereço da rede onde está localizado o nó destino.

  Destination Node Address (**DA1**) (5) - Especifica o endereço do nó para onde o comando deverá a ser enviado. Este número é utilizado pelo *FINS* para distinguir as várias unidades que estejam ligadas à rede definida pelo **DNA.**

  Destination Unit Address (**DA2**) (6) - Especifica o endereço da unidade no nó de destino.

- Fonte *(Driver)*:

  Source Network Address (**SNA**) (7) **-** Indica o endereço da rede onde está localizado o nó fonte.

  Source Node Address (**SA1**) (8) - Especifica o endereço do nó local, sendo esse da mesma gama do **DA1.**

  Source Unit Address (**SA2**) (9) - Especifica o endereço da Unidade do nó fonte; a gama desse é a mesma do **DA2.**

Estes parâmetros constituem parte das tramas do protocolo, que são enviadas por UDP (*User Datagram Protocol*).

Após a configuração, é mostrada a janela de comunicação, figura 4.20, que permite visualizar as trocas de tramas, equivalente à janela de comunicação do *driver RS-232*.

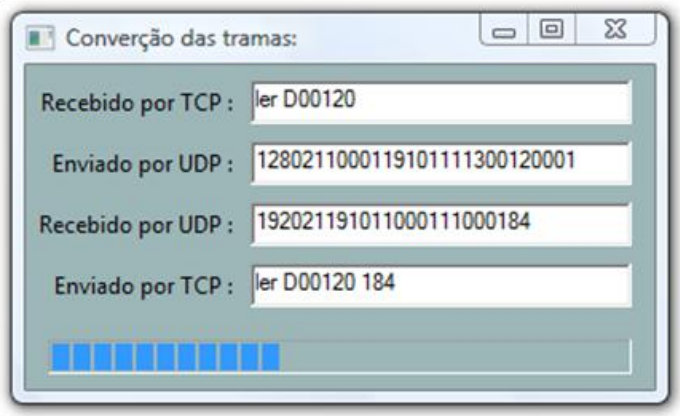
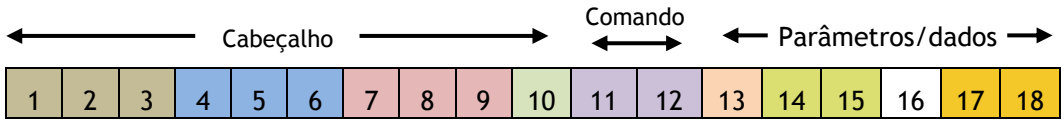


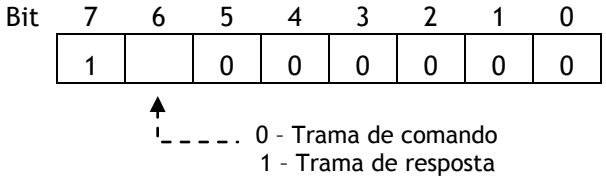
Figura 4.20 – Janela de comunicação do *Driver Ethernet*.

O *driver Ethernet* constrói as tramas *FINS* consoante o pedido:

- **Pedido de leitura:**



- (1) ICF (Information Control Fiel):



(2) RSV (Reserved by System): valor fixo zero;

(3) GCT (Permissible Number of Gateways): valor fixo 02 (hex)

- (4) DNA, (5) DA1, (6) DA2;

- (7) SNA, (8) SA1, (9) SA2;

- (10) SID (Service ID): é utilizado para identificar o processo, ou seja, defina-se qualquer número de 00 até FF, pois o mesmo número será devolvido na resposta, o quem permite combinar comandos e respostas.

- (11) MRC (Main request code) ;

(12) SRC (Sub-request code) : é utilizado para identificar o pedido, ou seja, caso se pretende um pedido de leitura esses deveram ser 1 1, caso seja um pedido de escrita esses deveram ser 1 2.

- (13) Identifica a área de memória que se pretende aceder, podendo ser:

Área:	CIO	HR	AR	DM	EM
Valor decimal	176	178	179	130	160

- Endereço de memória composto por dois bytes: (14) o mais significativo e (15) o menos significativo.

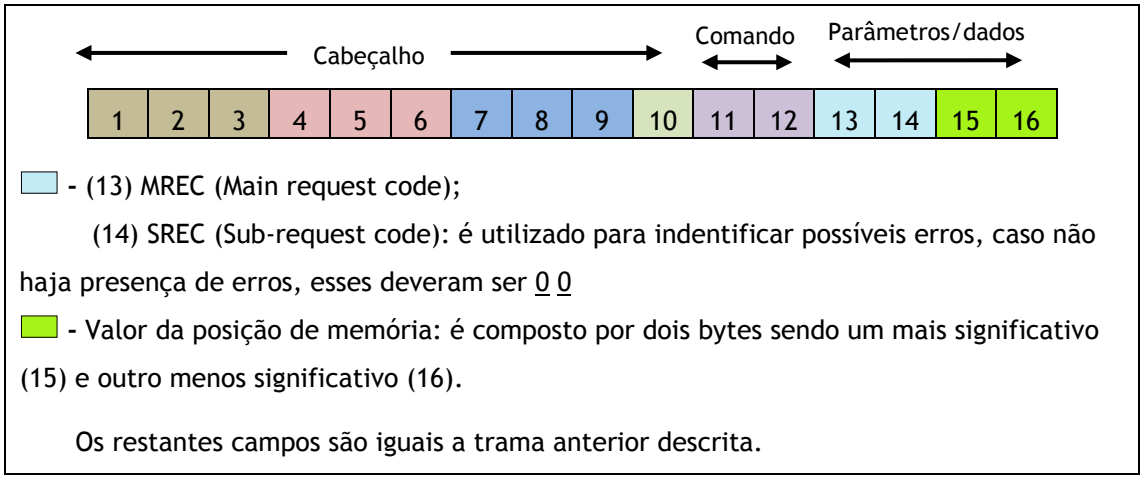
- (16) Tem o valor fixo de zero.

- Número de posições de memória que se pretende pedir, é composto por dois bytes, um mais significativo (17) e outro menos significativo (18). Neste driver estes bytes têm respectivamente o valor 0 e 1.

Esta trama é toda construída em hexadecimal.

Figura 4.21 – Estrutura da trama *FINS* para pedido de leitura.

A um pedido de leitura, o autómato responde com uma trama composta por 16 bytes, com o seguinte formato:



- (13) MREC (Main request code);

(14) SREC (Sub-request code): é utilizado para indentificar possíveis erros, caso não haja presença de erros, esses deveram ser 0 0

- Valor da posição de memória: é composto por dois bytes sendo um mais significativo (15) e outro menos significativo (16).

Os restantes campos são iguais a trama anterior descrita.

Figura 4.22 – Estrutura da trama *FINS* de resposta ao pedido de leitura.

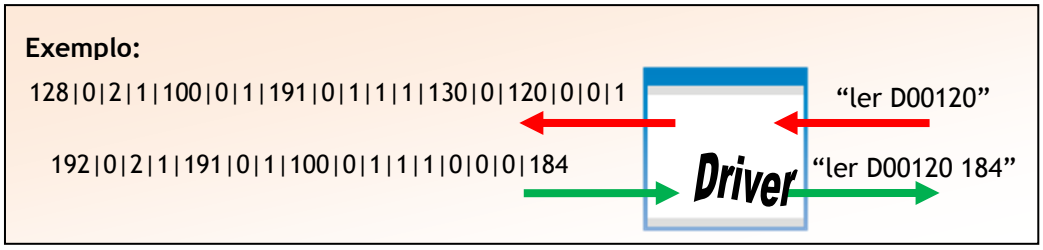


Figura 4.23 – Exemplo de um pedido de leitura ao *driver Ethernet*.

- **Pedido de escrita:**

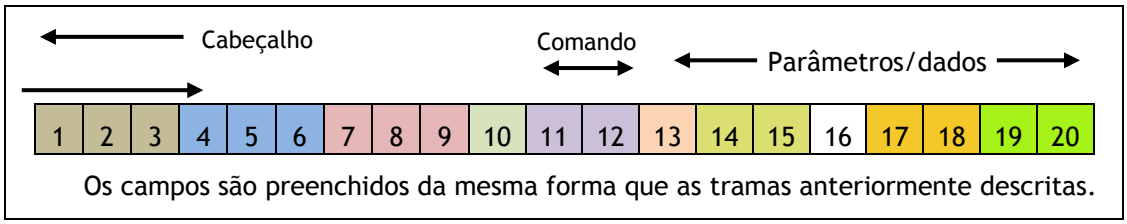


Figura 4.24 – Estrutura da trama *FINS* para pedido de escrita.

