Configuração de Sistemas de Controle Industriais baseada em Dispositivos Móveis Sem Fio

Yuri de Carvalho Gomes

Dissertação de Mestrado submetida à Coordenadoria do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande - Campus de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Processamento da Energia

Antonio Marcus Nogueira Lima, Dr.
Orientador

Campina Grande, Paraíba, Brasil

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL DA UFCG

G633c Gomes, Yuri de Carvalho

Configuração de sistemas de controle industriais baseada em dispositivos móveis sem fio / Yuri de Carvalho Gomes. — Campina Grande, 2007.

75 f. : il. color

Referências

Dissertação (Mestrado em Engenharia Elétrica) – Universidade Federal de Campina Grande. Centro de Engenharia Elétrica e Informática.

Orientador: Prof. Dr. Antônio Marcus Nogueira Lima.

1. Automação industrial 2. Configuração remota 3. Sistemas de Controle 4. Dispositivos Móveis I. Título.

CDU 621.3:65.011.56(043)

CONFIGURAÇÃO DE SISTEMAS DE CONTROLE INDUSTRIAIS BASEADA EM DISPOSITIVOS MÓVEIS SEM FIO

YURI DE CARVALHO GOMES

Dissertação Aprovada em 20.12.2007

ANTONIO MARCUS NOGUEIRA LIMA, Dr., UFCG
Orientador

ANGELO PERKUSICH, D.Sc., UFCG
Orientador

CURSINO BRANDÃO JACOBINA, Dr.Ing., UFCG
Componente da Banca

CAMPINA GRANDE – PB
DEZEMBRO – 2007

Dedicatória

Este trabalho é dedicado aos meus pais, José e Gilca, por estarem sempre ao meu lado quando preciso e por me oferecerem as melhores oportunidades para minha formação pessoal e profissional, aos meus irmãos Igor e Yegor, que apesar da distância sempre estarão presentes no meu coração, à toda minha família (tios e primos) que souberam entender minha ausência em muitas ocasiões, em especial, aos meus primos Augusto, Neto, Annie, Thaíse, Clarissa, Christina, Fernanda e Helinho, meu querido afilhado, e ainda à minha avó Socorro e ao meu padrinho Gilberto Gomes Batista (in memoriam), idealizador da minha formação como engenheiro.

Agradecimentos

Agradeço a Deus, pelo auxílio e conforto nos momentos de dúvida e angústia, e por sempre iluminar meu caminho.

Aos meus pais e meus irmãos, pelo apoio e confiança.

Ao professor Antonio Marcus Nogueira Lima pela orientação e colaboração nas discussões das idéias desta dissertação, sem as quais não seria possível realizar este trabalho. Aos professores Cursino Brandrão Jacobina, Maurício Beltrão de Rossiter Corrêa e Angelo Perkusich pelo apoio, atenção e colaboração nas atividades desenvolvidas no LEIAM e no EMBEDDED.

Aos companheiros do EMBEDDED, LIEC e LEIAM pela amizade e descontração durante o período de desenvolvimento desta dissertação, em especial aos amigos Diego, Luiz Paulo, Cecília, Taciana, Fernanda, George, Marcus, Genildo, Thiago Onofre, Ádrian, José Luis, Olympio, Danilo, André, Mário, Paulo, Tomás, Jonas, Rafael, Isaac e Euzeli.

Aos amigos do dia-a-dia e de muitos anos de vida profissional e estudantil, em especial a Pedro Amorim, Thiago Amorim, Felipe Amorim, Cauê Colaço, Liliane Sena, Breno de Souza, Rodrigo Nóbrega, Roque da Costa e Moacir Delgado.

À minha família Irmãos Pela Fé.

Um agradecimento especial à minha namorada Giulianna.

Aos professores e funcionários do Departamento de Engenharia Elétrica que de uma certa forma contribuíram para o andamento do meu trabalho.

Resumo

A comunicação sem fio é considerada hoje em dia como mais um recurso disponível para auxiliar na automação e controle de processos. O desenvolvimento e implementação de infra-estruturas de rede sem fio em ambientes industriais vêm ocorrendo de forma gradual, no entanto toda a potencialidade da tecnologia sem fio ainda precisa ser explorada. Além disso, há uma contínua e crescente necessidade do setor industrial por ferramentas para configuração, análise e diagnóstico de processos industriais que proporcionem redução do tempo relacionado a atividades de configuração e programação de sistemas de controle, como também flexibilidade, portabilidade, confiabilidade e praticidade. Neste contexto, uma ferramenta para configuração remota de sistemas de controle industriais utilizando tecnologia sem fio pode ser desenvolvida, de modo a satisfazer estas características exigidas. Portanto, neste trabalho é apresentado um configurador remoto de algoritmos de controle para aplicações industriais, compatível com um sistema com conectividade sem fio, capaz de realizar geração automática de código e que pode ser acoplado ao sistema de controle. A arquitetura abordada para este sistema de configuração remota define uma estrutura cliente-servidor baseada em duas entidades, Estação Base e Estação Remota, comunicando-se entre si por um padrão de tecnologia sem fio. A parte cliente (Estação Remota) é usada para interface gráfica com o usuário e operação remota, consistindo em um *software* aplicativo executado em um dispositivo portátil (configurador remoto) responsável pela configuração do algoritmo de controle usando representação em diagrama de blocos. A parte servidor (Estação Base) envolve um sistema capaz de gerar código executável para o *hardware* de controle da aplicação a partir de modelos em diagrama de blocos. Em geral, aplicações industriais nas quais sensores e atuadores são empregados para controlar parâmetros e/ou o estado do sistema de controle são favoráveis para a implementação da arquitetura do sistema de configuração remota, proporcionando conectividade e integração de forma prática e simples com estações remotas. Assim, o sistema de configuração remota proposto pode ser considerado uma aplicação em potencial para o setor industrial.

Abstract

Nowadays, wireless communication is considered as one more available resource to aid in the automation and process control. The development and implementation of wireless network infrastructures in industrial environments are increasing gradually, however all the potentiality of the wireless technology still needs to be explored. Furthermore, there is a continuous and increasing demand in the industry for configuration and management tools to control industrial processes that provide significative reduction of time in configuration and programming activities of control systems, as well as flexibility, portability, reliability and easiness. In this context, a remote configuration tool for industrial control systems using wireless technology can be developed to satisfy these required characteristics. Therefore, in this work, a remote configurator of control algorithms to industrial control applications, compatible with a system with wireless connectivity, capable of performing automatic code generation and which can be incorporated to the control application, is presented. The used architecture for the remote configuration system defines a client-server structure based on two entities, Base Station and Remote Station, communicating each other by a standard wireless technology. The client part (Remote Station) is used for graphical user interface and remote operation, consisting of a software running in a portable device (remote configurator) responsible for configuration of control algorithm using block diagram representation. The server part (Base Station) consists of a system capable of performing automatic code generation from block diagram models. In general, industrial applications in which sensors and actuators are used to control parameters and/or the state of the control system are favorable to implementation of the architecture of the remote configuration system, providing connectivity and integration in a simple and practical way with remote stations. Therefore, the remote configuration system proposed can be considered a potential application in the industry.

Índice

Lista de Símbolos e Abreviaturas

a, b, c	Terminais de saída do inversor
A/D	Conversor de sinal analógico para digital
CA	Corrente Alternada
CB-PWM	*Carrier Based Pulse Width Modulation*
CC	Corrente Contínua
DEE	Departamento de Engenharia Elétrica
DSP	*Digital Signal Processor*
DVD	*Digital Versatile Disc*
D/A	Conversor de sinal digital para analógico
E	Tensão do barramento CC do inversor de tensão
EMBEDDED	Laboratório de Sistemas Embarcados e Computação Pervasiva
ξ	Coeficiente de amortecimento de um sistema de segunda ordem
FPGA	*Field Programmable Gate Array*
GPL	*General Public License*
IEEE	*Institute of Electrical and Electronics Engineers*
IGBT	*Insulated Gate Bipolar Transistor*
ISA	*Industry Standard Architecture*
LED	*Light-Emitting Diode*
LEIAM	Laboratório de Eletrônica Industrial e Acionamento de Máquinas
MDL	Arquivo modelo do Simulink (*.mdl)
PC	*Personal Computer*
PDA	*Personal Digital Assistant*
PWM	*Pulse Width Modulation*
RCP	*Rapid Control Prototyping*
RISC	*Reduced Instruction Set Computer*
ROM	Read Only Memory
RTLIB	*Real-Time Library*

RTI	*Real-Time Interface*
RTW	*Real-Time Workshop*
SDRAM	*Synchronous Dynamic Random Access Memory*
SRAM	*Static Random Access Memory*
TLC	*Target Language Compiler*
T_S	período da modulação
T_1, T_2, T_3	Tempos de condução dos interruptores do inversor, obtidos com v_a, v_b, v_c
UFCG	Universidade Federal de Campina Grande
UML	*Unified Model Language*
UPS	*Uninterruptible Power Supply*
USB	*Universal Serial Bus*
VSI	*Voltage Source Inverter*
V/f	Volts/Hertz
v_a, v_b, v_c	Tensões senoidais de referência para modulação
Wi-Fi	*Wireless Fidelity*, tecnologia de rede sem fio baseada no padrão IEEE 802.11
WRC	*Wireless Remote Configurator*
ω_n	Frequência natural de um sistema de segunda ordem em rad/s
XML	*Extensible Markup Language*

Lista de Figuras

Capítulo 1

Introdução

Este trabalho está inserido no âmbito de Automação industrial, mais especificamente, relacionado a ferramentas para configuração remota de sistemas de controle industriais, como por exemplo, sistemas de acionamento de máquinas, de controle de processos térmicos, de controle de nível de reservatórios, robótica, entre outros. É perceptível no setor industrial a constante demanda por soluções eficientes para controle e automação de processos, e por isso surge o interesse por dispositivos de *hardware* cada vez mais sofisticados e por ferramentas de *software* para controle, monitoramento e configuração de sistemas de controle.

Neste contexto, sistemas computacionais embarcados, ou simplesmente sistemas embarcados (do inglês, *Embedded Systems*), conforme será abordado ao longo do texto, têm sido utilizados em uma ampla variedade de aplicações industriais para controlar praticamente qualquer tipo de equipamento ou planta. O avanço tecnológico e a redução de custos de produção de dispositivos eletrônicos, de forma geral, possibilitam o desenvolvimento de aplicações embarcadas cada vez mais complexas, desde sistemas de controle industriais e veiculares a eletrodomésticos e aparelhos eletrônicos mais sofisticados.

A grande diversidade de aplicações embarcadas torna difícil uma generalização clara e objetiva do que é um sistema embarcado. No entanto, as principais características destes sistemas podem ser identificadas. Trata-se de um sistema computacional, incluindo tanto recursos de *hardware* quanto de *software*, dedicado a realizar tarefas específicas de forma autônoma em um sistema mais abrangente [1–3]. Geralmente, um sistema embarcado interage com o ambiente no qual se encontra, através de sensores e atuadores, coordenando o funcionamento de componentes mecânicos e eletrônicos. Além disso, o sistema mais abrangente mencionado pode ter mais de um sistema embarcado em sua estrutura. Exemplos destes sistemas incluem aparelhos eletro-eletrônicos tais como os aparelhos celulares, terminais de caixa eletrônico, máquinas de refrigerante, calculadoras, televisores, aparelhos de DVD, fornos de microondas, máquinas de lavar roupa, impressoras, entre outros. Aplicações industriais de controle também utilizam sistemas embarcados para acionamento de máquinas elétricas, filtros ativos de potência, pré-

reguladores de fator de potência, controle de temperatura de sistemas térmicos, aquisição de dados, controle de nível de reservatórios, entre outras aplicações.

A estrutura básica de sistemas embarcados é semelhante a de um computador, mas o caráter específico das tarefas realizadas faz com que não sejam usados nem percebidos como um computador. Microcontroladores e microprocessadores são comumente usados como unidade central de processamento, aumentando a sofisticação de tais sistemas e influenciando novas aplicações industriais de controle. A maioria dos processadores hoje fabricados são utilizados em sistemas embarcados. Estima-se que cerca de cinco bilhões de microprocessadores estão em uso atualmente, representando 94% do mercado mundial, sendo os 6% restantes referentes aos processadores para plataforma PC (*Personal Computer*) [4].

Em geral, cada sistema embarcado, microprocessado ou microcontrolado, possui um *software*, usualmente denominado de *firmware*, que é armazenado como código objeto em memória não-volátil, tal como uma memória somente de leitura (do inglês, *Read Only Memory* - ROM). No *firmware* estão as instruções necessárias para o funcionamento de todo o sistema e para a comunicação e utilização dos componentes de *hardware* do sistema.

Um processo de atualização do código objeto pode ser necessário em situações envolvendo configuração de parâmetros, adição de novas funcionalidades e/ou correção de problemas no *firmware*. Alguns exemplos são: a alteração das funções de decodificação de vídeo e de áudio de um aparelho de DVD; atualização de uma versão mais estável do *firmware* de um *smartphone*, com novas funcionalidades adicionadas e/ou erros de funcionamento resolvidos; implementação de uma nova estratégia de controle para um sistema de controle de temperatura em ambiente industrial, entre outros exemplos. De modo geral, em sistemas embarcados nos quais não é possível que o usuário do sistema realize diretamente qualquer mudança funcional ou configuração paramétrica, o *firmware* tem que ser alterado manualmente e todo o processo de compilação, ligação (*linking*) e criação de um novo código objeto tem que ser realizado novamente. A operação de atualização é encerrada com o carregamento do código objeto no dispositivo de memória do sistema.

Por outro lado, ferramentas de *software* têm sido desenvolvidas com o intuito de automatizar todo este processo de atualização de *firmware*. Neste sentido, ferramentas de *software* para aplicações de controle e que realizam geração automática de código diretamente a partir de modelos em diagrama de blocos tornaram-se disponíveis. Uma delas é o *Real-Time Workshop* (RTW) [5], para uso com MATLAB/Simulink [6]. Estas ferramentas possibilitam o projeto do algoritmo de controle da aplicação usando representação em diagrama de blocos. A geração automática de código automatiza o processo de atualização de *firmware*, incluindo geração do código-fonte, compilação, ligação, criação do código objeto e carregamento deste no dispositivo de memória do *hardware* de controle da aplicação (uma placa DSP, por exemplo) [7]. Portanto, usando-se um conjunto de ferramentas de *software* como o MATLAB/Simulink/RTW é possível

gerar código executável para sistemas de controle a partir do algoritmo de controle da aplicação representado em diagrama de blocos, ou seja, pode-se realizar a elaboração/configuração do algoritmo de controle de modo mais prático e intuitivo através de uma abstração em alto nível.

A crescente complexidade de aplicações industriais tem conduzido projetistas e pesquisadores da área de ferramentas de apoio ao projeto de aplicações a elevar cada vez mais o nível de abstração de tarefas como especificação, desenvolvimento, validação e supervisão de sistemas de controle. Além disso, a tecnologia sem fio também tem sido usada com sucesso em aplicações industriais para monitoramento, manutenção preditiva, instrumentação, controle, etc. As principais vantagens incluem redução de custos, facilidade de manutenção e viabilidade de implementação em áreas remotas e/ou de difícil acesso [8–11].

O apelo pela eliminação de cabos, redução de custos, mobilidade e a crescente capacidade de processamento e armazenamento dos dispositivos portáteis também têm influenciado cada vez mais o emprego destes equipamentos em aplicações industriais, principalmente quando atividades de interface humano-máquina estão envolvidas. Em geral, estes dispositivos possuem conectividade sem fio, permitindo comunicação de modo mais prático. Portanto, o uso de dispositivos portáteis como ferramentas para configuração de sistemas de controle para aplicações industriais é bastante adequado.

Apesar do mercado industrial já demonstrar interesse na redução do custo de engenharia de configuração e programação, buscando diminuir a quantidade de horas necessárias para tais tarefas e oferecer maior praticidade para estas atividades [12], até o presente momento não foi encontrada na literatura uma linha de pesquisa abordando um sistema de configuração remota de algoritmos de controle para aplicações industriais baseando-se em comunicação sem fio e geração automática de código.

Esta tendência no desenvolvimento de ferramentas para configuração, análise e diagnóstico de processos industriais, aliada aos benefícios da tecnologia sem fio e da geração automática de código, com o intuito de proporcionar mobilidade, praticidade, facilidade, simplicidade e rapidez, é que motivou o desenvolvimento deste trabalho.

1.1 Objetivos

O objetivo principal deste trabalho é desenvolver uma ferramenta de *software* para configuração remota de algoritmos de controle para aplicações industriais a partir de dispositivos portáteis usando comunicação sem fio, ou seja, um configurador remoto que facilite e simplifique a interação entre o usuário e o sistema de controle através da configuração do algoritmo de controle em uma abstração de alto nível (diagrama de blocos).

O configurador remoto é compatível com um sistema com conectividade sem fio, capaz de realizar geração automática de código a partir de modelos em diagrama de blocos, e que pode

ser acoplado ao sistema de controle. No entanto, um sistema com tal capacidade dedicado para aplicações industriais ainda não existe, e por isso também faz parte dos objetivos deste trabalho a apresentação de uma arquitetura que torna possível a configuração de sistemas de controle industriais baseada em comunicação sem fio e geração automática de código.

Como objetivo específico, surge a montagem de uma plataforma de desenvolvimento experiemental baseada no sistema dSPACE, o qual possui uma infra-estrutura com recursos de *hardware* e *software* que viabiliza a implementação da Estação Base da arquitetura do sistema de configuração remota. A plataforma experimental também é utilizada para testar o funcionamento do configurador remoto e a implementação da arquitetura do sistema de configuração. A planta industrial utilizada na experimentação deste sistema de configuração consiste em um motor de indução alimentado por um inversor de tensão, de modo que diferentes estratégias de controle podem ser implementadas.

1.2 Revisão Bibliográfica

O estudo que despertou o interesse em seguir esta linha de pesquisa foi desenvolvido por Kesler *et al* [13], trabalho no qual um filtro ativo de potência paralelo foi projetado utilizando uma plataforma de prototipagem rápida baseada na geração automática de código do algoritmo de controle. O algoritmo é implementado através de um modelo em diagrama de blocos, elaborado no ambiente de desenvolvimento MATLAB/Simulink. Após a elaboração do algoritmo, o código executável é gerado e carregado no *hardware* de controle do sistema (uma placa DSP). Rapidamente, o sistema encontra-se pronto para funcionamento. O aspecto de destaque no referido artigo corresponde à facilidade de desenvolvimento do código gerado, tornando o processo de implementação do algoritmo completamente automático, em alto nível e testando-o no sistema de potência antes mesmo de definir sua versão final.

O termo Prototipagem Rápida (do inglês, *Rapid Prototyping*) surgiu há cerca de duas décadas e tem demonstrado ser uma técnica muito eficiente no auxílio ao desenvolvimento de novos produtos, equipamentos e sistemas [14]. O termo também é encontrado na literatura como *Rapid Control Prototyping* (RCP), quando envolvendo mais especificamente Engenharia de Controle. O aspecto chave da Prototipagem Rápida é a geração automática de código [15], que automatiza todo o processo de codificação do algoritmo de controle, incluindo geração do código-fonte, compilação, ligação, criação do código executável e carregamento deste no *hardware* de controle da aplicação.

