DISSERTAÇÃO

Mestrado em Engenharia Electrotécnica

Modelação e análise de um sistema flexível de produção

João Pedro Ferreira Gonçalves

Leiria, Abril de 2014

DISSERTAÇÃO

Mestrado em Engenharia Electrotécnica

Energia e Automação

Modelação e análise de um sistema flexível de produção

João Pedro Ferreira Gonçalves

Dissertação de Mestrado realizada sob a orientação do Professor Doutor Hugo Filipe Costelha de Castro e do Professor Doutor Carlos Fernando Couceiro de Sousa Neves, ambos professores da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria.

Leiria, Abril de 2014

Dedico esta dissertação a quem
me trouxe ao Mundo...
Não é por um motivo especial,
mas...
é por TUDO! Amo-te Mãe.
(João Pedro Gonçalves)

Agradecimentos

Quero agradecer aos meus orientadores, o Professor Doutor Hugo Costelha e Professor Doutor Carlos Neves que sempre me ajudaram e me encaminharam da melhor forma possível ao longo de todo o desenvolvimento. Despertaram em mim o espírito crítico que tornou possível a concretização desta dissertação.

Aproveito esta oportunidade para enaltecer todos os amigos que me ajudaram a concluir esta importante etapa da minha vida, com especial ênfase para os meus pais e irmã, pois foram eles que me proporcionaram todas as condições ao longo da minha vida académica.

Por último, um agradecimento à minha melhor amiga Nadine Castanheira, pois nunca me deixou perder a esperança e sempre me apoiou e incentivou.

Resumo

Esta dissertação insere-se na temática de sistemas flexíveis de produção. O objetivo principal é o desenvolvimento e integração de um sistema SCADA que, interligado a uma aplicação baseada em Redes de Petri, permite supervisionar e controlar o sistema CIM existente no laboratório de robótica da ESTG/IPL. Neste trabalho o nome dado ao sistema desenvolvido é SCADANet. Consegue-se desta forma aliar a funcionalidade de supervisão e monitorização dos sistemas SCADA, com a capacidade de modelação e análise das Redes de Petri, como sejam a previsão de bloqueios e a segurança, por exemplo.

As várias componentes do sistema interagem entre si através de uma base de dados centralizada, sendo as comunicações com esta realizadas via TCP/IP sobre ethernet. O sistema SCADANet é utilizado para a monitorização e visualização gráfica do estado do sistema. A rede de Petri pode também ser utilizada para monitorizar graficamente o estado do sistema, mas o seu papel principal consiste no controlo do processo e na sua utilização para análise *offline* do mesmo.

Para comprovar o funcionamento do sistema desenvolvido, foram realizados vários testes em simulação e com equipamento real, consistindo este num processo com um conjunto de células de produção numa escala académica.

Palavras-chave: Redes de Petri, SCADA, Linha de Produção, Sistemas Flexíveis de Produção, CIM.

Abstract

This thesis develops on the theme of flexible production systems. The main purpose is the development and integration of a SCADA system that, connected to an application based on Petri nets, is able to interact with the CIM system in the laboratory of robotics at ESTG. The name given to the system developed is SCADANet. In this way it is possible to combine the supervision and monitoring functionality of the SCADA system with the ability of modeling and analysis of Petri Nets, such as forecasting, deadlocks and security, for example.

The various components of the system communicate with each other through a centralized database, using TCP/IP protocol over ethernet. The SCADANet system is used to monitor and display the system state. The Petri net can also be used to monitor graphically the state of the system, but its main role is to control the process and be used for offline analysis.

To demonstrate the operation of the system developed, several tests were performed in simulation and with real equipment, consisting of a process with a set of production cells in an academic scale.

Keywords: Petri Nets, SCADA, Production Line, Flexible Manufacturing Systems, CIM

Índice

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

RdP Rede de Petri

PIPE *Plataform Independent Petri Nets Editor*

SCADA Sistemas de supervisão e aquisição de dados (*Supervisory Control And Data Acquisition*)

FMS Sistema flexível de fabrico (*Flexible Manufacturing System*)

CIM Fabrico assistido por computador (*Computer Integrated Manufacturing*)

BD Base de dados

SO Sistema Operativo

IDE Ambiente de desenvolvimento integrado (*Integrated Developement Environment*)

HMI Interface homem-máquina (*Human-Machine Interface*)

PLC Controlador lógico programável (*Programmable Logical Controller*)

CNC Controlo numérico computorizado (*Computer Numerical Control*)

ACL Linguagem avançada de controlo (*Advanced Control Language*)

ESTG/IPL Escola Superior de Tecnologia e Gestão, Instituto Politécnico de Leiria

Capítulo 1

Introdução

Com o avanço tecnológico das últimas décadas, torna-se necessário inovar os processos automatizados para se manterem economicamente rentáveis, sendo muitas vezes necessário recorrer a sistemas mais complexos. Com o aumento dessa complexidade, surge a necessidade de apostar em ferramentas que facilitem e suportem o processo nas suas várias fases: modelação, desenvolvimento e análise. Os sistemas industriais estão a tornar-se de tal modo complexos que a tecnologia tem evoluído no sentido de resolução de problemas associados ao desenvolvimento através de uma modelação e análises mais eficazes [7].

Com a necessidade de sistemas cada vez mais eficientes é relevante a introdução de um sistema de supervisão e controlo como, por exemplo, sistemas SCADA, que, além de efetuarem a supervisão do sistema, permitem o controlo pontual de alguns equipamentos. Este tipo de controlo normalmente acontece após a ocorrência de determinados eventos (temperaturas superiores ao estabelecido, nível de líquidos abaixo de determinado valor, etc.). Estes eventos provêm habitualmente de sistemas que são modelados com base em autómatos, podendo no entanto recorrer-se a outro tipo de abordagens, como a modelação com base em Máquinas de Estados ou Redes de Petri.

Num sistema de automação é comum basearmos-nos em eventos discretos, ou seja, acontecimentos que são discretos no tempo e que na prática correspondem a eventos gerados assincronamente. Estes sistemas podem ser descritos por modelos baseados em eventos discretos, sendo as duas principais técnicas utilizadas as Máquinas de estados e as Redes de Petri. As Rede de Petri (RdP), como ferramenta matemática que são [1], conseguem fazer uma análise formal do sistema e ainda uma simulação de todo o processo através dos seus eventos discretos.

Um caso prático da aplicabilidade das RdP consiste na modelação de uma linha de produção industrial, com o objetivo de melhorar o seu desempenho e prevenir falhas. Entre as falhas mais comuns encontram-se os bloqueios de sistema (*deadlocks*), por exemplo. As falhas do sistema, através das RdP, podem ser detetadas e evitadas com antecedência,

através da análise ou simulação da rede desenvolvida.

Os Sistemas de supervisão e aquisição de dados (*Supervisory Control And Data Acquisition*) (SCADA) permitem a recolha de informação de uma ou mais instalações e permitem o controlo de parte das instalações. Estes sistemas permitem uma deteção rápida e localizada de problemas pois, quando existe alguma anomalia nas instalações, transmitem essa informação através da interface que têm para o efeito [2].

O que é apresentado neste trabalho é uma aplicação de RdP [3] integrada com um sistema SCADA. A RdP permite a supervisão e controlo de todo o processo de produção, enquanto o sistema SCADA será utilizado apenas para monitorizar o processo e, quando em modo manual (isto é, caso não esteja a ser utilizada a RdP para o controlo do processo), permitirá ao utilizador interagir diretamente com o sistema.

Para que os vários componentes integrantes do sistema global possam comunicar entre si de forma segura e à distância (inclusive quando implementados em hardware distinto), foi utilizada uma Base de dados (BD). Deste modo, cada um dos sistemas trabalha independentemente, com o seu servidor associado e toda a informação está centralizada e acessível através da rede local. Para que as informações/ações provenientes do SCADA/RdP sejam transmitidas para o sistema real, foi estudada e desenvolvida uma aplicação para cada uma das células de trabalho do sistema físico. Assim, o objetivo de cada uma das aplicações é executar a comunicação entre os equipamentos físicos da respetiva célula de trabalho e os sistemas de supervisão/controlo, através da base de dados.

1.1 Objetivos

O objetivo do presente trabalho é desenvolver um sistema genérico centralizado, que permita analisar e controlar o processo produtivo de um Sistema flexível de fabrico (*Flexible Manufacturing System*) (FMS). Este objetivo pode ser dividido em três fases distintas: uma em que deve ser desenvolvido um sistema SCADA com a finalidade de monitorizar o processo produtivo em tempo real, outra que é a conceção de uma RdP que permita controlar, analisar e monitorizar o sistema global e, por fim, a integração de todos os sistemas.

1.2 Organização do documento

A estrutura desta dissertação está definida em 5 capítulos. O presente capítulo serve para enumerar os vários objetivos que foram previamente estabelecidos, bem como fazer uma introdução ao trabalho de investigação desenvolvido.

O Capítulo 2 faz referência e analisa algumas implementações/soluções existentes que

servem de base ao desenvolvimento, nomeadamente sistemas controlados pelo tempo ou por eventos, RdP, SCADA e Bases de dados.

O Capítulo 3 descreve a abordagem desenvolvida neste trabalho, nomeadamente em termos de estrutura proposta para os diferentes sistemas, interligação dos mesmos e a supervisão/controlo do processo produtivo com base no SCADA e nas RdP. É também feita, ao longo deste capítulo, uma comparação de diversos softwares de modo a selecionar os que são usados nesta dissertação.

O Capítulo 4 detalha a aplicação da abordagem desenvolvida (no Capítulo 3) a um caso prático de estudo. Aqui é descrita a abordagem adotada sobre a integração de um sistema SCADA com o sistema de controlo/monitorização baseado em redes de Petri e o sistema FMS existente no laboratório de robótica da Escola Superior de Tecnologia e Gestão, Instituto Politécnico de Leiria (ESTG/IPL).

Finalmente, no Capítulo 5, estão presentes os resultados obtidos nesta dissertação, são apresentadas conclusões e são ainda feitas algumas sugestões para trabalho futuro.

Este documento inclui ainda anexos que dizem respeito ao software IntegraXor, à BD PostgreSQL, ao software de modelação e análise de RdP (PIPE), software utilizado ao longo desta dissertação, estrutura das aplicações cliente e, por último, a RdP global de todo sistema. Nestes anexos é possível consultar informação mais específica e técnica, no que diz respeito à implementação dos sistemas desenvolvidos nos capítulos anteriores.

Capítulo 2

Sistemas Flexíveis de Fabrico

Ao longo deste capítulo são abordadas as áreas de estudo que servem de base para o desenvolvimento desta dissertação. Começa-se por fazer referência a sistemas FMS, aos quais podem ser aplicadas ferramentas de monitorização e análise, como é o caso dos temas introduzidos de seguida: RdP e SCADA. Por fim é feita referência aos tipos de BD existentes atualmente e é explicado como estas podem ser utilizadas para integrar as várias ferramentas estudadas.

O mercado mundial durante a década de 60 é caraterizado pelo aumento da produção pois, até à data, as produções não conseguiam ultrapassar o consumo [4]. A partir do final da década de 60 a relação produção/consumo voltou a mudar, em grande parte devido ao fator globalização e, nessa época, a oferta já superava a procura, o que deu origem a mercados mais refinados e exigentes. No final da década de 70 existiu uma nova mudança comportamental a nível global em que as pessoas começaram a focar a sua atenção sobre o que consumiam e o seu poder de compra aumentou significativamente [4]. De modo a atrair novos consumidores, os produtores tiveram de modernizar os processos de produção e reduzir o intervalo de tempo entre lançamentos de produtos para o mercado, o que impulsionou o desenvolvimento dos FMS (Flexible Manufacturing Systems).

O ciclo de vida de um FMS passa por várias etapas: planeamento do FMS, projeto, instalação, planeamento de produção, operação e afinações/melhorias ao longo da produção [5]. Ao longo do ciclo de vida existem formas de medir o desempenho do sistema, como por exemplo, através de taxas de utilização, taxas de produção ou grau de flexibilidade.

Os FMS são, por norma, um conjunto de máquinas, transportadores e dispositivos que têm funções de interação entre eles, sendo o seu processo, na maioria das vezes, cíclico[6]. Os FMS são caracterizados essencialmente pela sua flexibilidade pois podem mudar a sua configuração de acordo com o objetivo de produção. A flexibilidade inerente aos FMS reduz o tempo de execução dos processos, mas torna o seu controlo e modelação mais complexos. Ainda assim, um sistema automático fixo, otimizado para uma determinada

produção, tende a ser mais eficiente que um FMS, mas existem algumas vantagens que um FMS tem perante outros tipos de sistemas de produção [6, 7], como por exemplo:

- Maior rapidez e menores custos quando são necessárias alterações nos equipamentos do sistema, de modo a alterar a produção;
- Os custos de trabalho a médio/longo prazo são menores devido à redução do número de empregados;
- Possibilita melhorias na qualidade do produto, devido ao controlo automatizado, reduzindo a possibilidade de erro humano associado;
- Existência de uma maior produtividade (quando é necessário fazer alterações significativas à linha de produção em tempos relativamente reduzidos), logo o custo por unidade será menor nesses casos;
- Poupanças com custo indireto pois os erros de trabalho serão menores e o arranque, após falhas, será necessário menos vezes.

Algumas desvantagens identificadas:

- É necessária uma quantidade de tempo considerável para pré-elaborar o sistema automático;
- Elevado investimento inicial;
- São necessárias pessoas especializadas para manusear este tipo de sistemas pois, normalmente, são complexos.

2.1 Sistemas controlados pelo tempo ou por eventos

É entendido por evento o acontecimento instantâneo que pode levar o sistema a transitar de um dado estado para outro. Quando um sistema, sendo ele discreto ou contínuo, varia de acordo com o tempo, diz-se que é um sistema controlado pelo tempo. Por outro lado, quando o sistema não depende do tempo, mas sim de eventos diz-se que o sistema é controlado por eventos [1]. Na Figura 2.1 é possível verificar algumas possíveis classificações de sistemas.

2.1.1 Sistemas de eventos discretos

Num sistema de eventos discretos, as transições entre estados são dirigidas por eventos que ocorrem em determinados instantes de tempo sendo que, tipicamente, esse espaço de estados é discreto [1].

A modelação de sistemas é, cada vez mais, uma técnica utilizada quando se trata de sistemas com um grau de complexidade elevado. Dentro das técnicas utilizadas para modelação de sistemas de eventos discretos encontram-se, por exemplo, as máquinas de estados [8], os Grafcets [9] e as RdP. Cada um dos métodos de modelação tem as suas mais valias e particularidades, que devem ser exploradas, dependendo do sistema a modelar. As RdP destacam-se das restantes técnicas de modelação pelo fato de possuírem um nível de modelação superior e através dele se conseguir reduzir ou eliminar certas propriedades, como por exemplo a concorrência.

Figura 2.1: Possíveis classificações de sistemas tendo em conta as suas características [10].

2.2 Redes de Petri

Carl Adam Petri foi um matemático alemão que desenvolveu uma ferramenta de modelação, as Redes de Petri, em 1939, mas estas foram postas em prática apenas em 1962 na sua tese de doutoramento. A primeira vez que foi usado oficialmente o nome "Rede de Petri" decorria o ano de 1975, 13 anos após Petri se referir a esse formalismo, quando no *Massachusetts Institute of Technology* se realizou uma conferência chamada de "Conferência sobre Redes de Petri e métodos relacionados" [11].

Inicialmente as RdP tinham como objetivo descrever processos químicos mas, depois de ser observado que tinham capacidade para representar conceitos com um elevado grau de

abstração, a sua aplicação foi expandida para outro tipo de modelação de sistemas. Dada a sua aplicabilidade em modelação de sistemas de elevada complexidade, têm suscitado a atenção de diversos investigadores a nível mundial. As RdP começaram a ter uma maior aplicabilidade a partir da década de 80 quando surgiram extensões de alto nível, como por exemplo as RdP numéricas ou as redes coloridas [12].

Para além dos conceitos matemáticos sólidos em que se fundamentam, as RdP permitem verificar visualmente a evolução do sistema ao longo do tempo. Logo, permitem que uma pessoa que conheça os fundamentos e restrições das RdP, mesmo que não tenha conhecimentos do processo em si, consiga obter conhecimento acerca do funcionamento do sistema a descrever. Como esta ferramenta de modelação de sistemas deriva de conceitos teóricos que podem ser fundamentados matematicamente, é uma ferramenta eficaz no que diz respeito a avaliar qualitativamente e quantitativamente um sistema complexo, isto por si só já representa um avanço significativo para a previsão comportamental de sistemas.

As características referidas são apenas alguns dos motivos que tornam as RdP a ferramenta apropriada para modelar e representar sistemas dinâmicos, estando presente entre nas mais diversas áreas: robótica, software, hardware, processos químicos, sistemas computacionais, sistemas de comunicação, etc.

Nos parágrafos seguintes introduzem-se os conceitos básicos relacionados com as RdP, os quais se encontram com mais detalhe em [13]. As RdP podem ser classificadas de diversas formas, dependendo das respetivas características. Um método que vulgarmente se usa para agrupar as redes é através do seu grau de abstração, neste caso, de baixo nível ou de alto nível [14].

Nos últimos anos têm surgido diversas variantes de RdP, tendo grande parte delas nascido devido a alterações pelos projetos onde eram usadas. Essas alterações surgiram a partir das RdP ordinárias (de baixo nível), pois este modelo falha na representação de dois fatores que são: aspetos de temporização e demasiada complexidade gráfica quando o sistema é complexo.

As RdP podem ser divididas em dois grandes grupos, sendo que um é o conjunto de RdP Ordinárias e o outro o de RdP não Ordinárias [15]. Estes tipos de redes são caraterizados pelas suas marcas, sendo que, as RdP ordinárias se subdividem em:

- Binárias - Apenas permitem marcas de valor unitário e não nulo e no máximo existe uma marca por lugar.
- Lugar/Transição - É possível acumular várias marcas nos lugares e existem arcos com peso não unitário.

Nas redes de alto nível é possível distinguir o tipo de marcas em cada lugar e é permitida a agregação de várias marcas em grupos. Essa agregação pode ser feita através

de cores, dando origem às RdP Coloridas [12]. As redes não ordinárias não aumentam o poder de representação de um modelo, mas possibilitam uma visualização mais compacta do sistema, bem como variar o nível de abstração do modelo.

2.2.1 Componentes de uma Rede de Petri

As RdP são constituídas por 3 elementos básicos, visíveis na Figura 2.2. Os elementos são: lugares, transições e arcos, representados por círculos, barras e setas de ligação, respetivamente [16]. Para além destes elementos existe ainda um outro chamado *token* ou marca.

Embora possa haver outras abordagens, tipicamente os eventos dizem respeito a condições associadas a transições e os lugares dizem respeito a ações. O estado da RdP é definido pela sua marcação, isto é, pelo número de marcas em cada lugar da rede.

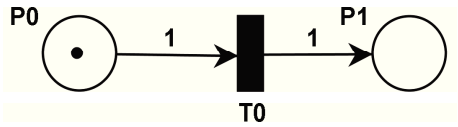


Figura 2.2: Representação dos lugares (P0 e P1), da transição (T0) e de arcos com peso unitário (interligam os lugares e as transições).

As marcas no sistema dão a possibilidade de representação dos estados, antes e depois de cada evento [17]. A marcação inicial do sistema faz parte da sua modelação e quaisquer falhas, podem traduzir-se em inconsistências futuras. A definição usada neste trabalho para RdP lugar/transição etiquetadas é:

- $R = (P, T, A, PA, M0, L)$, onde:
  - $P = P1, P2, ..., Pm$ é o conjunto de lugares;
  - $T = T1, T2, \ldots, Tn$ é o conjunto de transições;
  - $P \cap T = \emptyset \wedge P \cup T \neq \emptyset$ os conjuntos P e T não têm elementos comuns e a sua união é não nula;
  - $A : (P \text{ x } T) \cup (T \text{ x } P)$ é o conjunto dos arcos;
  - $PA : A \rightarrow \mathbb{N}$ são os pesos dos arcos;
  - $M0 : P \rightarrow \mathbb{N}_0$ é a marcação inicial;
  - $L = (L_p, L_t)$, com $L_p \cap L_t = \emptyset$, e onde L_p é uma função que representa a etiquetagem dos lugares e L_t é uma função que representa a etiquetagem das transições.

As mudanças de estado do sistema ocorrem com o disparo de transições, de acordo com algumas regras [13]:

- Uma transição t é dita disparável se cada lugar p de t está marcado com pelo menos $w(p,t)$ marcas, em que $w(t,p)$ é o peso do arco de p para t;
- O disparo de uma transição disparável t, retira $w(p_i,t)$ marcas de cada um dos lugares de entrada p_i de t e adiciona $w(t,p_j)$ marcas a cada um dos lugares de saída p_j de t, onde $w(p_i,t)$ é o peso do arco do lugar p_i para a transição t e $w(t,p_j)$ é o peso do arco da transição t para o lugar p_j.

Como indicado anteriormente, caso haja arcos com peso superior a 1 estamos perante uma RdP não-ordinária. As diferenças entre os dois tipos de redes são visíveis nas Figuras 2.3 e 2.4.

(a) Antes do disparo de T0. (b) Depois do disparo de T0.

Figura 2.3: *RdP com arcos com valor unitário.*

(a) Antes do disparo de T1. (b) Depois do disparo de T1.

Figura 2.4: *RdP com um arco de valor 2.*

A diferença que existe entre a RdP da Figura 2.3 e da Figura 2.4 é o peso do arco que liga a transição T0 e T1 ao lugar P2 e P5, respetivamente. Como o peso é diferente, a marcação final para cada uma das redes será diferente. A título de exemplo, indicam-se de seguida as marcações associadas a cada das redes exibidas acima, associadas ao estado M=(p1,p2,p3):

- Marcação inicial para a RdP da Figura 2.3: $M_i = (1, 1, 0)$;
- Marcação final para a RdP da Figura 2.3: $M_f = (0, 0, 1)$;

- Marcação inicial para a RdP da Figura 2.4: $M_i = (1, 1, 0)$;
- Marcação final para a RdP da Figura 2.4: $M_f = (0, 0, 2)$.

2.2.2 Propriedades das Redes de Petri

Como indicado anteriormente, o estado de uma RdP é dado pela marcação dos seus lugares, sendo que a marcação de cada lugar é o número de marcas nesse lugar (valor inteiro não negativo). Existem algumas propriedades associadas às RdP, entre elas encontram-se [18]:

- Alcançabilidade - Um estado de uma RdP é dito alcançável se, a partir do estado inicial da rede, existir uma sequência de transições que leve a esse estado;
- Limitação - Uma RdP é k-limitada se, para todos os estados alcançáveis, nenhum lugar contém mais que k marcas;
- Segurança - Uma RdP é segura se ela for 1-limitada;
- Persistência - Uma RdP é dita persistente se, para cada duas transições habilitadas, o disparo de uma não desabilita a outra;
- Reversibilidade - Uma rede é dita reversível se, para cada estado M_i no conjunto dos estados possíveis, o estado inicial puder ser novamente alcançado;
- Vivacidade - Uma RdP é dita viva se, a partir de qualquer estado alcançado após o estado inicial da rede, existir uma trajetória em que uma ou mais transições possam disparar (não existam bloqueios do sistema). Uma RdP diz-se bloqueada, quando existe *deadlock* se, após uma dada sequência de disparos a partir da estado inicial, esta se encontra numa situação em que não há qualquer transição "disparável". Existem diferentes tipos de vivacidade da RdP de acordo com o número possível de disparos das transições em causa [13].

Quando um sistema não bloqueia completamente mas, uma certa região da rede tem um possível ciclo infinito de disparos de transições, diz-se que existe a possibilidade de *livelock*, caso o sistema bloqueie completamente, diz-se que existe um *deadlock* na RdP.

As RdP etiquetadas são uma sub-categoria de RdP, em que são atribuídos caracteres de um alfabeto para algumas ou para todas as transições existentes na rede [19]. Neste tipo de RdP, a cada transição é associada uma etiqueta. Este tipo de etiquetas estabelece uma relação direta entre as RdP e os eventos associados ao sistema [20], permitindo a identificação de cada um dos elementos da rede que simbolizam eventos. Posteriormente, derivado da etiquetação, existe uma visualização gráfica mais percetível da rede global e é possibilitada a integração de sistemas externos através da utilização dessas etiquetas.

2.3 Sistemas SCADA

SCADA, é um acrónimo que significa *Supervisory Control And Data Aquisition*. Os programas SCADA dizem respeito a sistemas que recolhem dados de vários sensores espalhados pelas instalações (fábricas, sistemas de transportes, etc.), sendo que, de seguida, processa os dados e atua em conformidade [21]. Para além da recolha de dados, este tipo de programas incluem o processo de decisão, daí a palavra 'supervisão' no acrónimo. Tipicamente um SCADA é utilizado para desempenhar várias funções, as quais podem ser divididas em quatro grupos distintos: aquisição de dados, comunicação de dados, apresentação de dados e controlo.

Atualmente os sistemas SCADA utilizam-se para controlar, sobretudo ao nível da decisão e *set-points*, uma grande variedade de equipamentos. Este tipo de sistemas é tipicamente utilizado para automatizar processos industriais onde, por exemplo, o controlo do processo por um ser humano seria impraticável. Utilizam-se, por exemplo, em sistemas onde existem muitos fatores a controlar ou quando esse controlo é demasiado rápido para um humano executar confortavelmente em tempo real. Existe controlo SCADA em empresas de geração, transporte e distribuição de energia elétrica, estações de tratamentos de águas, sinalização luminosa em tráfego automóvel, edifícios e instalações ou em transportes coletivos, entre outros. São apenas alguns exemplos onde os sistemas SCADA podem ser aplicados pois, de dia para dia, a automação vai permitindo aumentar a eficiência de execução de tarefas, abrangendo assim um maior leque de possibilidades [1].