Com este pedido de escrita, o autómato responde com uma trama composta por 14 *bytes*, tendo o seguinte formato:

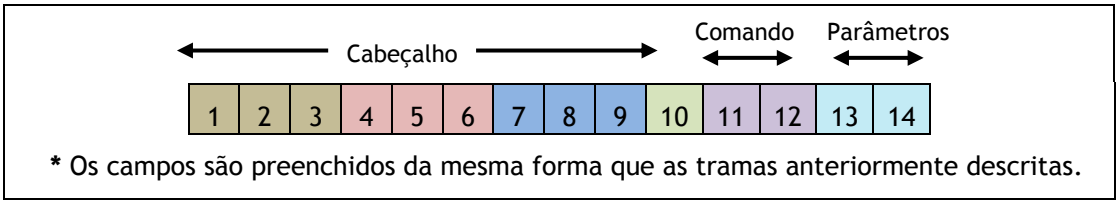


Figura 4.25 – Estrutura da trama *FINS* de resposta ao pedido de escrita.

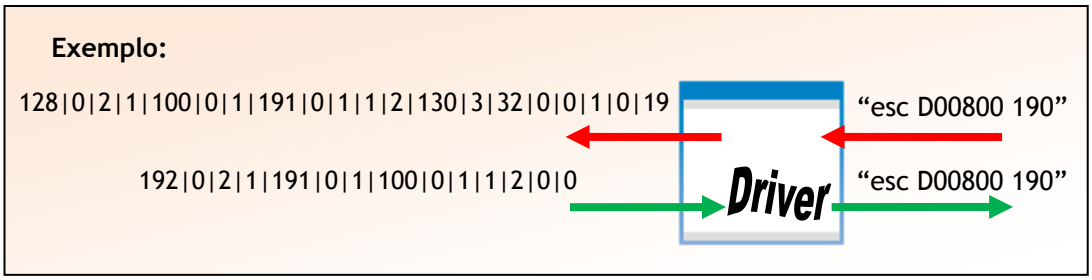


Figura 4.26 – Exemplo de um pedido de escrita ao *driver Ethernet*.

4.3.2 – Algoritmo de controlo

Em todas as aplicações deve haver um algoritmo de controlo que seja capaz de detectar/reparar possíveis erros e/ou problemas, esse algoritmo é importante para que uma aplicação seja viável e funcional.

Foi desenvolvido um algoritmo de detecção e reparação de tramas inválidas, seja por motivos de perda de dados durante a transferência ou porque foram enviados errados, nesta situação o *driver* ignora a trama, ficando a espera de uma trama valida. No entanto podem surgir outros problemas no que respeita ao envio de tramas por parte do *driver*, sendo essas tramas destinadas ao autómato ou à aplicação que fez o respectivo pedido. Caso a trama não chegue ao autómato, o driver nunca receberá a trama de resposta ao seu pedido, devendo repetir o pedido. O *driver*, consegue intervir nestes tipos de problemas, pois possui um *timer* que é inicializado à chegada de uma trama valida, esse não será reinicializado caso surge uns dos problemas referenciados anteriormente. Quando o *timer* chega ao fim da sua "contagem", verifica o estado em que se encontra agindo em conformidade:

- Envia pedido de leitura ao autómato;
- Envia pedido de escrita ao autómato;
- Envia resposta ao pedido de leitura da aplicação;
- Envia resposta ao pedido de escrita da aplicação.

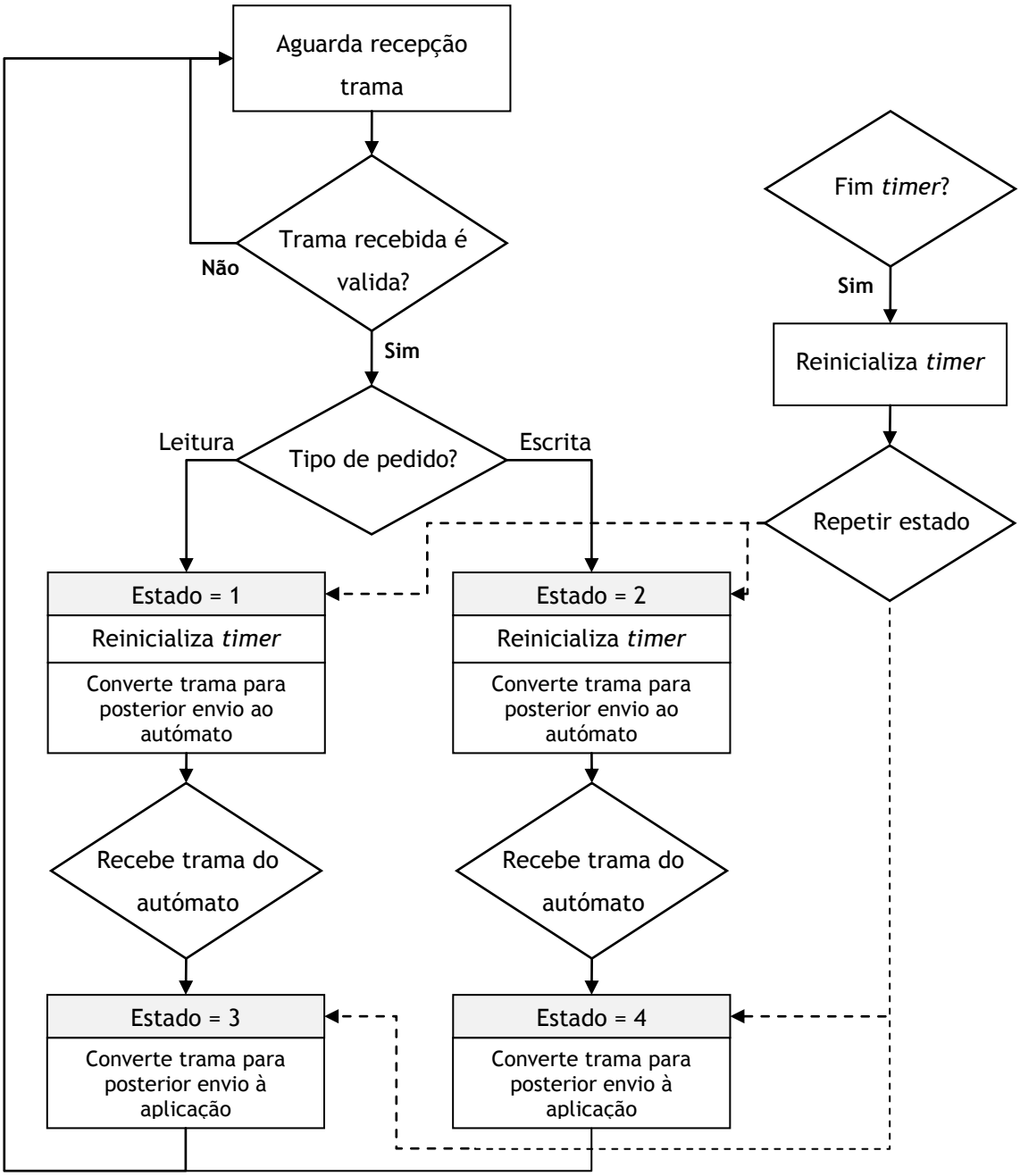


Figura 4.27 – Diagrama de blocos da estrutura de controlo.

Com este algoritmo o *driver* torna-se numa aplicação estável, consegue responder a inúmeros erros e/ou problemas, o que lhe permite assegurar a comunicação entre a aplicação de monitorização e controlo e o sistema de aquisição.

4.5 - Interface

Como já aqui foi referenciado, este projecto consiste essencialmente no desenvolvimento de aplicações capazes de fazer a monitorização e o controlo de um determinado sistema, empregando apenas ferramentas *freeware*. Para tal, foram desenvolvidas duas soluções, uma baseada em *Freepascal* e outra em *Java Applet*.

4.4.1 – Interface *FreePascal*

Foi desenvolvido uma interface em *FreePascal* (Lazarus) que permite ao utilizador monitorizar e controlar a experiência de ***Medição e Controlo de Nível***. Esta aplicação comunica com o sistema de aquisição (autómato) através de um *driver*, desenvolvido para o efeito. A interface foi desenvolvida de modo a apresentar uma interface "clara" e de fácil utilização, em que a informação fosse bem distribuída e a navegação intuitiva.

Quando a aplicação é iniciada, dá origem a uma janela, Figura 4.28, onde o utilizador configura a porta pela qual quer estabelecer a comunicação.

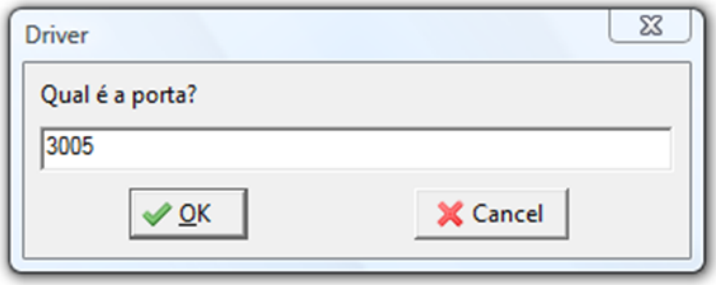


Figura 4.28 – Janela de configuração da porta.