Todavia, ferramentas de *software* para aplicações de controle em tempo real e que realizam geração automática de código diretamente a partir de modelos em diagrama de blocos tornaram-se disponíveis. Ferramentas como VisSim (Visual Solutions Inc.), MATRIXx (National Instruments), RIDE (Hyperception Inc.) e MATLAB/Simulink (The MathWorks Inc.)

possibilitam o projeto de sistemas de controle através de diagrama de blocos. Entre eles, o MATLAB/Simulink é provavelmente o mais conhecido e amplamente usado programa de simulação. O módulo RTW (*Real-Time Workshop*) [5] do Simulink pode realizar geração automática de código em linguagem C ou C++ a partir do diagrama de blocos construído. O RTW é capaz de gerar código executável para uma ampla variedade de sistemas operacionais com diferentes tipos de *hardware*, incluindo computadores pessoais, DSP's e microcontroladores. Ele também funciona como uma ferramenta de compilação cruzada, ou seja, o código pode ser compilado em uma plataforma (desenvolvimento) para ser executado em uma outra (alvo). Entretanto, uma plataforma integrada e completa (*hardware* e *software*) é necessária para oferecer ao projetista um ambiente de desenvolvimento desde a fase inicial de modelagem do sistema até os passos finais de geração de código e carregamento deste no *hardware* de controle do sistema.

Diante disso, algumas plataformas para prototipagem rápida têm sido propostas usando o ambiente MATLAB/Simulink/RTW e recursos de *hardware* personalizados ou disponíveis comercialmente, baseados em DSP's e microcontroladores.

Rebeschieβ [16] apresenta uma plataforma baseada em microcontrolador, integrando um conjunto de ferramentas para sistemas de controle em tempo real com o Simulink (MIRCOS). MIRCOS permite programação em alto nível usando o Simulink e operação em tempo real com o microcontrolador 80C166 (Siemens *Microelectronics*).

Hercog e Jezernik [17] apresentam um ambiente para prototipagem rápida, baseado em DSP e utilizando o MATLAB/Simulink, que fornece uma rápida e estável transição de uma simulação *off-line* no Simulink para a operação em tempo real sobre a placa DSP do sistema. Além disso, a visualização dos dados e um ajuste dos parâmetros *on-the-fly* são possíveis. Hong *et al* [18] descrevem uma implementação de algoritmos para DSP usando o *software* MATLAB e uma plataforma de *hardware* da Texas Instrument, TMS32OC30 *Evaluation Module*, sendo a mesma plataforma utilizada por Gan *et al* [15], os quais apresentam esta arquitetura como ambiente de prototipagem rápida educacional e de baixo custo.

Hanselmann [7] apresenta um "ambiente de desenvolvimento total" para prototipagem rápida que inclui MATLAB, Simulink, uma infra-estrutura de *hardware* baseada em DSP, além de um excelente conjunto de ferramentas de *software* para visualização de dados em tempo real. Trata-se da plataforma dSPACE [19], cujas placas controladoras são completamente programáveis no ambiente Simulink, a exemplo das placas DS1103 e DS1104.

A plataforma dSPACE pode ser e, de fato, é utilizada como parte da plataforma de desenvolvimento experimental deste trabalho, correspondendo ao sistema capaz de realizar geração automática de código. Conectividade sem fio é agregada ao dSPACE com o intuito de viabilizar o sistema de configuração remota e testar o funcionamento do configurador remoto.

A tecnologia sem fio tem sido usada em aplicações industriais para monitoramento remoto, instrumentação e controle com sucesso. Ramamurthy *et al*, em [8–11], apresentam o

estudo, projeto e implementação de uma plataforma com conectividade sem fio direcionada para aplicações industriais. Esta plataforma permite operação, monitoramento e configuração de parâmetros remotamente através de comunicação sem fio. A plataforma baseia-se em uma arquitetura adequada para uma variedade de aplicações industriais, envolvendo automação e controle. Tal arquitetura consiste em um conjunto de sensores, e atuadores comunicando-se com uma unidade central de controle usando um padrão de tecnologia sem fio. A contribuição chave do trabalho é a versatilidade do sistema desenvolvido e a habilidade de ser configurado para diversas aplicações.

Em [12], o foco do estudo é o impacto que o uso de ferramentas de configuração e diagnóstico tem causado em sistemas de acionamento industriais, destacando como principais fatores o crescente emprego de microprocessadores em aplicações de acionamento e o uso de tecnologias de rede que permitem controle e configuração remota. A tendência no setor industrial, segundo a pesquisa, é de crescimento no desenvolvimento de aplicações baseadas na plataforma PC, funcionando como ferramentas de configuração e monitoramento.

Outros pesquisadores também discutem o uso da Internet para supervisão e operação remota de dispositivos e sistemas de controle na indústria e no ensino de engenharia [20–23].

Apesar da iniciativa de linhas de pesquisa envolvendo o uso de tecnologia sem fio em ambientes industriais, ainda não foram contempladas nos estudos ferramentas para configuração remota de aplicações industriais agregando os benefícios proporcionados por um sistema que permite geração automática de código, como este trabalho apresenta.

A presente revisão bibliográfica visa um levantamento teórico fundamentando de maneira consistente o trabalho desenvolvido e contextualizando os principais tópicos abordados nesta dissertação.

1.3 Sinopse dos Capítulos

Esta dissertação está organizada em cinco capítulos, cujos conteúdos são apresentados, resumidamente, a seguir.

No Capítulo 2, é apresentada uma visão geral do sistema de configuração remota de algoritmos de controle para aplicações industriais baseado em geração automática de código e comunicação sem fio, e discutida toda a infra-estrutura da arquitetura abordada. Por fim, um cenário de aplicação para o sistema de configuração é apresentado, no qual uma plataforma de desenvolvimento experimental pode ser utilizada para testar o funcionamento de todo o sistema.

No Capítulo 3, a plataforma de desenvolvimento experiemental utilizada para testar o funcionamento do configurador remoto e a implementação da arquitetura do sistema de configuração é descrita, apresentando toda a infra-estrutura utilizada, tanto *hardware* quanto *software*,

com destaque para o sistema dSPACE, representando um sistema embarcado capaz de realizar geração automática de código. Também são apresentados os detalhes de projeto e implementação das aplicações cliente e servidor, incluindo a integração com a plataforma dSPACE e o funcionamento de toda a plataforma experimental construída.

No Capítulo 4, são descritos os testes realizados com a plataforma experimental, de forma a apresentar o funcionamento do configurador remoto em conjunto com toda a estrutura desenvolvida. Dois experimentos são realizados. O primeiro aborda um simples experimento de aquisição de dados utilizando os conversores A/D e D/A do *hardware* de controle da plataforma experimental, visando apresentar o funcionamento do processo de configuração remota de maneira detalhada. O segundo envolve o cenário de aplicação apresentado no Capítulo 2, consistindo em um motor de indução alimentado por um inversor de tensão no qual é possível a implementação de diferentes estratégias de controle, sendo abordado o acionamento baseado no princípio Volts/Hertz.

No Capítulo 5, uma conclusão geral do trabalho desenvolvido é apresentado de maneira concisa, e as perspectivas para futuros trabalhos a fim de continuar as pesquisas iniciadas nesta dissertação são citadas.

No Apêndice A é apresentada uma descrição da estrutura da plataforma experimental montada em laboratório.

Capítulo 2

Sistema de Configuração Remota

2.1 Introdução

Neste capítulo é apresentada uma visão geral do sistema de configuração remota de algoritmos de controle para aplicações industriais usando geração automática de código, e discutida toda a infra-estrutura da arquitetura abordada. Um cenário de aplicação envolvendo o sistema também é apresentado.

2.2 Arquitetura do Sistema

O sistema de configuração remota compreende uma arquitetura que deve facilitar e simplificar a interação entre o usuário e o sistema de controle através da configuração do algoritmo de controle em uma abstração de alto nível (diagrama de blocos) a partir de dispositivos portáteis usando comunicação sem fio.

A arquitetura do sistema deve apresentar versatilidade visto que é desejado que uma ampla variedade de aplicações possam ser configuradas/re-configuradas remotamente. Dessa forma, tal arquitetura deve envolver elementos estruturais (módulos funcionais) similares aos encontrados em plataformas voltadas para prototipagem rápida de sistemas de controle, a exemplo do sistema dSPACE [19].

Tipicamente, a arquitetura destas plataformas consiste em uma infra-estrutura com recursos de *hardware* e *software* podendo ser decomposta em quatro módulos básicos: Interface com o Usuário, Geração Automática de Código, *Hardware* de Controle, e Planta Industrial (ou simplesmente Planta). Esta arquitetura típica é frequentemente adotada na literatura, conforme observado na maioria dos trabalhos pesquisados [7, 13, 15, 16, 18, 20–22, 24–32]. A arquitetura típica é ilustrada na Figura 2.1.

A arquitetura destas plataformas oferece ao projetista um ambiente de desenvolvimento desde a fase inicial de modelagem do algoritmo de controle até os passos finais de geração

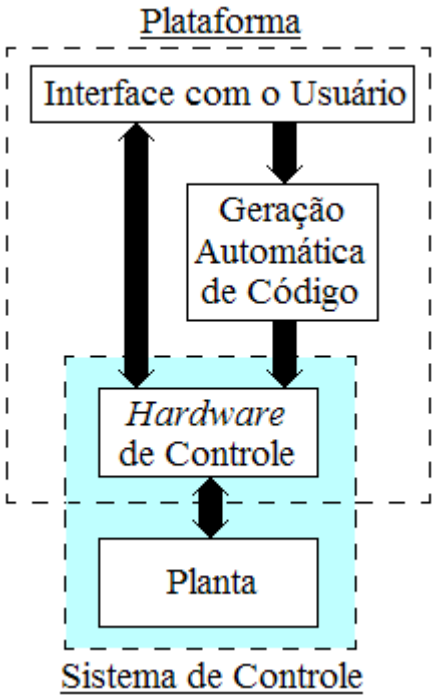


Figura 2.1: Arquitetura típica de plataformas para prototipagem rápida.

de código e carregamento deste no *hardware* de controle da aplicação industrial. Assim, tal arquitetura serviu de base para a definição da arquitetura do sistema de configuração remota, de modo a realizar uma configuração/re-configuração do algoritmo de controle em uma abstração de alto nível, sendo necessário, no entanto, agregar conectividade sem fio à estrutura.

Os módulos da arquitetura ilustrada na Figura 2.1 são descritos a seguir, destacando os relacionamentos e vínculos entre eles.

O módulo Interface com o Usuário consiste em um conjunto de ferramentas de *software* responsáveis pela interação entre o usuário e a plataforma, auxiliando na elaboração do modelo representativo do algoritmo de controle, bem como na visualização de variáveis e/ou parâmetros, isto é, monitoramento do sistema de controle (*hardware* de controle e planta industrial) em funcionamento. Este módulo relaciona-se com o módulo Geração Automática de Código para solicitar a geração de código executável a partir do modelo do algoritmo de controle. A ligação com o módulo *Hardware* de Controle está vinculada ao monitoramento e supervisão do sistema de controle.

O módulo Geração Automática de Código corresponde à ferramenta de *software* responsável por todo o processo de codificação do algoritmo de controle (geração do código-fonte, compilação, ligação, criação do código executável e carregamento deste no *hardware* de controle da aplicação). Este módulo gera o código executável a partir do modelo do algoritmo de controle elaborado no módulo Interface com o Usuário, e o carrega no módulo *Hardware* de Controle.

O módulo *Hardware* de Controle envolve todos os recursos de *hardware* utilizados pela

plataforma necessários para o processamento do algoritmo de controle, em geral, uma placa controladora baseada em microcontrolador ou microprocessador, e uma placa de interface com a planta industrial. Este módulo é responsável pela execução das rotinas em tempo real, além de estabelecer a interface entre os módulos Interface com o Usuário e Planta. Ou seja, o módulo *Hardware* de Controle permite que o módulo Interface com o Usuário tenha acesso a variáveis e parâmetros do modelo, tornando possível o monitoramento e atuação sobre a planta industrial.

O módulo Planta corresponde à planta industrial a ser controlada. Este módulo tem vínculo direto com o módulo *Hardware* de Controle, cujo *firmware* executa o algoritmo para controle da planta industrial em tempo real.

De fato, a arquitetura do sistema de configuração remota consiste na integração destes módulos, definindo-se uma estrutura cliente-servidor baseada em duas entidades, Estação Base e Estação Remota, que se comunicam através de um padrão de tecnologia sem fio. Além disso, são adicionados módulos responsáveis pela comunicação sem fio, tornando possível um procedimento de configuração remota do algoritmo de controle da aplicação industrial através de dispositivos portáteis.

O desenvolvimento de um sistema embarcado que implemente a Estação Base, ou seja, com conectividade sem fio e capaz de realizar geração automática de código, para ser acoplado a aplicações industriais de controle e interagir com o configurador remoto, pode ser considerado ideal para testar o funcionamento do configurador e a implementação da arquitetura, assim como uma solução em potencial para o setor industrial. Porém, o desenvolvimento de tal sistema embarcado demanda tempo e complexidades de projeto que inviabilizam sua implementação neste trabalho, sendo considerado, portanto, como uma das propostas para trabalhos futuros.

Contudo, é possível testar o funcionamento do configurador remoto e a implementação da arquitetura do sistema através de uma plataforma de desenvolvimento experimental baseada na arquitetura do sistema de configuração remota. Dessa forma, a referida plataforma experimental pode representar um sistema embarcado que satisfaz os requisitos da Estação Base.

2.2.1 Projeto do Sistema de Configuração Remota

O projeto do sistema de configuração remota segue uma metodologia semelhante à de projeto de sistemas embarcados, envolvendo os seguintes passos: requisitos, especificação, arquitetura, componentes e integração do sistema [1]. Os requisitos incluem os funcionais e os não funcionais. De fato, as funcionalidades do sistema a ser projetado são identificadas naturalmente, porém geralmente não são suficientes. Por isso requisitos não-funcionais devem ser incluídos, como desempenho, custo, tamanho, peso, consumo de energia, entre outros. Na fase de especificação, uma descrição mais detalhada do que o sistema deve fazer é estabelecida, declarando apenas como o sistema se comporta e não como é construído. A arquitetura é que irá apresentar mais informações do funcionamento interno do sistema, em termos dos componentes do

mesmo. Definidos os componentes necessários, o passo seguinte é o projeto e desenvolvimento dos mesmos, incluindo módulos de *software* e de *hardware*, se necessário. A integração dos componentes finaliza o procedimento de projeto do sistema.

Os requisitos do sistema de configuração remota são descritos a seguir:

1. Funcionalidade: o sistema é direcionado para aplicações industriais de controle, sendo capaz de realizar geração automática de código e permitindo a configuração do algoritmo de controle remotamente através de comunicação sem fio e em uma abstração em alto nível por meio de diagrama de blocos;

2. Interface com o usuário: o sistema utiliza um dispositivo portátil para realizar a interação entre o usuário e a sistema de controle por meio de uma aplicação que apresenta na tela do dispositivo o diagrama de blocos referente ao algoritmo de controle que deve ser configurado;

3. Desempenho: o sistema deve satisfazer os requisitos de tempo real da aplicação de controle, e o dispositivo portátil deve ser capaz de executar a aplicação cliente de interface com o usuário.

Em virtude da utilização de uma plataforma de desenvolvimento experimental, os requisitos não-funcionais como custo, tamanho, peso e consumo de energia não são considerados neste trabalho.

A especificação é definida com base na estrutura da arquitetura do sistema de configuração remota (Figura 2.2), incluindo seus módulos e o comportamento estabelecido entre eles, sendo estabelecida mais adiante, junto com o detalhamento da estrutura cliente-servidor.

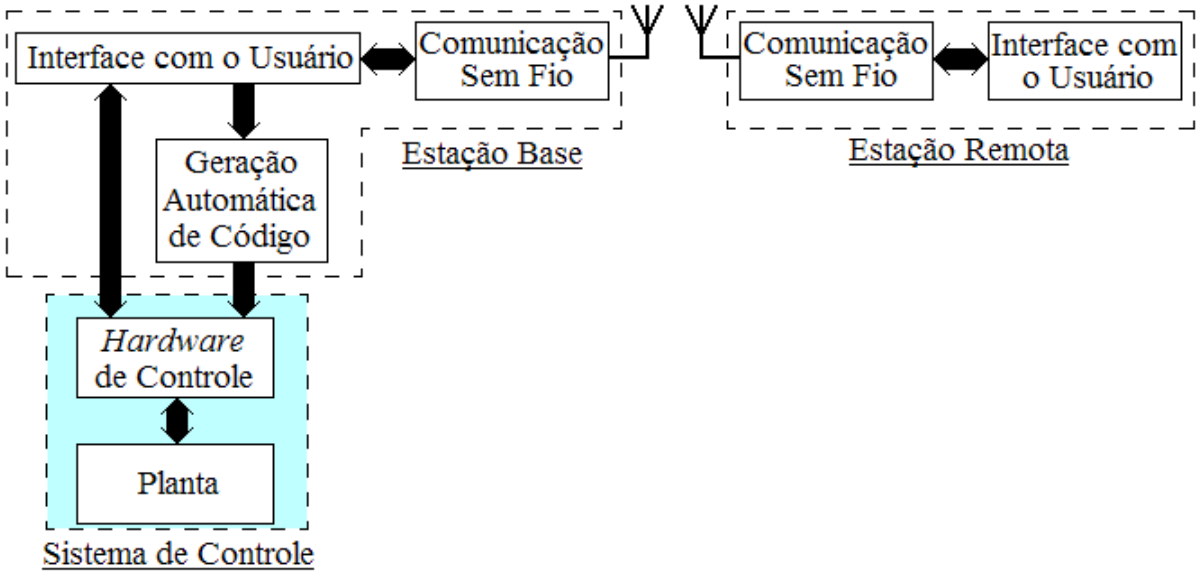


Figura 2.2: Arquitetura do sistema de configuração remota.

Arquitetura Cliente-Servidor

Arquitetura cliente-servidor é um modelo computacional envolvendo entidades separadas em clientes e servidores, mas interligadas entre si por uma rede. É um modelo comumente usado, podendo ser aplicado em sistemas onde entidades solicitam tarefas para serem feitas (clientes) e outras realizam o trabalho (servidores). Apesar do conceito ser usado em diversas áreas e aplicações, o modelo é praticamente o mesmo. Geralmente, o modelo cliente-servidor faz uso de protocolos de comunicação simples do tipo requisição/resposta. Há dois processos envolvidos: um no cliente e um no servidor. A fim de obter um serviço, um cliente envia uma mensagem de requisição (pedido) pela rede ao servidor. Este, por sua vez, executa as operações associadas ao serviço e envia uma mensagem de resposta ao cliente, contendo dados ou um código de erro caso o serviço não possa ser executado. Na Figura 2.3 é ilustrado o princípio de funcionamento do modelo cliente-servidor.

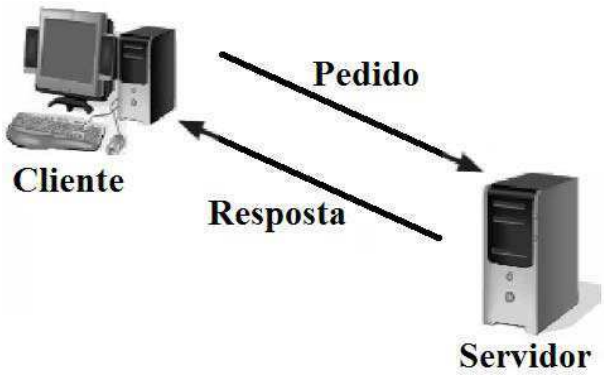


Figura 2.3: Modelo cliente-servidor.