Um sistema SCADA integra um conjunto de equipamentos reais e software, nomeadamente módulos de aquisição, controlo, visualização, interação homem-máquina e comunicação. É necessário por isso ter em conta diversos fatores no momento de escolher o sistema SCADA a utilizar. Por exemplo, existe uma grande dependência entre o Sistema Operativo (SO) e o sistema SCADA que corre numa determinada máquina. Se o sistema operativo deixa de ser suportado pelo fabricante, o SCADA que corre sobre esse SO pode tornar-se obsoleto. É aconselhável que a escolha do programa SCADA incida em padrões abertos, particularmente em termos de comunicação, que contenham um bom histórico e que tenham compatibilidade com os novos SO e com os protocolos dos equipamentos [22].

No passado, as máquinas que suportavam os sistemas SCADA eram, maioritariamente, baseadas em servidores Unix®[1] mas, com o aumento de processamento por parte dos computadores em geral e com a tendência dos utilizadores a utilizarem tecnologias baseadas em Microsoft Windows®, este cenário tem vindo a alterar-se.

Na Figura 2.5 é visível a relação do número de máquinas quanto à plataforma sobre a

[1]Marca registada, mas por motivos de simplificação de escrita, ao longo desta dissertação, o símbolo será omitido após a primeira utilização.

qual estão implementadas. Na mesma figura estão representados três tipos de equipamentos: a azul estão representados os servidores SCADA, a violeta os dispositivos remotos (RTU's, são normalmente equipamentos que concentram a informação de vários sensores numa dada localização física) e a amarelo os dispositivos eletrónicos inteligentes (IED's, que permitem a comunicação direta entre os RTU's e os servidores SCADA).

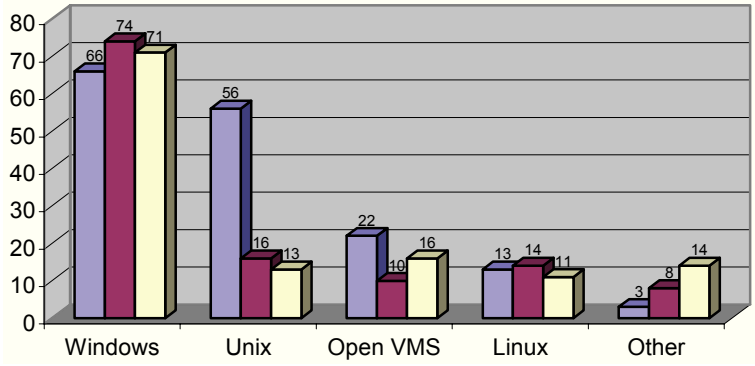


Figura 2.5: Percentagem de equipamentos do sistema SCADA em função dos SO nelas instalados [23].

A vida útil de um sistema SCADA termina, tipicamente, quando o hardware deixa de conseguir dar resposta às imposições feitas pelo sistema. Idealmente, deve ser feita uma atualização de hardware antes que este deixe de funcionar, tal como é feita a manutenção preventiva em todo o tipo de equipamentos. Apesar de ser aparentemente mais dispendioso custará menos que uma falha inesperada do sistema [22].

Um programa que tenha sido desenvolvido por um reduzido número de pessoas está mais vulnerável à estagnação pois, tipicamente,tem menor garantia de suporte. Existem ainda sistemas simples que não justificam a integração de uma ferramenta complexa. Nesses casos pode ser desenvolvido um programa à medida, mas deve ser equacionada a posterior expansão, caso contrário tornar-se-á um sistema solitário, sem apoio e que terá um número limitado de anos de vida [22].

Os programas SCADA têm algumas vantagens, entre as quais:

- Normalmente as estações físicas onde está a interface gráfica apenas necessitam de ligação à rede, sem existir necessidade de instalação de software exigente do ponto de vista de capacidade de processamento;

- Em caso de avaria consegue ter-se rapidamente uma noção do local da mesma;

- É fácil a integração de novas entradas e saídas ao sistema.

As desvantagens inerentes são [23]:

- Problemas relacionados com a comunicação entre sensores e servidor ou servidor e atuadores, independentemente da tecnologia usada;

- Normalmente é utilizado o método *pooling* para atualizar as variáveis do sistema (verificação periódica aos equipamentos sobre monitorização). Este método pode implicar um atraso relevante entre o pedido de informação, a receção da mesma e a atuação que lhe for devida;

- A reconfiguração do sistema pode ser morosa;

- Tem custos elevados e deve ser ponderada a sua utilização quando o sistema a modelar é pouco complexo.

2.4 Bases de dados

Uma base de dados é uma coleção de dados que está relacionada com um objetivo particular. Um sistema de gestão de BD (DBMS) é um sistema que guarda e devolve informação garantindo quer a gestão do acesso aos dados, quer o armazenamento dos mesmos [25]. Este é útil para ajudar a armazenar e organizar dados de acordo com o assunto a que estão remetidos, de modo a que seja fácil encontrar os dados pretendidos quando requeridos.

Nos sistemas flexíveis de produção há uma necessidade de gerir e centralizar um conjunto volumoso de informação, o que tipicamente acontece através de uma ou mais bases de dados. Foi feita uma análise acerca das características de diversas bases de dados e foi verificado que a oferta é bastante. De seguida são enumerados alguns servidores de BD que podem satisfazer as necessidades que a grande maioria dos projetos hoje tem: aumentar a segurança, aumentar a redundância e ter um sistema em que toda a informação está centralizada e acessível através da rede.

- Microsoft Access®
- MySQL®
- Oracle DataBase®
- SQL server®
- PostgreSQL®

No Capítulo 3.1 vai ser abordado com mais profundidade este tema, sendo aí selecionado um software para dar continuidade ao trabalho desenvolvido nesta dissertação.

Capítulo 3

Sistema desenvolvido - SCADANet

Neste capítulo descreve-se a abordagem desenvolvida para a integração de um sistema SCADA com um sistema de controlo/monitorização baseado em redes de Petri, para a monitorização, controlo e análise de um sistema flexível de produção. Este objetivo passa por fazer o dimensionamento da RdP através de um software de modelação e simulação de RdP, fazer o dimensionamento do sistema SCADA através de um programa para o efeito e, por fim, fazer a integração de ambos os sistemas. Inclui-se, por isso, neste capítulo uma análise de vários softwares e a respetiva seleção para cada uma funções descritas anteriormente. A Figura 3.1 representa o que é pretendido com a abordagem proposta nesta dissertação.

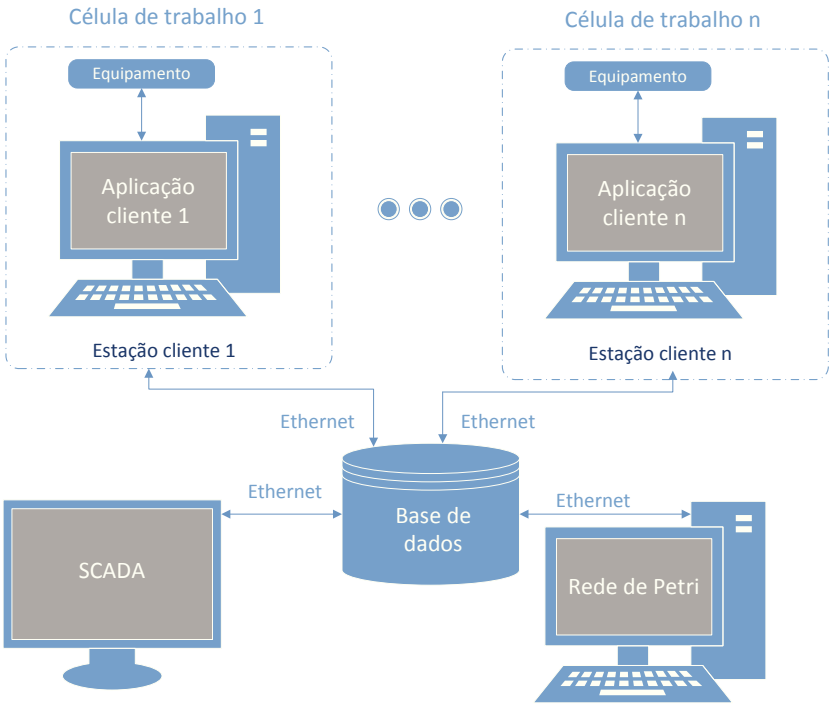


Figura 3.1: Esquema do sistema SCADANet.

Na abordagem proposta, o sistema de produção é supervisionado pela RdP e na Figura 3.2 é possível ver parte dessa abordagem e interligação de equipamentos. A RdP terá acesso aos eventos despoletados pelas diversas células de trabalho e utilizará essa informação para enviar ordens de produção de volta para essas mesmas células.

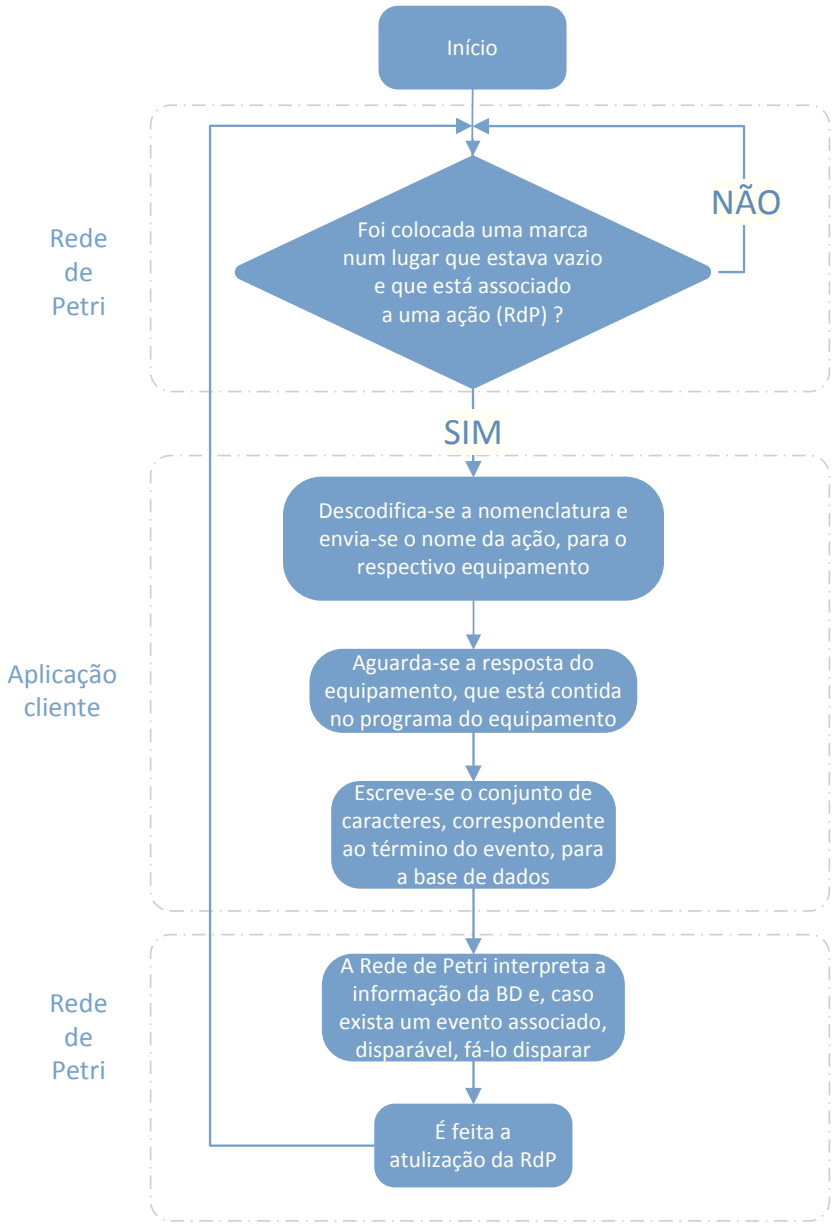


Figura 3.2: Filosofia de comunicação entre sistemas e equipamentos.

O sistema SCADA será utilizado para monitorizar o conjunto, possibilitando uma interface gráfica mais amigável para o utilizador. Todo o sistema comunica via base de dados centralizada, com comunicação baseada em Ethernet (TCP/IP).

As secções seguintes descrevem com mais detalhe cada um dos temas referidos neste capítulo.

3.1 Base de dados

Neste trabalho é vital a utilização de um sistema de armazenamento de informação global relativo a todos os processos e dispositivos, implementado através de uma base de dados. Esta base de dados é utilizada para leitura e escrita de dados, de acordo com as permissões dos clientes que requisitem ou que recebam a informação dela proveniente.

Após uma primeira análise, foram postos de lados os sistemas que não são de código-aberto, que o seu desenvolvimento não está ativo ou que contêm uma licença comercial, ou seja, apenas ficaram para uma posterior análise os softwares: MySQL e PostgreSQL.

Os requisitos recomendados pelo PostgreSQL (informação que não está disponibilizada oficialmente) podem ser algo a seu favor uma vez que são reduzidos, quando comparados com os do MySQL [1] [2]. Esta informação pode ser consultada nas Tabelas 3.1 e 3.2, relativamente ao PostgreSQL e ao MySQL, respetivamente.

Tabela 3.1: Requisitos recomendados para funcionamento do PostgreSQL.

Hardware	**Requisitos recomendados**
Memória Ram	2 GB
Processador	Processador 64 bits de 2 núcleos
Sistema Operativo	SO 64 bits

Tabela 3.2: Requisitos recomendados para funcionamento do MySQL.

Hardware	**Requisitos recomendados**
Memória Ram	$\geq$ 6GB
Processador	1 núcleo $\geq$ 3 GHz ou 2 núcleos $\geq$ 2GHz
Sistema Operativo	SO 64 bits

O resultado da comparação entre os recursos exigidos por cada um dos programas em estudo pode ser um dos motivos pelo qual o MySQL tem vindo a perder alguma liderança ao nível de sistemas de BD [3]. Com a aquisição do MySQL por parte da Oracle [4] os utilizadores procuraram outras alternativas gratuitas e de código-aberto. Este facto é de enorme importância pois os programadores gostam de desenvolver com a maior liberdade possível e sem estarem dependentes de terceiros ou de restrições ao nível de programa.

Neste trabalho a exigência que é pretendida da BD é relativamente reduzida. É necessário criar apenas algumas tabelas na BD e os acessos às mesmas podem ser considerados

[1] `http://www.mysql.com/support/supportedplatforms/workbench.html`

[2] `http://www.commandprompt.com/blogs/joshua_drake/2009/02/postgresql_mininum_requirements/`

[3] `http://vschart.com/compare/postgresql/vs/mysql`

[4] http://www.oracle.com/pt/index.html

reduzidos e com tempos de acesso confortáveis. Através da Figura 3.3 é possível verificar que, para o pretendido, o PostgreSQL supera em todos os testes o MySQL. A diferença entre tempos de criação de tabelas, através dos dois softwares analisados, é bastante significativa.

	512MB		1GB	
Módulo	**PostgreSQL**	**MySQL**	**PostgreSQL**	**MySQL**
Criação de Tabelas	0,02s	0,15s	0,03s	0,11s
Carga de Tabelas	4min	4min35s	8min	10min
Criação de Índices	10min	35min	37min	1h12min

Figura 3.3: Comparação PostgreSQL e MySQL em termos de tempo necessário para criação de tabelas na BD [27].

A BD que será usada ao longo desta dissertação é a PostgreSQL, pelos motivos referidos acima. Apesar disto foi equacionada a possível expansão para trabalhos futuros e verificou-se que é aconselhável a transição para MySQL, devido à robustez superior que esta apresenta relativamente ao PostgreSQL [27].

3.2 Sistemas SCADA

Como referido anteriormente, os sistemas SCADA são utilizados para monitorizar e supervisionar sistemas de produção. No caso particular deste trabalho, o sistema SCADA será utilizado maioritariamente para permitir uma interface gráfica amigável de monitorização do sistema, sendo o sistema supervisionado por uma RdP. Quando em modo de depuração, caso o utilizador pretenda intervir no sistema, ou simular um processo que se encontra parado, é possível também nesses casos utilizar o sistema SCADA para esse fim.

Foi efetuada a análise de alguns dos programas SCADA existentes no mercado, de modo a comparar as suas características e verificar qual se adapta melhor ao exemplo teórico. A análise está focada essencialmente em 3 sistemas: IntegraXor®, WinCC® e Genesis®. A pesquisa efetuada incidiu em questões como: custo da licença, credibilidade e histórico do desenvolvimento, compatibilidade de SO, linguagens de programação e flexibilidade.

SIMATIC® é um conjunto de programas de automação, que foi desenvolvido e que é distribuído pela marca alemã Siemens®. Este programa tem como principal objetivo permitir, a quem o manuseia, a verificação do estado de um determinado sistema automático e interagir com ele a partir de uma ou mais máquinas na qual está instalado [28]. Na gama de produtos Siemens SIMATIC existem duas aplicações importantes na área da automa-

ção, nomeadamente, os programas WinCC e STEP7®. O WinCC é um programa SCADA e HMI simultaneamente, desenvolvido para o SO Microsoft Windows e utiliza uma linguagem de perguntas estruturadas (SQL) para efetuar os registos. Por sua vez o STEP7 é uma ferramenta muito utilizada para manuseamento dos controladores (hardware) da Siemens e tem como objetivo prioritário permitir a sua configuração e parametrização. Existem diversos tipos de licenças que permitem a utilização do WinCC, desde as que são ilimitadas até às que são versões de amostra em que a utilização é parcialmente limitada.

Para além da oferta da Siemens, em que o WinCC [5] é o pacote para dar resposta aos sistemas SCADA, foi também estudada a utilização de programas alternativos como é o caso do Genesis e do IntegraXor. Estas soluções, apesar de serem menos utilizadas que o WinCC, têm características que permitem cumprir o que é pretendido nesta dissertação (Capítulo 1.1). Por conseguirem implementar a solução pretendida, tornam-se uma solução plausível pois apresentam custos e requisitos computacionais mais reduzidos. Os requisitos mínimos e aconselháveis para o IntegraXor [29], o Genesis [30] e o WinCC [31] podem ser visto nas Tabelas 3.3, 3.4 e 3.5, respetivamente.

Tabela 3.3: Requisitos mínimos de funcionamento do IntegraXor sobre o SO XP.

Hardware	**Requisitos mínimos**
Processador	Intel © Core ™ 2 @ 1.66GHz
Memória Ram	3 GB
Espaço livre em disco	$\simeq$ 25MB para Runtime
	$\simeq$ 200MB para Development
Sistema Operativo	SO Windows XP, Vista ou 7.

Tabela 3.4: Requisitos do sistema para funcionamento do Genesis64.

Característica	**Mínimo**	**Recomendado**
Processador	Dual Core 64 bits	Athlon, Phenom, Intel
Memória Ram	4 GB	10 GB
Espaço livre em disco	4 GB	4 GB
Sistema Operativo	SO 64 bits: Windows XP, Server, Vista, 7.	

Como se pode verificar os sistemas têm características semelhantes, no entanto o espaço em disco requerido pelo IntegraXor é bastante inferior quando comparado com os restantes softwares.

O software IntegraXor não apresenta problemas de compatibilidade, podendo as páginas Web ser consultadas remotamente em qualquer tipo de dispositivos (pc, *tablet* ou

[5] `http://www.automation.siemens.com/mcms/human-machine-interface/en/visualization-software/scada/simatic-wincc`

Tabela 3.5: Requisitos do sistema para funcionamento do WinCC.

Característica	Mínimo	Recomendado
Processador	Pentium 4 2,5 GHz	Pentium 4 3,5 GHz Dual Core
Memória Ram	2 GB	4 GB
Espaço livre em disco	1,5 GB	>10 GB
Sistema Operativo	Windows Server, XP e Windows 7	

smartphones iphone ou android) [6]. A visualização das páginas está otimizada para uma maior rapidez através das tecnologias HTML 5 e CSS 3. Existem ferramentas que se verificam bastante úteis como é o caso do ajuste automático da página Web à resolução do monitor/ecrã, de conter um poderoso editor de gráfico e de existir uma interface visual para fazer pedidos à BD, de modo a facilitar esta tarefa [6]. Olhando para os critérios de seleção de um ponto de vista mais global, obtém-se a Tabela 3.6.

Tabela 3.6: Critérios de seleção, por ordem decrescente de relevância atribuída.

Critério de seleção	WinCC	Genesis	IntegraXor
Custo	Alto	Alto	Baixo
Integração com outros sistemas	Média	Média	Alta
Recursos computacionais	Altos	Altos	Médios
Facilidade de uso	Média	Média	Alta
Documentação	Boa	Boa	Média
Credibilidade do fornecedor	Boa	Boa	Média

Tendo em conta os dados analisados no Capítulo 2, os programas WinCC e Genesis não foram utilizados para dar seguimento à pesquisa e desenvolvimento nesta dissertação pois contêm uma licença de usabilidade demasiado dispendiosa (dependendo do tipo de licença), o seu programa é fechado e pouco flexível [7] [8]. Como alternativa, foi equacionada a utilização do IntegraXor para o caso em estudo pois, para além de ser "gratuito"e conter requisitos menos exigentes, permite uma maior evolução e abertura à integração de novas funcionalidades.

Proveniente de uma empresa de programação da Malásia, o IntegraXor tem um leque de funcionalidades bastante abrangente, existindo um fórum disponível no próprio site que se baseia no conceito código-aberto para troca de informação e partilha de conhecimento, o que é uma mais valia [9].

[6] `http://www.integraxor.com/product.html`
[7] `http://www.automationsales.com.au/category.aspx?catID=42`
[8] `http://www.pcdsales.com/TrainingGENESIS64Basics.aspx`
[9] `http://www.integraxor.com/forum/`

O IntegraXor pode ser considerado gratuito para desenvolver e testar todo o sistema, apesar de obrigar a uma licença. Porém, é possível ter acesso à totalidade das funcionalidades durante duas horas após a inicialização do servidor. Esta situação não tem qualquer interferência com o desenvolvimento específico desta dissertação pois, como o programa irá ser utilizado para desenvolvimento e teste, não para produção contínua 24h/7, não é necessário estar mais de duas horas consecutivas ligado. Depois de o programa "terminar a sessão" devido à restrição da versão experimental, basta voltar a iniciar-se o servidor. Após o reinicio do servidor todas as funcionalidades voltam a estar disponíveis por um novo período de duas horas. O programa a utilizar para desenvolver o sistema será o IntegraXor (para ver detalhes acerca do seu funcionamento consultar o Anexo A) pois, cumpre integralmente os requisitos definidos para este trabalho.

A linguagem de programação usada nesta dissertação, nos programas IntegraXor e Inkscape (ver o Anexo A), é JavaScript. É usada esta linguagem porque é permitida por ambos os programas e é frequentemente utilizada na Web, o que irá reduzir a probabilidade de incompatibilidades futuras. Para a criação de etiquetas[10] (Tag's) no IntegraXor deve ter-se em atenção a referência em anexo A.3.

3.2.1 Conexão entre IntegraXor e base de dados

É necessário que exista uma ligação entre o servidor do sistema SCADA e o servidor de base de dados. Nesse sentido, a Figura 3.4, ilustra o esquema correspondente ao software IntegraXor, ao programa de edição gráfica Inkscape, à visualização feita através do navegador e à BD, isto é, o ambiente de desenvolvimento do sistema SCADA.

A Figura 3.5 representa um conjunto de ações que devem ser executadas de forma sequencial, de modo a criar uma conexão entre os programas IntegraXor e PostgreSQL. Este processo tem inerente o objetivo de permitir a posterior comunicação entre as aplicações cliente e o IntegraXor.

Uma vez que o programa de gestão da BD está escolhido e definido, deve ser criado um conjunto de caracteres que permitam a conexão (*string connection*), entre a base de dados e o software SCADA. Este conjunto de caracteres será específico para esta ligação e terá em conta vários parâmetros. Entre os parâmetros encontram-se: o driver de comunicação, o endereço do servidor, a porta utilizada para estabelecer a comunicação, o nome do servidor da BD e a palavra-passe (caso exista), sendo que estes dois últimos podem ser introduzidos por interface gráfica nos campos respetivos. Para o efeito pode ser consultado o Anexo A em que a sequência de conexão utilizada no IntegraXor foi a seguinte:

[10] As etiquetas referidas acerca do software IntegraXor não têm qualquer ligação com as etiquetas abordadas no âmbito das RdP etiquetadas.

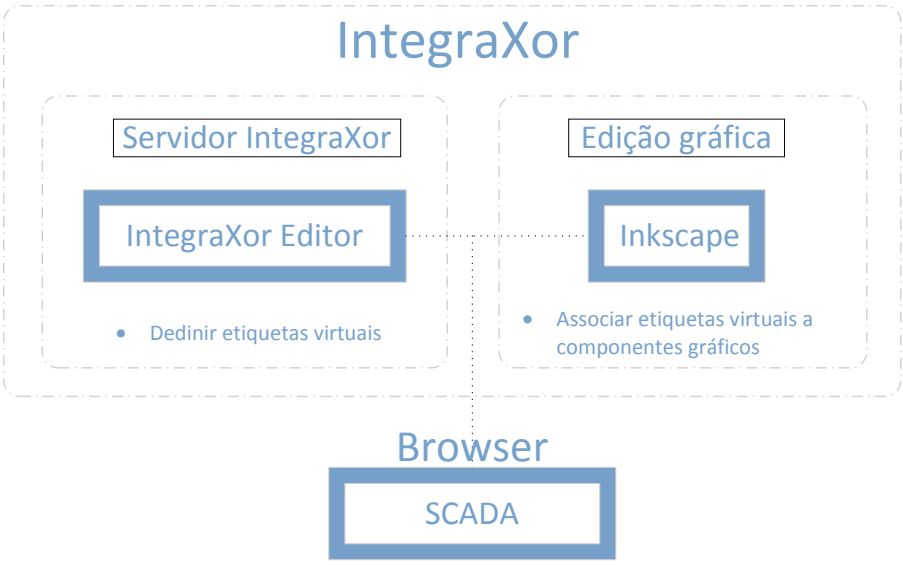


Figura 3.4: Diagrama que representa o IntegraXor, o ambiente de desenvolvimento de SCADA utilizado.