Após a configuração da porta, surge uma segunda janela de configuração, Figura 4.29, na qual o utilizador escreve o IP da máquina onde está alojado o *driver* (RS-232 ou *Ethernet*).

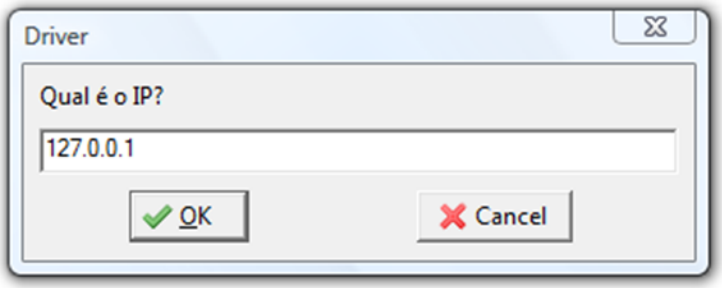


Figura 4.29 – Janela de configuração do IP.

Completadas as configurações da aplicação, é aberta a janela principal, Figura 4.30, onde o utilizador poderá monitorizar e controlar a experiência. Nesta janela é monitorizado o valor

dos transdutores e detectores, assim como a potência da bomba (visível apenas quando esta se encontra ligada).

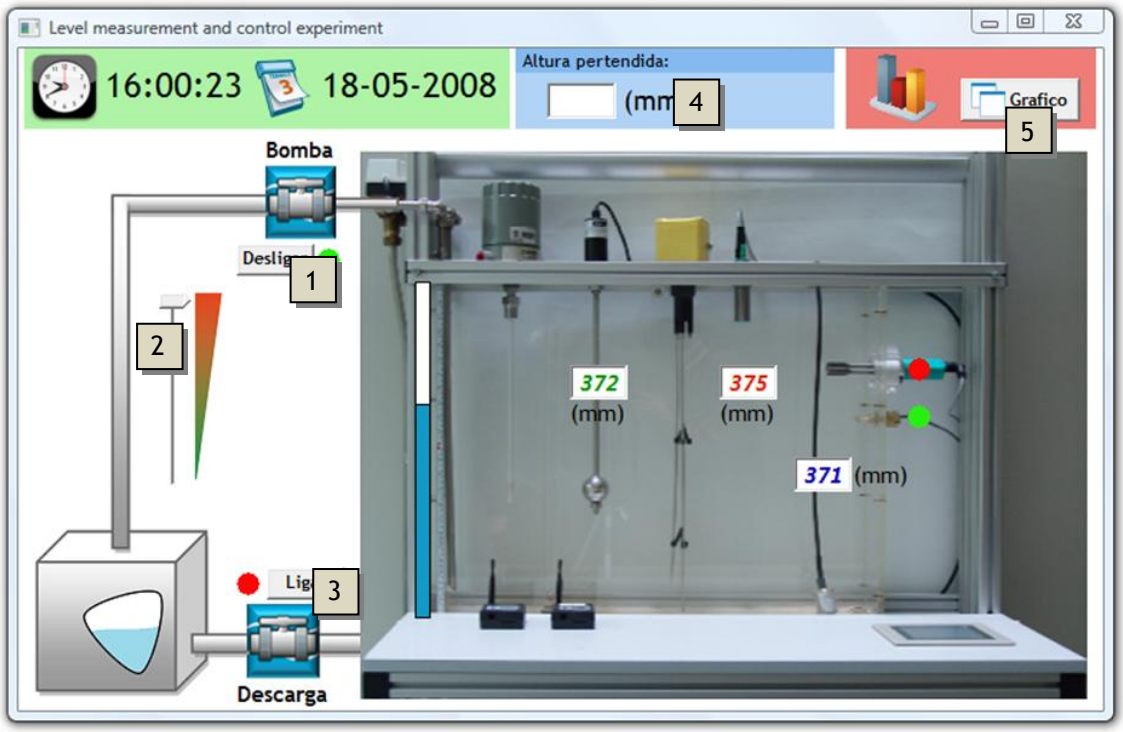


Figura 4.30 – Janela principal da interface *Lazarus*.

Esta interface permite ao utilizador ligar/desligar a bomba (1), controlar a potência desta através da "*Track Bar*" (2) (uma opção disponível somente quando a bomba se encontra ligada em modo manual). É ainda possível ligar/desligar a válvula de descarga (3), perceptível tanto nesta opção, como na anterior, através do indicador (🔴 ou 🟢), que permite informar o utilizador sobre o respectivo estado. Esta interface tem outras opções, como:

- Modo automático

  O utilizador pode ajustar o nível da água para um determinado valor, para tal deverá escrever o valor pretendido (4). Na eventualidade deste valor se encontrar fora da gama permitida, uma janela de informação é aberta, Figura 4.31, alertando para os limites definidos.

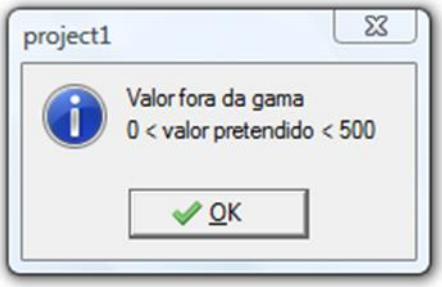


Figura 4.31 – Janela de informação.

Caso o valor esteja dentro da gama, surge um botão que permite ao utilizador dar início ao modo automático, abrindo uma nova janela onde é monitorizado o valor actual, assim como o desejado.

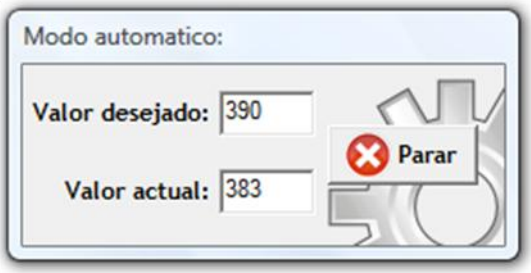


Figura 4.32 – Janela do modo automático.

- Histórico dos transdutores (5)

  O utilizador pode ver o histórico dos três transdutores e para tal deverá clicar no botão "Gráfico", onde lhe será mostrado uma nova janela com a evolução do nível da água medido pelos sensores.

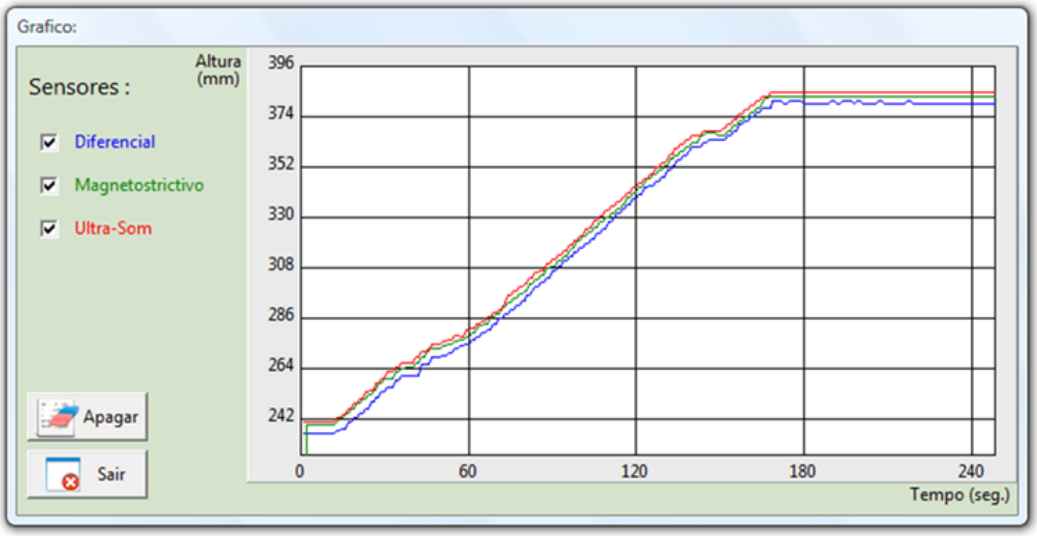


Figura 4.33 – Janela do histórico.

Nesta janela, é possível seleccionar os gráficos que se pretende observar, através das *Check Box*, como também reinicializar o gráfico através do botão "Apagar".

4.4.2 – Interface *Java Applet*

Esta metodologia desenvolvida a partir do *java applet* poderia utilizar um *driver* para estabelecer a comunicação com o autómato, ou então, comunicar directamente com esse. Apesar de ser uma boa solução, não seria de modo algum a melhor.

Quando é enviado um determinado "comando" a partir da interface (como por exemplo de abertura/fecho da válvula), para que o autómato actue na experiência, recebe na realidade um conjunto de tramas, Figura 4.34.