A arquitetura cliente-servidor do sistema de configuração remota estabelece o funcionamento da Estação Base como servidor enquanto a Estação Remota realiza a função de cliente. Neste esquema, usuários remotos (clientes) podem trocar dados com o servidor através de comunicação sem fio e acessar informações para trabalhar com elas à distância. Na Figura 2.4 é apresentado um cenário exemplo da arquitetura cliente-servidor adotada, destacando a possibilidade de interação do servidor (PC) com diferentes tipos de dispositivos portáteis capazes de realizar a função de cliente.

De acordo com a Figura 2.2, a Estação Remota (parte cliente) é usada para interface gráfica com o usuário e operação remota, sendo responsável pela configuração do algoritmo de controle em uma abstração de alto nível usando representação em diagrama de blocos. Já a Estação Base (parte servidor) tem como função prover o serviço de configuração remota do modelo do algoritmo de controle da aplicação e realizar todo o processo de geração automática de código, carregando o código no *hardware* de controle responsável por interagir com a planta industrial a ser controlada.

Figura 2.4: Estrutura cliente-servidor adotada.

A Estação Base envolve os módulos Interface com o Usuário e Geração Automática de Código (Figura 2.1), acrescentando-se ainda um novo módulo denominado Comunicação Sem Fio, que tem a finalidade de estabelecer a comunicação remota entre as estações.

O módulo Interface com o Usuário transfere a responsabilidade pela interação entre o usuário e o sistema de controle para o configurador remoto, isto é, a configuração do modelo representativo do algoritmo de controle. A funcionalidade de visualização de variáveis e/ou parâmetros do sistema de controle pode ser mantida para finalidades de verificação e monitoramento local da planta. Este módulo relaciona-se com os módulos Geração Automática de Código e *Hardware* de Controle da mesma forma como na arquitetura típica (Figura 2.1). Com o módulo Comunicação Sem Fio a Interface com o Usuário disponibiliza serviços ao nível de sistema, como operações de entrada e saída, execução de comandos e acesso aos dados do sistema de controle.

O módulo Comunicação Sem Fio é responsável pela implementação de todos os recursos necessários para a comunicação sem fio, segundo um padrão de tal tecnologia. Este módulo atua como um servidor, provendo o serviço de configuração remota. É permitida a conexão de apenas um cliente por vez, visto que uma operação de atualização de *firmware* será realizada, e portanto deve-se evitar qualquer situação que possa gerar inconsistência de dados e, conseqüentemente, problemas de funcionamento do sistema de controle. O módulo Comunicação Sem Fio relaciona-se com o módulo Interface com o Usuário conforme o parágrafo anterior.

O funcionamento do módulo Geração Automática de Código é o mesmo daquele da arquitetura típica (Figura 2.1), sem alterações.

A Estação Remota, por sua vez, possui apenas dois módulos: Interface com o Usuário e Comunicação Sem Fio. O módulo Interface com o Usuário consiste em um *software* aplicativo responsável por permitir a interação entre o usuário e a aplicação de controle. Esta interação ocorre através da representação em diagrama de blocos do algoritmo de controle da aplicação, sendo possível a configuração do modelo do algoritmo. O módulo Comunicação Sem Fio tem a mesma função daquela da Estação Base, porém este módulo atuará como cliente, fazendo

uso do serviço de configuração remota. Este módulo deve implementar o mesmo padrão de tecnologia sem fio adotado pela Estação Base. Ou seja, de fato a Estação Remota corresponde ao configurador remoto.

Quanto à especificação, uma descrição do funcionamento do procedimento de configuração remota pode ser utilizada.

Operação Típica do Sistema de Configuração Remota

Supondo que o sistema de controle encontra-se desativado, inicialmente, o operador do sistema deve colocar a planta industrial em condições para entrar em funcionamento. Em seguida, deve-se inicializar o sistema de configuração remota, mais precisamente, apenas iniciar a execução da aplicação servidor da Estação Base. O servidor, então, anuncia um serviço de configuração remota e aguarda pela conexão de uma aplicação cliente (Estação Remota, configurador remoto). Neste momento, o próprio operador do sistema, um supervisor ou algum usuário capaz de utilizar adequadamente o configurador remoto pode proceder com a configuração do sistema de controle.

A partir do configurador remoto, o usuário deve executar a aplicação cliente e tentar conectar-se ao servidor. Primeiramente, é realizada uma busca por serviços, mais especificamente pelo serviço de configuração remota oferecido pelo servidor do sistema de controle que se deseja configurar. A busca pode ser realizada pelo nome do serviço oferecido, e também por um número de identificação, e/ou outros recursos que fornecem uma maior segurança, caso seja necessário.

Após descoberto o serviço, um pedido de conexão é emitido automaticamente ao referido servidor. A conexão é estabelecida desde que o servidor não esteja conectado à uma aplicação cliente. Caso haja tentativa de conexão de múltiplos clientes ao mesmo tempo, apenas um conseguirá: aquele que se conectar primeiro.

Estabelecida a conexão, o servidor aguarda por comandos do cliente (configurador remoto). O usuário pode realizar operações como iniciar ou parar a execução do sistema de controle, encerrar a conexão com o servidor, solicitar ao servidor o envio do modelo atual do algoritmo de controle armazenado na Estação Base (operação de *Upload*), ou ainda enviar para o servidor um modelo de algoritmo de controle para ser implementado no sistema de controle (operação de *Download*).

Para o caso de um procedimento típico de configuração do sistema de controle, supõem-se a solicitação ao servidor do envio do modelo atual do algoritmo de controle (*Upload*). O servidor ao receber este comando, recupera o modelo e o envia para o cliente. Por sua vez, a aplicação cliente recebe o modelo, apresentando na tela do configurador remoto o diagrama de blocos correspondente.

Após a configuração do modelo recebido, o modelo alterado deve ser enviado para a Es-

tação Base (*Download*). Então, o servidor confirma que está preparado para receber o novo modelo configurado pela aplicação cliente. Ao recebê-lo, o servidor armazena o modelo no sistema de arquivos da Estação Base e solicita os serviços dos módulos responsáveis pela geração automática de código e execução do novo algoritmo de controle imediatamente.

Um diagrama de sequência UML (do inglês, *Unified Modeling Language*) apresentado na Figura 2.5 torna possível um melhor entendimento da especificação conceitual descrita.

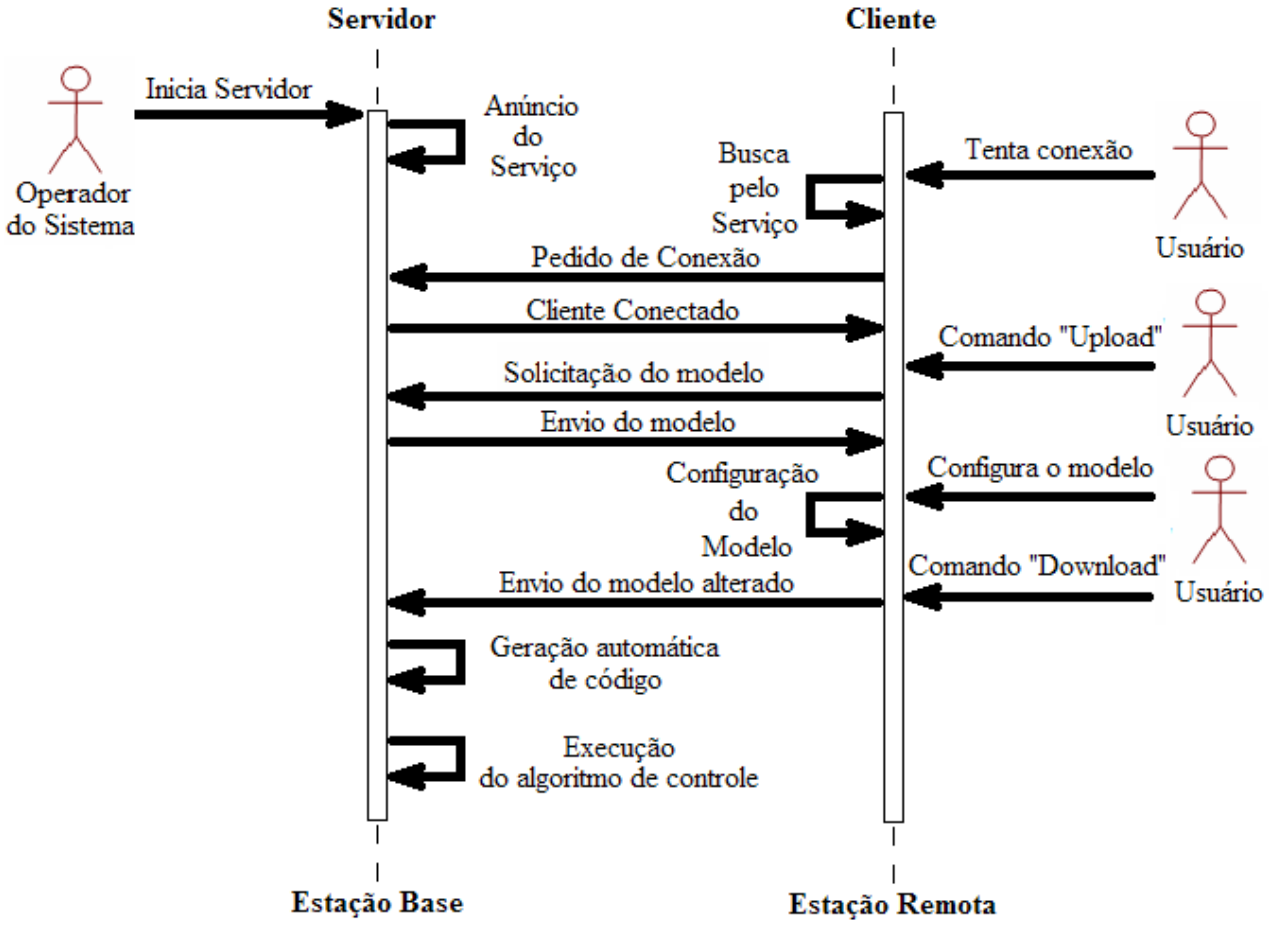


Figura 2.5: Diagrama de sequência UML para uma sequência típica do procedimento de configuração.

Recursos Utilizados na Implementação da Arquitetura

Com base na arquitetura do sistema, os principais componentes e recursos (incluindo *hardware* e *software*) para implementação dos módulos funcionais podem ser definidos. A tecnologia Bluetooth [33] é adotada como padrão de comunicação sem fio em virtude de características como facilidade de implementação, simplicidade e baixo custo. A Estação Remota pode ser implementada utilizando qualquer dispositivo portátil com conectividade Bluetooth e capaz de executar o *software* aplicativo cliente, para configuração do algoritmo de controle. O N800 *Internet Tablet* é escolhido para exercer a função de configurador remoto. Para a implementação

da Estação Base é utilizada a plataforma dSPACE e o pacote de ferramentas de *software* associado, em virtude desse conjunto oferecer todos os recursos necessários para a implementação dos módulos funcionais da Estação Base. O módulo *Hardware* de Controle é implementado utilizando-se a placa controladora e o painel de conectores do dSPACE, o módulo Geração Automática de Código é implementado pelas ferramentas MATLAB/Simulink/RTW, o módulo Interface com o Usuário é implementado pela ferramenta ControlDesk e pelo *software* aplicativo servidor, e por fim o módulo Comunicação Sem Fio é implementado agregando-se conectividade Bluetooth ao dSPACE através de um adaptador USB.

No entanto, vale salientar que a plataforma dSPACE é utilizada apenas por possuir os recursos que viabilizam a implementação de módulos da arquitetura do sistema. O dSPACE não é o sistema ideal ou solução final para implementação da Estação Base.

2.3 Cenário de Aplicação

Um cenário de aplicação onde o sistema de configuração remota pode ser utilizado é um ambiente industrial no qual a aplicação de controle consiste em um motor de indução alimentado por um inversor de tensão, no qual é possível usar diferentes estratégias de controle para o acionamento da máquina. Na Figura 2.6 é apresentado o referido cenário de aplicação.

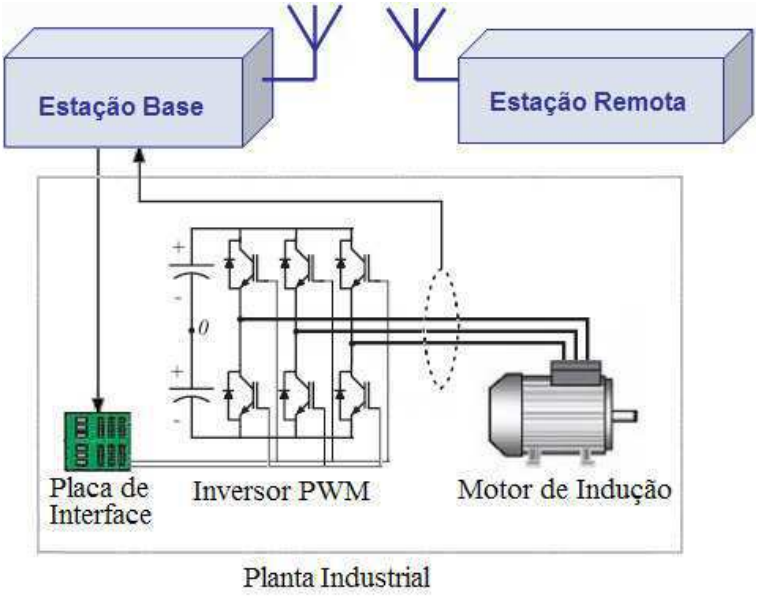


Figura 2.6: Cenário de aplicação para o sistema de configuração remota.

Neste cenário, um operador do sistema (usuário) pode implementar um novo algoritmo de controle através da Estação Remota(configurador remoto). Em geral, aplicações industriais de controle onde sensores e atuadores são empregados para controlar parâmetros e/ou o estado do sistema, são favoráveis para a implementação dessa arquitetura. Com base nos componentes da

arquitetura abordada e nos elementos definidos para implementação do sistema de configuração, uma plataforma de desenvolvimento experimental pode ser construída para representar este cenário de aplicação e testar o funcionamento do sistema. Na figura 2.7 é ilustrado o cenário de aplicação representado pela plataforma de desenvolvimento experimental.

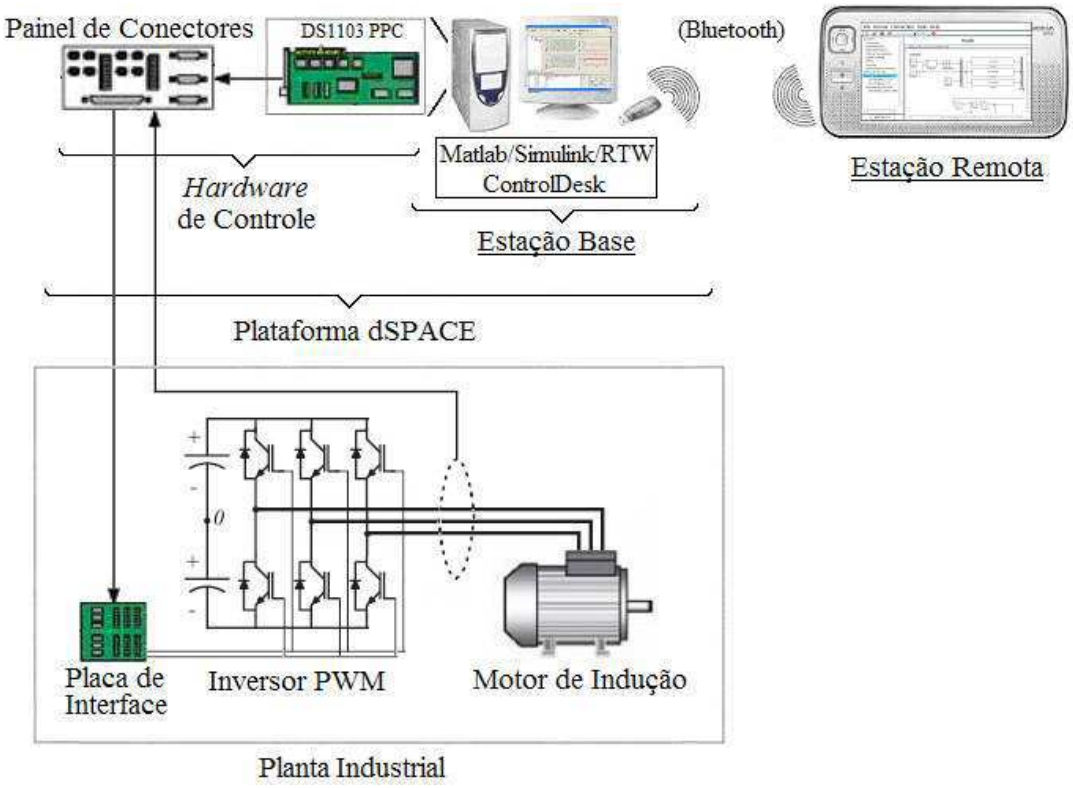


Figura 2.7: Cenário de aplicação representado pela plataforma de desenvolvimento experimental.

A estrutura e o funcionamento da plataforma de desenvolvimento experimental são detalhados no capítulo seguinte.

2.4 Sumário

Neste capítulo foi apresentada uma visão geral do sistema de configuração remota de algoritmos de controle para aplicações industriais usando geração automática de código e comunicação sem fio. A definição da estrutura, organização e funcionamento da arquitetura foram baseados na infra-estrutura típica de plataformas para prototipagem rápida de sistemas de controle encontradas na literatura. Os principais módulos funcionais foram identificados e um projeto arquitetural foi discutido, apresentando requisitos, especificação e a estrutura da arquitetura abordada. Por fim, um cenário de aplicação para o sistema de configuração foi apresentado, no qual uma plataforma de desenvolvimento experimental pode ser utilizada para testar o funcionamento de todo o sistema.

Capítulo 3

Plataforma de Desenvolvimento Experimental

3.1 Introdução

Este capítulo tem como objetivo descrever a plataforma de desenvolvimento experimental utilizada para testar o sistema de configuração de sistemas de controle baseada em dispositivos móveis sem fio. A plataforma é constituída por um sistema dSPACE, representando a Estação Base da arquitetura, e um dispositivo portátil realizando a função de configurador remoto do sistema de controle, i.e., representando a Estação Remota. A tecnologia Bluetooth é utilizada como padrão de comunicação sem fio entre as estações.

De fato, a plataforma dSPACE é utilizada por possuir os recursos que viabilizam a implementação dos módulos Interface com o Usuário (ControlDesk), Geração Automática de Código (MATLAB/Simulink/RTW) e *Hardware* de Controle (placa controladora e painel de conectores) da arquitetura do sistema.

No entanto, é necessário o desenvolvimento da estrutura cliente-servidor da arquitetura. Para isso, são desenvolvidos dois aplicativos de *software*: um é o cliente (Estação Remota), responsável pela configuração do modelo do algoritmo de controle da aplicação, e o outro servidor (Estação Base), incluído no módulo Interface com o Usuário e responsável pela integração entre os demais módulos da Estação.

Os detalhes de implementação das aplicações cliente e servidor são apresentados, incluindo a integração com a plataforma dSPACE e o funcionamento de toda a plataforma experimental construída.