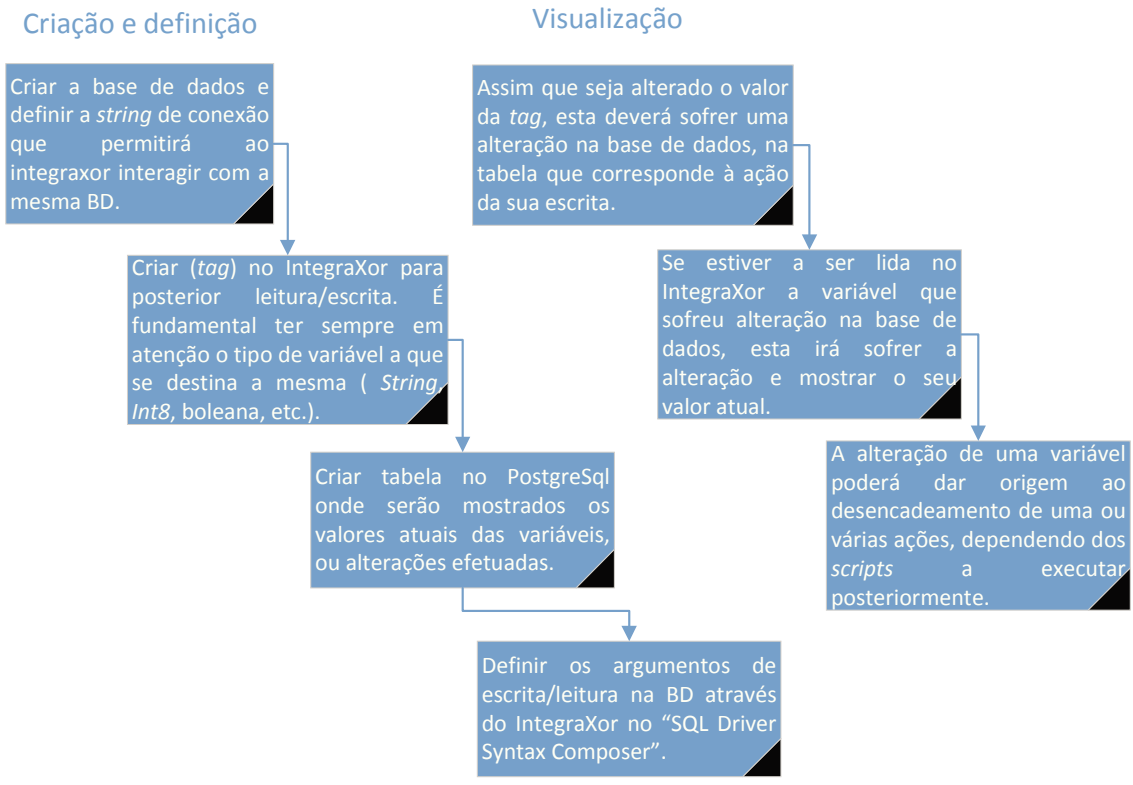


Figura 3.5: Representação das etapas para ler/escrever uma variável do/no IntegraXor.

- $Driver = PostgreSQLUNICODE; Server = 127.0.0.1; Port = 5432; Database = postgres$;

  - **Driver de comunicação:** PostgreSQLUNICODE.

- **Servidor:** 127.0.0.1 [11]

- P**orta de comunicação:** 5432 (porta pré-definida).

- **Base de dados:** postgres

3.3 Supervisão baseada em Redes de Petri

Ao longo deste trabalho a ferramenta escolhida e usada para modelação do sistema foi RdP, em detrimento das outras abordagens, porque as RdP permitem uma análise funcional à priori da execução do sistema e têm ferramentas que permitem detetar falhas no sistema, como por exemplo *deadlocks*. Existem várias variantes de RdP, mas ao longo deste trabalho foi usado o tipo de RdP etiquetadas, tal como abordadas no Capítulo 2. Este tipo de redes permite que se faça a integração de um sistema real com a RdP, sendo deste modo possível que, para além de simulação, se possa executar todo o processo utilizando a RdP como controlador do sistema. Este controlo é feito associando parte das transições da rede a eventos externos (gerados pelos equipamentos), enquanto alguns lugares estão associados a ações a executar pelos vários equipamentos.

Uma RdP, na abordagem utilizada ao longo desta dissertação, pode estar a ser executada em dois modos distintos. Pode ser executada no modo de simulação (*offline*) ou em modo *online*. Em ambos os modos os lugares da rede estão associados a ações e as transições associadas a eventos. Quando uma rede está em modo de simulação, esta não está interligada com o sistema físico, logo todas as transições vão estar associadas a eventos internos da RdP, não dependendo dos eventos nem executando ações propriamente ditas, apenas se simula a execução da rede.

Em modo online, as transições associadas a eventos externos, desde que verificadas as condições de disparo em termos de marcas dos seus lugares de entrada, podem disparar se o evento externo ocorrer. Os lugares associados a ações externas despoletam essas ações sempre que a marcação passa de 0 para 1 no respetivo lugar. A lógica de disparo de transições deste tipo está demonstrada na Figura 3.6.

[11] O servidor da BD está instalado na máquina local. Caso contrário deve ser feito o processo de adição de permissão no PostgreSQL. Este processo permite aceder ao servidor da BD através da rede.

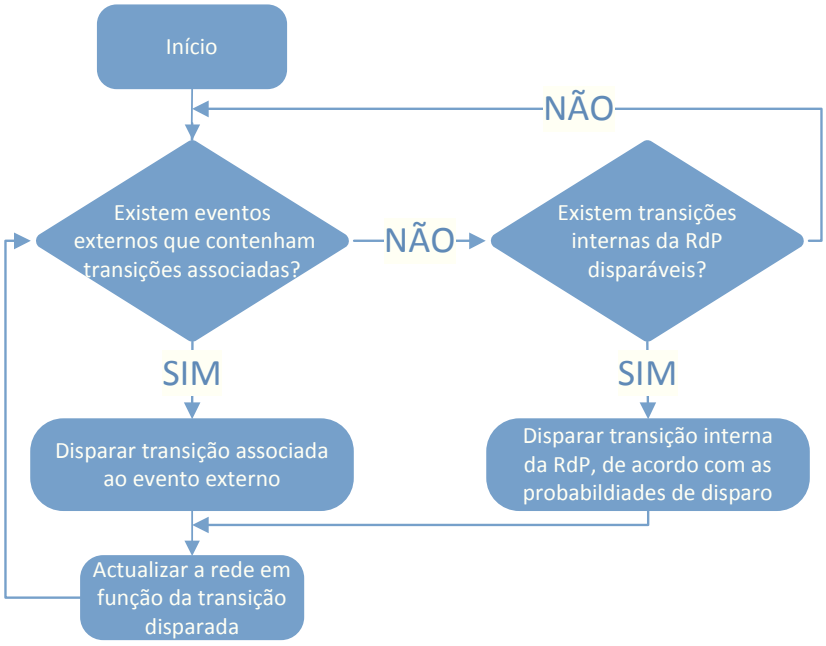


Figura 3.6: Sequência que dá prioridade ao disparo de transição associada aos eventos externos.

3.3.1 Nomenclatura dos lugares e transições

Quando se cria uma RdP, é comum associar-lhe uma nomenclatura, associando a cada um dos componentes da rede nomes sugestivos e que, por si só, permitam perceber o estado dos componentes físicos associados [32]. Este princípio tem como objetivo a melhor interpretação da RdP, para além de facilitar a possível integração com outros sistemas externos, como por exemplo, uma base de dados. Os nomes sugestivos ou etiquetas possibilitam ainda a diferenciação entre ações ou eventos e permitem que se saiba qual o equipamento de origem ou destino.

As RdP que têm uma nomenclatura específica associada, são vulgarmente designadas de RdP etiquetadas, tal como descrito na Secção 2.2. Cada elemento da rede deverá estar etiquetado de acordo com a nomenclatura, existindo diferenças entre a nomenclatura associada a uma ação ou a um evento, que estão representados nas RdP por lugares e transições, respetivamente. As etiquetas associadas a lugares seguem o seguinte formato:

- [**Ação**].[**Equipamento**].[**Ação a executar**]

Quando se tratam de lugares que não estão associados a nenhuma ação externa, o primeiro bloco da nomenclatura não é incluído e o segundo é opcional. As etiquetas associadas a transições seguem o seguinte formato:

- [**Evento**].[**Equipamento**].[**Evento associado**]

Neste caso, se a transição não estiver associada a nenhum evento externo, o primeiro bloco não é incluído e o segundo bloco é opcional.

Uma das grandes vantagens das RdP é a sua capacidade de análise do sistema modelado, nomeadamente de propriedades como as indicadas na Secção 2.2.2, existindo neste momento um número elevado de ferramentas de edição e análise de RdP ([12]). Neste trabalho pretende-se ter uma ferramenta que permita não só editar e analisar uma RdP, como supervisionar e controlar o sistema real. Neste sentido, e de forma a acelerar o desenvolvimento, efetuou-se um estudo de ferramentas de código aberto que pudessem ser utilizadas e/ou adaptadas para este fim. O processo de modelação de um sistema de RdP, pode e deve cumprir várias etapas.

Foi feita uma pesquisa acerca dos vários softwares de dimensionamento, análise e simulação de Redes de Petri [33, 34]. Entre os nomes selecionados para uma análise posterior, devido a conterem as funcionalidades pretendidas, encontram-se: *Plataform Independent Petri Nets Editor* (PIPE) [35, 36], VisObjNet++ [17] e o HPSim. Na Tabela 3.7 são visíveis as principais características destes programas.

Tabela 3.7: Características dos softwares de RdP analisados.

	Platform Independent Petri Net Editor	**Visual Object Net ++**	**HPSim**
Ambientes	Java	Microsoft Windows	Microsoft Windows
Redes de Petri suportadas	•Redes Lugar/Transição	•Redes Lugar/Transição •Redes de Petri temporais •Redes dinâmicas híbridas e redes híbridas de objetos	•Redes Lugar/Transição
Componentes	•Editor gráfico •Jogo de animação de marcas •Simulação rápida •Espaço de estados •Espaço de estados condensado •Lugares invariantes •Transições Invariantes •Análise estrutural •Análise de desempenho simples •Interligação de formatos de ficheiro •Módulos de análise estendida e de formato de ficheiros	•Editor gráfico •Jogo de animação de marcas •Simulação rápida •Análise estrutural •Análise de desempenho simples •Suporte a hierarquias de objetos	•Editor gráfico •Jogo de animação de marcas •Simulação rápida •Análise de desempenho simples

Utilizando a informação presente neste capítulo é possível concluir que o programa que se adequa melhor ao pretendido nesta dissertação, acerca de RdP, é o PIPE. De acordo com isso, foi definido que o PIPE seria o programa a utilizar pois a sua interface gráfica é bastante apelativa e o conjunto de componentes para criar, editar e analisar as RdP é o mais completo dos três conjuntos, sendo que é o único multi-plataforma. Para além disso o PIPE é de código-aberto e é desenvolvido em Java o que é uma vantagem, pois permite a integração com a base de dados de modo relativamente fácil. Podem ser consultados mais detalhes acerca desta escolha no Anexo C.1. As funcionalidades que tiveram de ser

[12] http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/quick.html

implementadas no PIPE de modo a adaptar a aplicação existente ao pretendido neste trabalho foram:

- Desenvolver código que permita comunicar com a BD, o que implicitamente permite interagir com as aplicações cliente e, por esse meio, interagir com os equipamentos;
- Monitorizar eventos externos registados na BD;
- Registar ações e fazer o registo dos eventos (para efeito de diário de acontecimento) da RdP, na BD;
- Permitir que o PIPE funcione em modo *online* e *offline*, consoante seja para controlar/supervisionar o sistema real ou simular o sistema, respetivamente.

Acerca da linguagem de programação a utilizar para alteração do código fonte do PIPE foram considerados dois ambientes de desenvolvimento integrado (IDE) distintos: Netbeans [13] e Eclipse [14]. A restrição mais importante é o facto dos IDE permitirem desenvolvimento em Java, pois este é um requisito previamente definido e indispensável. Outro requisito importante é o facto de o programa ser gratuito de modo a eliminar gastos com licenças.

A utilização do Netbeans deve-se ao facto de, por definição, este já integrar ferramentas de criação de interfaces gráficas para o utilizador (GUI's) que vão ser utilizadas ao longo desta dissertação. O Netbeans já integra, por exemplo, a ferramenta *GUI builder*, enquanto essa funcionalidade para o Eclipse está apenas disponíveis a partir da instalação de extensões [15]. O PIPE necessita de muitas dependências de ficheiros externos e como o número de extensões (*plugins*) do Eclipse é superior e mais maduro, a sua integração é facilitada através deste software, sendo utilizado por isso mesmo. No anexo C.1 é possível verificar, com mais detalhe, algumas das alterações necessárias ao código fonte do PIPE.

Através do descrito no Capítulo 2 foi desenvolvida uma nomenclatura seguindo um esquema semelhante ao das RdP etiquetadas tal como indicado acima, de modo a permitir a interpretação de eventos e a executar ações associados às células de trabalho. Neste caso específico, as aplicações cliente utilizam a nomenclatura para ler ações da BD e registar eventos na BD, de modo a serem controlados pela RdP.

Nos lugares associados a ações, a rede envia ordens às células de trabalho de acordo com as etiquetas associadas, mas a ordem é enviada para a célula apenas quando a marcação deixa de ser nula, e não é enviada nenhuma ordem quando esta passa a nula (assume-se que há um evento externo associado ao fim da ação).

[13] `https://netbeans.org/downloads/`
[14] `https://www.eclipse.org/downloads/`
[15] `http://www.theserverside.com/feature/Whats-the-Big-IDE-Comparing-Eclipse-vs-NetBeans`

Caso exista uma transição disparável, correspondente a um evento externo, esta tem prioridade de disparo sobre qualquer outra. Existem lugares e transições que não estão associados a este tipo de nomenclatura pois são lugares ou transições internos cuja informação é apenas utilizada pela RdP. As prioridades de disparo das transições são um tema que tem influência no desenrolar de todo o processo de fabrico de peças. Deste modo tiveram de ser definidas algumas regras de prioridade de transições:

- As transições associadas a eventos externos têm prioridade sobre transições internas da RdP;
- Caso existam transições internas disparáveis, o sistema escolhe uma aleatoriamente e executa o seu disparo. A aleatoriedade com que é feito o disparo das transições (quando existem várias em simultâneo) é proporcional ao peso relativo das transições disparáveis em cada momento. Essa probabilidade pode ser controlada alterando o peso das transições, controlando desta forma o processo.

3.4 Células de Trabalho

Entende-se por célula de trabalho o local onde existe uma ou mais estações de trabalho e equipamento associado. Cada célula de trabalho deve ser capaz de interagir com a base de dados para comunicar eventos ocorridos e receber ordens. Isso pode ser feito tendo um PC para o efeito, ligado ao equipamento em particular, ou através de outro hardware, por exemplo, um autómato.

Entenda-se por estação de trabalho, um equipamento físico que pode ser, por exemplo, um computador. Em cada uma das estações está a correr uma aplicação Java para permitir a comunicação com o respetivo equipamento (aplicação cliente). Tem de existir uma conexão entre cada uma das aplicações cliente e o PostreSQL, pois cada uma delas irá ler e escrever na BD. Para efetuar este tipo de comunicação estão descritas as etapas de parametrização necessárias em B.1.

3.4.1 Aplicações cliente

De modo a escolher uma linguagem de programação para desenvolver a aplicação do cliente para cada um dos dispositivos que fazem parte do sistema, fez-se uma pesquisa acerca das linguagens mais utilizados no mercado de acordo com o objetivo e dependendo das características das mesmas. Obviamente que este fator não deve ser o único a ser tido em conta mas, ainda assim, pode ter alguma influência na decisão.

A linguagem escolhida para o desenvolvimento das aplicações cliente foi o Java, pelo facto de ser multi-plataforma e ter um bom suporte. As aplicações, desenvolvidas em Java,

são muito semelhantes, funcionando de acordo com o fluxograma ilustrado na Figura 3.7. Estas aplicações, presentes em cada uma das células de trabalho, permitem o controlo dos equipamentos locais enviando os eventos detetados para a base de dados e recebendo as ordens de execução da base de dados.

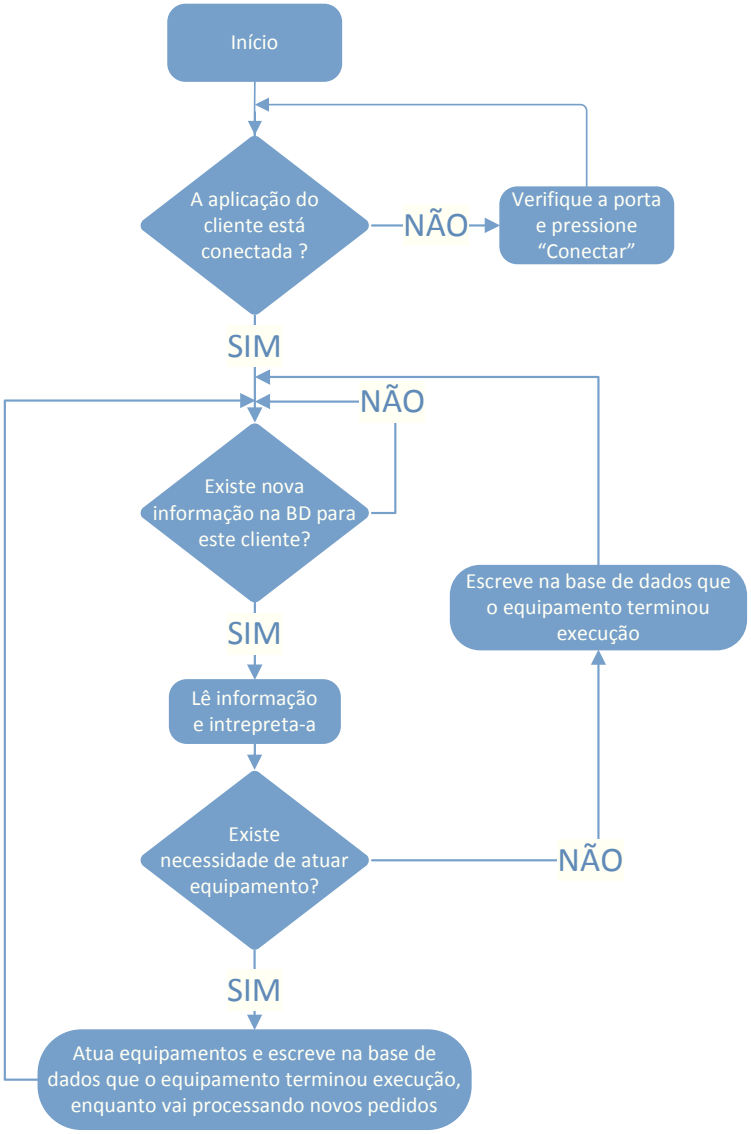


Figura 3.7: Diagrama geral de funcionamento das aplicações Java dos clientes.

Capítulo 4

Caso de estudo e implementação prática

Este capítulo foca-se essencialmente na descrição da implementação da abordagem descrita no Capítulo 2 a um caso prático. Tal como descrito anteriormente, a implementação tem por base um sistema SCADA com objetivo de monitorização do processo produtivo e um sistema de RdP para o controlo e supervisão de um FMS baseado, neste caso, no equipamento existente no laboratório de robótica da ESTG/IPL. Neste capítulo, descreve-se o equipamento utilizado, as funcionalidades do FMS e as várias aplicações envolvidas.

Na Figura 4.1 pode verificar-se como é constituído o FMS, enquanto a Figura 4.3 mostra o diagrama de blocos de todo o sistema. Em ambos podem ser observados os diversos equipamentos que integram o sistema.

De acordo com as estações de trabalho existentes, que juntamente com os equipamentos formam as células de trabalho, a abordagem escolhida para este trabalho é:

- Aplicação cliente para controlo do robô 1;
- Aplicação cliente para controlo do robô 2;
- Aplicação cliente para controlo do tapete transportador;
- Aplicação cliente para Unidade de Controlo de Qualidade (UCQ).

Na Figura 4.2 encontra-se uma representação esquemática do sistema, em que estão representados os locais: do tapete, *buffers*, armazéns de paletes, robôs, alimentador de peças em bruto, CNC ou unidade de processamento, unidade de controlo de qualidade e a saída de peças.

- A célula de trabalho 1 permite gerir vários armazéns: o alimentador de peças em bruto (*Feeder*), armazém de peças manufaturadas (*Rack* 1 e *Rack* 2), armazém de

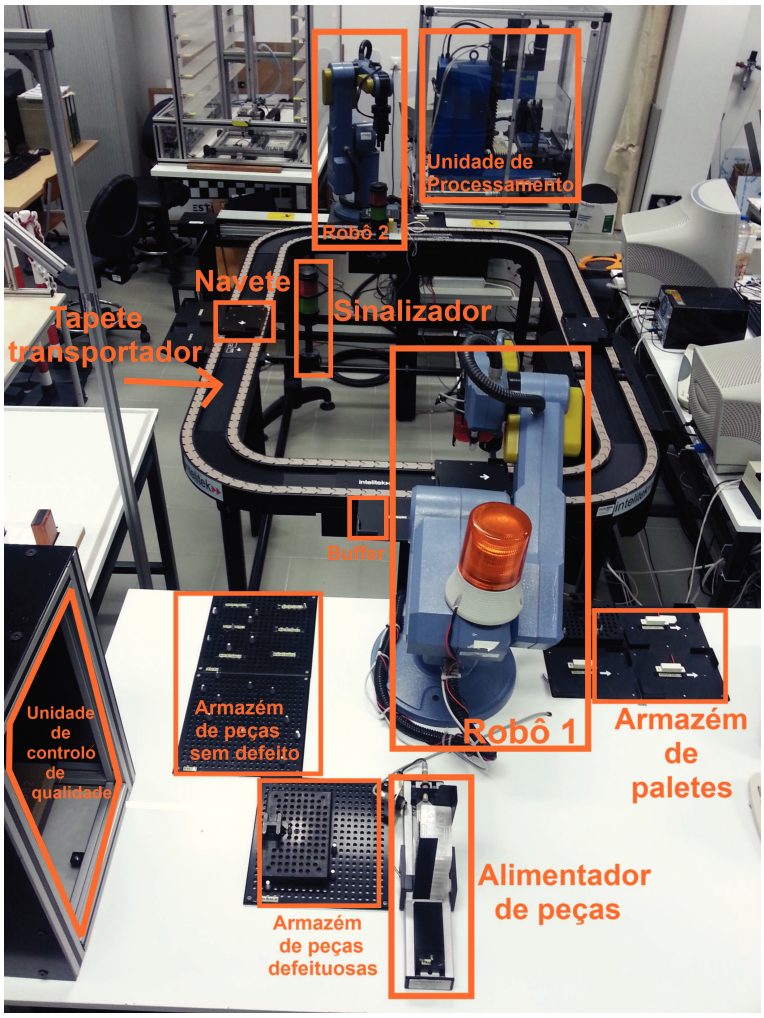


Figura 4.1: Equipamento do sistema FMS presente no laboratório de robótica.

peças com defeito (*Rack* 3) e o armazém de paletes, onde estão as paletes. Nesta célula de trabalho encontra-se o controlador do robô 1 e encontram-se as estações de trabalho que permitem controlar o robô 1 e a unidade de controlo de qualidade;

- A célula de trabalho 2 inclui o robô 2 (com guia linear), o qual permite o transporte de peças de/para a fresadora que, neste caso, simula uma determinada ação de manufatura nas peças. Nesta célula de trabalho encontram-se ainda o controlador do robô 2, a fresadora e a estação de trabalho que permite controlar o robô 2. A fresadora com CNC está ligada à rede local (*LAN*), pois existe a possibilidade de os programas de fabricação das peças em bruto poderem estar armazenados num disco partilhado, localizado numa das outras estações;

- A célula de trabalho 3 permite a visualização e controlo de todo o sistema. É a estação onde está a correr o servidor SCADA, o servidor de BD e o software de RdP;

- A célula de trabalho 4 contém a estação de trabalho que está ligada a um autómato, o qual comunica com a respetiva aplicação cliente, permitindo o controlo do tapete de transporte.

Na Figura 4.2 foram omitidas as estações de trabalho para simplificação da figura. A célula de trabalho 3 não está presente pois não tem equipamentos físicos associados.

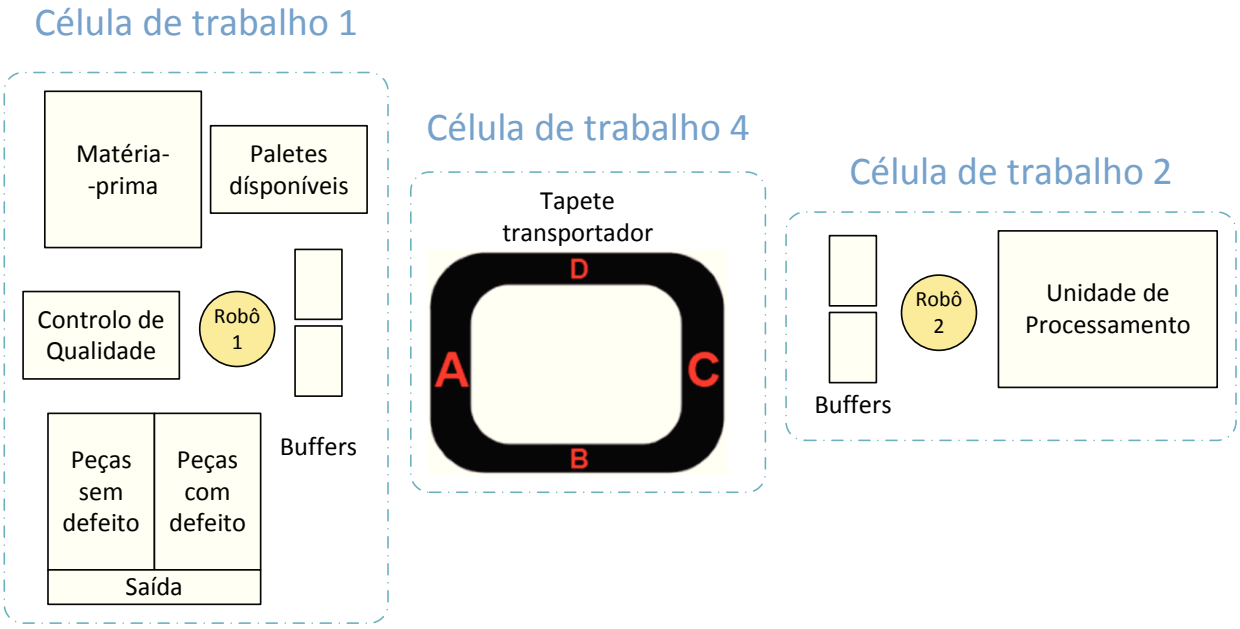


Figura 4.2: Diagrama funcional das células de trabalho existentes.