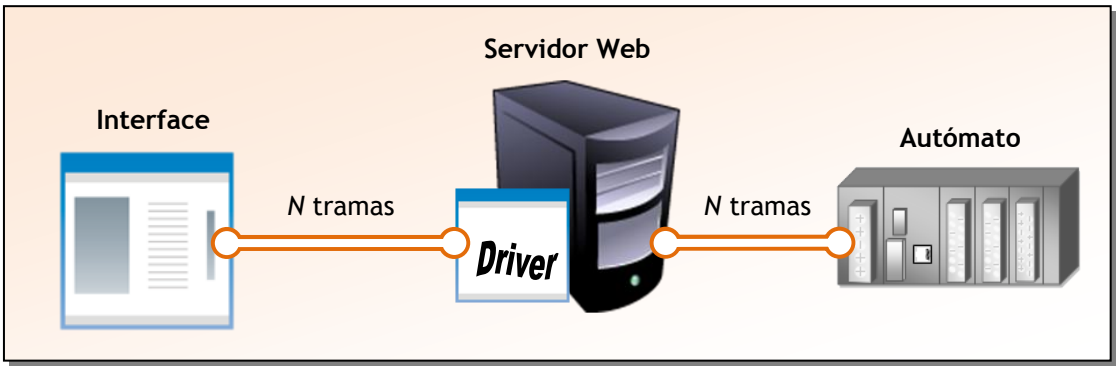


Figura 4.34 – Envio de uma ordem ao *driver*.

Apesar de ser um processo rápido, este torna-se ainda mais rápido caso o "comando" enviado pelo utilizador seja composto por apenas uma trama. Sendo interpretado por um programa (CGI) alojado no servidor, que por sua vez envia o respectivo conjunto de tramas ao autómato.

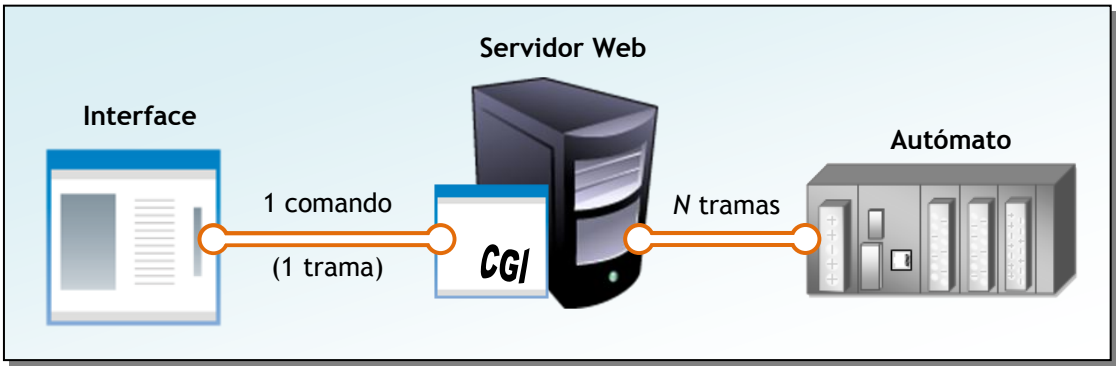


Figura 4.35 – Envio de uma ordem ao *CGI*.

Esta metodologia é vantajosa em dois aspectos principais: aumento da velocidade e melhoria da estabilidade da comunicação. A razão pela qual esta troca de informação tem tempos diferentes deriva do facto do *laboratório* e servidor Web se encontrarem geralmente

numa *intranet*, enquanto a interface (isto é o utilizador) estará em geral distante, ligado pela *internet*.

A estabilidade é melhorada, pois com esta metodologia, há um menor conjunto de tramas a "circular" na *internet*, reduzindo assim a probabilidade dos seus dados sofrerem erros ou serem perdidos.

Para implementar este conceito, foi desenvolvida uma aplicação *Lazarus* (CGI) que é alojada no servidor Web (poderá ser alojado noutro computador) que permite decifrar os "comandos" enviados pela interface (*java applet*) para posterior envio ao autómato.

Esta aplicação quando inicializada exibe uma janela de configuração, onde é possível configurar os vários parâmetros de comunicação com o autómato (protocolo FINS), assim como a porta de recepção da aplicação, Figura 4.36. Esta janela de configuração tem as mesmas propriedades que a janela de configuração do *Driver Ethernet*.

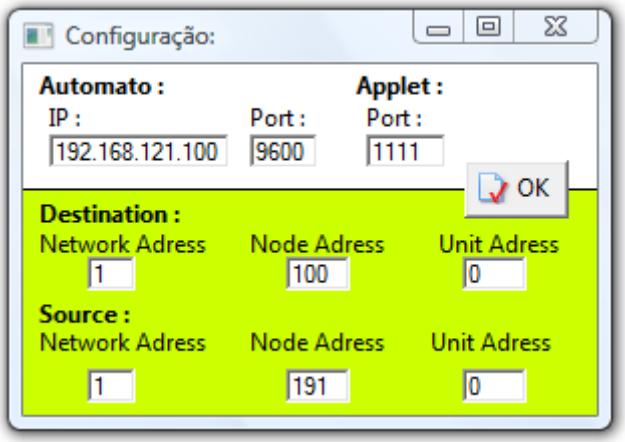


Figura 4.36 – Janela de configuração do "*CGI*".

Após a configuração da comunicação, é iniciada a aplicação, Figura 4.37, onde é possível visualizar as variáveis da experiência, como o valor dos sensores, estado dos detectores e actuadores. Existem nesta janela duas barras de progresso, que indicam a troca de dados entre a aplicação e o autómato, assim como a troca de dados entre a aplicação e o *Java Applet*. A aplicação não permite uma interacção directa com a experiência, ou seja, esta interage com a experiência caso receba um determinado comando (enviados pelo aplicação *Java Applet*).

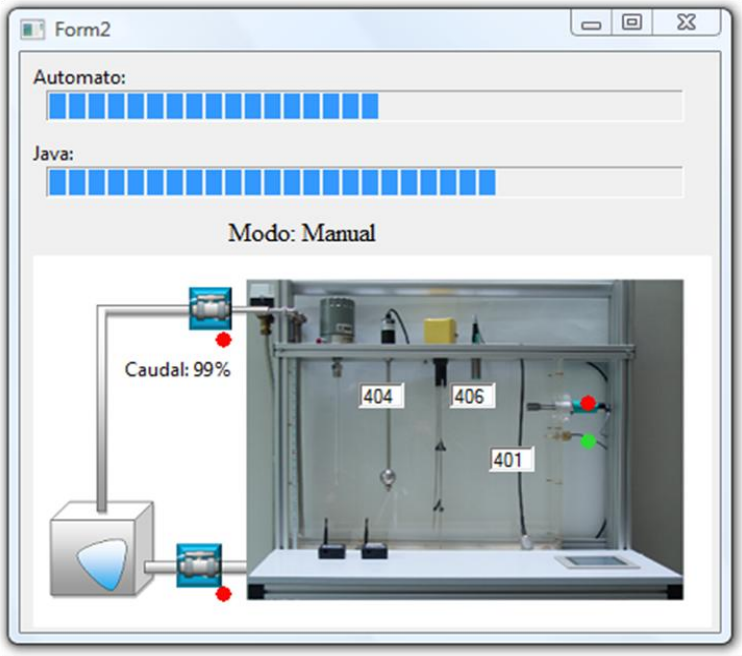


Figura 4.37 – Janela principal do "*CGI*".

Esta aplicação é constituída por um programa principal (processo de monitorização e controlo) e três algoritmos que apoiam este programa na detecção de anomalias, garantindo a estabilidade da aplicação, figura 4.38. Estes algoritmos permitem à aplicação responder a erros/problemas como a perda parcial/total dos dados trocados entre o autómato e a aplicação, assim como os dados trocados entre a interface (*java applet*) e a aplicação.