3.2 Plataforma dSPACE

A plataforma dSPACE (acrônimo para *Digital Signal Processing And Control Engineering*) é um sofisticado ambiente de desenvolvimento para prototipagem rápida de sistemas de controle. O sistema dSPACE é excelente para atividades de pesquisa e desenvolvimento em laboratórios de engenharia. Utilizando recursos de *hardware* de alto desempenho e um aprimorado conjunto de ferramentas de *software* para simulação e experimentação de sistemas de controle, o dSPACE auxilia todas as etapas do processo de desenvolvimento, desde o projeto até a implementação de aplicações de controle [19].

Esta plataforma surgiu como solução diante da necessidade por ferramentas que facilitem o processo de desenvolvimento e simulação de sistemas de controle, proporcionando também confiabilidade e flexibilidade. Neste sentido, o sistema dSPACE oferece ferramentas para modelagem, simulação e experimento em tempo real de sistemas de controle [34].

3.2.1 Estrutura

A plataforma dSPACE consiste em uma infra-estrutura envolvendo recursos de *hardware* e de *software*. A plataforma está disponível comercialmente em duas configurações de *hardware*: *single-board*, na qual o processador principal e os módulos de interface de entrada e saída de dados encontram-se em uma única placa; e modular, na qual o *hardware* é composto de um ou mais processadores e módulos de interface de entrada e saída de dados, alocados em placas diferentes.

O sistema dSPACE utilizado neste trabalho encontra-se disponível no Laboratório de Eletrônica Industrial e Acionamento de Máquinas (LEIAM-DEE-UFCG). Consiste em um PC equipado com uma placa controladora DS1103 PPC (configuração *single-board*) [35], painel de conectores e ferramentas de *software* específicas para as funções de simulação e experimentação.

A placa controladora DS1103 PPC é especificamente projetada para desenvolvimento de controladores digitais multivariáveis com elevada capacidade de processamento e para simulações de sistemas de controle em tempo real. O processador principal (mestre) da placa DS1103 é um PowerPC PPC604e. Algumas características deste processador são listadas a seguir [35]:

- Microprocessador RISC (*Reduced Instruction Set Computer*);
- Implementa a arquitetura PowerPC (acrônimo para *Power Optimization With Enhanced RISC - Performance Computing*);
- Arquitetura de 32 bits;
- Processa dados do tipo inteiro de 8, 16 e 32 bits;
- Processa dados do tipo flutuante de 32 e 64 bits;

- Unidade de gerenciamento de memória separada para instruções e dados.

A placa DS1103 inclui ainda um processador DSP escravo baseado no microprocessador TMS320F240 da Texas Instruments, membro da família de controladores DSP baseados na geração TMS320C2xx de processadores digitais de sinais de 16 bits em ponto fixo. Esta família é otimizada para aplicações de controle digital de motores, combinando baixo custo, elevada capacidade de processamento e diversos componentes periféricos sofisticados.

Algumas especificações da placa controladora DS1103 PPC [35] são apresentadas a seguir:

- Processador PowerPC PPC604e;
- 2 MB de memória SRAM local e 128 MB de memória SDRAM global;
- Controlador de interrupção com 22 fontes de interrupção e 4 interrupções externas;
- Sensor de temperatura para o processador PPC;
- 4 Conversores A/D multiplexados, cada um com 4 conversores A/D de 16 bits e tempo de conversão de $4\mu s$ (16 canais);
- 4 Conversores A/D de 12 bits e tempo de conversão de $800ns$;
- 8 Conversores D/A de 14 bits com $5\mu s$ de tempo de ajuste;
- 1 Codificador Incremental Analógico;
- 6 Codificadores Incrementais Digitais;
- Unidade digital de entrada e saída de dados de 32 bits;
- Microprocessador DSP escravo TMS320F240.

A referida placa é apresentada na Figura 3.1 a seguir.

Figura 3.1: Placa controladora dSPACE DS1103 PPC.

A placa DS1103 é compatível com o padrão ISA, podendo ser conectada diretamente a um PC ou então inserida em uma caixa de expansão comunicando-se com o PC através de uma placa de comunicação (DS811, por exemplo) ou via Ethernet [35]. Para propósitos de prototipagem

rápida de sistemas de controle, painéis de conectores específicos fornecem fácil acesso a todos os sinais de entrada e saída da placa. Com isso, o computador *host* (PC) é transformado em um excelente ambiente de desenvolvimento de sistemas de controle bastante adequado para vários campos de aplicação, tanto na área industrial quanto na acadêmica. Na Figura 3.2 é ilustrada uma visão geral da arquitetura da placa DS1103 e seus módulos funcionais.

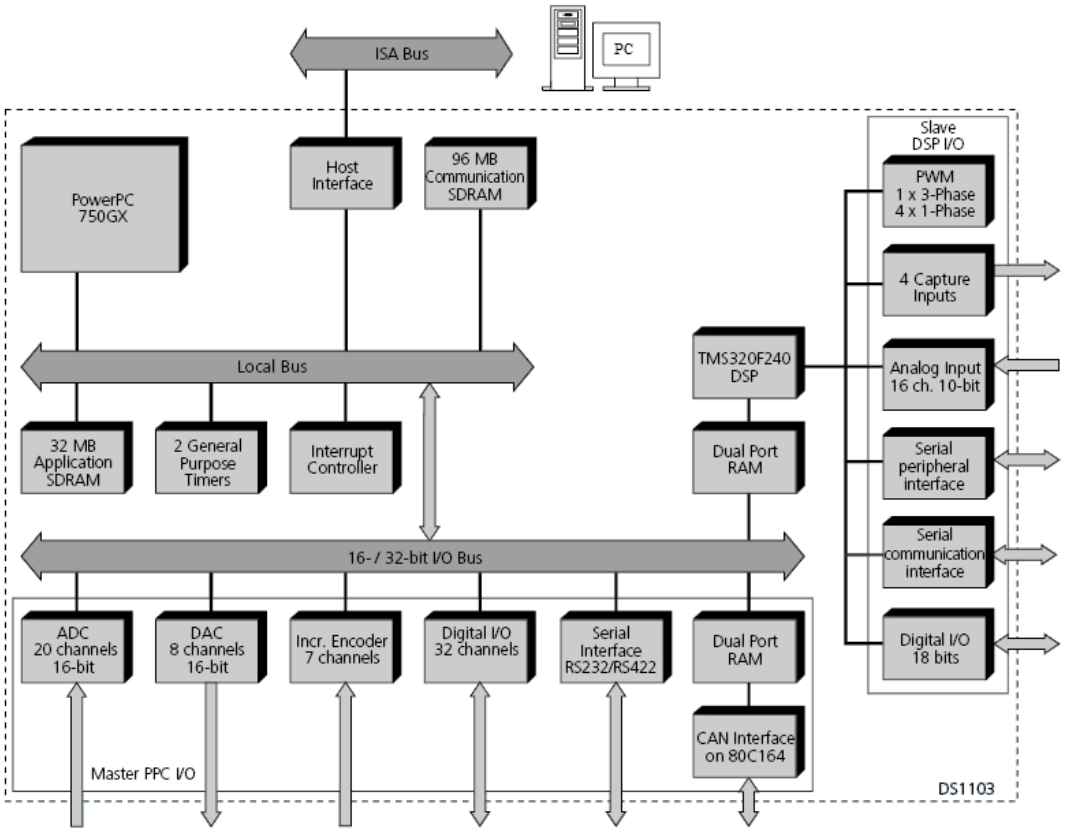


Figura 3.2: Visão geral da arquitetura da placa controladora DS1103 PPC.

A placa DS1103 é responsável pela execução das rotinas de controle em tempo real e por estabelecer a interface entre o *software* aplicativo para monitoramento e controle (interface gráfica com o usuário) e a planta. Por isso ela é considerada o núcleo da plataforma dSPACE.

Dentre o conjunto de ferramentas de *software* da plataforma pode-se destacar a ferramenta ControlDesk [36], que representa a principal interface gráfica com o usuário do sistema. Esta plataforma também pode trabalhar em conjunto com ferramentas como o MATLAB e o Simulink, ideais para a fase de modelagem de sistemas de controle e que incluem a ferramenta RTW [5], responsável pela geração automática de código. O pacote de *software* específico da plataforma dSPACE corresponde ao *release 4.0* e o pacote MATLAB/Simulink/RTW, ao *release R13*, operando no sistema operacional Windows [34].

3.2.2 Modelagem, Simulação e Experimento

A placa DS1103 é completamente programável no ambiente MATLAB/Simulink [6]. Dessa forma, o desenvolvimento de estratégias de controle pode ser baseado em diagrama de blocos. Diagramas de blocos elaborados na plataforma são convertidos em programas executáveis, sendo em seguida carregados e executados na placa controladora. O processo de desenvolvimento de sistemas de controle na plataforma dSPACE é realizado em três etapas: modelagem, simulação e experimento em tempo real do algoritmo de controle.

O MATLAB e o Simulink constituem excelentes ferramentas de modelagem e simulação, e podem ser utilizadas em conjunto com a plataforma dSPACE. O MATLAB é uma ferramenta para o desenvolvimento de algoritmos, visualização e análise de dados usando programação textual (*hand coding*), enquanto que o Simulink é uma ferramenta interativa para modelagem, simulação e análise de sistemas dinâmicos baseada em diagrama de blocos [6].

A modelagem de sistemas de controle é realizada de forma mais prática, simples e fácil com estas ferramentas, principalmente quando recursos específicos da plataforma, isto é, da placa controladora estão disponíveis como bibliotecas de funções no próprio ambiente de desenvolvimento (Figura 3.3). Assim, com o auxílio de bibliotecas como a RTI (*Real-Time Interface*) [37] para interface em tempo real com a placa controladora da plataforma dSPACE, pode-se projetar sistemas de controle a partir da construção de modelos em diagrama de blocos do próprio Simulink. A biblioteca RTI da plataforma dSPACE (específica para a placa DS1103) é apresentada na Figura 3.4.

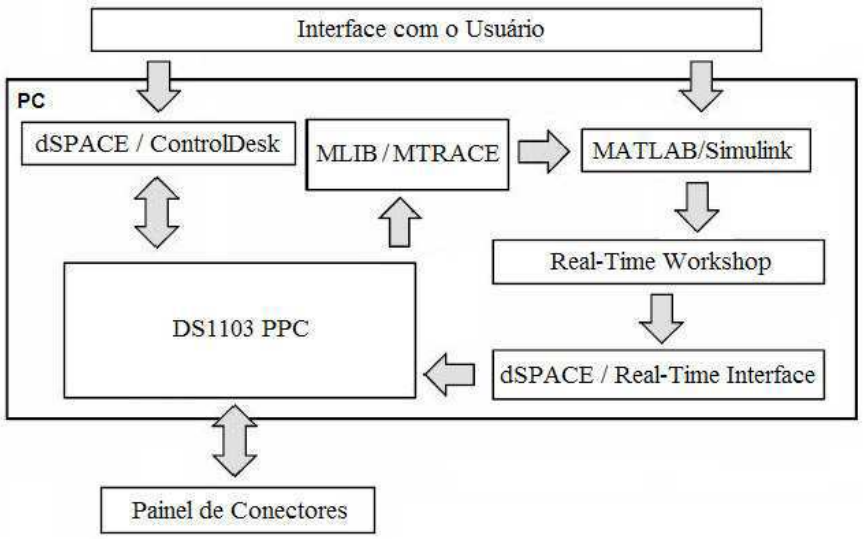


Figura 3.3: Infra-estrutura do ambiente de desenvolvimento da plataforma dSPACE.

Seguindo o processo de desenvolvimento, o próximo passo consiste na implementação do modelo construído no *hardware* de controle, ou seja, na placa controladora do dSPACE. As ferramentas RTI e RTW trabalham em conjunto para tornar possível a implementação do

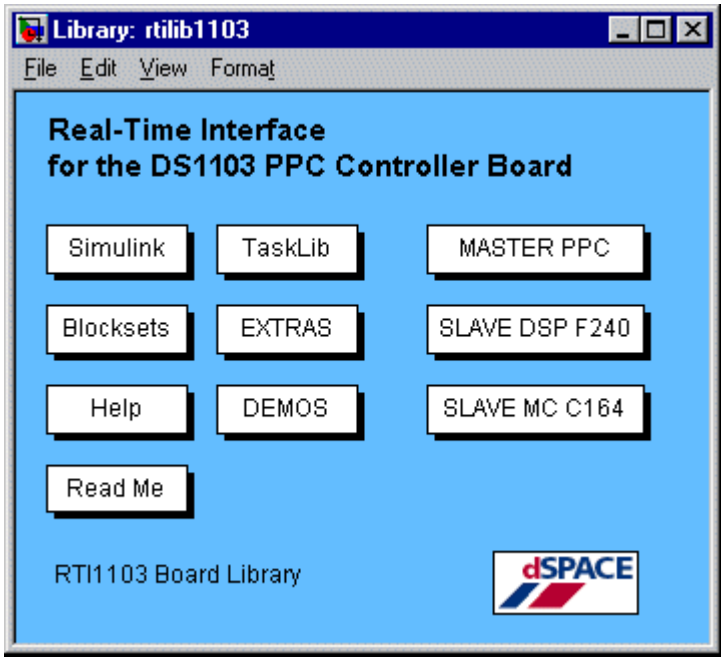


Figura 3.4: Biblioteca RTI da plataforma dSPACE (placa DS1103).

algoritmo de controle desenvolvido. Dois modos de desenvolvimento e implementação do algoritmo de controle da aplicação são possíveis: via Simulink ou através da programação manual do código-fonte (*hand coding*) seguindo padrões pré-estabelecidos. Na implementação via Simulink, os blocos existentes na biblioteca RTI possibilitam o acesso direto aos recursos de *hardware* da placa controladora, mantendo a programação em nível de diagrama de blocos. Através do módulo RTW, todos os arquivos necessários à implementação do algoritmo de controle (TRC[1], PPC[2], MAP[3], SDF[4] e PAR[5]) são gerados automaticamente. Já na implementação manual do código-fonte, o algoritmo de controle é desenvolvido utilizando linguagem C e funções pré-definidas disponíveis para programação da placa controladora através da biblioteca RTLIB (*Real-Time Library*) [38]. Neste modo de implementação, os arquivos TRC e SDF devem ser criados pelo projetista manualmente, seguindo padrões de convenção e sintaxe pré-estabelecidos na biblioteca da plataforma, enquanto que os arquivos PPC, MAP e PAR são gerados após a compilação do código-fonte utilizando o compilador Microtec PowerPC [34].

[1] *Arquivo que inclui a descrição das variáveis do modelo (nome, grupo, tipo, entre outras características) que serão utilizadas pelo ControlDesk.*

[2] *Arquivo executável do processador PowerPC da placa controladora do dSPACE, gerado a partir do compilador Microtec PowerPC.*

[3] *Arquivo contendo o mapa de endereçamento físico das variáveis do modelo, gerado a partir do compilador Microtec PowerPC.*

[4] *Arquivo de descrição do sistema, fazendo referência aos arquivos TRC e PPC, e que deve ser carregado no ControlDesk. Este arquivo indica o arquivo executável e todos os outros relacionados que devem ser utilizados pelo dSPACE.*

[5] *Arquivo contendo informações das variáveis paramétricas do modelo, que podem ser alteradas pelo usuário em tempo real.*

Após a geração do arquivo executável, procede-se com o carregamento do mesmo na placa controladora. Então, a execução do algoritmo é iniciada, ocorrendo integralmente em tempo real no *hardware* de controle, ou seja, na placa controladora DS1103 PPC. Através do painel de conectores da plataforma, o qual permite o acesso aos dispositivos de entrada e saída de dados existentes na placa, pode-se estabelecer uma interface com a planta a ser controlada. Várias aplicações utilizando a plataforma dSPACE estão relacionadas com as áreas de acionamento de máquinas, mecatrônica, controle automotivo e sistemas de eletrônica de potência de uma forma geral.

O dSPACE disponibiliza uma ferramenta de *software* que permite o monitoramento e controle da placa controladora, ou seja, permite a visualização de dados em tempo real. Trata-se do ControlDesk [36], que possui recursos para visualizar e armazenar valores de variáveis do modelo, alterar parâmetros do modelo em tempo real, entre outros. Assim, pode-se controlar em tempo real a atividade experimental com o sistema de controle. Além disso, a plataforma dSPACE possui bibliotecas, MLIB/MTRACE, que permitem o acesso à placa controladora diretamente do MATLAB, tornando possível o uso de todos os recursos do MATLAB durante a etapa de implementação do algoritmo.

3.2.3 Geração Automática de Código

A geração automática de código é sem dúvida uma das principais características da Prototipagem Rápida. Ela transforma o processo de desenvolvimento do algoritmo de controle em um procedimento automático, incluindo geração do código-fonte, compilação, ligação (*linking*), e carregamento na placa controladora da plataforma, isto é, desde a criação do código-fonte (a partir do diagrama de blocos) até a execução em tempo real do algoritmo de controle na placa. Esta automatização do processo permite que mudanças de projeto possam ser feitas diretamente no diagrama de blocos e estarem prontas para novos testes em segundos.

A ferramenta de *software* responsável pela geração de código executável é o *Real-Time Workshop* (RTW) [5], cujo processo de construção automática do código é apresentado na Figura 3.5.

O processo de construção do programa executável é iniciado a partir do *software* Simulink. Com o modelo elaborado e após um ajuste adequado nos parâmetros de simulação do modelo [19], o código-fonte do modelo é automaticamente gerado usando-se uma ferramenta de *software* chamada *Target Language Compiler* (TLC) integrada ao RTW [5]. O RTW utiliza um arquivo de extensão *.tmf (*Template Makefile*, TMF), para construir o arquivo executável a partir do código-fonte gerado. O arquivo TMF deve especificar o compilador adequado e as opções de compilação para o processo de criação do código objeto. Um arquivo de configuração *makefile* (*.mk) é criado a partir do arquivo TMF, através da cópia de cada linha do referido arquivo. Um *makefile* é um arquivo de configuração que define o local dos arquivos-fonte, como

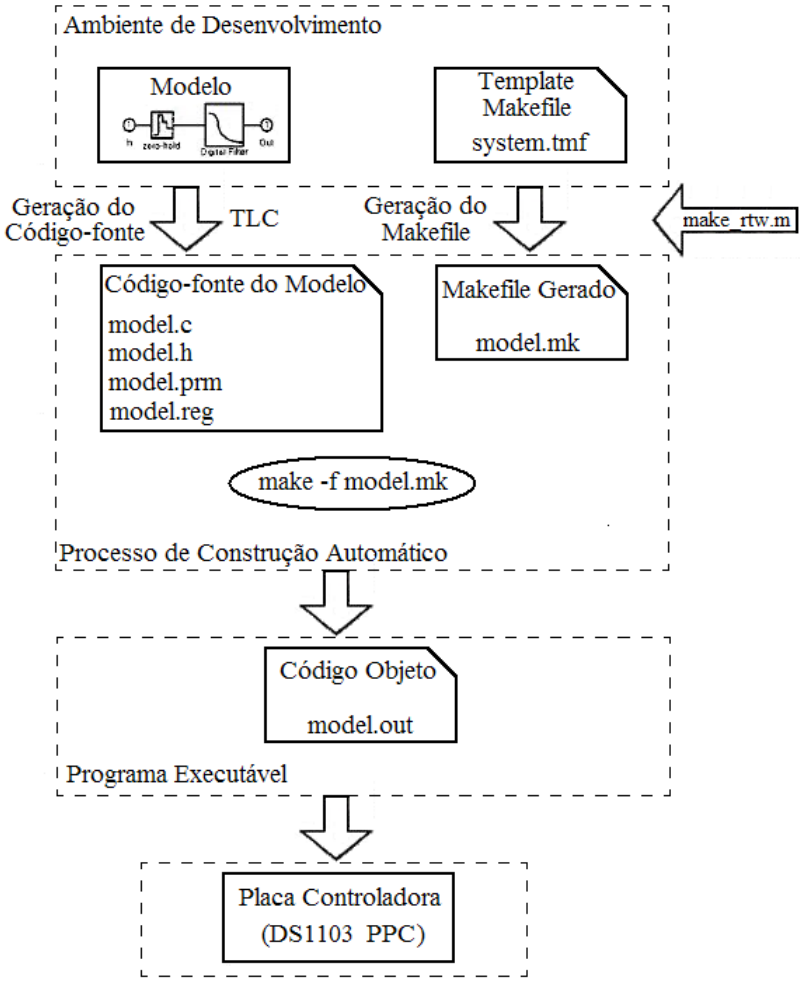


Figura 3.5: Processo de construção automática do código.

eles serão compilados e ligados (*linking*) para criar um programa executável. Em seguida, uma ferramenta de *software* (*make utility*) constrói um arquivo executável (*.out) a partir do conjunto de arquivos especificados no arquivo *makefile* (model.c, model.h, etc). Por fim, o arquivo executável é carregado na placa controladora da plataforma. O projetista pode ainda configurar o processo de construção modificando o arquivo TMF.