As quatro células encontram-se ligadas em rede através de TCP/IP e cada uma delas tem comunicação com o respetivo controlador/autómato de acordo com a Figura 4.3 [37]. Descrevem-se ao longo deste capítulo, com mais detalhe, cada dos equipamentos referidos acima, bem como a sua integração no sistema [37, 38].

Cada estação de trabalho tem, pelo menos, uma aplicação cliente que controla um equipamento e que permite a comunicação com a BD. A Figura 4.4 representa a interface gráfica das várias aplicações cliente. Todas as aplicações têm a mesma lógica de funcionamento, independentemente do equipamento, como abordado no Capítulo 3.4, sendo que o que difere são, por exemplo, as configurações para comunicação com estes equipamentos. A interface gráfica da aplicação cliente contém vários elementos distintos, entre eles, dois campos de texto, um para visualização de toda a comunicação do porto RS232, por exemplo, e outro para mensagens de erro existentes durante a comunicação.

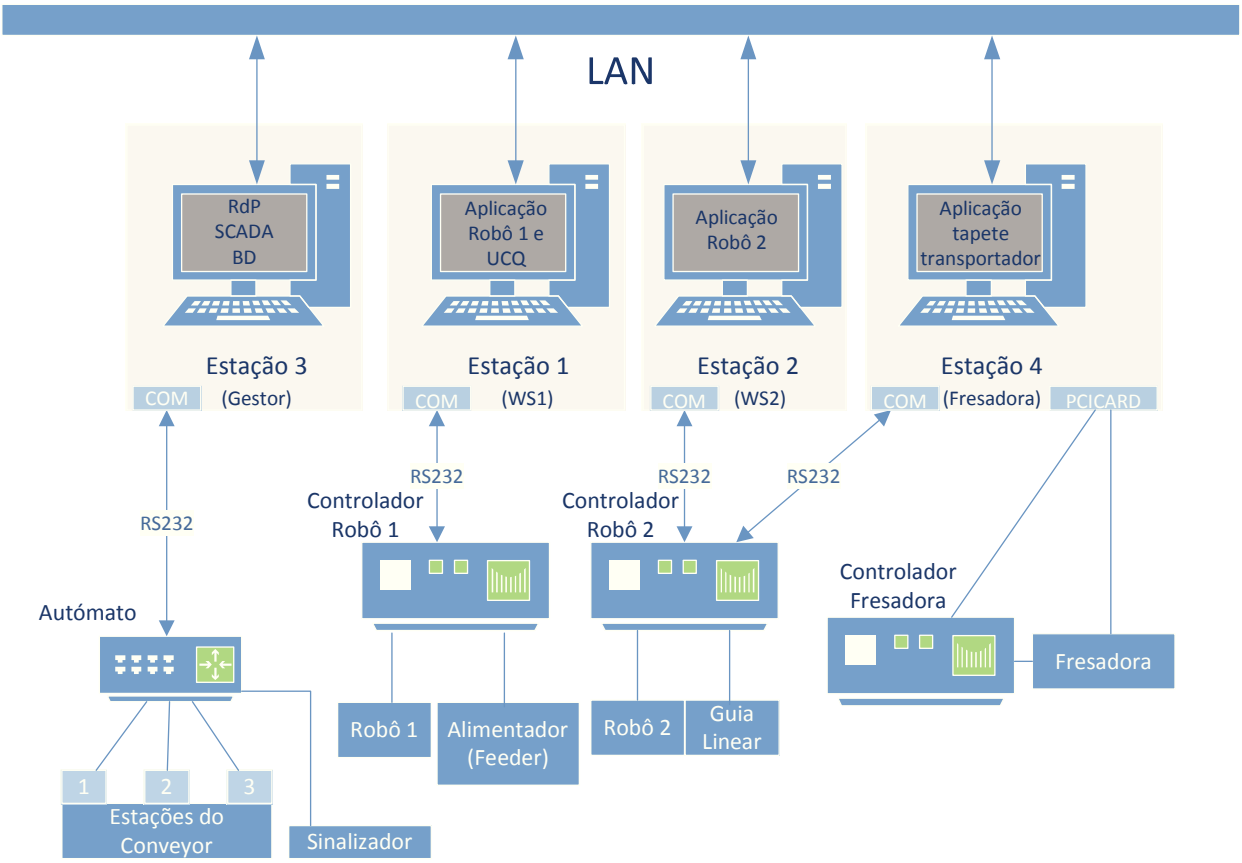


Figura 4.3: Diagrama de ligações do FMS existente no laboratório de robótica.

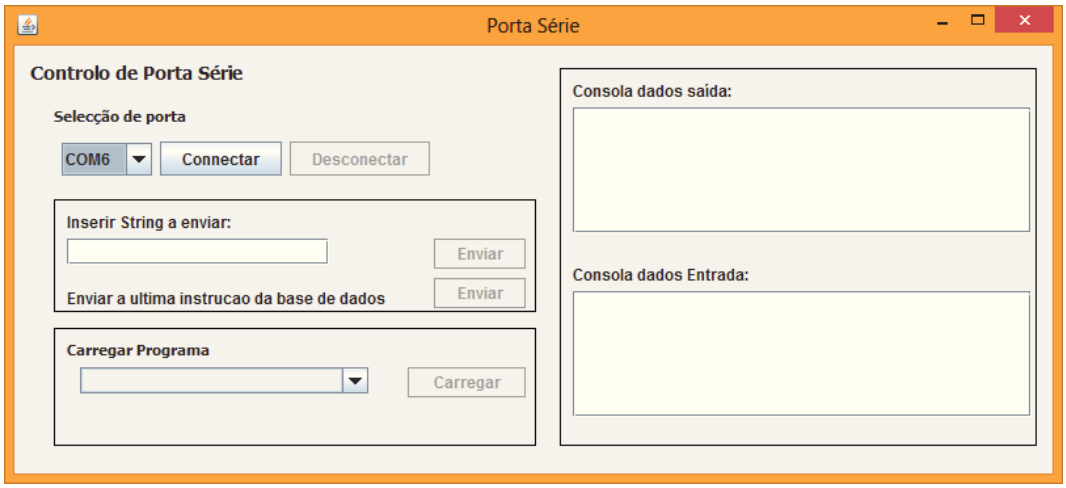


Figura 4.4: Interface das aplicações cliente.

Existe também um botão designado "Conetar"que serve para fazer a conexão da aplicação à BD e para iniciar a comunicação com a porta série.

A aplicação tem uma *drop down list* que torna possível a escolha da porta que se pretende utilizar para a realização da comunicação série. A lista mostra todas as portas

séries existentes na máquina, sendo que, caso alguma já se encontre ligada a um outro dispositivo, a consola de saída mostrará uma mensagem, a vermelho, com o erro "PORT COM is already in use".

Na mesma aplicação existe também um campo de entrada, do tipo texto, que só ficará disponível para escrita após efetuada a ligação e que permite a escrita de caracteres diretamente no porto RS232. A informação escrita no porto será guardada na tabela de registos disponível na BD.

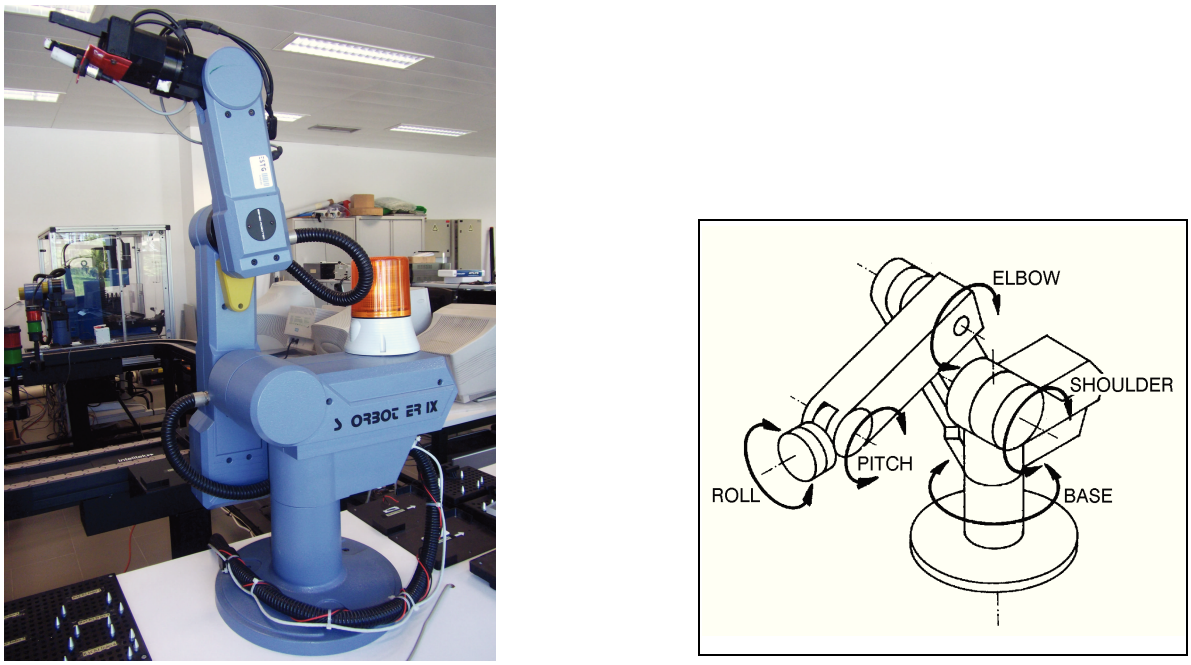
A *drop down list*, integrada no campo "Carregar Programas", serve para listar todos os programas existentes na memória da máquina (diretoria: C:/prog_tese/). Para fazer o carregamento de um programa, para o respetivo cliente, basta selecionar um elemento na lista e pressionar "Carregar".

Nesta dissertação foram desenvolvidas 4 aplicações distintas, cada uma delas com o objetivo de estar numa das células de trabalho utilizadas. Podem ser obtidas mais informações acerca das aplicações cliente no Anexo E.1.

4.1 Célula de trabalho 1

A Célula de trabalho 1 é constituída por um armazém de paletes, um *buffer* de peças de entrada, dois locais de saída (um para peças com defeito e outro para peças sem defeito), uma unidade de controlo de qualidade (UCQ) e um alimentador de peças em bruto (*feeder*), os quais se descrevem de seguida.

Na Figura 4.5 é visível o robô SCORBOT ER-IX [39]. Este robô é um manipulador robótico que tem 5 juntas, funcionamento elétrico e um elemento terminal, por exemplo uma garra, que pode ser alterado conforme as necessidades.



(a) Aspecto físico do robô.

(b) Representação das juntas.

Figura 4.5: *Robô SCORBOT-ER IX. [39].*

A Tabela 4.1 fornece informação acerca das juntas do SCORBOT. Através desta figura pode ser observado o esquema físico do robô e informação técnica adicional em função de cada uma das juntas.

Tabela 4.1: Especificações do SCORBOT [39].

Especificações do braço robótico		
Movimento do eixo	**Gama de rotação**	**Velocidade efetiva**
Eixo 1: Rotação de base	270°	79° a 112° / seg
Eixo 2: Rotação de ombro	145°	68° a 99° / seg
Eixo 3: Rotação de cotovelo	210°	76° a 112° / seg
Eixo 4: Rotação de pulso *pitch*	196°	87° a 133° / seg
Eixo 5: Rotação de pulso *roll*	737°	166° / seg

No sistema FMS estão disponíveis dois robôs e existe um controlador para cada um deles. Os robôs devem estar conectados ao respetivo controlador, que tem como objetivo efetuar o controlo dependendo das instruções recebidas no porto de comunicação RS232. A configuração para permitir a comunicação, através da porta série, está definida na Tabela 4.2.

Tabela 4.2: Parâmetros do porto RS232 para comunicação com os SCORBOT.

Parâmetro	Valor
Bits por segundo	9600
Bits de dados	8
Paridade	Nenhum
Bits de paragem	1
Controlo de Fluxo	Nenhum

O sistema do SCORBOT-ER IX contém um software que utiliza a linguagem de programação (ACL) [40]. Este software, entre outras particularidades, permite fazer o carregamento dos programas para o robô.

Foi verificado que existe uma limitação no número de caracteres associados ao nome do programa que os SCORBOT conseguem ler. Caso esse nome exceda 8 dígitos, o programa não é reconhecido, dando um erro a expressar tal situação. Assim sendo, foi criada um tabela de BD que faz a tradução dos nomes associados ao programa, com os nomes associados à ação a executar (representada através da etiqueta da RdP). Na Figura 4.6 é possível observar os campos da tabela e os respetivos tipos de variáveis.

	Id [PK] integer	transicao_petri character varying	a_disparar character var
1	2	A.R1.transf_locpal_bufent	GT016
2	3	A.R1.transfpecent_bufent	GT015
3	4	A.R1.transfbufent_pallocl	PT001
4	5	A.R1.transflocl_bufucq	GT001
5	6	A.R1.transfbufucq_ucq	PT011
6	7	A.R1.transfbufucq_locpal	PT002
7	8	A.R1.transfucq_saidaBoas	PT003
8	9	A.R1.transfucq_saidaDefeituosas	PT004
9	10	A.R2.transfloc3_bufup	GT006
10	12	A.R2.transfpec_bufup_up	PT023
11	14	A.R2.transfbufup_loc3	PT016
*			

Figura 4.6: Tabela de tradução entre nomes de ações a executar e programas carregados nos SCORBOT.

Na Figura 4.7 é possível visualizar os vários equipamentos físicos necessários para que o SCORBOT esteja operacional.

A consola de programação, *Teach Pendant*, permite efetuar o controlo manual do SCORBOT através da operação e controlo dos eixos ligados ao controlador. Além disso possui também um botão de emergência.

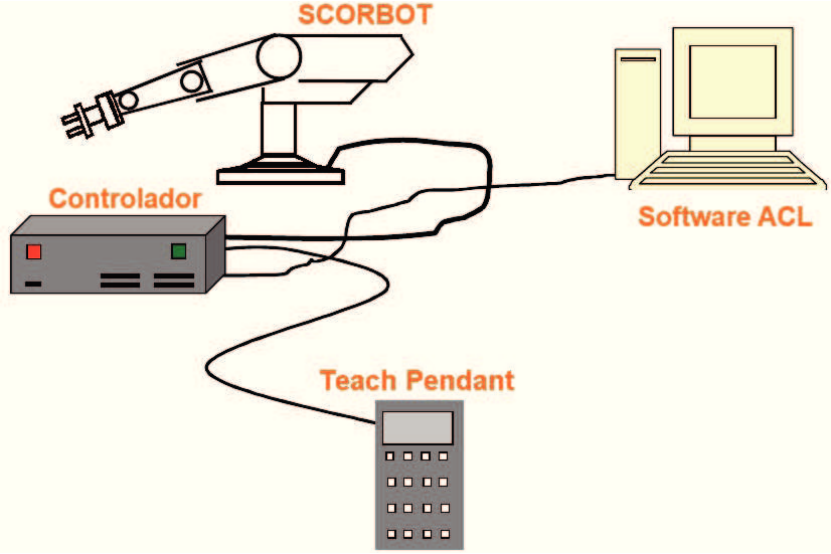


Figura 4.7: Equipamento físico necessário para funcionamento do SCORBOT [41].

O alimentador de peças (*feeder*) é um equipamento que disponibiliza peças em bruto ao sistema, para posteriormente serem maquinadas na fresadora CNC. O alimentador de peças é considerado o ponto de entrada do sistema e pode ser visto na Figura 4.8.

Figura 4.8: *Feeder* com as respetivas peças em bruto.

A Figura 4.9 representa o armazém de peças manufaturadas que existe no sistema real. Existem dois *racks* para as peças manufaturadas sem defeito e um outro para as peças defeituosas.

Figura 4.9: *Rack* ou armazém de peças manufaturadas.

A unidade de controlo de qualidade (UCQ) que se pode ver na imagem 4.10 serve para certificar se as peças processadas estão dentro dos parâmetros desejados, ou seja, para fazer uma distinção entre peças sem defeito e as peças defeituosas.

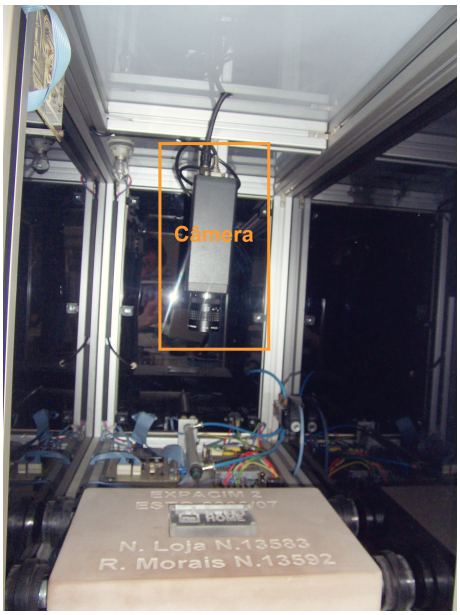


Figura 4.10: Unidade de controlo de qualidade para peças processadas.

Relativamente à célula de trabalho 1, as RdP associadas podem ser consultadas através das Figuras 4.11 e 4.12. Os lugares que se encontram associados a ações, estão enumerados nas Tabelas 4.3 e 4.5, sendo que a última diz respeito à unidade de controlo de qualidade. Os eventos externos que estão associados a transições da rede podem ser consultados na Tabela 4.4 e na Tabela 4.6, respetivamente para o local A e para a unidade de controlo de qualidade.

Esta célula de trabalho interage com vários equipamentos e contém duas aplicações. Existe a aplicação cliente do robô e a aplicação cliente da UCQ. O funcionamento geral desta célula de trabalho baseia-se na entrada e saída de peças do sistema. Para a entrada de peças o robô 1 vai buscar uma palete vazia ao armazém de paletes e coloca-a no *buffer* de entrada, de seguida vai buscar uma peça ao alimentador de peças em bruto e coloca-a na palete. Posteriormente a palete com peça é colocada no tapete transportador. Para a saída de peças do sistema o robô 1 começa por retirar a palete com peça processada do tapete transportador e coloca-a no *buffer* de UCQ, depois retira a peça e coloca-a na UCQ. A última etapa é enviar a peça inspecionada para a saída respetiva, caso esteja com ou sem defeitos.

Tabela 4.3: Descrição dos lugares, com ações associadas ao local A.

Descrição dos lugares da RdP	Ações
A.R1.transf_locpal_bufent	Transferir palete do armazém de paletes para buffer de entrada.
A.R1.transfpecent_bufent	Transferir peça em bruto para buffer de entrada, que contém palete.
A.T.pararLoc1	Parar navetes nas posições A e C.
A.R1.transfbufent_palloc1	Transferir palete, com peça em bruto, para navete no Local A.

Tabela 4.4: Descrição das transições, com eventos externos associados ao local A.

Descrição das transições externas da RdP	Eventos
E.R1.fimtransf_locpal_bufent	Fim da transferência de palete do armazém de paletes para o *buffer* de entrada.
E.R1.fimtransf_pecent_bufent	Fim da transferência de peça em bruto para palete no *buffer* de entrada.
E.T.fim_pararLoc1	Parar navetes nas posições A e C.
E.R1.fimtransf_bufent_loc1_1	Fim da transferência de peça do *buffer* de entrada para navete 1, no local A.
E.R1.fimtransf_bufent_loc1_2	Fim da transferência de peça do *buffer* de entrada para navete 3, no local A.

Tabela 4.5: Descrição dos lugares, com ações associadas à unidade de controlo de qualidade.

Descrição dos lugares da RdP	Ações
A.T.pararLoc1_ucq	Parar navetes que se encontram no tapete transportador.
A.R1.transfloc1_bufucq	Transferir palete, com peça processada, do local A para o *buffer* de UCQ.
A.T.libertarLoc1_ucq	Libertar navetes que se encontram no tapete transportador.
A.R1.transfbufucq_ucq	Transferir peça processada do *buffer* de UCQ para unidade de controlo de qualidade.
A.UCQ.activa	Iniciar o teste à peça, na unidade de controlo de qualidade (UCQ).
A.R1.transfbufucq_locpal	Transferir palete vazia do *buffer* de UQC para o armazém de paletes.
A.R1.transfucq_saidaBoas	Transferir peça da unidade de controlo de qualidade para a saída de peças boas.
A.R1.transfucq_saidaDefeituosas	Transferir peça da unidade de controlo de qualidade para a saída de peças defeituosas.

Tabela 4.6: Descrição das transições, com eventos externos associados à unidade de controlo de qualidade.

Descrição das transições externas da RdP	Eventos
E.T.fim_pararLoc1_ucq	Tapete parado com navetes nos locais A e C.
E.R1.fimtransfloc1_bufucq	Fim da transferência de palete, com peça processada, do local A para o *buffer* de entrada.
E.R1.fimtransfbufucq_ucq	Fim da transferência de peça, do *buffer* de entrada, para a unidade de controlo de qualidade.
E.UCQ.fim_activa	Iniciar o controlo de qualidade da peça, na unidade de controlo de qualidade (UCQ).
E.R1.fim_transfbufucq_locpal	Fim da transferência de palete vazia, do *buffer* de UCQ, para o armazém de paletes.
E.R1.fimtransfucq_bufucq	Fim da transferência de peça analisada, da unidade de controlo e qualidade para a saída.

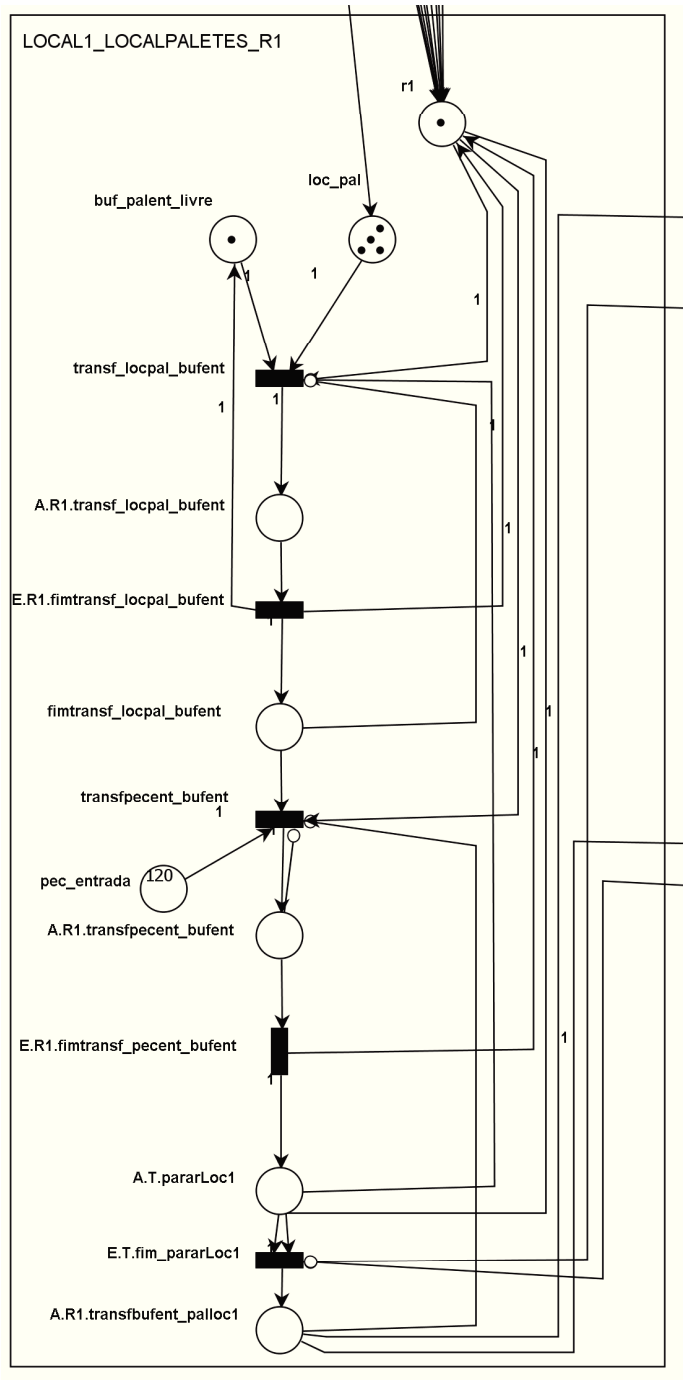


Figura 4.11: Representação da RdP correspondente ao local A e *buffer* de entrada.