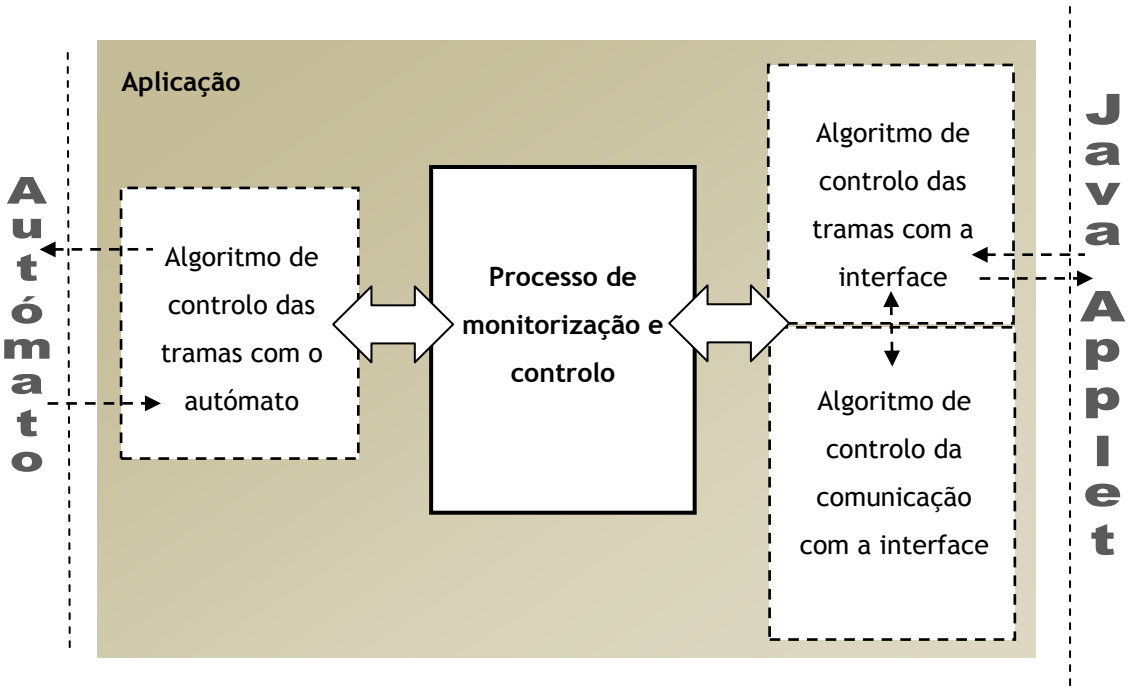


Figura 4.38 – Estrutura da aplicação.

Após a configuração, a aplicação aguarda pela recepção de um comando que lhe indica o começo do processo de monitorização e controlo, que só será finalizado quando receber na sua entrada o respectivo comando de finalização, figura 4.39.

Quando o processo de monitorização e controlo se encontra a decorrer, envia vários pedidos de leitura ao autómato, possibilitando a leitura das variáveis da experiência, que são guardadas na aplicação. Em paralelo a este "ciclo de leitura", a aplicação responde aos pedidos de leitura vindos do *java applet*, este processo é interrompido caso a aplicação receba um pedido de "acção" ou "finalização". Com a chegada de um pedido de "acção", a aplicação aguarda pela resposta do seu último pedido feito ao autómato para dar início ao ciclo de "acção", que consiste num conjunto de tramas que permite satisfazer o respectivo pedido. Terminado o ciclo de "acção", a aplicação retoma o ciclo de "leitura" e envia para o *java applet* um comando que indica **o existo da** "acção". Com a presença de um comando de paragem (enviado pelo *java applet* quando é encerrado), a aplicação aguarda a resposta do pedido de leitura ou escrita feita ao autómato, para dar início ao ciclo de "finalização". Este ciclo é composto por duas fases, uma primeira onde a experiência é reiniciada e uma segunda onde é feita a conexão/desconexão da "porta" da aplicação. A primeira fase consiste em desactivar os elementos actuadores do sistema caso se encontram activos, como fechar a válvula de escoamento e desligar a bomba, assim como desactivar o modo automático (caso se encontre ligado). Este ciclo é muito importante, visto que um utilizador pode fechar a página de interface esquecendo-se de desligar os elementos referidos.

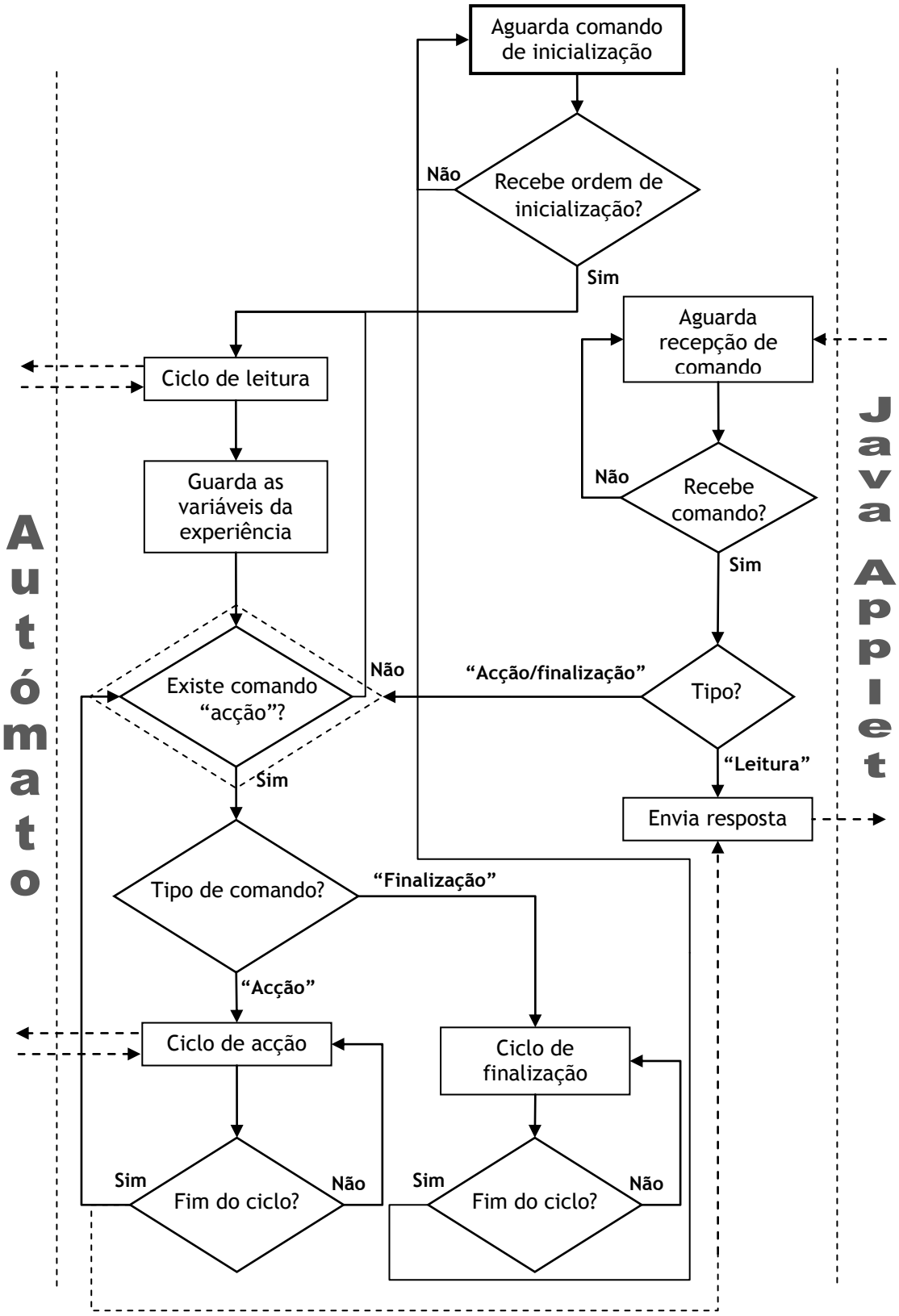


Figura 4.39 – Princípio de funcionamento do programa principal.

Os três algoritmos que apoiam este programa principal, são:

- Controlo das tramas do autómato

  Este algoritmo tem como finalidade garantir a correcta troca de dados entre a aplicação e o autómato, figura 4.40.

  Quando a aplicação envia uma determinada trama ao autómato é iniciado um *timer*, na recepção de uma trama (enviada pelo autómato) é feito a paragem deste. Quando o timer está activo ("fim *timer*"), significa que a aplicação não recebeu a resposta do pedido feito ao autómato, devendo por isso repeti-lo.

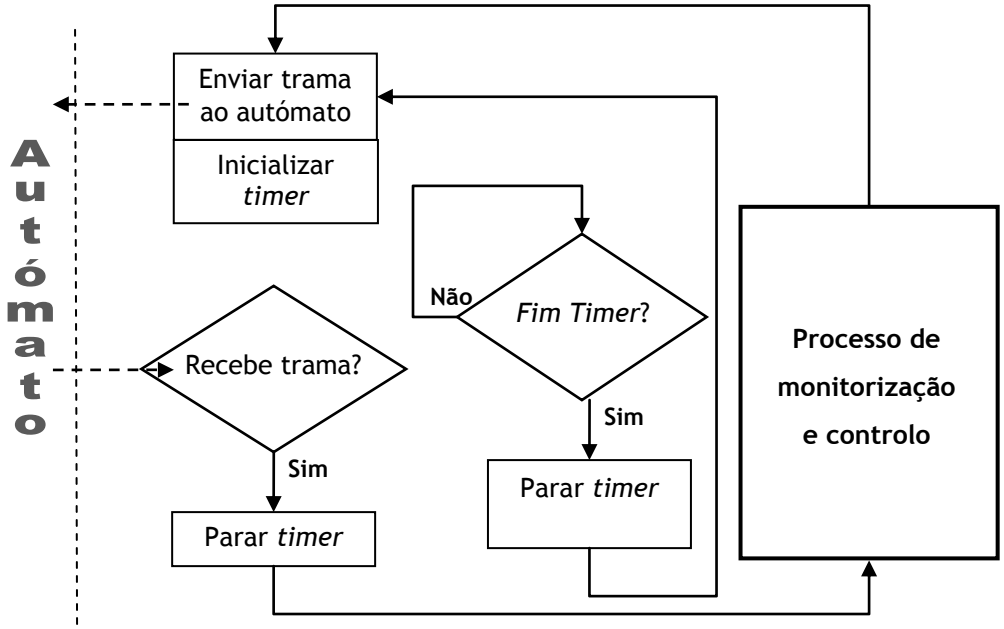


Figura 4.40 – Algoritmo de controlo da troca de tramas com o autómato.