3.2.4 Ferramentas para Simulações e Experimentos em Tempo Real

A principal ferramenta para interface gráfica com o usuário da plataforma dSPACE é o *software* ControlDesk. Esta ferramenta oferece todas as funções para controle, monitoramento e automação da atividade experimental, tornando o desenvolvimento de sistemas de controle mais eficiente. Utilizando este *software*, o usuário da plataforma pode montar um painel de instrumentação virtual adequado para seu experimento, utilizando recursos como botões, barras de rolagem (*sliders*), *displays* e gráficos para visualização dos valores das variáveis do modelo, entre outros. Também é possível alterar parâmetros do modelo em tempo real, aprimorando a

atividade experimental com o sistema de controle.

Na Figura 3.6 é ilustrada a interface gráfica do *software* ControlDesk durante um experimento realizado com a plataforma dSPACE.

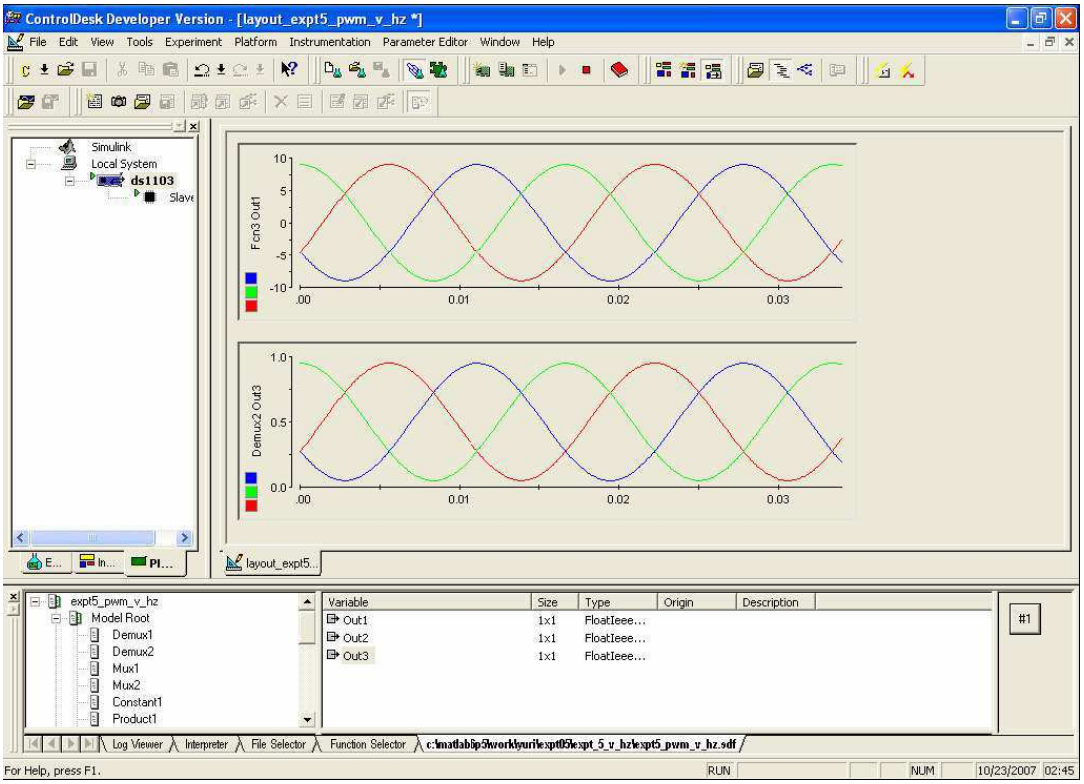


Figura 3.6: Vista da interface gráfica do *software* ControlDesk.

Vale salientar que simulações e depuração também podem ser desenvolvidas no ambiente MATLAB/Simulink, antes mesmo da geração automática de código, permitindo uma avaliação inicial do algoritmo de controle ainda em nível de *software*.

No ControlDesk, a placa controladora DS1103 PPC assim como o Simulink são consideradas como "plataformas de simulação" para o algoritmo de controle desenvolvido [36]. Na placa DS1103, a simulação ocorre em tempo real no próprio *hardware* de controle. De modo semelhante, o Simulink é utilizado para executar simulações virtuais, isto é, em nível de *software*. Portanto, ambas "plataformas de simulação" podem ser utilizadas pelo ControlDesk. Os mesmos recursos de instrumentação virtual são válidos para qualquer uma dessas "plataformas".

Outra importante característica da ferramenta ControlDesk é a capacidade de automação da atividade experimental por meio de uma interface de programação que permite o controle remoto deste *software*. O ControlDesk incorpora um interpretador para linguagem de programação Python [39], permitindo tanto a execução de linhas de código ou comandos em Python quanto de *scripts* Python (arquivos de extensão *.py). A automação do *software* é realizada através da programação de *scripts* Python para serem executados pelo interpretador, tornando automática a execução de determinadas tarefas, como por exemplo uma variação paramétrica

em tempo real programada para avaliar o desempenho do sistema de controle em operação. O sistema dSPACE inclui todas as bibliotecas padrões de Python e módulos específicos para a plataforma que permitem o controle do *software* (Figura 3.7)

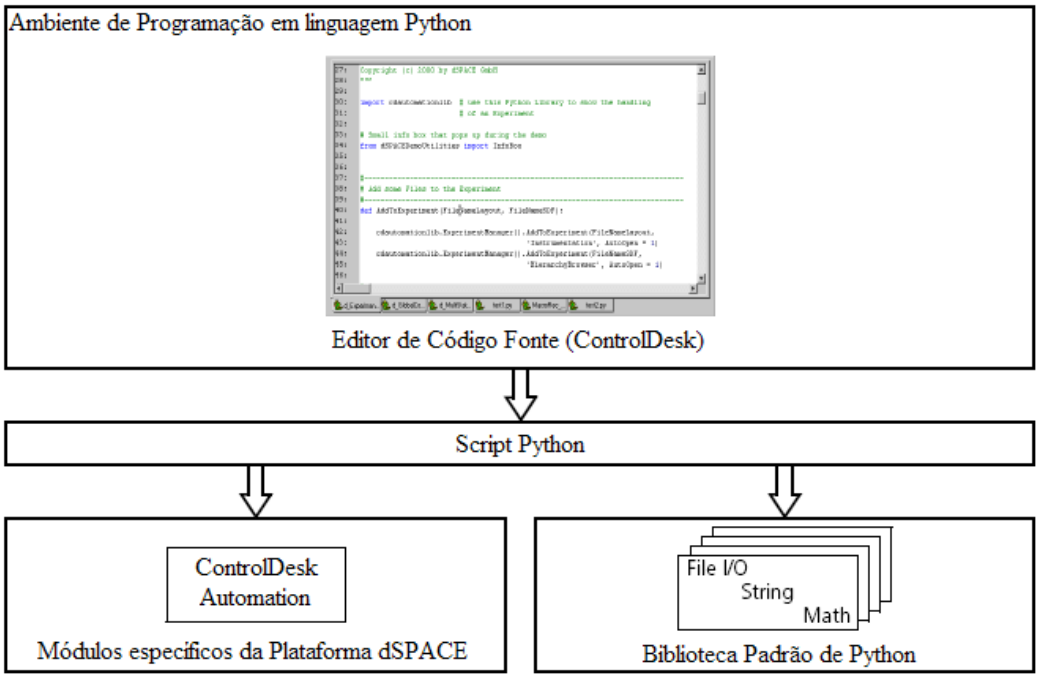


Figura 3.7: Esquema do ambiente de programação para linguagem Python na plataforma dSPACE.

As bibliotecas disponíveis na plataforma permitem o uso de vários recursos da ferramenta ControlDesk e do sistema dSPACE. Algumas capacidades são listadas a seguir:

- Controle de forma geral de características do ControlDesk, incluindo alterações de aparência da área de trabalho;
- Associação de varáveis do modelo do sistema de controle aos instrumentos virtuais;
- Controle da atividade experimental de forma geral, criando, abrindo ou salvando experimentos, adicionando ou removendo arquivos e *layout's* de instrumentação virtual ao experimento, entre outras funções;
- Manipulação do *hardware* de controle em tempo real, carregando diferentes aplicações (programas executáveis com os algoritmos controle) e listando as placas controladoras disponíveis na plataforma;
- Manipulação das diferentes "plataformas de execução" (placas controladoras ou Simulink), carregando, iniciando ou parando a execução de aplicações.

Toda esta estrutura de programação torna a plataforma extensível, isto é, com capacidade de aprimoramento pela adição de novos componentes e recursos, tanto para *software* quanto para *hardware*. Dessa forma, com todos os recursos disponíveis pela plataforma, é possível agregar novas funcionalidades à plataforma, como por exemplo conectividade sem fio.

3.3 Desenvolvimento da Estrutura Cliente-Servidor

A arquitetura abordada neste trabalho para a configuração remota de algoritmos de controle para aplicações industriais usando comunicação sem fio e geração automática de código consiste em um modelo cliente-servidor no qual são definidas duas entidades, Estação Base e Estação Remota, que se comunicam através de um padrão de tecnologia sem fio. A configuração do algoritmo de controle é realizada através de um *software* aplicativo executado em um dispositivo portátil, usando representação em diagrama de blocos e simplificando, portanto, a interação entre o usuário e o sistema de controle.

Para testar a implementação da arquitetura é necessário o desenvolvimento de toda a estrutura cliente-servidor, incluindo o desenvolvimento do *software* aplicativo cliente, responsável pela configuração remota do algoritmo de controle, e do *software* aplicativo servidor, responsável pela integração entre os módulos da Estação Base da arquitetura, isto é com a plataforma dSPACE. Daí surge a plataforma de desenvolvimento experimental, baseada na arquitetura do sistema de configuração.

O desenvolvimento de uma estrutura cliente-servidor implica na especificação do funcionamento do modelo de comunicação, definindo quais tarefas devem ser executadas no cliente e quais no servidor. Esta especificação influencia fatores como custo, robustez e segurança da estrutura como um todo, assim como flexibilidade e extensibilidade do projeto diante de uma necessidade de modificação, ou ainda interoperabilidade com outra plataforma.

Um ponto importante diz respeito ao projeto do *software* aplicativo do cliente: o quão específico ele deve ser. O uso de programas padronizados pode economizar custos de desenvolvimento, visto que não é preciso desenvolver um novo aplicativo, entretanto a estrutura deve aceitar as limitações impostas pela padronização.

No modelo clássico cliente-servidor em duas camadas, a camada cliente e a camada servidor devem distribuir entre si as funcionalidades do modelo, isto é, as responsabilidades sobre a interface com o usuário, a lógica de negócio e o armazenamento de dados. Assim, este modelo cliente-servidor pode ser baseado em três modos: cliente magro (*thin client*), cliente gordo (*thick client*) ou um modo híbrido dos dois primeiros [40]. No modo cliente gordo, a camada cliente trata da lógica de negócio e da interface com o usuário, enquanto a camada servidor trata dos dados, utilizando por exemplo um sistema gerenciador de banco de dados. Ou seja, no cliente gordo, todo o funcionamento lógico da aplicação é processado no cliente, que utiliza, por sua

vez, o servidor apenas como repositório de dados. Já no modo cliente magro, a camada cliente trata apenas da interface gráfica com o usuário, enquanto a camada servidor fica responsável pela lógica de negócio e pelo armazenamento e gerenciamento dos dados.

O modelo cliente-servidor adotado para a arquitetura do sistema de configuração é baseado no modo híbrido, ou seja, combinando características do modo cliente magro com algumas do cliente gordo, visto que parte da lógica de negócio é processada no cliente e outra parte no servidor. Além da função de interface com o usuário, a camada cliente do modelo (Estação Remota) também é responsável pelo procedimento de configuração remota, enquanto que a camada servidor (Estação Base) é responsável pela geração automática de código e implementação do algoritmo de controle no sistema de controle.

Com base na arquitetura do sistema de configuração remota, um *software* aplicativo cliente deve ser desenvolvido para ser executado por um dispositivo portátil. Para realizar a interação entre o usuário e o sistema de controle é necessário um dispositivo portátil com os seguintes recursos: tela, teclado, dispositivo apontador (*mouse* ou tela sensível ao toque) e capacidade de processamento suficiente para lidar com recursos gráficos e comunicação sem fio exigidas pela aplicação cliente.

O dispositivo N800 *Internet Tablet* da Nokia [41] é bastante adequado e satisfaz os requisistos necessários. Este aparelho é um "computador multimídia" utilizando sistema operacional Linux, com excelente desempenho, praticidade e mobilidade. Ele possui conectividade Bluetooth, Wi-Fi, USB, tela sensível ao toque (*touch screen*) e boa capacidade de armazenamento de dados (compatibilidade com cartões de memória até 8 GB). Há uma preferência por este dispositivo visto que modelos deste aparelho encontram-se disponíveis no Laboratório de Sistemas Embarcados e Computação Pervasiva (EMBEDDED-DEE-UFCG) para desenvolvimento de aplicações.

Como ainda não existe um aplicativo padrão que funcione como uma ferramenta de configuração adequada aos requisitos da arquitetura proposta e que seja compatível com o dispositivo portátil N800 utilizado pela plataforma experimental, um *software* aplicativo cliente, denominado WRC-Client (do inglês, *Wireless Remote Configurator - Client*) foi desenvolvido para satisfazer todos os requisitos da aplicação cliente da arquitetura. A implementação da camada servidor da arquitetura consiste no desenvolvimento de um *software* aplicativo, denominado WRC-Server (do inglês, *Wireless Remote Configurator - Server*), para prover o serviço de configuração remota, ou seja, para anunciar tal serviço, receber, processar e responder aos comandos do aplicativo cliente. O WRC-Server também é responsável pela interação com as ferramentas referentes aos módulos Geração Automática de Código (MATLAB/RTW) e Interface com o Usuário (ControlDesk) da arquitetura.

O conjunto dispositivo portátil mais o aplicativo cliente WRC-Client constitui o configurador remoto da plataforma experimental. De acordo com a arquitetura do sistema, o configurador

remoto pode conectar-se com outras plataformas que implementem a mesma arquitetura, ou seja, um configurador remoto é suficiente para configurar mais de um sistema de controle, entretanto, um de cada vez.

3.3.1 Aplicação Cliente

A aplicação cliente corresponde ao *software* aplicativo executado pelo dipositivo portátil da plataforma experimental, constituindo a Estação Remota da arquitetura do sistema de configuração, ou seja, o configurador remoto da plataforma.

A metodologia de desenvolvimento utilizada para a construção da aplicação cliente consiste na análise e projeto orientado a objetos do *software* aplicativo. A linguagem UML (do inglês *Unified Modeling Language*) [42] é utilizada para criar modelos de análise e de projeto do *software*, com o intuito de modelar o problema e a solução para a criação do aplicativo.

As fases de um processo de desenvolvimento de *software* envolvem de modo geral: planejamento e elaboração (definição de requisitos e especificação), construção do sistema (codificação e testes) e implantação (sistema em funcionamento).

Na fase de planejamento e elaboração, as principais etapas são o levantamento dos requisitos funcionais e não funcionais do sistema, elaboração de um modelo conceitual do funcionamento, definição da especificação e projeto da arquitetura do *software*.

A linguagem UML é bastante útil para visualização, especificação, construção e documentação de artefatos de um *software* em desenvolvimento, visando prover uma representação parcial do sistema, incluindo diversos tipos de diagramas. Um dos tipos de diagrama UML é o de Caso de Uso (*Use Case*), que descreve um cenário mostrando as funcionalidades do sistema do ponto de vista do usuário. Um diagrama de Caso de Uso referente ao aplicativo cliente é apresentado na Figura 3.8.

O diagrama de Caso de Uso é utilizado para ajudar na definição dos requisitos de operação do aplicativo. Os requisitos funcionais do aplicativo cliente são listados a seguir:

- Interface com o usuário: apresentar o diagrama de blocos referente ao algoritmo de controle implementado pelo sistema de controle, e uma lista de blocos disponíveis para configuração do diagrama;
- Operações disponíveis sobre o modelo em diagrama de blocos: adição e remoção de blocos, conexões e ramificações de conexões, alteração de parâmetros de configuração dos blocos, salvar ou abrir um modelo e criar um novo modelo;
- Operações disponíveis para interação com o sistema de controle: receber e enviar modelos do algoritmo de controle, iniciar e parar o funcionamento do sistema de controle;
- Comunicação sem fio: conectar-se e desconectar-se do servidor;

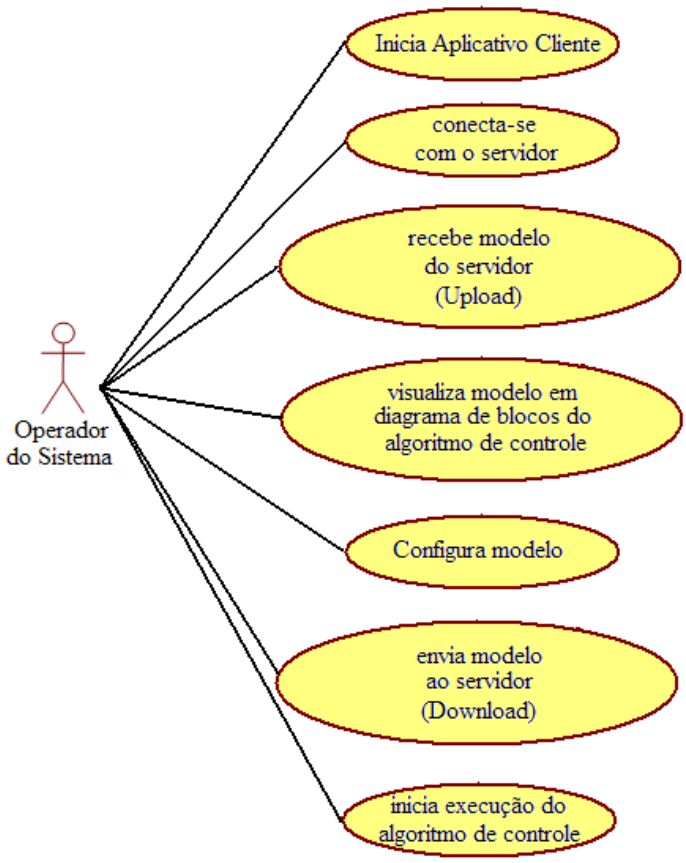


Figura 3.8: Diagrama de Caso de Uso para o *software* aplicativo cliente.