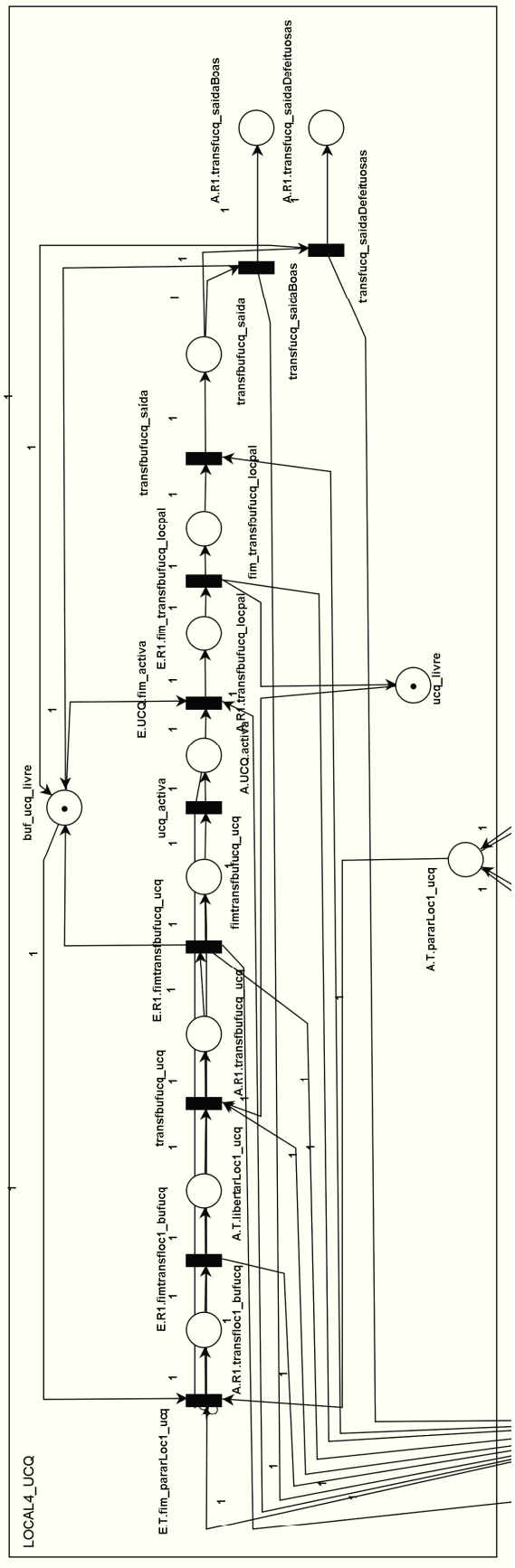


Figura 4.12: Representação da RdP correspondente à Unidade de Controlo de Qualidade.

4.2 Célula de trabalho 2

A célula de trabalho 2 contém uma guia linear, representada na Figura 4.13, que é utilizada como método de deslocamento horizontal para o robô 2, acrescentando um sexto grau de liberdade. O robô 2 é semelhante ao robô 1 mas, neste caso, permite a interação entre o local C, o *buffer* e a unidade de processamento (CNC).

Figura 4.13: Guia linear.

Relativamente às células de trabalho 2 e 4, a RdP associada pode ser consultada através da Figura 4.15. Os lugares que se encontram associados a ações estão descritos na Tabela 4.7. Os eventos externos que estão associados a transições da rede podem ser consultados na Tabela 4.8.

Tabela 4.7: Descrição das transições, com eventos externos associados ao local C.

Descrição dos lugares da RdP	Ações
A.T.pararLoc3	Parar navetes que se encontram no tapete transportador.
A.R2.transfloc3_bufup	Transferir palete, com peça em bruto, do local C para o *buffer* do local C.
A.T.libertarLoc3	Libertar navetes que se encontram no tapete transportador.
A.R2.transfpec_bufup_up	Transferir peça em bruto, do buffer do local C, para a unidade de processamento.
A.CNC.up_activa	Iniciar o processamento da peça, na unidade de processamento (CNC).
A.R2.transfpecup_bufup	Transferir peça processada, de unidade de processamento, para *buffer* com palete.
A.R2.transfbufup_loc3	Transferir palete, com peça processada, para navete no Local C.

Tabela 4.8: Descrição dos lugares, com ações associadas ao local C.

Descrição das transições externas da RdP	Eventos
E.T.fim_pararLoc3	Tapete parado com navetes nos locais A e C.
E.R2.fimtransfloc3_bufup	Fim da transferência de palete, com peça, do local C para o buffer.
E.R2.fimtransfbufup_up	Fim da transferência de peça em bruto do *buffer* para a unidade de processamento.
E.R2.fimtransfpecup_bufup	Fim da transferência de peça processada da unidade de processamento para o *buffer*.
E.T.fim_pararLoc3	Parar navetes nas posições A e C.
E.R2.fimtransfbufup_loc3	Fim da transferência de palete, com peça processada, do *buffer* para o local C.

O equipamento de processamento, uma fresadora de comando numérico, representada na Figura 4.14, permite manufaturar as peças em bruto. Após a transformação, estas saem da fresadora CNC (transportadas pelo robô 2) como peças processadas. Esta fresadora Controlo numérico computorizado (*Computer Numerical Control*) (CNC) tem um controlador suportado por um PC dedicado.

O funcionamento desta célula de trabalho baseia-se no processamento de peças na fresadora CNC, sendo que as peças que entram nela são peças em bruto e as peças que saem são peças processadas. O transporte das peças do tapete para a CNC e vise-versa é feito através do robô 2.

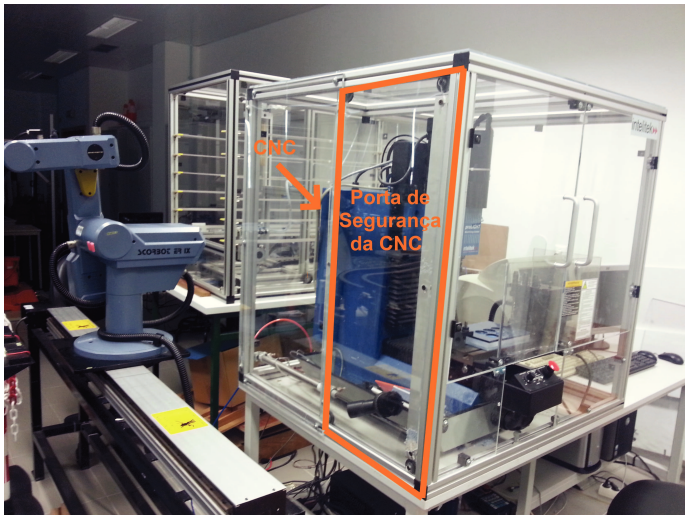


Figura 4.14: CNC que transforma peça em bruto em peça processada.

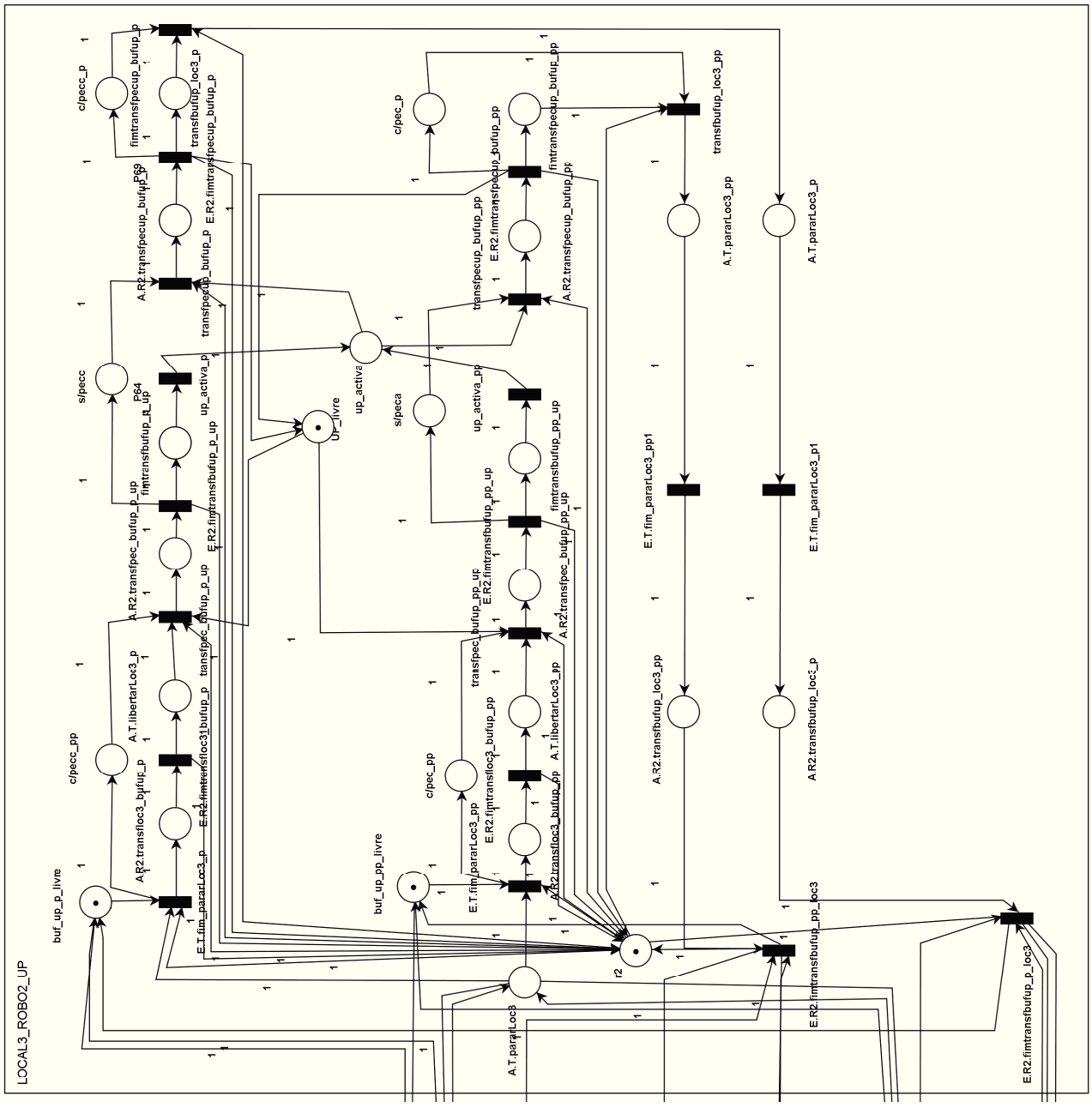


Figura 4.15: Representação da RdP correspondente ao local C e Unidade de Processamento.

4.3 Célula de trabalho 3

A célula de trabalho 3 é constituída pela estação de trabalho central e que, vulgarmente, se encontra fisicamente afastada do local de operações. Esta estação está encarregue de suportar os serviços centrais do sistema. Neste caso aloja o servidor de BD (PostgreSQL), o servidor do sistema SCADA (IntegraXor) e o programa de modelação e controlo do sistema baseado em RdP (PIPE).

De modo a centralizar o sistema através de um servidor de BD, foram criadas duas tabelas. Uma serve para fazer todos os registos do sistema (inicializações, ações e eventos que ocorrem) e outra para fazer a gestão das variáveis associadas a componentes gráficos (etiquetas) do IntegraXor. Através das Figuras 4.16 e 4.17, respetivamente para registo dos acontecimentos do sistema e para armazenamento dos valores das variáveis associadas ao SCADA, é possível verificar a estrutura das tabelas de BD.

	id [PK] serial	event_action character varying	time_stamp timestamp without time zone	others character varying
1	1	A.R1.transfloc3_bufup_p	2014-03-06 23:39:29.818	done
2	2	E.R1.fimtransfloc3_bufup_p	2014-03-08 14:18:52.091	done
*				

Figura 4.16: Aspeto da tabela de BD que guarda os registos de acontecimentos do sistema.

	id [PK] serial	name character varying	value integer	valuestring character varying(30)
1	1	pec_ent	0	
2	2	r1_ocup	0	
3	3	r1_disp	1	
4	4	r2_ocup	0	

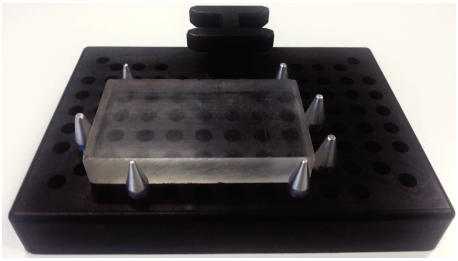
Figura 4.17: Aspeto da tabela de BD que serve para gerir as variáveis gráficas associadas ao IntegraXor (etiquetas).

4.4 Célula de trabalho 4

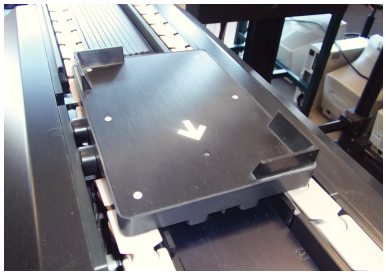
A Célula de trabalho 4 é constituída por vários equipamentos e também pela aplicação que faz o controlo do tapete transportador.

A palete, representada na Figura 4.18, é um suporte que tem como objetivo servir de base para o transporte, pelos robôs, das peças presentes no sistema. As peças pode ser processadas ou por processar. Para que uma peça seja transportada através do tapete transportador tem de estar colocada numa palete.

A navete, objeto que pode ser visualizado na Figura 4.18, é uma bandeja que viaja sobre o tapete transportador e que foi projetada para efetuar o transporte de paletes. As navetes estão encarregues de efetuar diversos tipos de transporte no tapete transportador, seja de palete com peça por processar (em bruto), com peça processada ou ainda sem qualquer palete.



(a) Palete para transporte de peças em bruto ou peças processadas.



(b) Navete.

Figura 4.18: Palete e navete utilizadas para o transporte de peças.

A comunicação como autómato que gere o funcionamento do tapete é feita através de porta série, cuja configuração se encontra na Tabela 4.9.

Tabela 4.9: Parâmetros do porto RS232 para comunicação com tapete.

Parâmetro	**Valor**
Bits por segundo	9600
Bits de dados	7
Paridade	Par
Bits de paragem	2
Controlo de Fluxo	Nenhum

Para controlar corretamente o tapete é necessário ter conhecimento das mensagens que o autómato programável industrial que comanda o funcionamento do transportador é capaz de interpretar e usar. Estas mensagens encontram-se nas Figuras 4.10, 4.11 e 4.12, respetivamente para transportar navete para determinada posição, terminar o transporte de determinada navete e libertar navete de determinada posição [38].

Tabela 4.11: Mensagem a enviar para o transporte de navetes:

Função	**Mensagem a enviar**
Terminar transporte Navete 1	@00WD000900995A*
Terminar transporte Navete 2	@00WD0010009952*
Terminar transporte Navete 3	@00WD0011009953*

Tabela 4.10: Mensagem a enviar para transporte de navete e respetiva confirmação.

Função mover	Mensagem a enviar	Mensagem de confirmação
Navete 1 para Posição 1	@00WD000900015B*	@00EX0001000100015C*
Navete 1 para Posição 2	@00WD0009000258*	@00EX0002000100015F*
Navete 1 para Posição 3	@00WD0009000359*	@00EX0003000100015E*
Navete 2 para Posição 1	@00WD0010000153*	@00EX0001000200015F*
Navete 2 para Posição 2	@00WD0010000250*	@00EX0002000200015C*
Navete 2 para Posição 3	@00WD0010000351*	@00EX0003000200015D*
Navete 3 para Posição 1	@00WD0011000152*	@00EX0001000300015E*
Navete 3 para Posição 2	@00WD0011000251*	@00EX0002000300015D*
Navete 3 para Posição 3	@00WD0011000350*	@00EX0003000300015C*

Tabela 4.12: Mensagem a enviar para libertar navete de determinada posição.

Função	Mensagem a enviar
Libertar Navete na Posição 1	@00WD004800015E*
Libertar Navete na Posição 2	@00WD004900015F*
Libertar Navete na Posição 3	@00WD0050000157*

A RdP associada pode ser consultada através da Figura 4.20. O funcionamento do tapete transportador é feito através de quatro transições, que dizem respeito a eventos externos à RdP. Os eventos externos que estão associados a transições da rede podem ser consultados na Tabela 4.13.

Tabela 4.13: Descrição das transições, com eventos associados ao tapete transportador.

Descrição das transições da RdP	Eventos
E.T.arrancouLoc1	Tapete libertou navetes do local A e do local B está em rotação.
E.T.chegouLoc1	Chegada de peças. Uma peça no local A e outra no local B.
E.T.arrancouLoc2	Tapete libertou navetes do local B e do local A e está em rotação.
E.T.chegouLoc2	Chegada de peças. Uma peça no local B e outra no local A.

A gestão deste comportamento é parcialmente feita pela a aplicação cliente, sendo esta responsável por gerar os eventos externos para a BD em função das mensagens enviadas pelo controlador do tapete, bem como enviar as mensagens para o controlador do tapete em função das ações recebidas da BD. O seu funcionamento foi idealizado e definido, para este trabalho, de acordo com as etapas que podem ser verificadas na Figura 4.19.

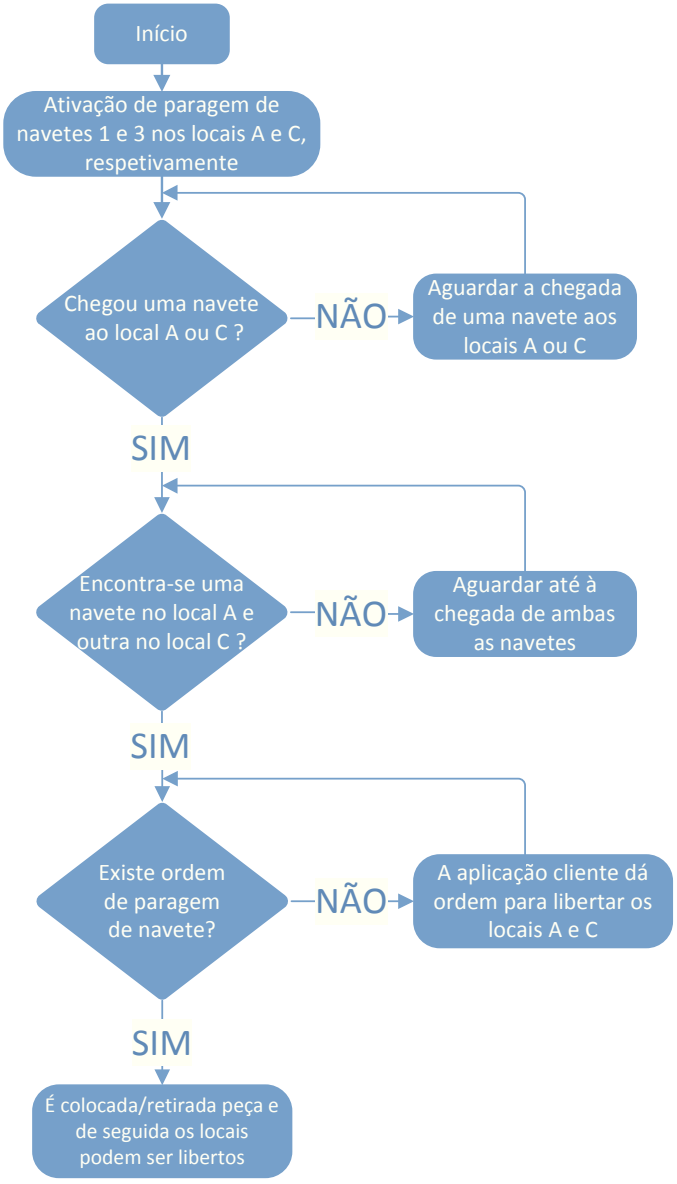


Figura 4.19: Diagrama que representa o funcionamento do tapete transportador.

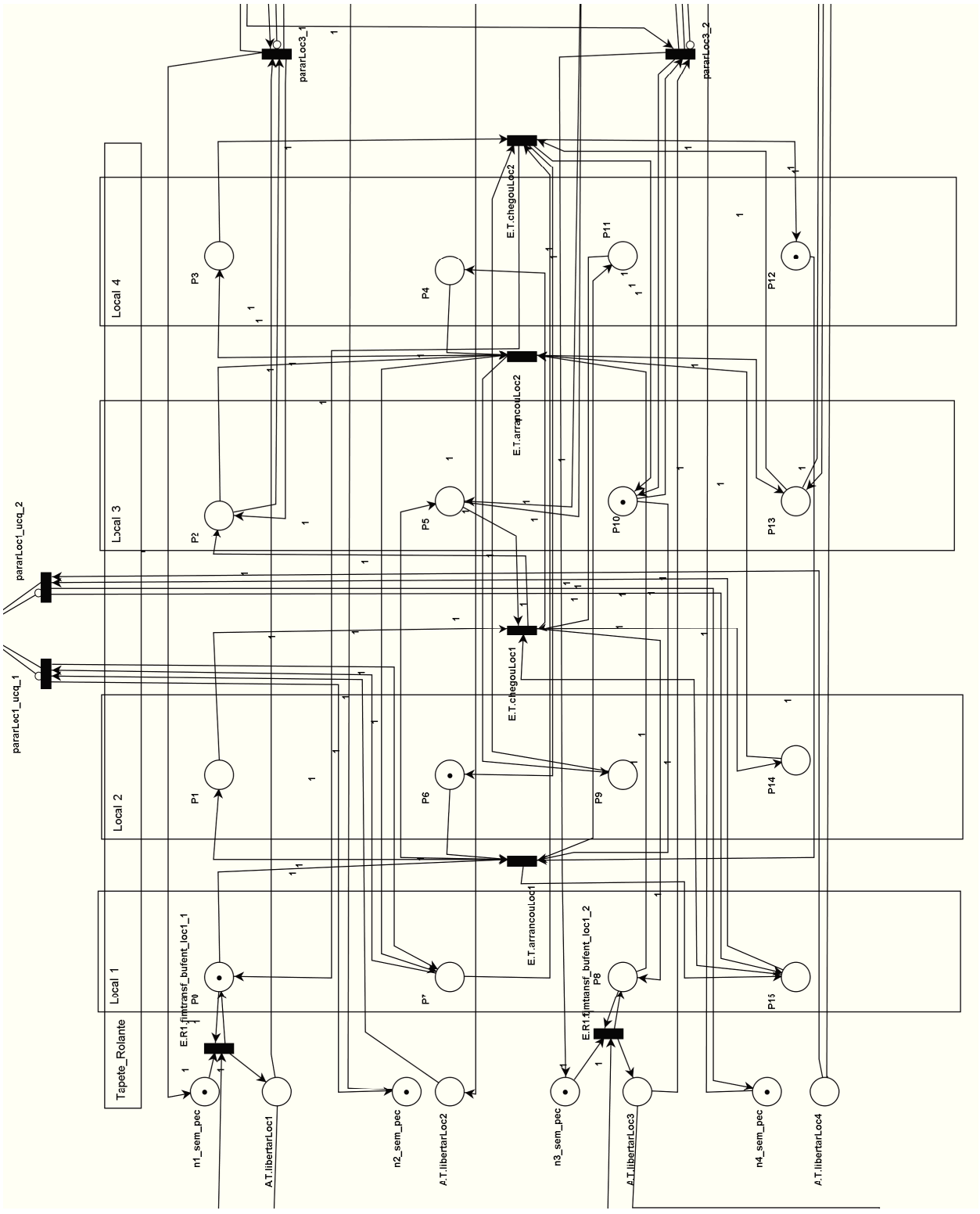


Figura 4.20: Representação da RdP correspondente ao tapete transportador.

4.5 Descrição do funcionamento do FMS modelado

Os diversos locais do tapete estão numerados de acordo com o definido para a modelação do sistema. O tapete está em constante movimento e, para se efetuar a paragem de uma navete nos locais local "A"ou "B", é necessário enviar ao autómato ordem para tal. O mesmo acontece quando se pretende libertar um local, se este contiver uma navete parada.

Existem ainda locais de passagem no sistema que são denominados de *Buffers*. Estes servem para armazenamento temporário das paletes e/ou peças. O processo de manufatura de uma peça tem várias fases, e as mesmas têm uma ordem lógica de execução para que as peças sejam manufaturadas.

O primeiro passo para dar inicio à produção é a inicialização do sistema. De seguida o robô 1 vai buscar uma palete ao armazém ASRS e coloca-a no *Buffer*. Caso a palete não tenha peça em bruto, o mesmo robô vai buscar a peça ao alimentador de peças e coloca-a na palete. Cada uma das navetes está numerada e o tapete transportador tem capacidade de deteção das mesma, nos quatro locais de passagem (A,B,C ou D). De seguida o sistema aguarda a paragem de uma navete número 1 ou 3, pois foi definido que estas estariam encarregues do transporte de peças em bruto. Assim que o robô recebe a informação que a navete está disponível, coloca a palete na navete. Quando a mesma navete passar diante do robô 2, este, dependendo da ordem dada pela RdP, retira ou não a palete e coloca-a no *Buffer*.

A fresadora CNC, como modo de proteção, tem uma porta de segurança. Após a abertura da mesma, o robô vai buscar a peça em bruto à palete que se encontra no *Buffer* e coloca-a na fresadora, fechando a porta de seguida. No fim da peça estar manufaturada, a porta de segurança é aberta e o robô 2 vai buscar a peça à fresadora, colocando-a na palete. Posteriormente é colocada a palete na navete e, quando a mesma passar diante do robô 1, este retira-a e coloca-a no *Buffer*. Uma vez que a peça processada se encontra no *Buffer*, o robô poderá efetuar o seu transporte para a unidade de controlo de qualidade e, no fim desse processo, para a saída, existindo uma distinção entre peças com e sem defeito.

A partir do FMS existente no laboratório, foram feitas algumas opções de forma a simplificar e estruturar melhor a modelação e controlo do processo, nomeadamente:

- O sistema terá 4, e apenas 4, navetes no tapete transportador;
- As navetes 1 e 3 transportam apenas peças por processar, enquanto as navetes 2 e 4 transportam peças processadas;
- Não faz parte do foco da dissertação a maquinação das peças na CNC, pelo que o

processo de transformação de peça em bruto para peça processada é simulado com um tempo de espera (*delay*), depois da peça ser introduzida na máquina;

- A tomada de decisão de peça com ou sem defeito, a cargo da unidade de controlo e a qualidade, é simulada com base numa probabilidade pré-definida.