- Controlo das tramas da interface

  Este algoritmo tem como funcionalidade garantir a troca de dados entre a interface e a aplicação, figura 4.41.

  Quando a aplicação recebe um determinado comando da interface, pára o *timer*, que é iniciado na recepção de um comando. Quando o *timer* se encontra activo, implica que a interface não recebeu o comando enviado, pelo que é reenviado o comando.

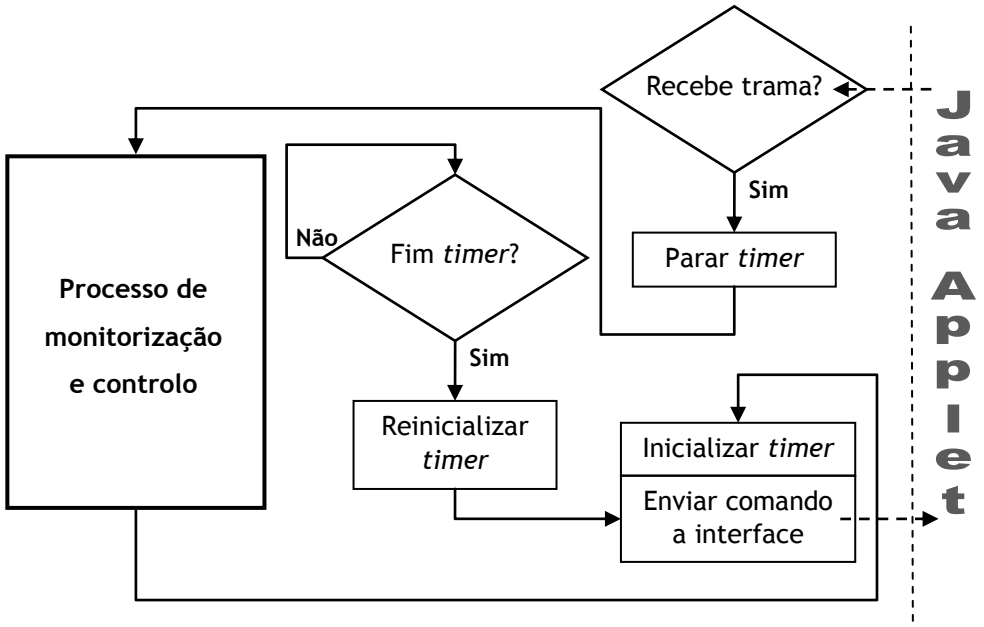


Figura 4.41 – Algoritmo de controlo da troca de tramas com a interface.

- Controlo da ligação com a interface

  Este algoritmo é primordial para esta aplicação, visto que a aplicação só termina o processo de monitorização e controlo caso receba na sua entrada o respectivo comando de finalização. O comando é enviado quando é encerrada a interface, caso este não chegue ao destino (servidor Web) a aplicação não termina o processo de monitorização e controlo. Dois problemas surgem neste caso, primeiro a aplicação não porá fim a experiência, o que é um inconveniente caso o utilizador deixe a bomba ligada e/ou a válvula de escoamento aberta. O segundo problema, reside na conexão entre a aplicação e a interface (associada à máquina do utilizador). Caso o comando de "finalização" não chegue à aplicação esta guardará a última configuração da conexão, impedindo assim que outros utilizadores usufruem da interface. Estes são os factores que mostram a importância da realização do ciclo de "finalização", para tal foi desenvolvido este algoritmo que garante a execução desse ciclo.

  Tendo em conta que a aplicação é de monitorização, ou seja, existe uma troca constante de dados entre a interface e a aplicação, a aplicação "assume" que caso não receba uma trama durante um determinado período de tempo, é porque o utilizador encerrou a interface (não chegando o comando de finalização) ou perdeu a conexão, executando o "ciclo de finalização". Este algoritmo, figura 4.42, tem um funcionamento similar ao anterior. Na recepção de um comando (enviado pela interface) o *timer* é reiniciado, caso este fique activo (fim do *timer*), é desligado o *timer* e iniciado o ciclo de "finalização".

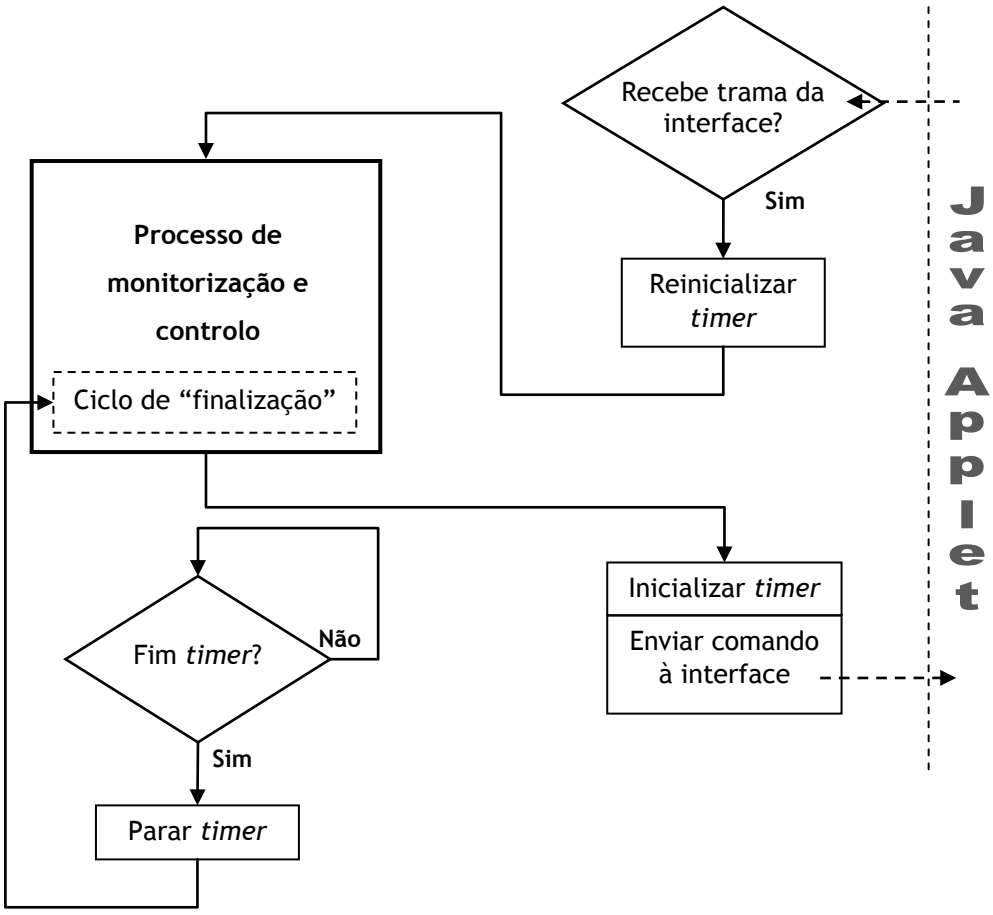


Figura 4.42 – Algoritmo de controlo da comunicação com a interface.

É importante realçar que este *timer* é distinto do *timer* que rege o algoritmo anteriormente descrito, este é um *timer* mais extenso, permite assim a aplicação repetir o envio do comando e assegurar-se que o utilizador já não se encontra *on-line*.

Estes são os algoritmos que estão integrados na aplicação, permitindo obter uma grande estabilidade e robustez.

A interface *java applet* também é dotada de um algoritmo de controlo, que tem um funcionamento similar ao algoritmo de controlo das tramas do autómato. A aplicação inicia um *timer* no envio de um comando, sendo este finalizado na recepção da resposta.

Para aceder à interface basta ao utilizador abrir um *browser* colocando o respectivo endereço, onde lhe será mostrado uma página Web (html), composta pela interface *Java Applet* e o vídeo em tempo real proveniente de uma câmara IP, figura 4.43.