O aplicativo cliente possui os seguintes requisitos não funcionais:

- Usuário: operador ou supervisor do sistema de controle sob configuração, com conhecimento adequado para configuração do algoritmo de controle;
- Equipamento: dispositivo portátil provido de tela, teclado, dispositivo apontador (*mouse* ou tela sensível ao toque), conectividade sem fio e com capacidade de processamento (sistema operacional) suficiente para executar aplicativos e bibliotecas de *software* necessárias para a implementação dos requisitos funcionais;
- Compatibilidade com outros aplicativos: compatível com o *software* MATLAB/Simulink, de modo que seja possível interpretar (ler e escrever) arquivos MDL (de extensão *.mdl) da referida ferramenta, que contêm o modelo do algoritmo de controle da aplicação;
- Robustez: garantir confiabilidade no processo conexão com o servidor e na transmissão dos dados referentes ao modelo do algoritmo de controle;
- Portabilidade: compatibilidade multi-plataforma (Linux, Windows, Mac OS);
- Facilidade de uso: configuração do algoritmo de controle em uma abstração de alto nível (diagrama de blocos);

- Flexibilidade: capacidade de adaptação de funcionalidades para satisfazer novos requisitos rapidamente;
- Extensibilidade: capacidade de adição de novas funcionalidades.

O modelo conceitual envolve a elaboração das idéias e conceitos básicos que determinam os elementos fundamentais do *software* em questão. Ele exerce influência sobre a interface com o usuário e a arquitetura do *software*. Ele envolve a elaboração da maneira como o usuário pode interagir para realizar suas tarefas, a escolha dos objetos de interfaces (botões, menus, caixas de texto, etc.), o *layout* de janelas e telas, etc. A interface deve garantir boa usabilidade do *software*, fator fundamental para o êxito do aplicativo. O modelo conceitual da aplicação cliente é apresentado na Figura 3.9, tendo como base o funcionamento do *software* Simulink.

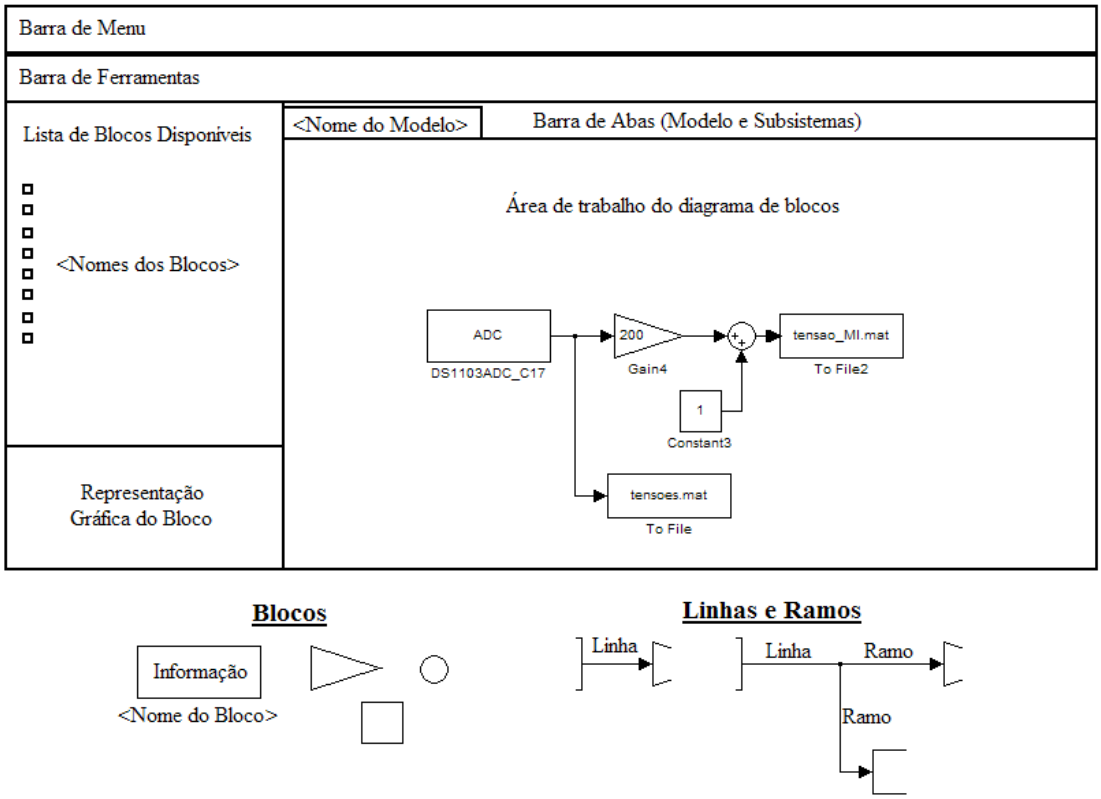


Figura 3.9: Modelo conceitual do aplicativo cliente.

A partir do modelo conceitual e do diagrama de Caso de Uso é possível definir as especificações de funcionamento do *software*. O usuário da aplicação cliente (operador do sistema de controle) utiliza o configurador remoto para alterar o algoritmo de controle do sistema. Primeiramente, a aplicação cliente deve conectar-se ao servidor, o qual anuncia o serviço de configuração remota. Implicitamente, o aplicativo cliente deve realizar uma busca pelo serviço de configuração para, em seguida, proceder com o estabelecimento da conexão. A conexão será estabelecida desde que o servidor não esteja conectado à um cliente. Esta condição é imposta

visto que se trata da alteração do algoritmo de controle do sistema e, portanto, deve-se evitar uma possível inconsistência de dados que acarretaria um funcionamento inadequado do sistema de controle. O usuário deve, então, realizar uma operação de *Upload*, solicitando ao servidor o envio do modelo atual do algoritmo de controle presente na Estação Base. Ao receber o modelo, o configurador remoto apresenta na tela o diagrama de blocos correspondente. Neste momento, o usuário dispõem de operações para adicionar ou remover blocos, conexões e ramificações de conexões (linhas e ramos), alterar parâmetros de configuração dos blocos, e para salvar, abrir ou criar um novo modelo. Após a configuração do modelo, a nova versão do algoritmo deve ser enviada para a Estação Base através de uma operação de *Download*. Em seguida, o usuário pode enviar comandos ao servidor para executar ou parar a execução do algoritmo de controle contido na plataforma experimental, além de poder solicitar a desconexão com o servidor.

A arquitetura de *software* deve fornecer uma visão macroscópica do aplicativo em termos de componentes que interagem entre si. O projeto da arquitetura do *software* deve levar em consideração todos os artefatos definidos e informações obtidas para o planejamento do aplicativo. Um importante aspecto a ser considerado corresponde à compatibilidade com o *software* Simulink. O aplicativo cliente deve ser capaz de interpretar (ler e escrever) arquivos MDL, de modo a apresentar ao usuário o diagrama de blocos equivalente ao modelo do algoritmo de controle implementado no sistema de controle. Um arquivo modelo do Simulink (MDL), é um arquivo de texto estruturado hierarquicamente contendo pares de palavras-chave e valores paramétricos que descrevem os componentes do modelo em diagrama de blocos. A estrutura é apresentada na Figura 3.10.

Em [43] são descritos de forma clara e objetiva o funcionamento e a estrutura da ferramenta Simulink bem como os detalhes de criação de modelos, tornando-se referência fundamental para o desenvolvimento do aplicativo cliente de maneira compatível com o Simulink.

Outro aspecto importante diz respeito à transmissão de dados entre as aplicações cliente e servidor. O padrão XML [44] (do inglês, *Extensible Markup Language*) é utilizado para operações envolvendo o envio do modelo do algoritmo de controle (arquivos MDL). O intuito é agregar confiabilidade aos dados transmitidos. O padrão XML é uma excelente alternativa para a formatação e criação de arquivos com dados organizados de forma hierárquica. A idéia é utilizar a característica de portabilidade dessa linguagem, de forma que uma aplicação possa escrever um arquivo XML e uma outra possa ler os mesmos dados. No caso de um arquivo ser corrompido durante a transmissão dos dados, a aplicação que recebeu o arquivo reconhecerá a invalidez do mesmo. Com isso, a aplicação pode definir um tratamento adequado para tal situação, por exemplo, solicitar a retransmissão dos dados.

Com base nos artefatos obtidos e na estrutura básica de um arquivo MDL, foram identificadas e definidas as principais entidades componentes da arquitetura do *software* aplicativo cliente, conforme apresentada na Figura 3.11.

```
Model {
  <Model Parameter Name> <Model Parameter Value>
  ...
  BlockDefaults {
    <Block Parameter Name> <Block Parameter Value>
    ...
  }
  BlockParameterDefaults {
    Block {
      <Block Parameter Name> <Block Parameter Value>
      ...
    }
    ...
  }
  LineDefaults {
    <Line Parameter Name> <Line Parameter Value>
    ...
  }
  AnnotationDefaults {
    <Annotation Parameter Name> <Annotation Parameter Value>
    ...
  }
  System {
    <System Parameter Name> <System Parameter Value>
    ...
    Block {
      <Block Parameter Name> <Block Parameter Value>
      ...
    }
    Line {
      <Line Parameter Name> <Line Parameter Value>
      ...
      Branch {
        <Branch Parameter Name> <Branch Parameter Value>
        ...
      }
    }
  }
}
```

Figura 3.10: Estrutura de um arquivo Simulink (MDL).

A fase de construção do sistema envolve as etapas de codificação e teste do aplicativo. Nesta fase, cabe ao desenvolvedor dominar as características da linguagem de programação, ferramentas, *frameworks* e estruturas de dados para implementar as funcionalidades e satisfazer os requisitos identificados. Esta fase também envolve os testes de unidade, feitos durante o ciclo de desenvolvimento, para verificar se os componentes gerados atendem à especificação do projeto.

A linguagem de programação utilizada para codificação do aplicativo cliente foi Python e o ambiente de desenvolvimento foi o *software* Eclipse [45]. Python permite tanto programação estruturada quanto orientada a objetos, utilizando-se para isso o conceito de classe de objetos entre outros recursos da linguagem. Trata-se de uma linguagem multi-plataforma (Unix/Linux, Windows, Mac OS), interpretada, de sintaxe simples e clara. A programação em Python é

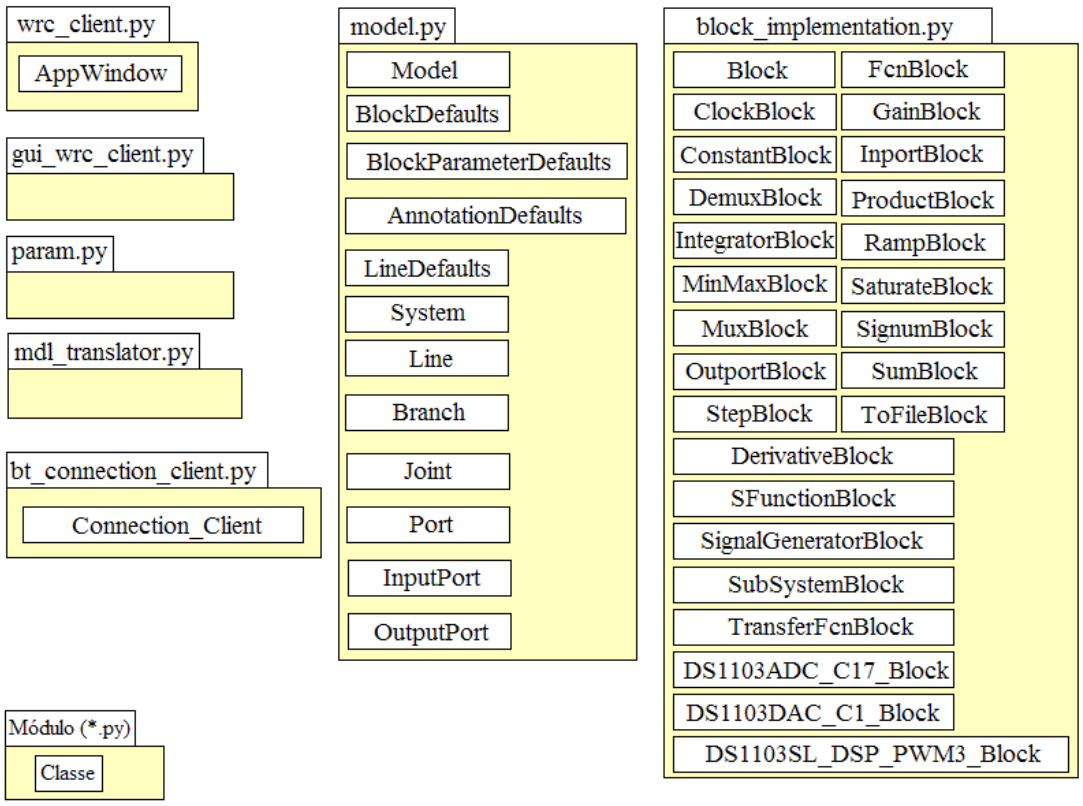


Figura 3.11: Entidades componentes da arquitetura do *software* aplicativo cliente.

realizada através de arquivos de texto com extensão *.py denominados de módulos ou *scripts*. Na programação orientada a objetos é possível definir mais de uma classe dentro de um mesmo módulo, diferentemente de outras linguagens de programação.

No desenvolvimento do aplicativo foram utilizadas bibliotecas para o desenvolvimento da interface com o usuário (PyGTK) e da comunicação sem fio baseada na tecnologia Bluetooth (PyBlueZ). Vale salientar que o padrão XML também está disponível através de bibliotecas para a linguagem Python, disponibilizando recursos úteis e facilitadores para implementação do padrão. A biblioteca PyBlueZ foi desenvolvida por Albert Huang em seu trabalho de mestrado [46]. Esta biblioteca torna mais fácil e rápido o desenvolvimento de aplicações utilizando Bluetooth, permitindo acesso aos recursos da pilha de protocolos Bluetooth utilizando a linguagem Python. PyBlueZ é compatível com as plataformas GNU/Linux e Windows XP, sendo gratuitamente distribuída sob a licença GPL (*General Public License*).

Na Figura 3.11 são apresentados sete módulos Python. O aplicativo cliente é executado a partir do módulo **wrc_client.py**, que possui uma única classe chamada **AppWindow**. Esta classe compreende a lógica de funcionamento para interação com o usuário, isto é, a implementação das funcionalidades relacionadas com a interface gráfica com o usuário. No módulo **gui_wrc_client.py** estão implementadas as funções para inicializar os recursos da interface gráfica. O módulo **param.py** é responsável por apresentar a janela de configuração

de parâmetros de cada bloco disponível pelo aplicativo. O módulo que implementa os blocos é o **block_implementation.py**, compreendendo 26 classes. O *software* aplicativo cliente deve ter a capacidade de interpretar o arquivo XML para obter o arquivo MDL, bem como criar um arquivo XML a partir de um arquivo MDL. O módulo **mdl_translator.py** implementa funções relacionadas a transformação de arquivos MDL em XML e vice-versa, e funções para criação de um arquivo MDL a partir do modelo apresentado na interface com o usuário e geração do o modelo em diagrama de blocos para a interface gráfica a partir de arquivos XML. O módulo **bt_connection_client.py** possui uma classe, **Connection_Client**, responsável pela implementação da comunicação Bluetooth. Por fim, o módulo **model.py** engloba parte considerável da lógica da aplicação baseada principalmente na estrutura de um arquivo Simulink (Figura 3.10).

Diagramas UML são mais adequados para representar a estrutura do código desenvolvido. A seguir, na Figura 3.12 é apresentado o diagrama de classes do *software* aplicativo cliente, destacando-se os relacionamentos entre as entidades componentes.

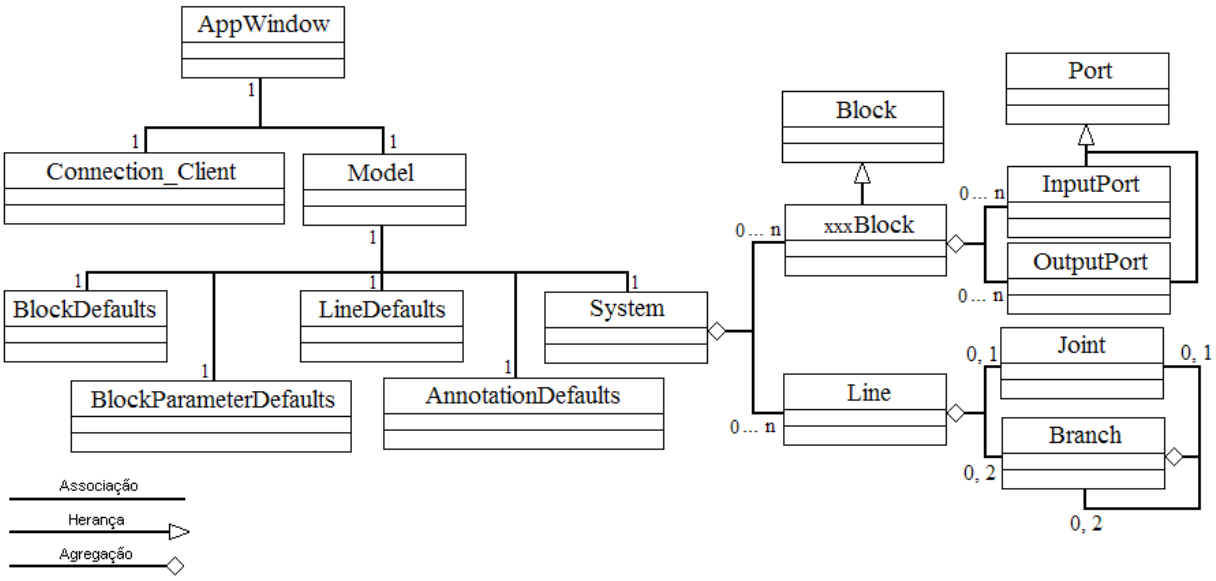


Figura 3.12: Diagrama de classes do *software* aplicativo cliente.

A relação de um para um entre as classes **AppWindow** e **Model** deve-se ao fato que o aplicativo cliente trata com apenas um modelo por vez. A comunicação Bluetooth é gerenciada pela classe **Connection_Client**, sendo necessário apenas uma instância desta classe para realizar tal função. Portanto, uma relação de um para um é estabelecida entre esta classe e a **AppWindow**.

De acordo com a estrutura de um arquivo MDL, a classe **Model** possui uma relação de um para um com as classes **BlockDefaults**, **BlockParameterDefaults**, **LineDefaults** e **AnnotationDefaults**, que encapsulam os parâmetros de configuração padrão do modelo relativos aos blocos, linhas e anotações do modelo, e com a classe **System**, visto que um modelo do

Simulink possui apenas um sistema [43], e, portanto a classe **Model** possui uma única instância de **System**. A classe **System** pode incluir instâncias das classes **xxxBlock** e **Line**, onde **xxx** refere-se ao tipo do bloco (Figura 3.11), por exemplo **ConstantBlock** que representa uma constante numérica.

Todos os blocos implementados são subclasses da classe **Block**. Um bloco pode ser composto por portas de entrada e/ou de saída, representadas pelas classes **InputPort** e **OutputPort**, respectivamente, e que são subclasses de **Port**. Quanto às conexões entre os blocos do diagrama, uma linha pode ser composta por uma junta (classe **Joint**) e dois ramos (classe **Branch**), e da mesma forma um ramo também pode ser composto por uma junta e dois ramos (Figura 3.9).