Ficou definido que a RdP global do sistema seria um conjunto de quatro RdP interligadas entre si, em que cada uma delas representa uma área de trabalho. Assim foram desenvolvidas 4 RdP com nomes para a sua designação de: Local1-Robo1, Local3-Robo2-UP, Tapete, Local1-UCQ (ilustradas previamente), e a RdP global.

No que diz respeito ao estado inicial do sistema (que é fundamental, como já foi referido anteriormente) devem ser cumpridas algumas regras de inicialização. Existem regras tanto para o sistema SCADA como para a RdP sendo que o objetivo é o mesmo para ambos, conhecer o estado inicial e a partir daí iniciar a produção de forma segura:

- Devem existir quatro paletes disponíveis no armazém de paletes;
- Devem existir peças em bruto no alimentador de peças;
- Os locais A,B,C e D devem estar com uma, e uma só, navete;
- Os *buffers*, bem como, a unidade de controlo de qualidade, a unidade de processamento, os robôs 1 e 2, o autómato e as saídas devem estar operacionais e livres.

4.6 O sistema SCADA desenvolvido

Esta secção descreve a funcionalidade implementada no sistema SCADA para interação com o utilizador. Ao inicializar o servidor do IntegraXor, por omissão é aberta, uma página do navegador de Internet. Inicialmente é visível uma página apenas com um botão, como demonstra a Figura 4.21. Este botão serve para iniciar todas as variáveis gráficas do sistema de modo a que a visualização esteja de acordo com o estado inicial do sistema físico real e da RdP.

Figura 4.21: Página Web para início do sistema SCADA.

No cabeçalho da página Web que está no navegador, é possível observar um separador denominado "Botões de Ordem". Este separador permite ao utilizador a simulação de eventos externos ao sistema, o que se pode tornar bastante útil caso se pretenda "simular", para efeito de testes, o funcionamento de todo o sistema.

Figura 4.22: Página Web para início do sistema SCADA.

Existe um outro separador que permite ao utilizador obter uma visualização geral de todo o sistema, em que a informação é atualizada em tempo real durante a sua execução. Neste separador está representado o sistema modelado através de imagem em duas dimensões. Ainda neste separador existem dois botões que servem para dar a ordem de marcha ou ordem de paragem ao sistema. Isto é uma das caraterísticas dos sistemas SCADA, permitir o controlo, pelo menos parcial, do sistema físico.

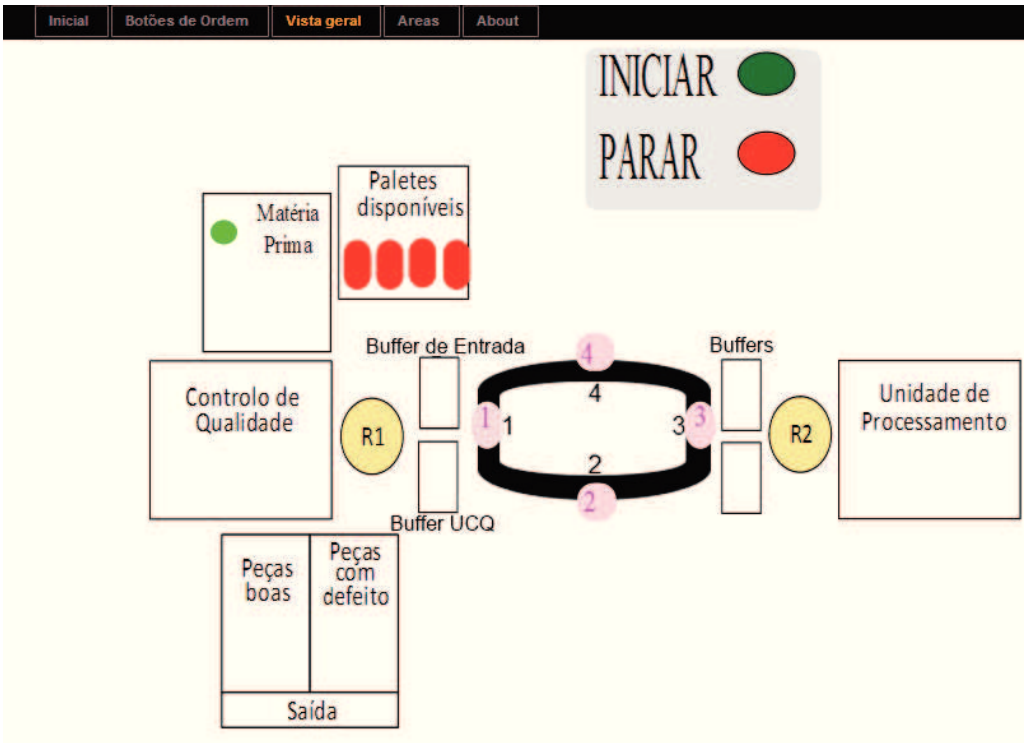


Figura 4.23: Interface de monitorização do sistema SCADA.

Capítulo 5

Resultados e Conclusões

Este trabalho consistiu no desenvolvimento de uma solução integrada de modelação, análise, controlo e monitorização de um sistema flexível de produção. Foi utilizado um sistema SCADA para monitorização e intervenção manual, um sistema baseado em RdP para a supervisão e controlo do sistema, aplicações cliente para o controlo local de cada célula de trabalho e um sistema de base de dados para interligar todos os componentes.

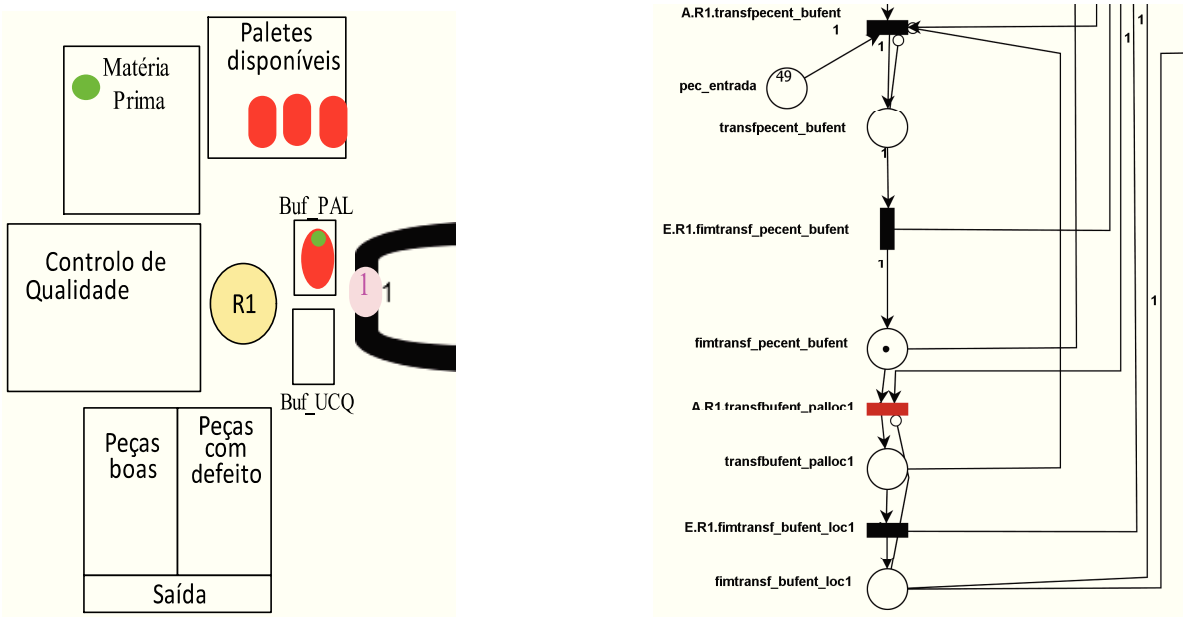
De modo a permitir a integração entre os vários elementos constituintes do sistema, flexível de produção (SCADA, RdP e aplicações cliente das células de trabalho), recorreu-se a um sistema centralizado de informação, utilizando um sistema de BD, baseado no PostgreSQL. Para que a RdP comunicasse com a BD, foi necessário alterar o código fonte da ferramenta PIPE, usada para a modelação e visualização da rede.

5.1 Resultados

Como referido anteriormente, o estado inicial do sistema real é relevante. As condições iniciais foram iguais em ambos os sistemas *online* e *offline*, sendo que apenas diferiram no número de matéria prima à entrada do sistema. Para o sistema em modo de simulação foram definidas 120 peças no alimentador de peças em bruto, enquanto as peças no sistema real foram limitadas a 10 unidades. Neste trabalho o estado inicial do sistema SCADANet correspondeu 10 peças em bruto no alimentador de peças, 4 paletes sem peças no armazém de paletes, os *buffers* e a saída de peças estavam vazios, existiam 4 navetes sem carga que estavam distribuídas por ordem ao longo do tapete transportador, a UCQ e a CNC não tinham peças no seu interior e, todos os equipamentos estavam operacionais e sem limitações.

Foram feitos testes ao sistema estando este em modo de execução (*offline*) e também em modo real (*online*), recorrendo ao programa de edição de RdP (PIPE). Através da

Figura 5.1 é possível verificar a RdP e o SCADA a serem executados em simultâneo, num determinado período de simulação, ou seja, estando o sistema em modo *online*.



(a) Sistema SCADA. (b) Rede de Petri.

Figura 5.1: Palete com peça no *buffer* de entrada, representada através dos sistemas SCADA e RdP.

Durante o período de simulação *offline*, foram fabricadas 100 peças. A simulação de peças boas ou com defeito foi efetuada definindo um peso associado a cada uma das transições internas da RdP associadas. O peso das transições de peça boa e peça má foram ajustados de forma a ter uma relação de 5% para as peças más e 95% para as peças boas, tendo no final da simulação obtido um valor de 5% de peças boas.

Relativamente à análise *offline* do sistema, efetuaram-se vários tipos de análises recorrendo aos módulos disponibilizados para o efeito pela ferramenta utilizada, o PIPE. Determinou-se que o sistema tem 1080 estados alcançáveis. Existem módulos que permitem a verificação de, por exemplo, três propriedades associadas vulgarmente às RdP (limitação, segurança e bloqueios). Verificou-se que não existem *deadlocks* (bloqueios) no sistema, dado que as transições do tapete rolante poderem disparar sempre, independentemente dos outros estados do sistema. Significa assim que, apesar de não haver *deadlocks*, existem *livelocks*, esperado para um número finito de peças de entrada.

De modo a efetuar a comunicação com os equipamentos do FMS existente no laboratório, foram desenvolvidas com sucesso quatro aplicações cliente, similares. Estas aplicações têm como objetivo a leitura e escrita na base de dados de acordo com os acontecimentos da respetiva célula de trabalho bem como controlar o equipamento de acordo com as e das ordens provenientes da RdP (ou do sistema SCADA).

5.2 Conclusões

Olhando para a RdP de forma global do ponto de vista do sistema *offline*, esta não tem bloqueios, mesmo sabendo que certas regiões podem ficar dependentes de fatores externos, como por exemplo, de peças em bruto na entrada. O sistema SCADANet não apresentou um bloqueio geral (*deadlock*), apresentando apenas o *livelock* que era esperado devido à ação do tapete transportador. Assim, o comportamento do sistema SCADANet está de acordo com o simulado e previsto anteriormente, através de simulação.

Sendo o SCADANet a integração dos vários sistemas referidos ao longo desta dissertação, foram efetuados testes ao sistema global, tendo os resultados ido ao encontro dos objetivos definidos e demonstrando que este tipo de sistemas é viável. O sistema foi desenvolvido e aplicado com sucesso, cumprindo os objetivos inicialmente propostos e servindo de elo de ligação entre o SCADA, as RdP e o sistema FMS, através da base de dados. Desta forma consegue-se um sistema integrado que não só permite supervisionar o sistema de produção, como permite modelar e analisar o mesmo sem que para isso seja necessário a utilização do sistema real.

5.3 Trabalho Futuro

Uma área de estudo que é identificada como de grande interesse para dar seguimento a esta dissertação é a de integração automática entre as ferramentas SCADA e RdP. O pretendido no futuro será efetuar as alterações no projeto que utiliza uma dessas ferramentas e conseguir, de forma possivelmente automática, que a alteração possa ser repercutida na outra ferramenta.

Uma outra área a explorar incide no facto de existirem extensões de RdP que podem ser mais vantajosas, nomeadamente em termos de visualização gráfica e desenho da RdP. Foi identificado ao longo do trabalho desenvolvido nesta dissertação que a extensão de RdP coloridas seria uma mais valia pois simplificaria a rede do ponto de vista visual.

Dando continuação ao trabalhado elaborado seria importante, futuramente, implementar mecanismos de paragem forçada, não permitidos pelas restrições atuais impostas à definição da RdP utilizada nesta dissertação. Na implementação atual são enviadas ordens de arranque aos equipamentos, ficando o sistema à espera que estes completem essas ordens (com ou sem sucesso), situação essa indicada através do envio de eventos externos. Como tal, neste momento não é possível parar um equipamento imediatamente em qualquer instante, exceto se for através do envio de uma ordem específica para o efeito. Tal significa que teríamos que ter em toda a rede deteções para essa situação, o que tornaria a rede ainda mais confusa. Sugere-se que no futuro haja uma definição automática

desses mecanismos incluída na RdP, não visível na vista normal, garantindo assim essa funcionalidade sem complicar a rede visualmente.

Neste momento toda a comunicação é feita usando TCP/IP sobre Ethernet. Dado que a comunicação é apenas de informação em tempo discreto, com baixa frequência, não será crítico para o sistema. De qualquer forma a utilização de comunicação baseada em Ethernet/IP [42] poderia ser uma mais valia em termos de fiabilidade do sistema. O desenvolvimento de clientes em hardware dedicado também é perfeitamente possível e permitiria utilizar soluções de custo mais acessível sem ser necessário recorrer a um PC. A solução desenvolvida poderá também ser implementada recorrendo a outros sistemas SCADA, mais comerciais e mais utilizados a nível industrial, desde que o acesso aos mesmos não seja fechado em termos de comunicações com os restantes sistemas.

Bibliografia

[1] A. Montezano, "Modelo de Rede de Petri de um sistema de automação de elevador de passageiros," Tese de Mestrado, Escola Politécnica da Universidade Federal do Rio de Janeiro, 2009.

[2] S. Boyer, *SCADA: Supervisory controlo and data aquisition*. ISA - The Instrumentation, Systems, and Automation Society, 2004.

[3] C. Petri, "Communication with Automata, Technical Report RADC-TR-65-377," Relatório técnico, 1966.

[4] R. Santos, "Modelagem e análise de performance de sistemas flexíveis de manufatura baseado em redes de Petri temporizadas: estudo de caso na indústria automobilística." Tese de Mestrado, Escola Politécnica da Universidade de São Paulo, 2008.

[5] R. Suri, "An overview of evaluative models for flexible manufacturing systems," *Annals of Operations Research*, vol. 3, no. 1, pp. 13–21, 1985.

[6] H. Shivanand, M. Benal, e V. Koti, *Flexible Manufacturing System*. New Age International(P) Limited, Publishers, 2006.

[7] E. Babovic e J. Velagic, "Lowering scada development and implementation costs using ptp concept," in *Information, Communication and Automation Technologies, 2009. ICAT 2009. XXII International Symposium on*, Oct 2009, pp. 1–7.

[8] D. Thomas e A. Hun, *State Machines*. Los Alamitos, CA, USA: IEEE Computer Society Press, Nov. 2002, vol. 19.

[9] F. Schumacher e A. Fay, "Transforming time constraints of a grafcet graph into a suitable petri net formalism," in *Industrial Technology (ICIT), 2013 IEEE International Conference on*, Feb 2013, pp. 210–218.

[10] N. Sakurada e D. Miyake, "Aplicação de simuladores de eventos discretos no processo de modelagem de sistemas de operações de serviços," *Gestão e Produção*, 2009.

[11] L. Gomes, "In memoriam: Prof. carl adam petri [society news]," *Industrial Electronics Magazine, IEEE*, vol. 4, no. 4, pp. 52–52, Dec 2010.

[12] K. Jensen, "Coloured Petri nets," in *Petri Nets: Central Models and Their Properties*, ser. Lecture Notes in Computer Science, W. Brauer, W. Reisig, e G. Rozenberg, Eds. Springer Berlin Heidelberg, 1987, vol. 254, pp. 248–299.

[13] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, Apr 1989.

[14] Marranghello, Norian, *Redes de Petri: Propriedades e Análise.* Universidade Estadual Paulista - UNESP, 2005.

[15] Francês, Carlos, "Introdução às redes de petri," Universidade Federal do Pará - UFPA, Relatório técnico, Agosto de 2003.

[16] R. David, "Modeling of Dynamic Systems By Petri Nets," Tese de Mestrado, Escola Politécnica da Universidade de São Paulo, 2008.

[17] D. Penha, H. Freitas, e C. Martins, "Modelagem de Sistemas Computacionais usando Redes de Petri: aplicação em projeto, análise e avaliação," Pontifícia Universidade Católica de Minas Gerais, Relatório técnico, 2004.

[18] "Deadlock and liveness properties of petri nets," in *Supervisory Control of Concurrent Systems*, ser. Systems and Control: Foundations and Applications. Birkhäuser Boston, 2006, pp. 125–151.

[19] J. Hayman, "Petri net semantics," University of Cambridge, Computer Laboratory, Relatório técnico UCAM-CL-TR-782, Jun. 2010.

[20] J. Peterson, "Petri Nets," *ACM computing surveys*, 1977.

[21] W. Vianna, P. Bringhenti, e L. Martins, *Sistema SCADA supervisório.* Instituto Federal Fluminense de Educação Ciência e Tecnologia, 2008.

[22] S. Boyer, *Supervisory Control and Data Acquisition.* Int.Society For Measurement and Control, Universidade do Estado da Pensilvânia, 1999.

[23] Barnes, Ken and Johnson, Briam and Nickelson, Reva, *Review Of Supervisory Control And Data Acquisition (SCADA) Systems.* Idaho National Engineering and Environmental Laboratory, 2004.

[24] P. Carbonnelle, "PYPL PopularitY of Programming Language index," acedido em 3-03-2014. [Online]. Disponível: https://sites.google.com/site/pydatalog/pypl/PyPL-PopularitY-of-Programming-Language

[25] R. Ramakrishnan e J. Gehrke, *Database Management Systems*, 2nd ed. Berkeley, CA, USA: Osborne/McGraw-Hill, 2000.

[26] E. Corporation, "Postgres Plus 8.4 vs. MySQL 5.5 - Feature Comparison and Commentary," EnterpriseDB Corporation, Relatório técnico, 2010.

[27] C. Pires, R. Nascimento, e A. Salgado, "Comparativo de Desempenho entre Bancos de Dados de Código Aberto," *Centro de Informática – Universidade Federal de Pernambuco (UFPE)*, 2006.

[28] *Simatic - Programming with STEP 7*, 5th ed., Siemens AG, 2010.

[29] *User Guide - Ecava IntegraXor 4.1*, Ecava Integraxor, 2013.

[30] *Genesis64 - Getting Started*, Iconics, 2010.

[31] *WinCC - Configuration Manual*, Siemens AG, 1999.

[32] K. Jones, "Automated Abstraction of Labeled Petri Nets," Tese de Mestrado, University of Utah, 2011.

[33] R. Mendes, *Modelagem e controle de sistemas a eventos discretos*. Faculdade de Engenharia Elétrica - UNICAMP, 1994.

[34] J. Oliveira, "Modelação e integração em sistemas flexíveis de produção," Tese de Mestrado, Universidade Nova de Lisboa, 1995.

[35] N. Dingle, W. Knottenbelt, e T. Suto, "Pipe2: A tool for the performance evaluation of generalised stochastic petri nets," *SIGMETRICS Perform. Eval. Rev.*, vol. 36, no. 4, pp. 34–39, 2009.

[36] N. Anastasiou e W. Knottenbelt, *Deriving Coloured Generalised Stochastic Petri Net Performance Models from High-precision Location Tracking Data*, 2013.

[37] S. Filipe, "Sistema CIM do Laboratório de Robótica Manual Simplificado do Utilizador," IPL - ESTG, Relatório técnico, 2010.

[38] A. Rolo, "Adaptação de um Software Livre de Controlo de Sistemas Integrados de Fabrico (CIM) ao Sistema Existente no Laboratório de Robótica," Tese de Mestrado, Escola Superior de Tecnologia e Gestão de Leiria, 2011.

[39] "SCORBOT-ER IX, User's manual," Eshed Robotec, Relatório técnico, 1996.

[40] *ACL (Advanced Control Language) Versions 1.43, F1.44 and ATS (Advanced Terminal Software) Version 1.44: Reference Guide for Controller A.* Eshed Robotec, 1991.

[41] A. Ferrolho e M. Crisóstomo, "Software Development to Control the Scorbot ER VII Robot With a PC," *Proceedings of the 5th WSEAS International Conference on Mathematical Methods and Computational Techniques in Electrical Engineering*, 2003.

[42] J. Warren, "Ethernet/ip applications for electrical industrial systems," in *Industry Applications Society Annual Meeting, 2009. IAS 2009. IEEE*, Oct 2009, pp. 1–5.

Anexo A

IntegraXor

A.1 Guia de criação de um novo projeto

Para criar um novo projeto no IntegraXor abre-se a aplicação *IntegraXor Editor* e, de seguida, executam-se os seguintes passos:

- File (Figura: A.1);
- New File;
- Preenchem-se os campos: Name e Location (Figura:A.2);
- Clicar em OK.

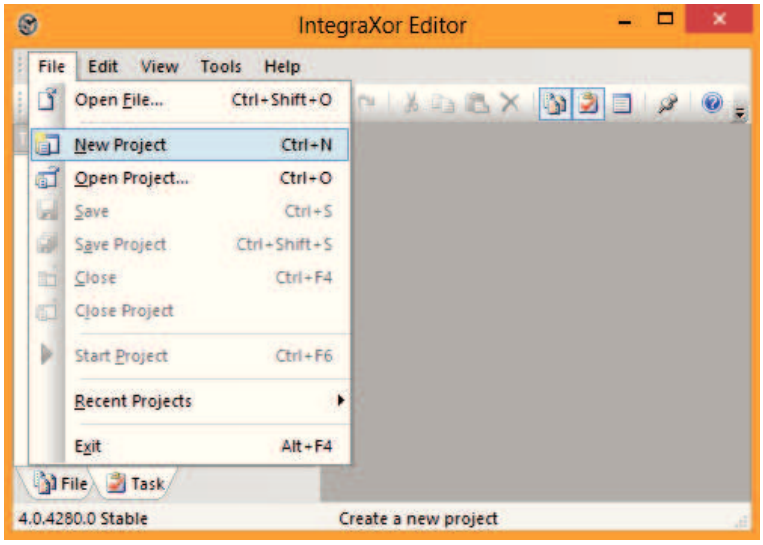


Figura A.1: Criação do novo projeto no IntegraXor.

Após estes passos estará criado o novo projeto.

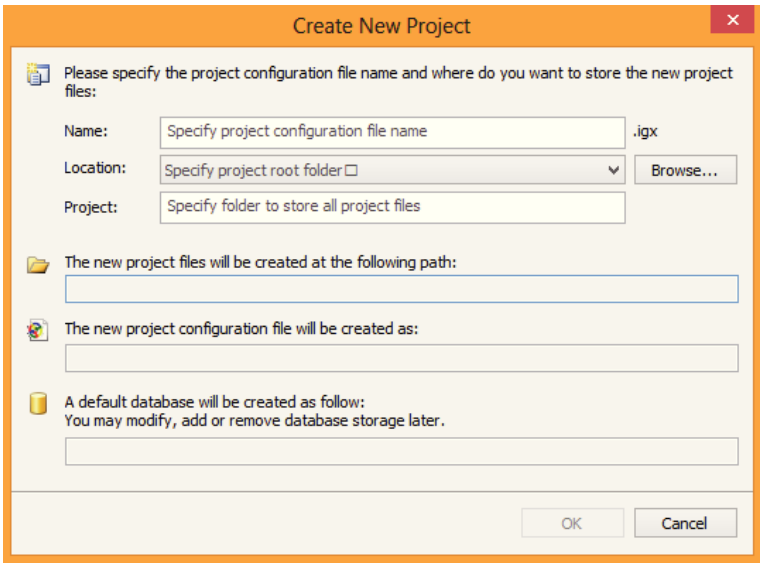


Figura A.2: Preenchimento dos campos de configuração do projeto.

A.2 Adicionar/alterar a Base de Dados

Como pode ser verificado na Figura A.3, por predefinição, a base de dados do IntegraXor é um ficheiro do tipo .MDB, ou seja, associado ao Microsoft Access. No entanto, o IntegraXor permite utilizar qualquer base de dados que suporte Open Database Connectivity (ODBC) logo, podem ser usados diferentes sistemas gestores de bases de dados, entre os quais se encontram:

- Microsoft SQL 2005 Express Edition [1]
- Oracle SQL Developer [2]
- PostgreSQL (utilizado) [3]

Existem duas maneiras distintas de fazer a interligação entre a base de dados e o IntegraXor: ou através do nome da fonte de dados ODBC ou através da *string* de conexão. Nesta dissertação a comunicação do software PostgreSQL com o IntegraXor foi feita através da *string* de comunicação, como pode constatar pela Figura A.3.

Para eliminar possíveis falhas na conexão deverá ser efetuado o teste de comunicação. Este teste faz-se clicando no botão "*Test Connection*", após a introdução da *string* de conexão pretendida.