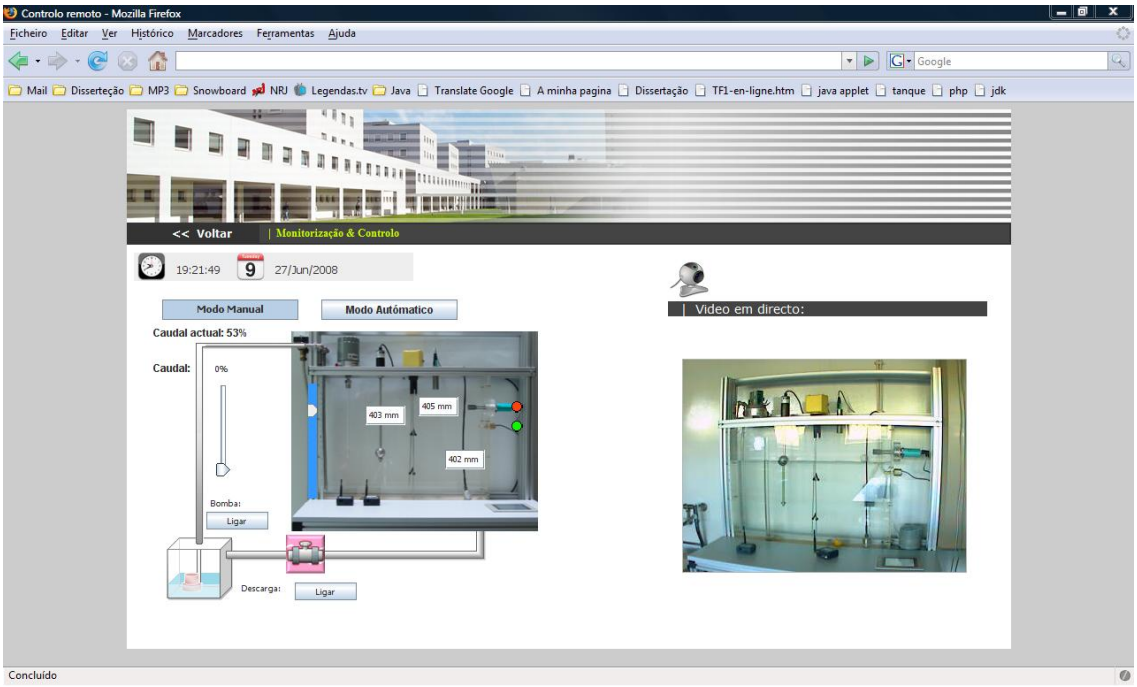


Figura 4.43 – Página Web da interface Java Applet.

A interface é constituída por dois "modos":

- Enchimento manual

  Este é por omissão o painel escolhido quando é feito o *download* da interface, figura 4.44, neste modo, o utilizador actua na experiência de forma manual, ou seja, tem a liberdade de ligar/desligar a bomba (6), assim como variar o seu caudal a partir do *"Slider"* (7) e pode ainda ligar/desligar a válvula de descarga (8). Na interface é sempre feita a monitorização dos valores dos vários transdutores e detectores: transdutor magnetostrictivo (1), transdutor ultra-som (2), transmissor de pressão diferencial (3), detector ressonante (4) e detector infra-vermelho (5). Para os transdutores e o transmissor são apresentados os respectivos valores na interface, para os detectores é mostrado um círculo, com cor verde caso se encontra activo e vermelho caso contrário.

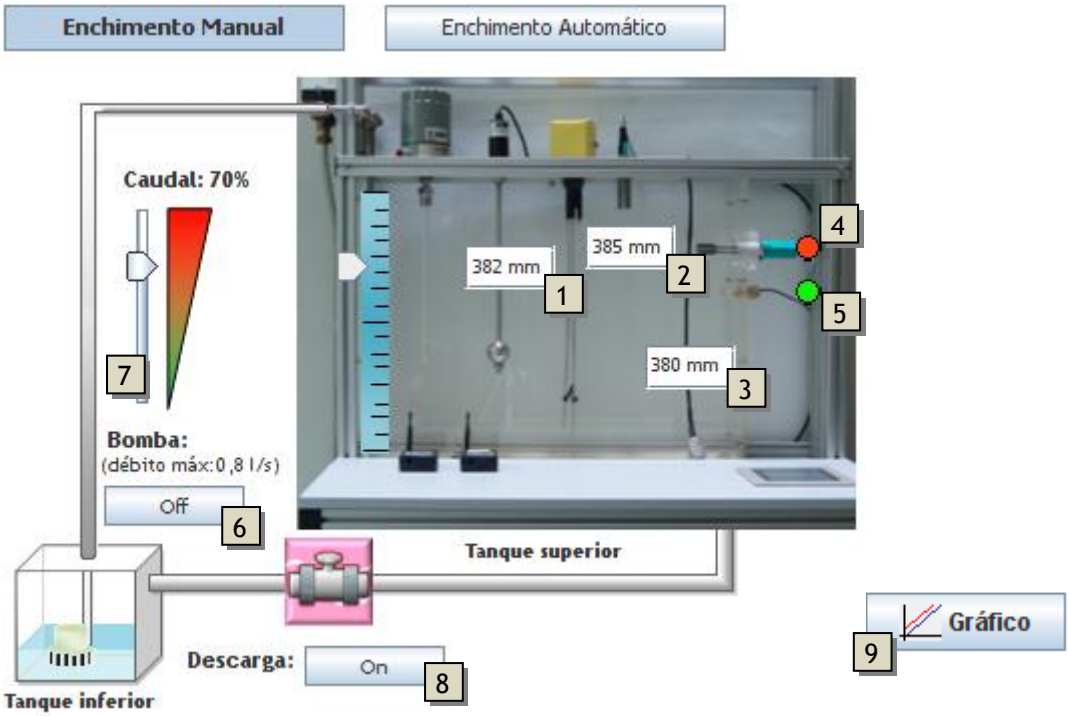


Figura 4.44 – Enchimento manual.

É possível ainda visualizar o histórico dos três transdutores, figura 4.45, para tal deve-se seleccionar o botão "Gráfico" (9). Neste gráfico o utilizador pode ampliar uma determinada área, imprimir o gráfico, assim como alterar o seu aspecto.

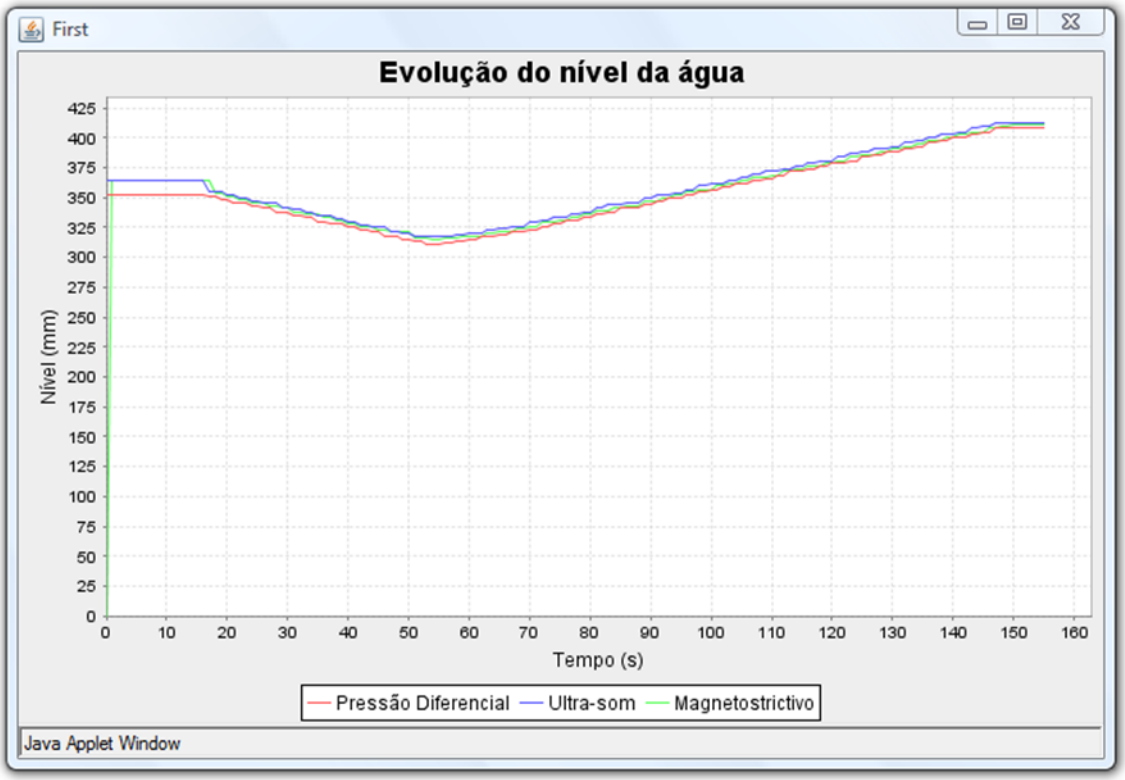


Figura 4.45 – Gráfico da evolução do nível.

- Enchimento automático

  Este painel, figura 4.46, permite ao utilizador escolher um valor para o qual pretende ajustar o nível da água, sendo este ajuste feito de forma automático. O valor para o qual se pretende ajustar o nível da água (2) é escolhido a partir do *"slider"* (1), este é validado a partir do botão "validar" (3). O ajuste automático pode ser interrompido pressionando o botão "parar" (3). Caso o utilizador troque para o painel de manual, o ajusto automático é encerrado (caso se encontre ligado).