As principais operações realizadas pelo aplicativo são detalhadas através de diagramas de sequência que ilustram o fluxo de execução do programa (figuras 3.13, 3.14, 3.15 e 3.16).

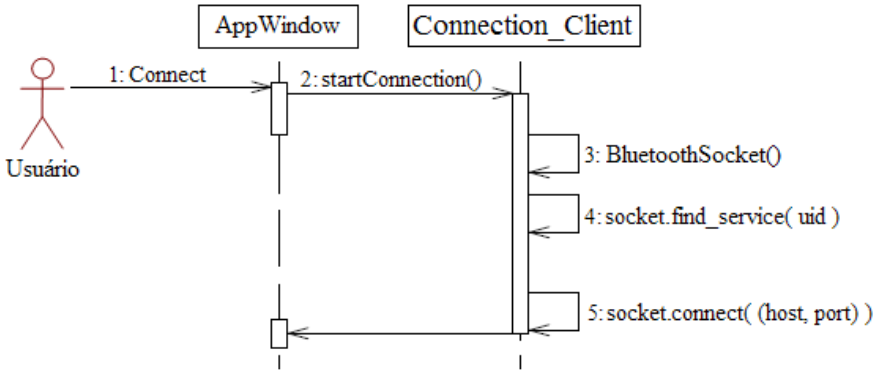


Figura 3.13: Diagrama de sequência referente ao pedido de conexão com o servidor.

Na Figura 3.13 é apresentado o fluxo de execução durante uma operação de conexão com o servidor. O aplicativo utiliza recursos fornecidos pela biblioteca PyBlueZ para implementar a comunicação Bluetooth, fazendo uso de funções para a descoberta de serviços e conexão com o servidor.

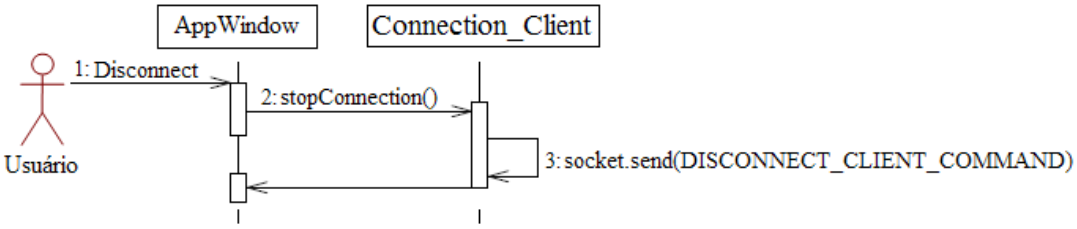


Figura 3.14: Diagrama de sequência referente ao pedido de desconexão com o servidor.

Na Figura 3.14 é apresentado o fluxo de execução referente à uma operação de desconexão com o servidor, baseado no envio de um comando que solicita a desconexão do cliente (**DISCONNECT_ CLIENT_COMMAND**).

Na Figura 3.15 é ilustrado o processamento decorrente de uma operação de *Upload* solicitada pelo usuário. Após o envio do respectivo comando (**UPLOAD_COMMAND**), o aplicativo

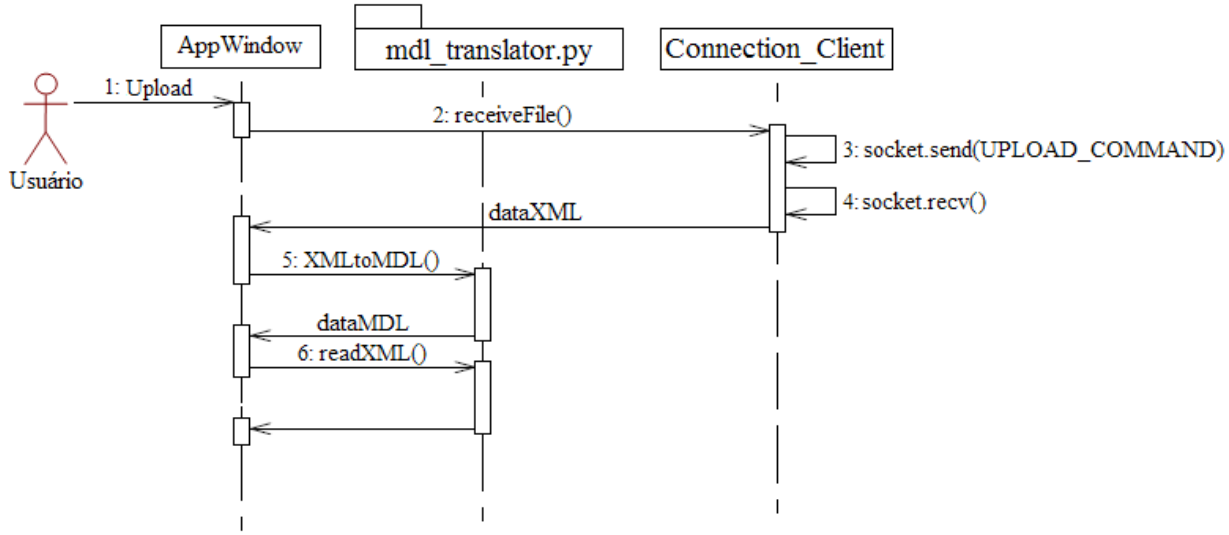


Figura 3.15: Diagrama de sequência referente ao comando *Upload*.

aguarda pelo recebimento do arquivo XML (**dataXML**) contendo o modelo do algoritmo de controle, i.e. encapsulando o arquivo MDL. De posse do arquivo XML, duas operações são sucedidas: uma para obter o arquivo MDL a partir do XML recebido (função **XMLtoMDL()**), e outra para interpretar o arquivo XML recebido e apresentar o diagrama de blocos correspondente na interface gráfica com o usuário.

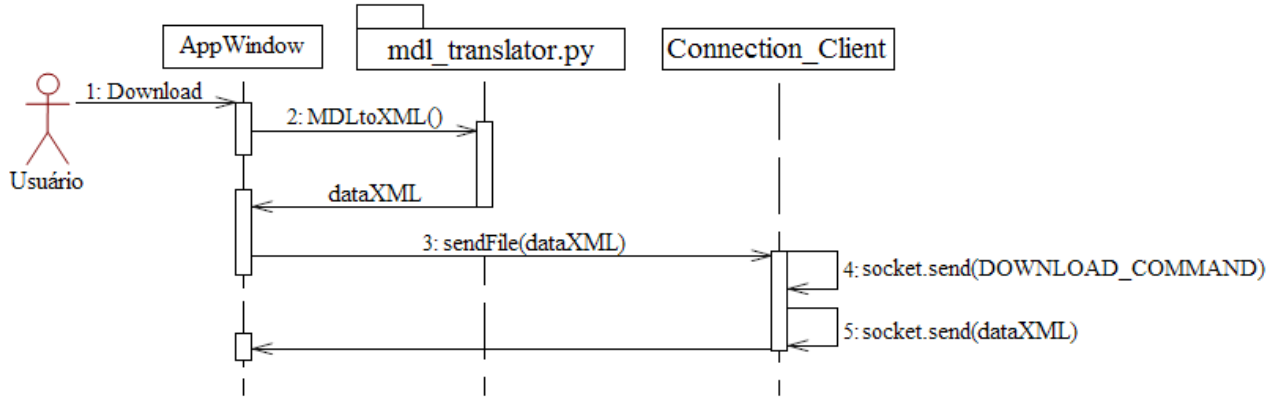


Figura 3.16: Diagrama de sequência referente ao comando *Download*.

Na Figura 3.16 é apresentado o fluxo de execução referente ao comando de *Download* para envio do modelo configurado para a Estação Base. Primeiramente, é criado um arquivo XML correspondente ao modelo configurado, apresentado na interface gráfica (função **MDLtoXML()**). Em seguida, o comando **DOWNLOAD_COMMAND** é enviado ao servidor para informar que uma operação de atualização do algoritmo de controle será efetuada. Então, o arquivo XML gerado é enviado ao servidor via Bluetooth.

A interface gráfica do *software* aplicativo cliente (WRC-Client) desenvolvido, sendo executado no dispositvo portátil N800, é apresentada na Figura 3.17.

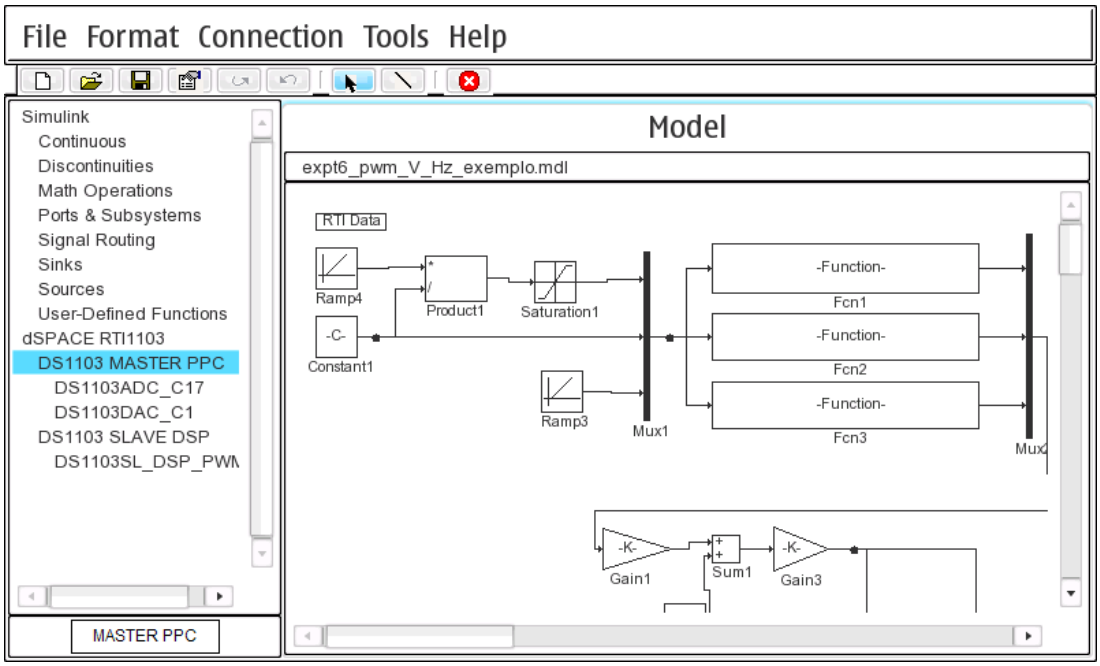


Figura 3.17: Vista da interface gráfica do *software* aplicativo cliente (WRC-Client).

O aplicativo cliente disponibiliza suas funções por meio de um *menu* de opções ou da barra de ferramentas. Para adicionar um bloco ao modelo basta arrastar o respectivo nome do bloco da lista de blocos e soltar na área de trabalho do diagrama. A remoção é realizada selecionando-se o bloco desejado e pressionando-se o botão **X** da barra de ferramentas. Nesta barra também estão presentes dois botões que indicam o modo de funcionamento do aplicativo: modo de seleção (cujo símbolo é uma seta) no qual o usuário pode selecionar os blocos e movê-los na área de trabalho, e o modo de conexão (cujo símbolo é uma contra-barra) no qual o usuário pode conectar os blocos criando linhas e ramos.

A fase de implantação do sistema corresponde à integração do aplicativo cliente com o aplicativo servidor e demais recursos, para constituir a plataforma experimental.

3.3.2 Aplicação Servidor

A aplicação servidor corresponde ao *software* aplicativo executado pelo PC da plataforma experimental, constituindo parte da Estação Base da arquitetura proposta.

A metodologia de desenvolvimento utilizada para a construção da aplicação servidor é semelhante à da aplicação cliente, consistindo na análise e projeto orientado a objetos do *software* aplicativo. A linguagem UML também é utilizada para criar os modelos de análise e de projeto do *software* servidor. Na Figura 3.18 é ilustrado o diagrama de Caso de Uso referente ao aplicativo servidor.

Os requisitos funcionais do aplicativo servidor foram identificados e são listados a seguir:

- Interface com o usuário: interface para iniciar e parar a execução do servidor;

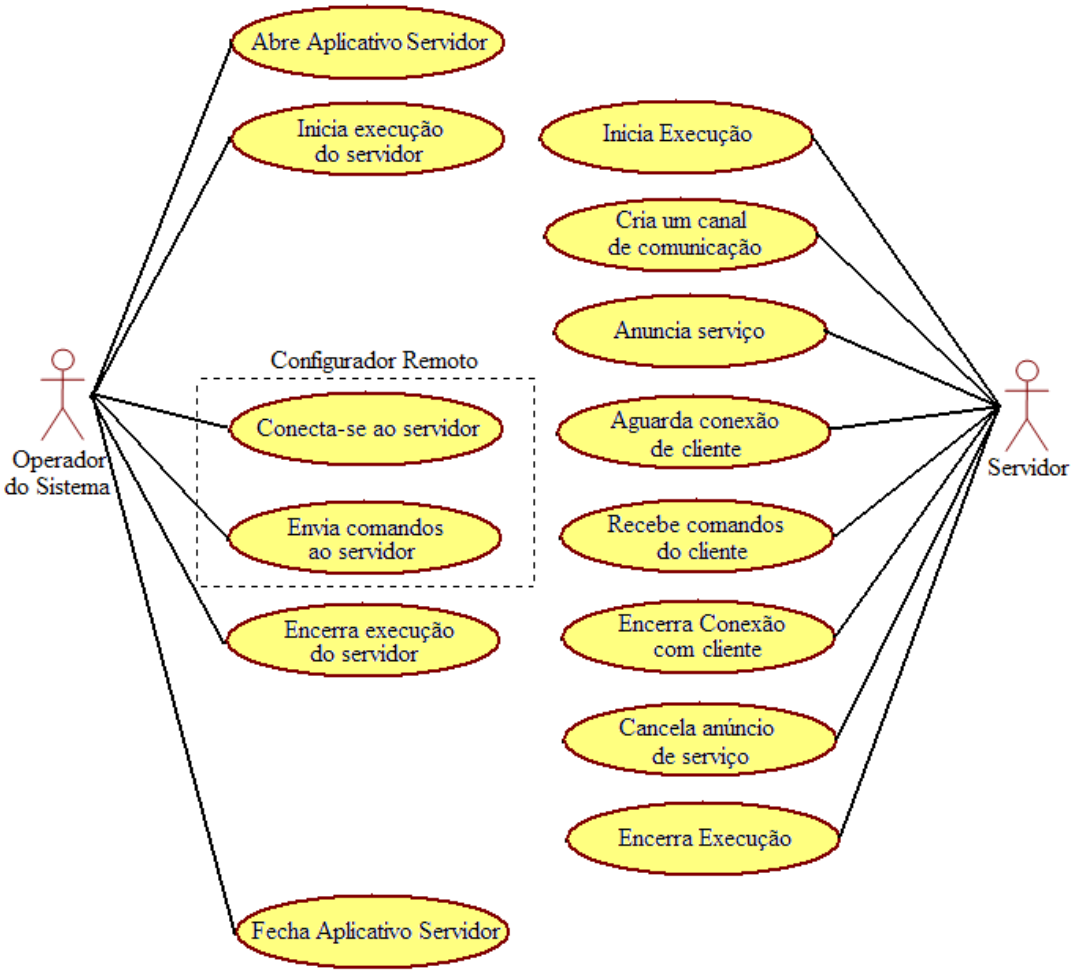


Figura 3.18: Diagrama de Caso de Uso para o *software* aplicativo servidor.

- Interação com o configurador remoto: interpretar comandos enviados pelo configurador referentes às funções de receber e enviar modelo do algoritmo de controle, iniciar e parar o funcionamento do sistema de controle, e conectar e desconectar do servidor;
- Comunicação sem fio: iniciar e cancelar o anúncio do serviço de configuração remota;
- Integração com módulos da Estação Base: interagir com o módulo Geração Automática de Código para solicitar a geração do código executável, e com o módulo Interface com o Usuário para iniciar e parar o funcionamento do sistema de controle, de acordo com o comando recebido do configurador remoto.

Como requisitos não funcionais, são identificadas as seguintes características:

- Usuário: operador ou supervisor do sistema de controle sob configuração, com conhecimento adequado para iniciar e parar a execução do servidor;
- Equipamento: sistema computacional com conectividade sem fio e com capacidade de

processamento (sistema operacional) suficiente para executar ferramentas e bibliotecas de *software* necessárias para a implementação dos requisitos funcionais;

- Robustez: garantir confiabilidade no processo de conexão com o cliente e na transmissão dos dados referentes ao modelo do algoritmo de controle;
- Portabilidade: compatibilidade multi-plataforma (Linux, Windows, Mac OS);
- Flexibilidade: capacidade de adaptação de funcionalidades para satisfazer novos requisitos rapidamente;
- Extensibilidade: capacidade de adição de novas funcionalidades.

O modelo conceitual da aplicação servidor é apresentado na Figura 3.19.

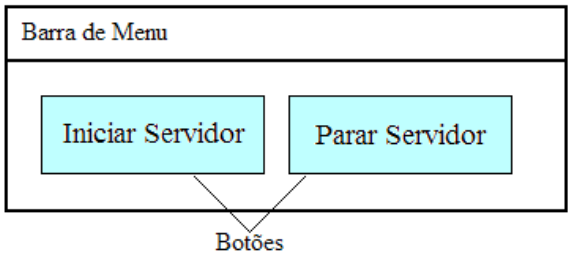


Figura 3.19: Modelo conceitual do aplicativo servidor.

A especificação de funcionamento do *software* é definida a partir do modelo conceitual elaborado. O operador do sistema de controle deve, a partir do aplicativo, iniciar a execução do servidor pressionando o botão Iniciar Servidor. O *software*, por sua vez, cria um *socket* (que pode ser entendido como uma porta para um canal de comunicação), anuncia o serviço de configuração remota e aguarda pela conexão de um cliente. O servidor deve conectar-se com apenas um cliente por vez, visto que se trata da alteração do algoritmo de controle do sistema e, portanto, deve-se evitar uma possível inconsistência de dados. Após a conexão de um cliente, o servidor está apto a receber comandos envolvendo operações de *Upload*, na qual o cliente solicita o envio do modelo atual do algoritmo de controle, *Download*, na qual o cliente envia o modelo do algoritmo de controle modificado, e operações para iniciar ou parar a execução do sistema de controle, além do comando de desconexão.

Entretanto, no atual estágio de implementação do aplicativo, as operações referentes aos comandos para iniciar e parar a execução do sistema de controle ainda não estão integradas por completo com o *software* ControlDesk da plataforma experimental, ou seja, estas funções não estão disponíveis ao usuário no aplicativo cliente. Apesar da execução do algoritmo de controle iniciar imediatamente após a geração automática de código decorrente de um comando de *Download*, para interromper a execução é necessário utilizar o respectivo comando manualmente através do ControlDesk.

Com base nos artefatos obtidos, as principais entidades componentes da arquitetura do *software* aplicativo servidor são apresentadas na Figura 3.20, e o diagrama de classes na Figura 3.21.