Antes de ser criada uma etiqueta no IntegraXor deve saber-se qual a sua finalidade: deste modo é mais percetível o tipo de variável apropriado para o efeito. Neste projeto

[1] http://www.microsoft.com/en-us/download/details.aspx?id=21844
[2] http://www.oracle.com/technetwork/developer-tools/sql-developer/downloads/index.html
[3] http://www.postgresql.org/download/

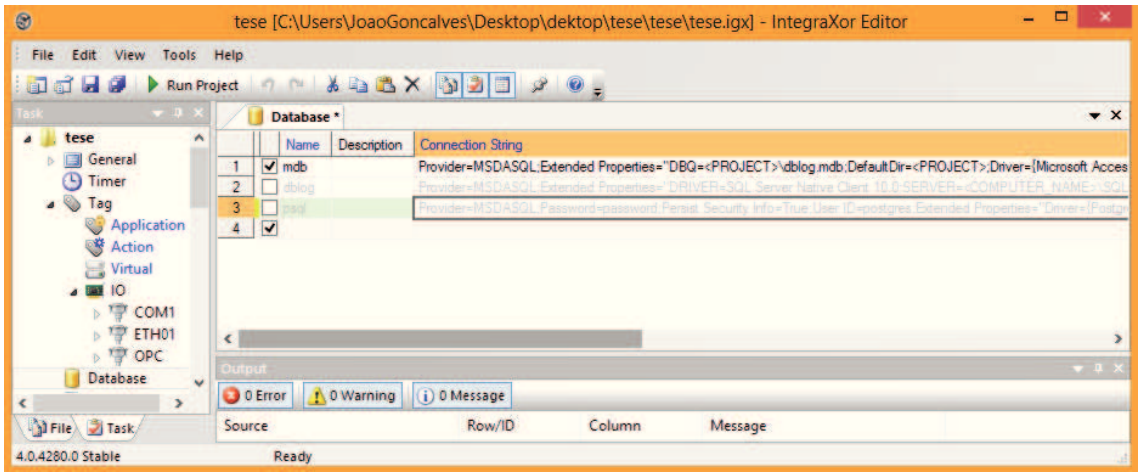


Figura A.3: *String* que permite a conexão entre o IntegraXor e o PostgreSQL.

existem diversos tipos de variáveis: booleanas, vetores de caracteres ou inteiras, dependendo da função que estão a desempenhar e para a qual foram criadas. A secção A.3 demonstra qual o conjunto de ações que devem ser executadas para criar e/ou editar as etiquetas, aqui referidas.

A.3 Criar/Editar e eliminar etiquetas

De modo a criar, editar e/ou eliminar etiquetas virtuais (*tag's*), do projeto SCADA, deverão ser executadas as seguintes tarefas, como mostra a Figura A.4:

- Na barra de tarefas, à esquerda, deverá ser selecionada a subsecção "*Tag*".

- Clicar em "Virtual", na mesma barra de tarefas, visível na Figura A.4.

De seguida irá ficar acessível a tabela relativa a todas as etiquetas virtuais definidas no projeto. Esta poderá ser editada e guardada de acordo com as necessidades.

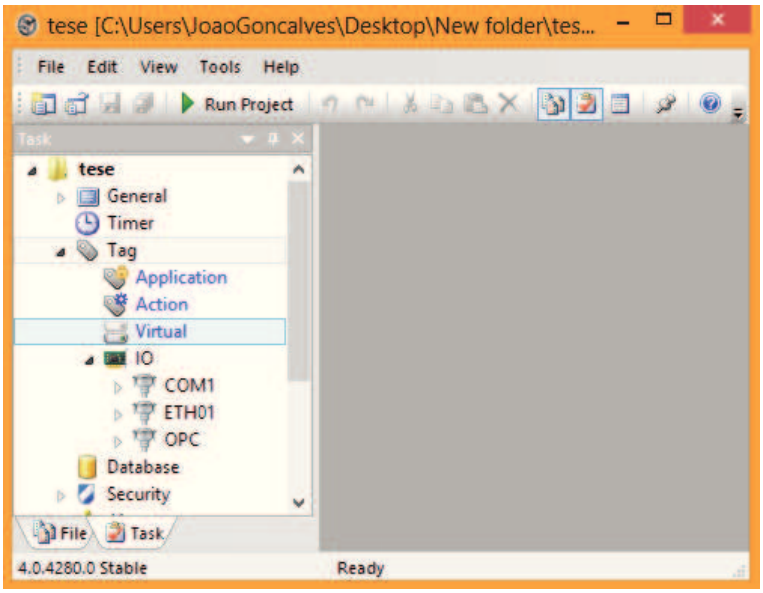


Figura A.4: Criação, edição e/ou eliminação de etiquetas virtuais.

Cada um dos componentes gráficos contém uma etiqueta (variável) associada que faz com que o mesmo possa ser acedido posteriormente na aplicação do cliente. A aplicação do cliente gere, de acordo com o estado do processo, se esse componente deve ou não estar visível. Exemplos de código associado a um componente gráfico no Inkscape e da caixa visualizadora das suas propriedades estão visíveis na Figura A.5.

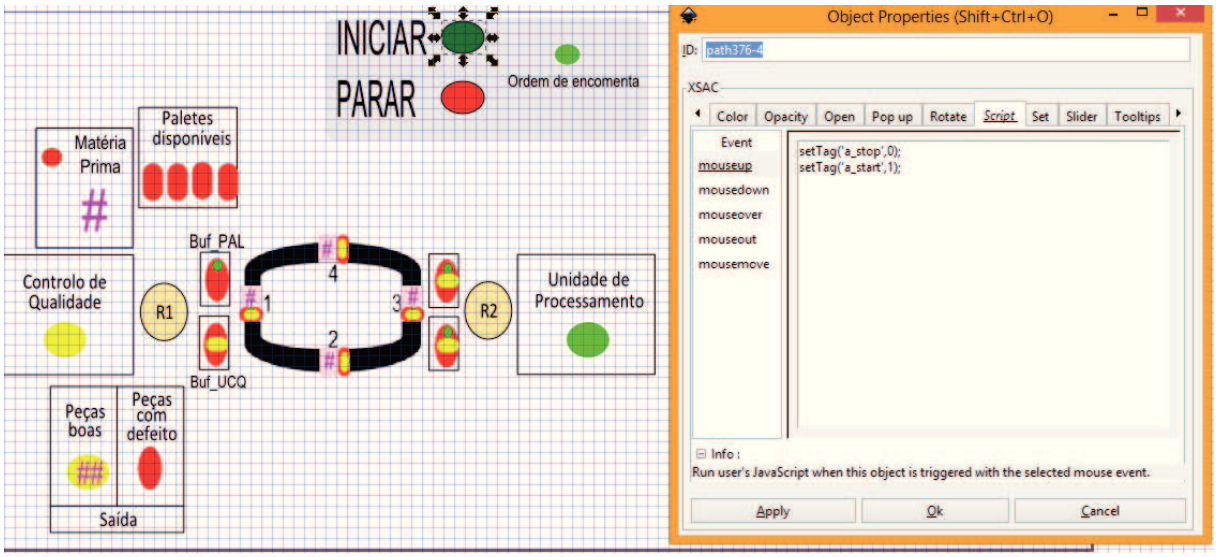


Figura A.5: Interface do Integraxor para associar um componente à *tag* anteriormente criada A.3.

Botão "Iniciar":

- Script *mouseup* (ao premir o botão):

  - setTag('a_stop',0);
  - setTag('a_start',1);

Como o objetivo é permitir ou não a visibilidade de um componente gráfico, a *tag* em questão deve ser definida como uma variável do tipo booleana A.3, como demonstra a Figura A.6.

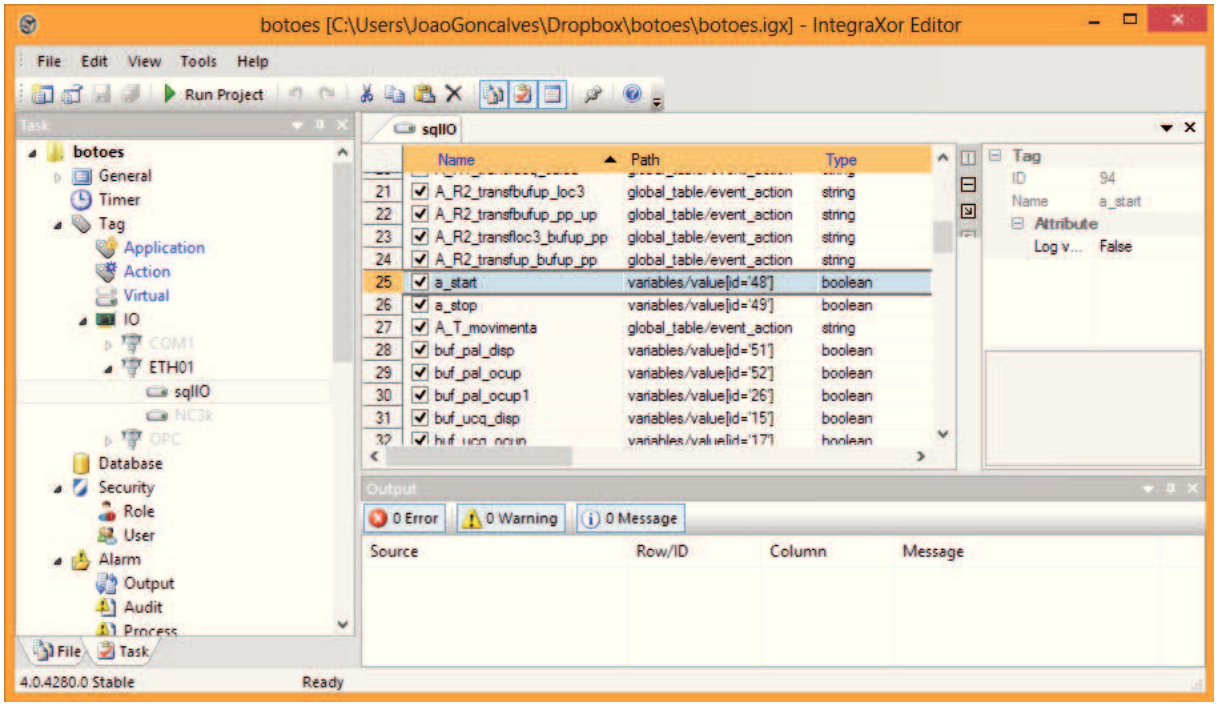


Figura A.6: Definição das *tags* no IntegraXor.

Para informações detalhadas acerca das funcionalidades dos componentes e efeitos de animação, por exemplo, deve ser consultado o manual de utilização do IntegraXor [29].

No anexo A.1 encontra-se uma descrição mais detalhada do processo de criação de um novo projeto. Estão representadas as várias etapas a executar, quando se pretende criar, configurar e visualizar uma nova etiqueta no IntegraXor.

Para concluir com um contexto geral, a Figura A.7 demonstra a ideologia da criação de uma etiqueta no IntegraXor.

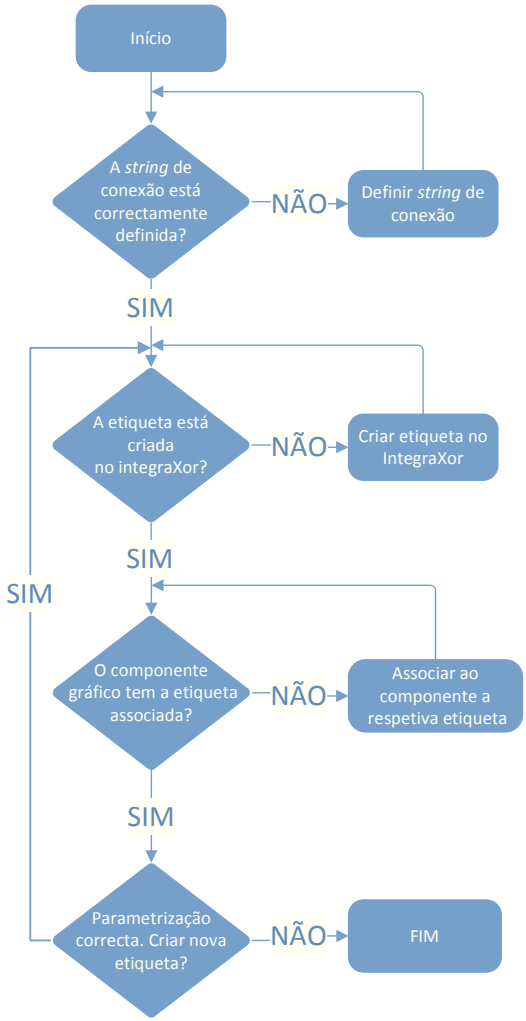


Figura A.7: Fluxograma para criar e configurar uma nova etiqueta no IntegraXor.

A.4 Adicionar *scripts*

O IntegraXor permite código em linguagem javascript. Assim, sendo podem ser adicionados *scripts* ao projeto.

Para se adicionarem *scripts* ao projeto executam-se as seguintes tarefas:

- Na barra de tarefas seleciona-se "*Script*".
- Faz-se a definição do nome do *Script* na coluna "*Name*".
- Por último coloca-se a localização do ficheiro em formato javascript (.JS) na coluna que diz *"File Name"*.

A leitura dos *scripts* pelo servidor é executada basicamente através de duas formas, ou quando ocorre a alteração de uma etiqueta ou em determinados intervalos de tempo

previamente definidos. Esta escolha é feita através da sequência representada na Figura A.8:

- Na barra de tarefas, deverá ser selecionada a subsecção *"Script"*.
- E na coluna *"Trigged by"*, deverá ser selecionada a opção *"Timer"* ou *"Tag"*, dependendo da finalidade pretendida.

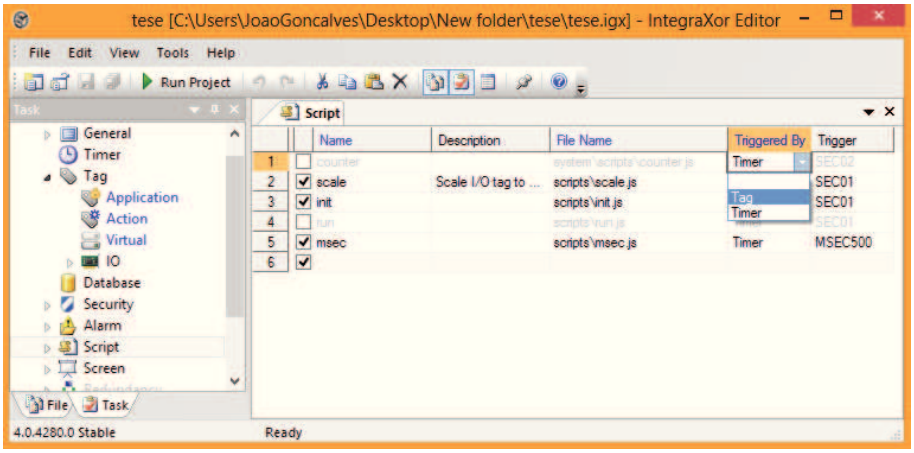


Figura A.8: Definições de execução dos *scripts*.

A.5 Adicionar/ocultar página Web ao projeto SCADA

É possível adicionar diversas páginas Web ao projeto SCADA. Normalmente estas são ficheiros do tipo .HTML ou .SVG, dependendo se para a sua programação foi utilizada a linguagem *Hyper Text Markup Language* ou *Scalable Vector Graphics*, respetivamente. A página HTML pode ser elaborada recorrendo a vários editores, como, por exemplo, o Notepad. Ao longo da elaboração desta dissertação foi utilizado o software Netbeans 7.0 (livre) como Ambiente de desenvolvimento integrado (*Integrated Developement Environment*) (IDE), para desenvolvimento das aplicações cliente, o Visio 2013 (licença IPL) para elaborar esquemas e diagramas e ainda um software de edição gráfica para produzir a *interface* SCADA, o Inkscape (livre).

Ao Inkscape foi ainda adicionado um complemento chamado SAGE [4] que permite integrar animações em tempo real, caso se guarde o ficheiro em formato .SVG.

Para se adicionar uma página ao projeto devem ser executadas as seguintes tarefas, como mostra a Figura A.9:

[4] `http://sourceforge.net/projects/sage-scada/files/SAGE%20v4.07/inkscape0484_sage407.exe/download`

- Clicar no menu lateral esquerdo *"Screen"*.
- Premir com a tecla direita do rato em cima de um elemento da coluna *"Menu"* e selecionar a opção *"New Menu"* ou *"New sub-Menu"*, a primeira criará um menu principal, a segunda um submenu correspondente ao menu selecionado.

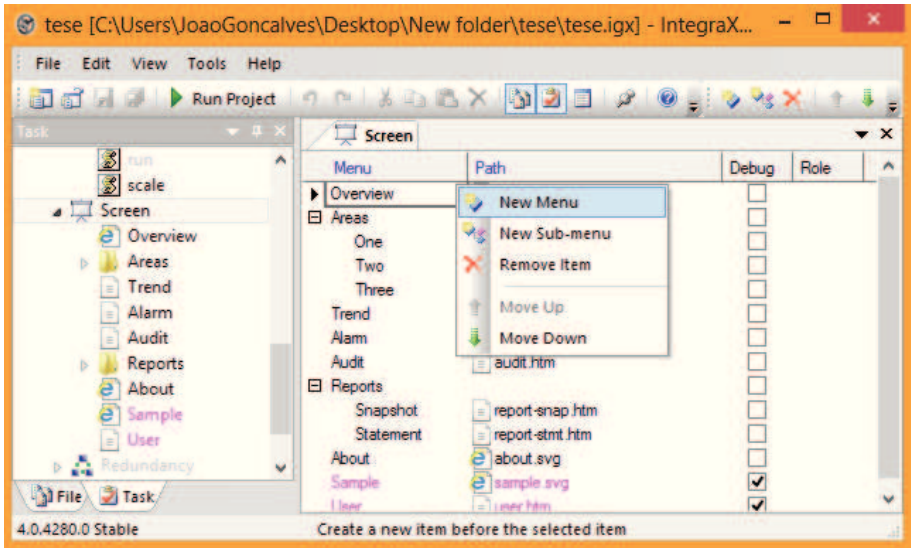


Figura A.9: Criação de uma nova página no projeto.

- Para alterar o nome da página acede-se à coluna *"Menu"*.
- O caminho de origem do ficheiro deverá ser colocado na coluna *"Path"* como mostra a Figura A.10:
- Para a página estar visível a opção *"Debug"* deverá estar desabilitada, caso contrário esta fará parte do projeto mas não será visível no *browser* (cliente).

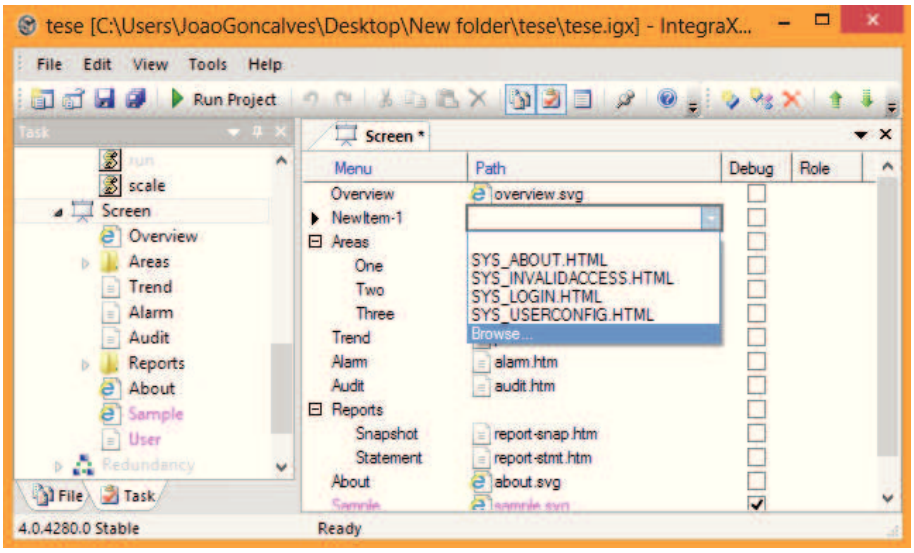


Figura A.10: Adicionar o caminho de origem da página web.

A.6 Analisar estado das variáveis do sistema (etiquetas)

O IntegraXor tem uma ferramenta que demonstrou ser muito útil para fazer depuração durante o desenvolvimento do SCADA. Essa ferramenta permite ver o valor atual de todas as etiquetas do sistema e tem o nome de *watch*. A Figura A.11 demonstra o aspeto visual da ferramenta, acessível a partir da janela do servidor (clicando em *"View"*, seguindo-se de *"Watch Windows"*).

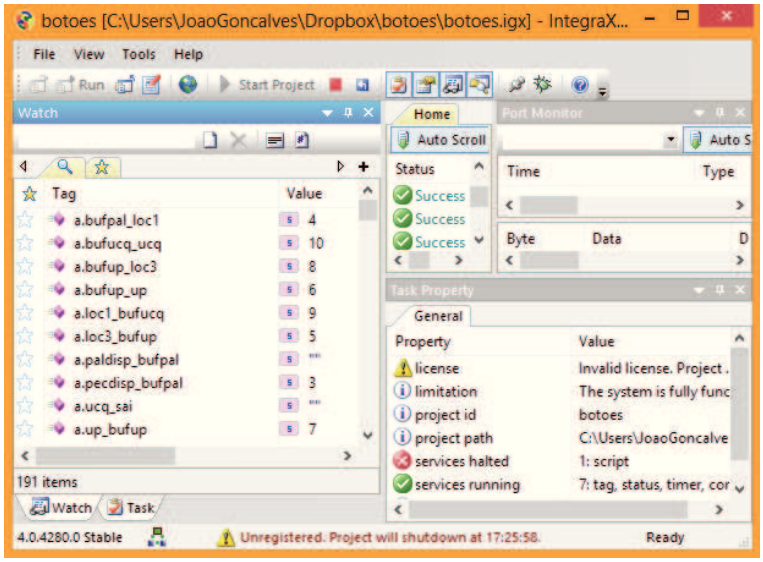


Figura A.11: Ferramenta disponibilizada para ver o valor das etiquetas em tempo real.

Por vezes, pode ser necessário resolver alguns problemas do sistema e/ou serem efetu-

adas algumas ações para dar início ao processo. Essas tarefas estão descritas em forma de diagrama, através da Figura A.12.

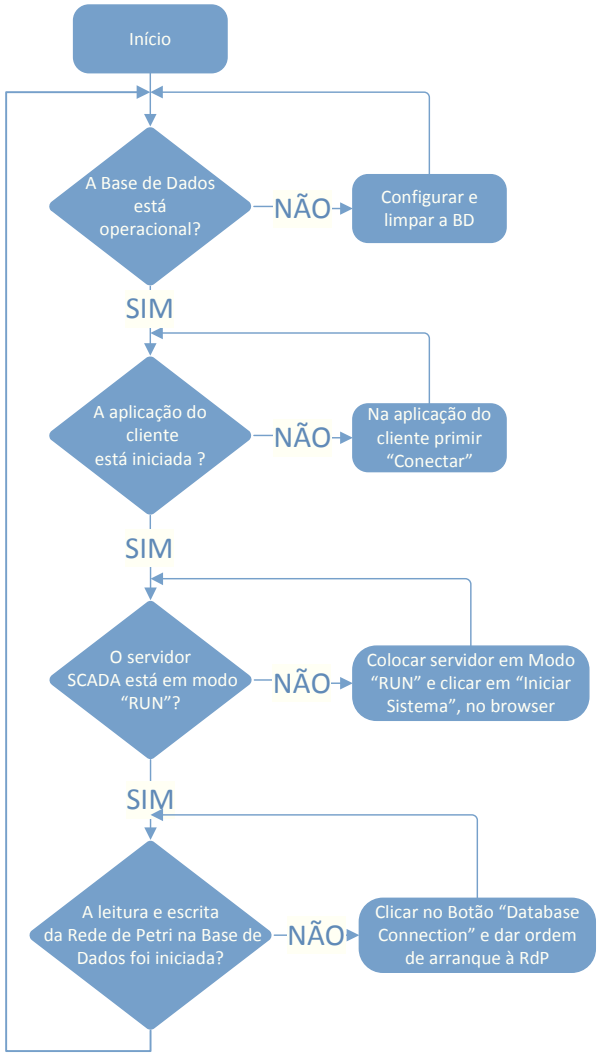


Figura A.12: Diagrama de verificação e correção de problemas de arranque.

Anexo B

PostgreSQL

B.1 Adicionar conexão de cliente ao servidor

Na elaboração desta dissertação o software SCADA e o servidor da BD encontram-se ambos instalados na mesma máquina do FMS, no WS3 ou Manager. Assim sendo, todos os clientes que se pretendam ligar ao servidor do PostgreSQL devem ter o seu endereço no respetivo ficheiro de configuração. Para executar essa operação devem ser efetuadas as seguintes etapas:

- Ir à pasta de instalação do PostgreSQL e abrir: *PostgreSQL* > *9.2* > *data* > *pg_hba*.
- No ficheiro de configuração *pg_hba* adicionar todos os endereços pretendidos, como mostra a Figura B.1.
- Gravar o ficheiro.

Após a alteração do ficheiro *pg_hba*, existe um outro ficheiro que é necessário alterar. Esta alteração tem como objetivo colocar o servidor PostegreSQL a escutar todos os endereços que a ele tentem aceder. A Figura B.2 representa o estado final do ficheiro de configuração, obtido através dos seguintes passos:

- Colocar o parâmetro $< listen\_addresses = ' * ' >$.
- Colocar o parâmetro $< tcpip\_socket = true >$.

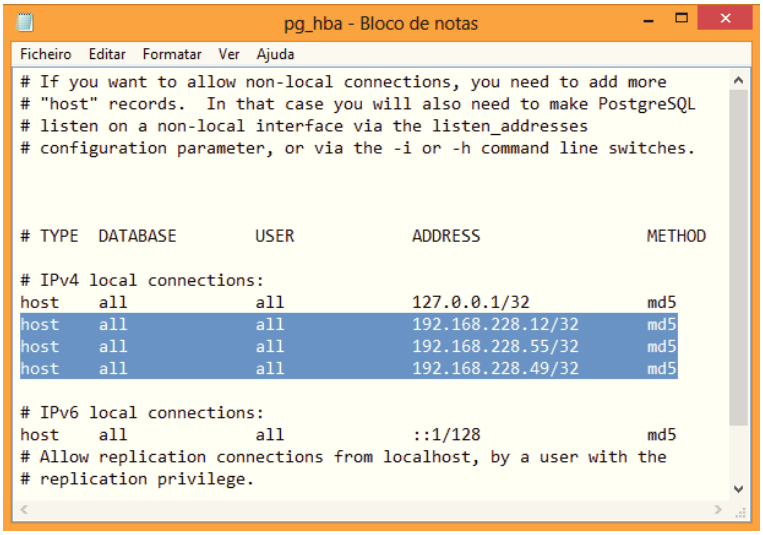


Figura B.1: Ficheiro de configuração *pg_hba*.