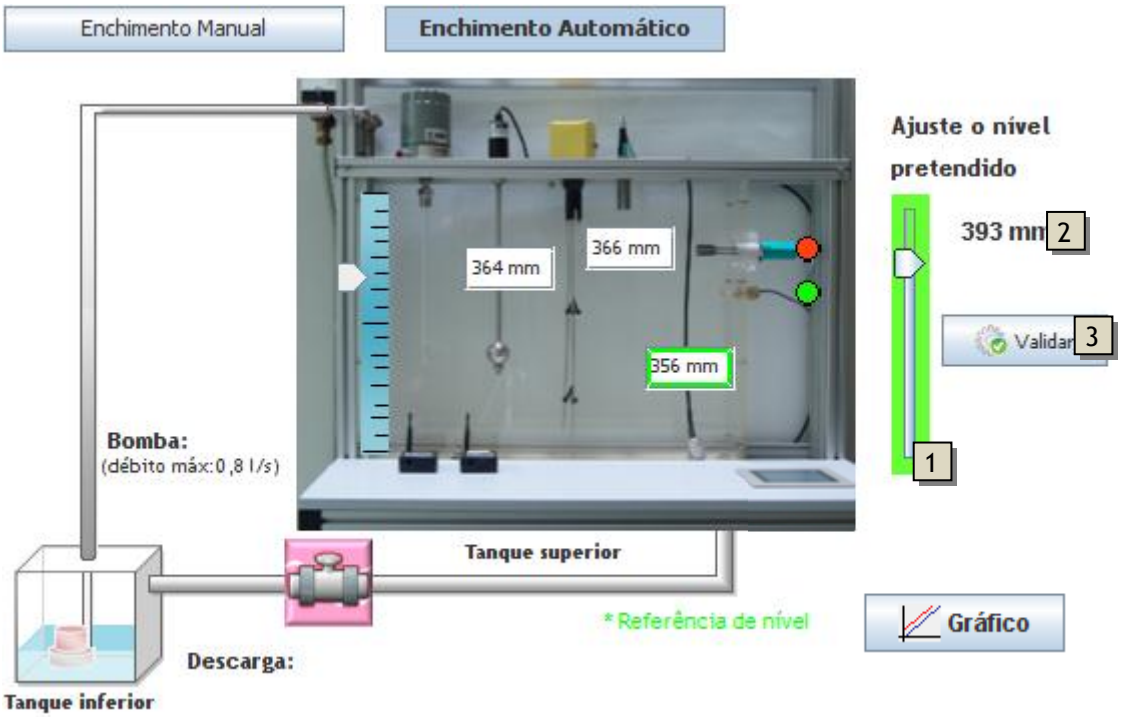


Figura 4.46 – Enchimento automático.

4.4.3 – Vantagens/desvantagens das interfaces Lazarus e Java

O principal problema da interface *Lazarus*, reside na sua dimensão, pois a aplicação poderá atingir uma dezena de mega bytes, um inconveniente caso o utilizador se encontre fora da rede local (internet) dado que o processo de transferência da interface será demoroso. A interface *java applet* consegue superar este inconveniente, visto que a dimensão da interface se resume a uma centena de kilobytes. Contudo, se os utilizadores se encontrarem apenas dentro da rede local (intranet), visto que a transferência é um processo muito rápido, ambas as interfaces parecem ser óptimas soluções. No entanto, com a interface *Lazarus*, se cada utilizador/computador possuir a aplicação (executável) o servidor Web será dispensável, mas este caso não se aplica nos laboratórios remotos. No entanto, para

ambientes industriais em que o contacto com o processo poderá ser necessário apenas dentro da rede local, será uma solução interessante.

Das metodologias desenvolvidas neste projecto, o *java applet* revelou-se uma óptima escolha, permitindo satisfazer os vários requisitos/objectivos pretendidos para uma interface de monitorização e controlo "leve".

Capítulo 5

Conclusões

Este projecto propõe uma nova metodologia para a implementação de laboratórios remotos que permite uma elevada robustez e fiabilidade, reduzindo simultaneamente os custos de hardware/software e de manutenção.

Os aspectos mais relevantes relativos à metodologia desenvolvida são:

- software de monitorização e controlo com detecção/correcção de eventuais erros/problemas que surjam na comunicação;
- comunicação optimizada para uma gestão eficiente do tráfego gerado;
- solução baseada em ferramentas *freeware*, factor relevante para a redução de custos de implementação.

Esta metodologia foi usada para criar um modo alternativo de acesso à experiência "Medição e controlo de nível", disponível no Laboratório de Instrumentação para Medição, permitindo algumas melhorias:

- com a arquitectura implementada, o autómato comunica directamente com o servidor Web, conduzindo a uma redução de custos, consumo energético e espaço ocupado. Assim, o computador local (junto da experiência) deixou de ser necessário;
- a comunicação entre o servidor Web e o autómato foi melhorada, permitindo uma comunicação directa, pois deixou de ser necessário o computador local e a conversão série - Ethernet. Acresce ainda que o servidor Web ao comunicar directamente com o autómato por Ethernet permite uma taxa de transferência de dados mais elevada do que por interface série;
- independentemente das aplicações de monitorização e controlo, a comunicação Ethernet (protocolo *FINS*) do autómato, torna a comunicação mais robusta e estável, face à comunicação série (utilizada anteriormente);

- todas as aplicações de monitorização e controlo que se encontravam desenvolvidas em LabVIEW foram substituídas por aplicações equivalentes desenvolvidas com ferramentas *freeware*, reduzindo assim o custo da solução;
- a aplicação de monitorização e controlo implementada inclui um extenso conjunto de verificações de erros de modo a garantir a robustez da aplicação.
- outro aspecto importante na utilização de uma aplicação remota reside no tempo de transferência da interface para o computador do utilizador. Neste ponto a solução proposta reduz o tamanho do ficheiro de 10 vezes.

A solução implementada mostrou ser robusta e eficiente, permitindo um conjunto importante de mais-valias em relação à solução anterior.

A interface da aplicação foi testada por um conjunto de utilizadores (professores e alunos) que sugeriram pequenas alterações, que permitiram torná-la mais agradável e intuitiva.

Como trabalhos futuros sugere-se a integração no *Moodle* e num sistema de agendamento (booking) automático que permita reservar a experiência. Isto é necessário dado que em cada instante só é possível a interacção da experiência com um único utilizador.

Referências

[1] Maria Teresa Restivo, Fernando Gomes de Almeida, Maria de Fátima Chouzal, Joaquim Mendes e António Mendes Lopes, Laboratorios de Instrumentação para Medição/Instrumentation for Measurement Laboratories, bilingual edition, Editora UPorto, ISBN: 978-972-8025-67-0, March 2008

[2] Placa de aquisição LabJack UE9. Disponível em http://www.labjack.com/labjack_ue9 .php?prodId=56 . Acesso em Fevereiro/2008.

[3] Placa de aquisição Web Server – WS10. Disponível em http://www.novus.com.br/ artigosnoticias/arquivos/folheto%20webserver.pdf. Acesso em Fevereiro/2008.

[4] Interface I/O da Schneider Electric. Disponível em http://www.schneiderelectric.pt/ index.htm. Acesso em Fevereiro/2008.

[5] Joaquim Gabriel Mendes, Maria Teresa Restivo, Jorge Reis, Controlo Remoto de um Porshe em Modelo Reduzido, National Instruments Corporation, Outubro 2003, http://digital.ni.com/worldwide/portugal.nsf/web/all/D3F72DBFCAFC709280256D95002EB8EE?OpenDocument&node=174321_pt.

[6] Maria Teresa Restivo, Isabel Carvalho, Joaquim Mendes, Ricardo Magalhães, Gyula Gróf, Remote visiting academic Labs using ICTs, Proceedings of SEFI 34 th Annual Conference, Session 4, pp. 64, Upsala, Sweden, 28th june/1stJuly, 2006.

[7] Maria Teresa Restivo, Jaime Villate, Fernando Almeida, Fátima Chouzal, Ricardo Magalhães and Isabel Menezes, The Michelson Interferometer: a learning aoject, CD-Rom Proceeding of the International Conference on Remote Engineering and Virtual Instrumentation – REV2007, 24-26 June 2007, FEUP, Portugal.

[8] Manuais Omron. Disponíveis em http://industrial.omron.eu/dlc/browse.do;jsessionid=E18B9D7E729C89C8ADC6A519950D9A3E?path=IAB%2FProducts.

[9] Lazarus. Disponível em http://www.lazarus.freepascal.org/

[10] Tutoriais de HTML e PGP. Disponível em http://www.w3schools.com/.

[11] Java Applet. Disponível em http://java.sun.com/developer/onlineTraining/ e http://www.realapplets.com/tutorial/.