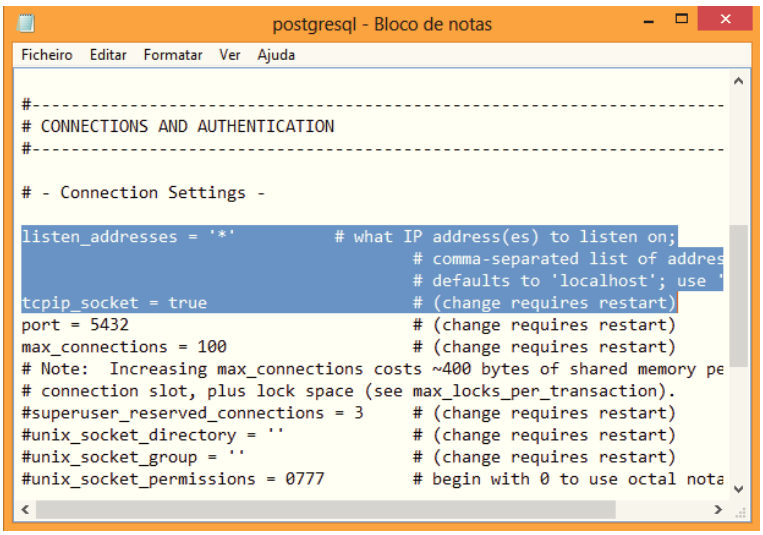


Figura B.2: Ficheiro de configuração *PostgreSQL*.

Anexo C

PIPE

Existiu a necessidade de fazer algumas alterações à aplicação PIPE, de modo a que esta pudesse comunicar com a aplicação cliente desenvolvida ao longo desta dissertação. Para que tal fosse possível recorreu-se ao Eclipse para depurar e alterar o código fonte. Os passos de configuração podem ser consultados na Secção C.1. Na Figura C.1 é possível ver o aspeto gráfico do programa.

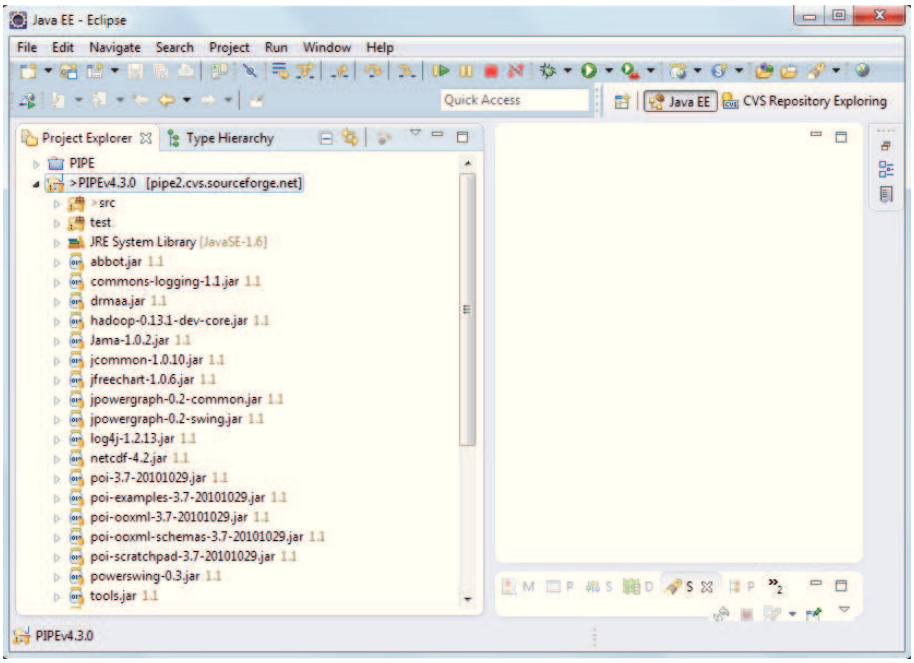


Figura C.1: Ambiente de trabalho do programa Eclipse.

C.1 Aceder ao código fonte do PIPE

O PIPE está acessível através de um repositório CVS (Concurrent Versions System). A parametrização do repositório, passo a passo, pode ser consultada de seguida. É descrita

a sequência de tarefas que devem ser executadas para a importação do código fonte do PIPE, no Eclipse, de modo a permitir a posterior edição, execução e compilação.

1. Descarregar o Eclipse [1].

2. Seleccionar 'Window' - Open Perspective - CVS Repository Exploring'.

3. No painel à esquerda, 'CVS Repository', clicar com a tecla direita do rato e premir 'New - Repository Location'.

4. Na caixa de diálogo que aparece devem ser colocados os seguintes parâmetros:

   - Host: pipe2.cvs.sourceforge.net
   - Repository path: /cvsroot/pipe2

   Para utilizadores registados devem inserir-se as suas próprias credenciais, caso contrário devem ser colocadas as seguintes:

   - Username: Anonymous
   - Password: -deixar em branco-

   No tipo de conexão selecionar: 'pserver', para tipos Anonymous CVS ou 'extssh' para outros tipos. As portas devem ser deixadas com o valor por omissão e a confirmação 'leave validate connection' deve ser permitida, clicando de seguida em 'Finish'.

5. Se tudo correu como previsto, a conexão foi validada e apareceu uma nova entrada no painel CVS Repositories. Deve expandir-se o painel recente usando o item à sua esquerda e de seguida deve ser expandido também o painel denominado 'HEAD' . Ao efetuar estes passos deve ser agora possível ver um painel chamado 'PIPE'.

6. Seleccionar o painel 'PIPE' com a tecla direita e escolher a opção 'Check out as...'. Na caixa de diálogo que aparecer, premir 'Check out as a project configured using the New Project Wizard'. Nota: Esta opção só estará disponível caso não esteja criado nenhum projeto na diretoria escolhida para o projeto.

7. Seleccionar 'Java Project' a partir da lista do assistente de criação de projeto.

8. Denominar o projeto 'pipe' e clicar em 'Finish'. Depois deste passo deve alterar-se a visualização do projeto para a perspetiva Java [2] e clicar-se em 'Yes'.

[1] `http://www.eclipse.org/downloads/`

[2] Um dos modos de visualização otimizada das ferramentas, presentes na *interface* disponível no IDE utilizado

9. Durante alguns segundos o processo de ‘checked out from CVS’ será executado e após esse processo terminar serão exibidos alguns erros no painel.

10. Clique no projeto ‘pipe’ no explorador de projeto (painel esquerdo) e selecione ‘Project Properties’ a partir do menu principal do Eclipse.

11. Clicar na lista da esquerda e selecionar ‘Java Build Path’ depois selecionar o separador ‘Source’. Clicar em ‘Add folder...’ e permitir as diretorias ‘Resources’, ‘src’ e ‘test’.

12. Ainda no separador ‘Source’:

    - Mudar a pasta de saída que se encontra por omissão para ’pipe/build/app’;
    - Permitir o campo ’Allow output folders for source folders’;
    - Expandir a entrada ’test/pipe’ na lista de pastas fonte e clicar no botão ‘Edit’. Selecionar a opção ’Specific output folder’ e colocar “build/test” na caixa. Clicar em ‘Ok’.

13. No separador ‘libraries’ clicar em ‘JARs’. Selecionar ’junit.jar’ e em ’abbot.jar’ a partir da diretoria ’pipe/test/lib’ e clicar em ‘Ok’.

14. Ir a ‘Window > Preferences’ depois ‘Ant > Run-time’ e clicar em ‘Global Entries’ que se encontra no separador ‘Classpath’. Clicar em JARs e adicionar ‘junit.jar’, ‘xalan.jar’ e ‘serializer.jar’ a partir da diretoria ’pipe/test/lib’ e clicar ’Ok’ e de seguida ’Ok’ novamente.

15. Vai aparecer no ecrã uma mensagem de aviso sobre a remoção das fontes geradas. Clicar ‘Yes’ para seguir.

16. Agora, novamente através de uma perspetiva Java, e uma vez que o projeto esteja recompilado, os erros que apareceram anteriormente devem estar extintos. Pode ser feito neste momento o ‘Run’ do projeto.

17. Seleccionar ‘Run > Run’ presente no menu principal do Eclipse. Na janela que aparecer selecionar ‘Java Application’, depois clicar em ‘New’.

18. Alterar o nome de ‘New configuration’ para ‘Pipe’. Clicar em no botão ‘Search’ presente na secção ‘Main Class’. Selecionar ‘RunGui’ no diálogo que aparecer e depois clicar ‘Apply’ seguido de ‘Run’.

Nesta fase deverá aparecer a interface da aplicação Pipe, para isso basta clicar no botão de ‘Run pipe’ que se encontra na barra de ferramentas.

C.2 Guia de criação de um novo projeto

Como referido anteriormente, o software usado para a elaboração da Rede de Petri representativa do sistema flexível de produção foi o PIPE v4.3.0. Através da Figura C.2 é possível ver o aspeto visual da aplicação PIPE.

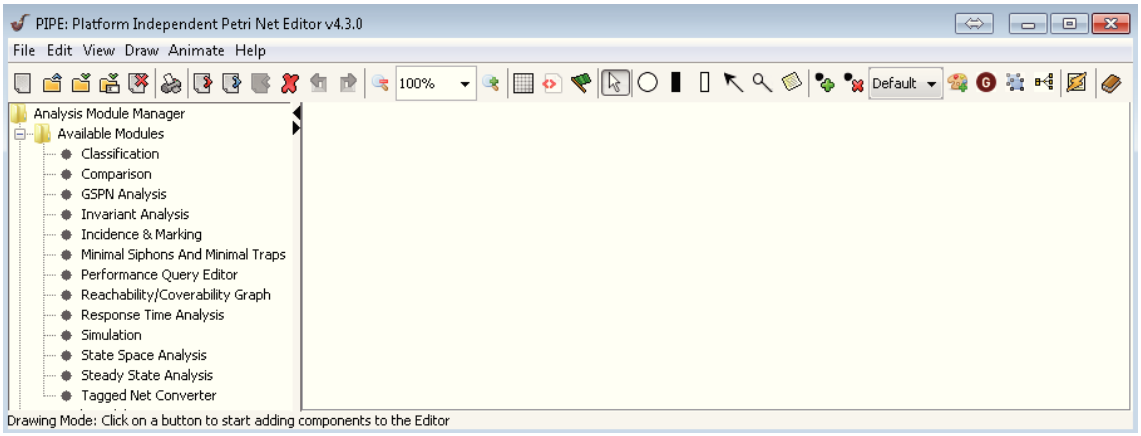


Figura C.2: Ambiente gráfico do Software PIPE v4.3.0.

Por omissão, ao iniciar a aplicação esta inicia um novo projeto, mas pode sempre criar-se um novo através dos seguintes passos:

- File;
- New (CRTL + N).

Para abrir um ficheiro já existente os passos serão:

- File;
- Open;
- Seleciona-se o ficheiro e prime-se Abrir.

Depois de aberto o ficheiro, e caso se deseje simular o sistema em modo *offline*, os passos a seguir são:

- Na barra de ferramentas selecionar o ícone "Toggle Animation Mode", se desejar correr o sistema em modo de simulação, representado pela Figura C.3.

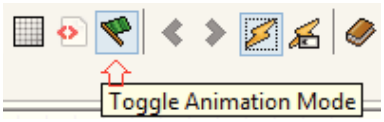


Figura C.3: Depois de pressionado "Toggle Animation Mode ", pressionar "Randomly fire a transistion"para disparo de uma transição aleatória.

- Para disparar uma transição aleatória deve selecionar-se o ícone *"Randomly fire a transition"* ou, caso se pretenda disparar um determinado número de transições aleatórias, deve selecionar-se o ícone "Randomly fire a number of transitions".
- Caso se queira disparar um número de transições aleatórias devem ainda ser introduzidos alguns dados, tais como o número de transições a disparar e o tempo entre transições, como se verifica através da Figura C.4. Este tempo (expresso em milissegundos) pode permitir visualizar todo o processo, caso não seja muito curto.

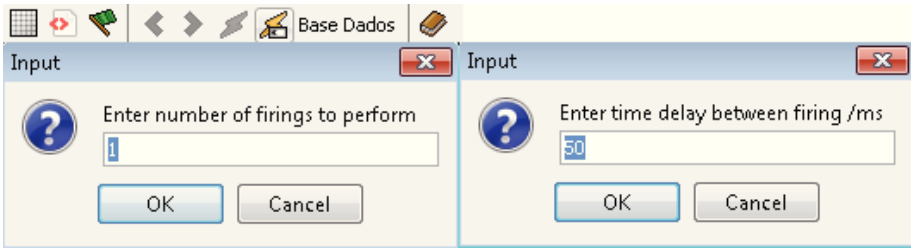


Figura C.4: "Randomly fire a number of transistions". Disparo de várias transições aleatórias em que o número é em unidades e o tempo em milissegundos.

- É possível disparar uma transição específica, clicando duas vezes sobre a mesma.

Depois de aberto o ficheiro, e caso se deseje simular o sistema em modo *online*, os passos a seguir são:

- Carregar no botão "*Database*"e de seguida selecionar "*Yes*", como é possível verificar através da Figura C.5.

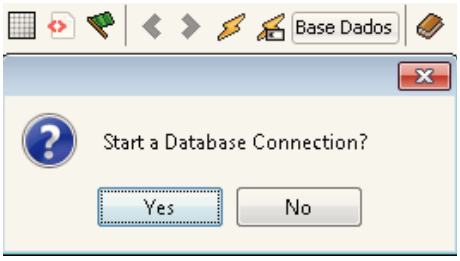


Figura C.5: Ligação do PIPE à BD, através da funcionalidade implementada.

Anexo D

Software utilizado

Ao longo da elaboração desta dissertação foram utilizados vários programas que podem ser adquiridos através das seguintes ligações:

1. O NetBeans IDE 7.0.1 utiliza-se como IDE para desenvolvimento de uma aplicação. Esta aplicação foi desenvolvida em tecnologia Java e o seu principal objectivo é fazer a comunicação entre os vários componentes do sistema e uma base de dados centralizada e comum a todo o sistema [1].

2. O Eclipse é um software que contém um ambiente base de desenvolvimento, denominado *workspace*, e para além disso também disponibiliza um sistema de extensões, plugins, que permitem personalizar esta plataforma à medida das necessidades do programador [2].

3. O Inkscape0.48.1 com a extensão SAGE v3.03 permite criar e editar a interface gráfica dos conteúdos SCADA do sistema. Estes conteúdos são geridos num outro software SCADA com o nome de Integraxor [3].

4. Java Development Kit 7 é a plataforma Oracle que, no caso desta dissertação em específico, deve estar instalada em todas as máquinas do sistema, sejam elas clientes ou servidores, para que seja possível posteriormente existir comunicação entre as mesmas numa linguagem que todas consigam perceber. A Oracle aconselha vivamente a que esta plataforma esteja sempre na versão mais actualizada de modo a evitar anomalias [4].

5. Para facilitar o acesso à base de dados PostgreSQL e possibilitar o ambiente gráfico

[1] `https://netbeans.org/downloads/7.0.1/index.html`
[2] `http://www.eclipse.org/downloads`
[3] `http://www.afterdawn.com/software/general/download_splash.cfm/inkscape_portable`
[4] `http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html`

e visível da mesma, é instalado este software de gestão de bases de dados, PGAdmin [5].

6. A base de dados utilizada ao longo do desenvolvimento desta dissertação foi a PostgreSQL 9.2 [6].

7. Winrar é um software que permite a compressão e descompressão de ficheiros e serve para suprimir algumas necessidades que vão surgindo, nesse âmbito [7].

8. Microsoft Visio Professional 2013 é um software Microsoft que se foca na edição de diagramas profissionais, com o objectivo de transformar informação complexa em algo que seja de fácil leitura e compreensão [8].

9. Durante a execução da aplicação surge a necessidade de recorrer aos serviços da porta série. Para que tal seja possível tem de ser instala uma biblioteca nativa Java que fornece um pacote de compilação nativo para uma máquina Windows de 64 bits [9], denominada RXTX para Java.

10. Relacionado com as RdP existe o PIPE, software que serve para criar e analisar Redes de Petri. Permite, por exemplo, a simulação em tempo real da RdP e a posterior análise através de relatórios gerados pelo próprio PIPE [10].

[5] http://www.postgresql.org/ftp/pgadmin3/release/v1.16.1/win32/
[6] http://www.postgresql.org/ftp/source/v9.2.0/
[7] http://www.win-rar.com/download.html&?L=0
[8] http://office.microsoft.com/en-001/visio/
[9] http://mfizz.com/oss/rxtx-for-java
[10] http://sourceforge.net/projects/pipe2/files/latest/download

Anexo E

Aplicação cliente

E.1 Aplicação cliente

Nesta dissertação existiu a necessidade de desenvolver uma aplicação cliente para cada um dos equipamentos a controlar, pois têm parametrizações diversificadas. Uma vez que todas as aplicações cliente são semelhantes, às Tabelas E.1,E.2 e E.3 permite visualizar a estrutura sobre as quais foram desenvolvidas.

Tabela E.1: Estrutura da aplicação relativamente à comunicação com os equipamentos.

	Procurar portas série	Conexão porta série	Escrever na consola	Escrever nos controladores
Comunicação com equipamento	Verifica as portas série do computador e cria uma lista de portas válidas para posterior conexão	Verifica a porta série selecionada na lista de disponíveis e inicia a comunicação série	Verifica as portas série do computador e cria uma lista de portas válidas para posterior conexão	Verifica as portas série do computador e cria uma lista de portas válidas para posterior conexão

Tabela E.2: Estrutura da aplicação relativamente à interface gráfica.

	Iniciar aplicação	Terminar aplicação	Enviar instruções a partir do terminal	Consola de instruções
GUI - Interface	Faz a atualização da interface gráfica de acordo com a ordem de início da conexão	Faz a atualização da interface gráfica de acordo com a ordem de fim da conexão	Verifica o campo disponível no terminal da aplicação e caso esteja preenchido envia os carateres pelo porto RS232	Faz o desenho gráfico que permite visualizar a interface gráfica da aplicação

Tabela E.3: Estrutura da aplicação relativamente à comunicação com a BD.

	Observador	Notificador
Comunicação com a BD	Verifica a base de dados em intervalos de tempo pré-definidos e permite efetuar leituras e escritas na mesma	Devolve a notificação de quando ouve alterações na base de dados, para que se possa atuar em conformidade

São agora indicados excertos de código, escritos em Java, que servem de base para o desenvolvimento de algumas funções abordadas.

Procurar portos de comunicação na estação de trabalho:

```
public void searchForPorts() {
       ports = CommPortIdentifier.getPortIdentifiers();

       while (ports.hasMoreElements()) {
           CommPortIdentifier curPort = (CommPortIdentifier)
               ports.nextElement();
           if (curPort.getPortType() == CommPortIdentifier.PORT_SERIAL) {
               window.cboxPorts.addItem(curPort.getName());
               portMap.put(curPort.getName(), curPort);
           }
       }
   }
```

Escrita de carateres no porto de comunicação RS232:

```
    public void writeData(String b) {
        String c = "";
        try {
            for (int i = 0; i < b.length(); i++) {
                output.write(b.charAt(i));
            }
            output.write(13);
        } catch (Exception e) {
            //    logText = "Falhou ao escrever data. (" + e.toString() + ")";
            window.txtLogOut.setForeground(Color.red);
            window.txtLogOut.append(logText + "\n");
        }
    }
```

Leitura de carateres através do porto de comunicação RS232:

```
  public void serialEvent(SerialPortEvent evt) {
        if (evt.getEventType() == SerialPortEvent.DATA_AVAILABLE) {
           try {
```

```
                byte singleData = (byte) input.read();
                if (singleData != NEW_LINE_ASCII) {
                    logText = new String(new byte[]{singleData});
                    total = total.concat(logText);
                    window.txtLogOut.append(logText);
                } else {
                    window.txtLogOut.append("\n");
                }
            } catch (Exception e) {
                logText = "Falhou ao ler data. (" + e.toString() + ")";
                window.txtLogIn.setForeground(Color.red);
                window.txtLogIn.append(logText + "\n");
            }
        }
    }
```

Conexão com a base de dados PostgreSQL:

```
public class Conexao {

    public static String nome = "postgres";
    public static String senha = "tese";
    public static String url =
        "jdbc:postgresql://192.168.228.120:5432/postgres";
    public static String driver = "org.postgresql.Driver";
    public static Connection con;

    public Conexao() {
    }

    public Connection conexaoPostgres() {
        try {
            Class.forName(driver);
            con = DriverManager.getConnection(url, nome, senha);
        } catch (Exception e) {
            System.out.println("Erro: CONEXAO" + e);
        }
        return con;
    }
```

```
    public void fechaConexao() {

        try {

            con.close();
        } catch (Exception ex) {

            System.out.println("Erro: " + ex);
        }
    }
}
```

Funções para efetuar o acionamento dos eixos, colocar o controlo do robô em modo automático e colocar o robô na posição inicial e de modo operacional:

```
public void conModeMeth(Communicator communicator) {
        communicator.writeData("CON");
        do {
            this.scorboting = communicator.total;
        } while (!scorboting.contains("CONTROL ENABLED"));
    }

    public void autoModeMeth(Communicator communicator) {
        this.scorboting = "";
        communicator.writeData("AUTO");
        do {
            this.scorboting = communicator.total;
        } while (!scorboting.contains("O.K"));
    }

    public void runhomeModeMeth(Communicator communicator) {
        this.scorboting = "";
        communicator.writeData(run_homes);
        do {
            this.scorboting = communicator.total;
        } while (!scorboting.contains("Homing complete"));
    }
```

Anexo F

Rede de Petri Global

O conjunto das RdP de cada uma das células de trabalho forma uma RdP global do sistema SCADANet, representada através da Figura F.1.

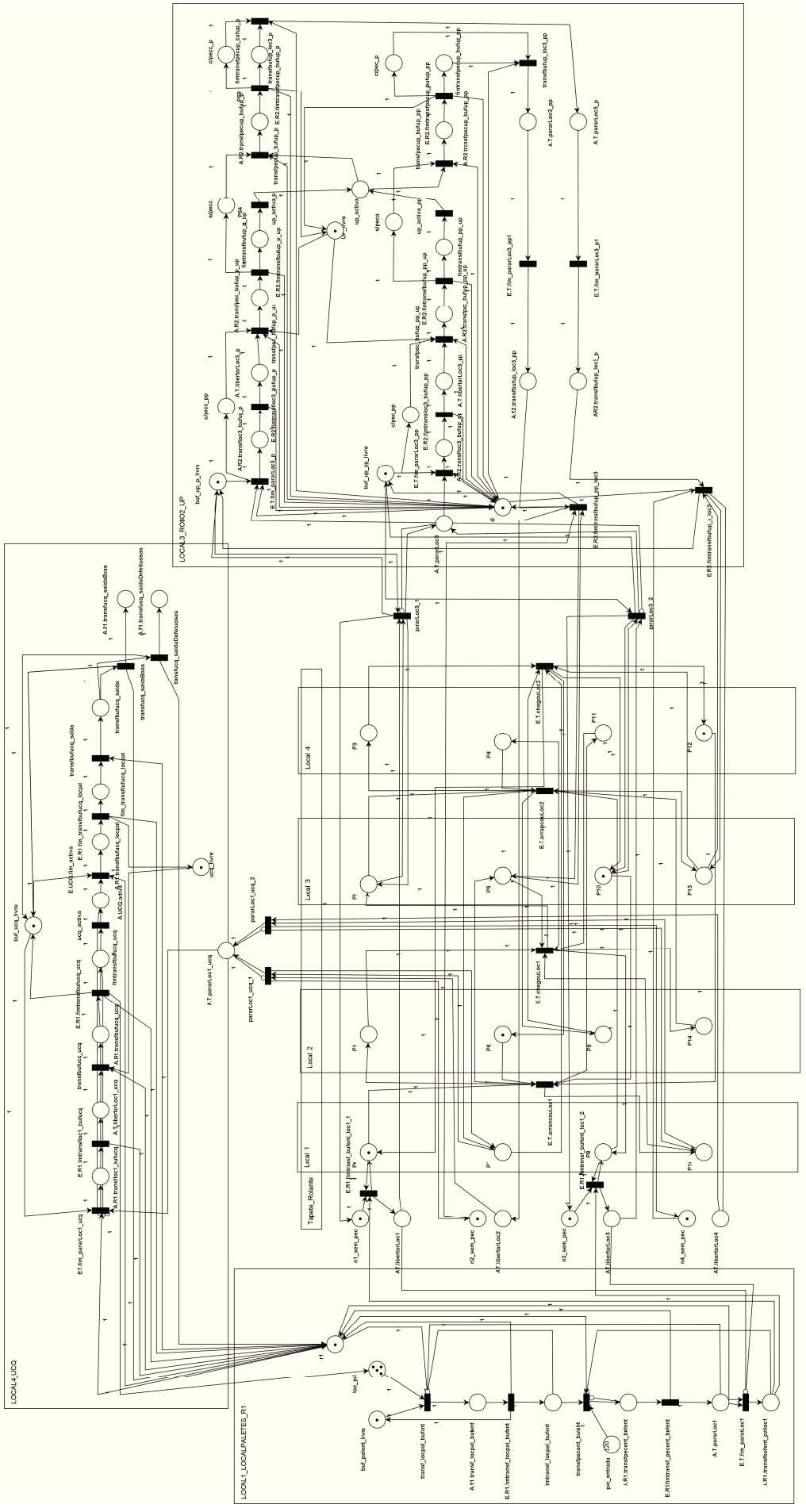


Figura F.1: Rede de Petri global do sistema SCADANet.