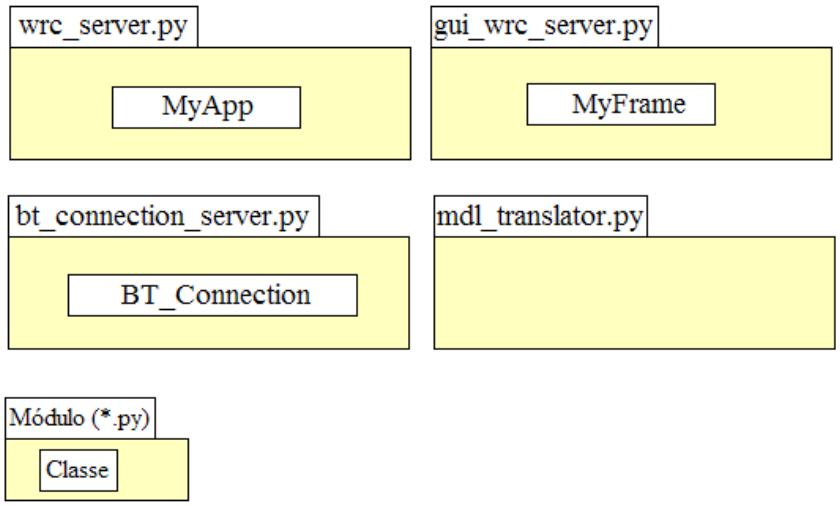


Figura 3.20: Entidades componentes da arquitetura do *software* aplicativo servidor.

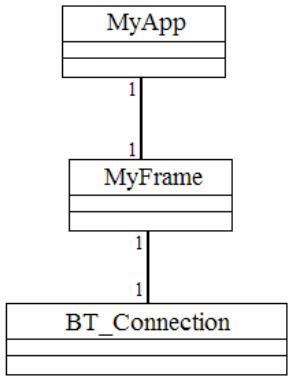


Figura 3.21: Diagrama de classes do *software* aplicativo servidor.

O *software* aplicativo servidor envolve quatro módulos Python. O módulo responsável pelo início da execução do aplicativo é o **wrc_server.py**. Este módulo possui uma única classe, **MyApp**, utilizada para incializar o aplicativo, que faz uso do módulo **gui_wrc_server.py** para a construção da interface gráfica com o usuário. O módulo **gui_wrc_server.py** possui a classe **MyFrame** que gerencia o funcionamento da interface e inicializa o servidor através do módulo **bt_connection_server.py**, ou seja, da classe **BT_Connection** que estabelece a comunicação com o cliente via Bluetooth. A classe **BT_Connection** também utiliza funções do módulo **mdl_translator.py** para interpretar arquivos XML e obter arquivos MDL, e vice-versa.

Assim como o aplicativo cliente, o servidor possui um módulo **mdl_translator.py** para realizar as funções de leitura e escrita de arquivos XML. Após receber o arquivo XML contendo

a informação do modelo do algoritmo de controle, o aplicativo deve obter o arquivo MDL e armazená-lo no sistema de arquivos da plataforma de modo que a ferramenta de *software* responsável pela geração automática de código (RTW) possa acessar o modelo, e gerar o novo código executável para o *hardware* de controle da plataforma.

As principais operações realizadas pelo aplicativo são detalhadas através de diagramas de sequência que ilustram o fluxo de execução do programa (figuras 3.22, 3.23, 3.24 e 3.25).

Na Figura 3.22 é apresentado o fluxo de execução do aplicativo servidor quando uma operação de inicialização é requisitada pelo operador (usuário). Esta operação inicia o servidor, cria um *socket*, inicia o anúncio do serviço de configuração remota, e aguarda pela conexão de um cliente. Após a conexão de um cliente, o servidor está apto a receber comandos do cliente e realizar as tarefas correspondentes.

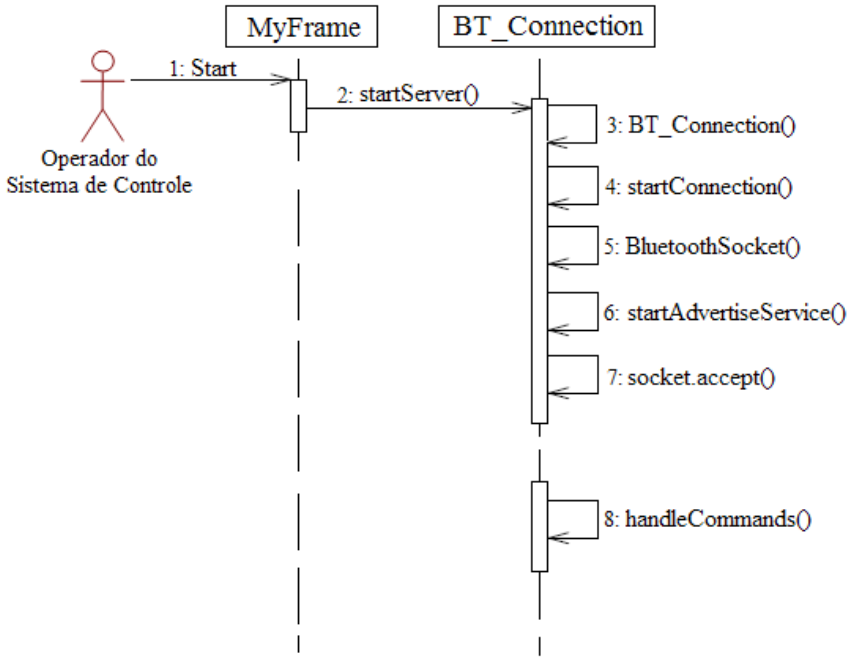


Figura 3.22: Diagrama de sequência referente ao comando para iniciar o servidor.

Na Figura 3.23 é ilustrado o processamento de um comando para encerrar o funcionamento do servidor. Primeiro, é cancelado o anúncio do serviço de configuração remota, e em seguida o *socket* para comunicação com o cliente é fechado.

Na Figura 3.24, o fluxo de execução referente ao comando de *Upload* enviado pelo cliente é detalhado. Após receber o comando de *Upload*, o servidor solicita a geração do arquivo XML a partir do arquivo MDL contido no sistema de arquivos da Estação Base que corresponde ao algoritmo de controle implementado na plataforma. Então, o arquivo XML é enviado para o cliente, e o servidor volta a aguardar por novos comandos.

Na Figura 3.25 é apresentado o processamento realizado pelo servidor ao receber um comando de *Download* do cliente. Inicialmente, o servidor prepara-se para receber o arquivo

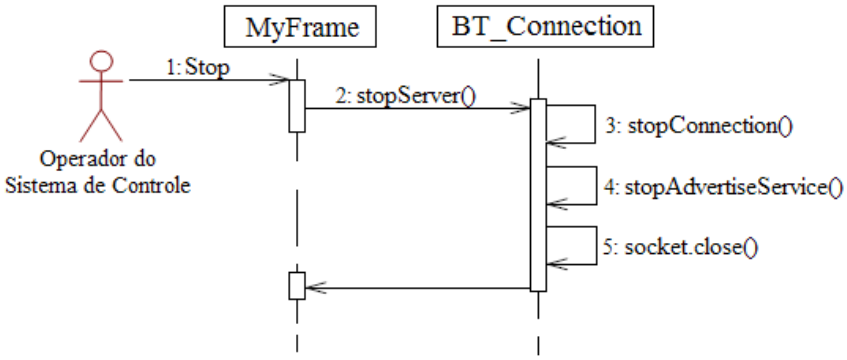


Figura 3.23: Diagrama de sequência referente ao comando para encerrar o servidor.

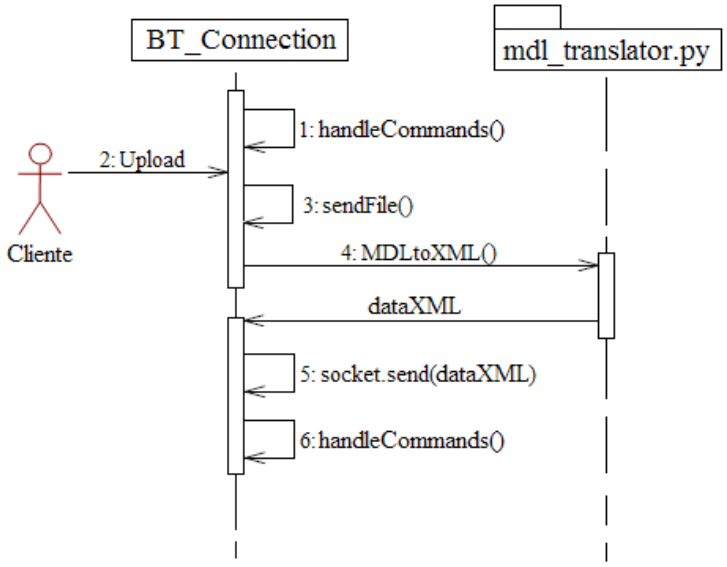


Figura 3.24: Diagrama de sequência referente ao processamento do comando *Upload* enviado pelo cliente.

XML que deve ser enviado pelo cliente. Ao receber o arquivo, o servidor utiliza a função **XMLtoMDL()** do módulo **mdl_translator.py** para obter o arquivo MDL que contém o novo algoritmo de controle que deve ser implementado na plataforma. De posse do arquivo MDL, a geração automática de código é solicitada e o servidor volta a aguardar por comandos do cliente.

Durante o processamento de um comando de *Download* enviado pelo cliente, o aplicativo servidor invoca o *software* MATLAB e executa um comando da ferramenta RTW (**rtwbuild model**) para iniciar o processo de geração automática de código. Todo o processamento ocorre em *background* na Estação Base. Junto com o comando é informado o nome do arquivo MDL que o RTW deve utilizar para geração do código executável. O aplicativo servidor faz referência a um único arquivo MDL de nome **model.mdl**, que é sempre acessado em um operação de *Upload* e sobrescrito após uma operação de *Download*. Dessa forma, apenas um arquivo MDL é armazenado no sistema de arquivos da Estação Base para ser manipulado nas operações com

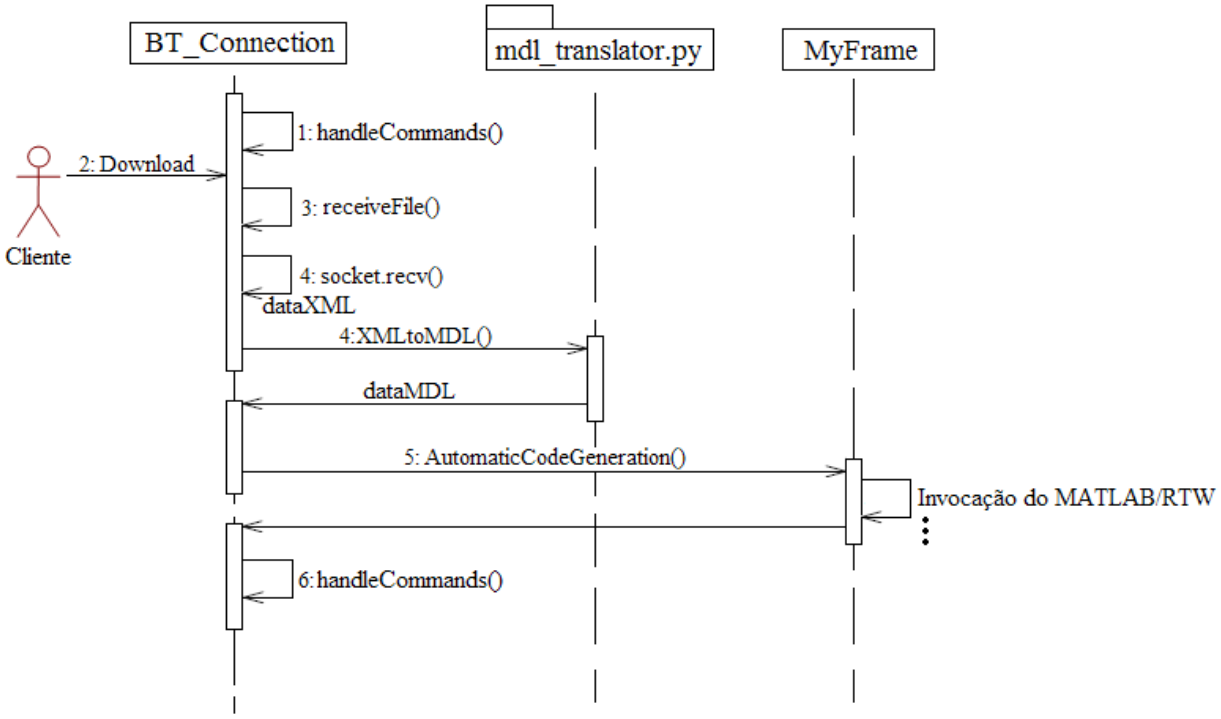


Figura 3.25: Diagrama de sequência referente ao processamento do comando *Download* enviado pelo cliente.

a Estação Remota, com o intuito de assegurar que apenas um modelo de algoritmo de controle está armazenado no sistema de controle.

A linguagem de programação Python e o *software* Eclipse também foram utilizados para o desenvolvimento do aplicativo servidor. Foram utilizadas as bibliotecas WxPython para o desenvolvimento da interface com o usuário, e PyBlueZ para comunicação sem fio usando tecnologia Bluetooth. A biblioteca WxPython, assim como PyGTK, fornece um conjunto de recursos para desenvolvimento de interface gráfica de modo fácil e rápido. Esta biblioteca foi adotada em virtude da sua disponibilidade pela plataforma dSPACE, mais precisamente a ferramenta ControlDesk, que inclui um interpretador para a linguagem Python e utiliza WxPython como biblioteca para recursos gráficos.

A interface gráfica do *software* aplicativo servidor (WRC-Server) desenvolvido, sendo executado no PC da plataforma experimental, é apresentada na Figura 3.26.

Figura 3.26: Interface gráfica do *software* aplicativo servidor (WRC-Server).

3.3.3 Integração com a Plataforma dSPACE

A plataforma experimental consiste na integração de todos os elementos que constituem a arquitetura do sistema de configuração remota, tanto recursos de *hardware* quanto de *software*. A plataforma experimental utiliza um configurador remoto (dispositivo portátil executando o aplicativo WRC-Client) que se comunica com um PC equipado com uma placa controladora DS1103 e um pacote de *software* que permite a geração automática de código e a comunicação com o configurador (WRC-Server). A comunicação utiliza o padrão Bluetooth de tecnologia sem fio, tornando possível um procedimento remoto de configuração. Um painel de conectores é utilizado para facilitar as conexões com a planta industrial a ser controlada.

Na Figura 3.27 é ilustrada a estrutura da plataforma experimental apresentando os elementos envolvidos em sua implantação.

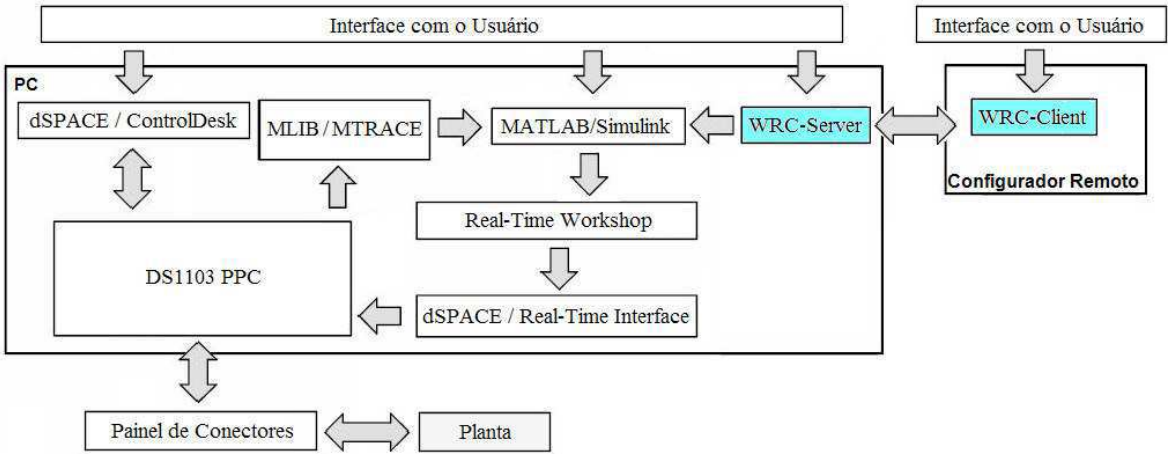


Figura 3.27: Estrutura da plataforma experimental.

De acordo com a arquitetura abordada e relacionando a Figura 3.27 com as estações Base e Remota, pode-se observar que os módulos Interface com o Usuário, Comunicação Sem Fio, Geração Automática de Código e *Hardware* de Controle, referentes à Estação Base, são implementados utilizando a plataforma dSPACE em conjunto com as ferramentas MATLAB/Simulink/RTW

e o aplicativo servidor WRC-Server. O módulo Planta corresponde ao sistema de controle acoplado a plataforma experimental, como por exemplo um motor de indução trifásico alimentado por um inversor de tensão utilizado como segundo experimento realizado com esta plataforma. Já na Estação Remota, os módulos Interface com o Usuário e Comunicação Sem Fio são implementados utilizando o dipositivo portátil N800 em conjunto com o aplicativo cliente WRC-Client e os recursos de comunicação Bluetooth, constituindo o configurador remoto da plataforma.

Procedimento de Configuração Remota

Considerando-se que o sistema de controle encontra-se pronto para iniciar seu funcionamento, inicialmente, o operador do sistema deve inicializar a plataforma experimental, mais precisamente, executar a aplicação servidor da Estação Base, i.e., o *software* WRC-Server. O servidor deve ser executado para disponibilizar o serviço de configuração remota. Em seguida, o usuário deve utilizar o configurador remoto para comunicar-se com a plataforma experimental. O configurador remoto consiste em um dispositivo portátil que executa o *software* WRC-Client para configuração do modelo do algoritmo de controle representado em diagrama de blocos.

O usuário, então, tenta conectar-se ao servidor. Um processamento de busca pelo serviço de configuração é realizado de forma transparente para o usuário que aguarda a conexão com o servidor. Quando conectado, o usuário deve solicitar uma operação de *Upload* do modelo (arquivo MDL). O servidor ao reconhecer o comando, procede com a criação de um arquivo XML contendo as informações do arquivo MDL. Os dados contidos no arquivo XML são lidos e divididos em pacotes para serem enviados ao configurador remoto. O configurador recebe os pacotes e reagrupa-os para obter o arquivo XML correspondente. A partir do conteúdo deste, o arquivo MDL é obtido e interpretado para apresentar o modelo em diagrama de blocos na tela do configurador remoto. Neste momento, o usuário pode configurar o modelo recebido.

Após configurado o modelo, o usuário deve realizar uma operação de *Download* para implementar o novo algoritmo na plataforma. Novamente, um arquivo XML contendo os dados do arquivo MDL do modelo configurado é criado, e os dados são divididos em pacotes e enviados para o servidor. O servidor ao receber os pacotes de dados, reagrupa-os para obter o arquivo XML correspondente. A partir do conteúdo deste arquivo XML, o arquivo MDL é obtido e salvo/sobrescrito com o nome ***model*** em um diretório do sistema de arquivos apropriado. O servidor, em seguida, invoca o MATLAB/RTW para realizar a geração automática de código do modelo recebido, transparentemente ao usuário. Por fim, o código executável gerado é carregado na placa controladora da plataforma e o funcionamento do sistema de controle pode ser acompanhado pelo ControlDesk.

3.4 Sumário

Neste capítulo foi apresentada a plataforma de desenvolvimento experimental utilizada para testar o sistema de configuração remota. A plataforma é constituída por um sistema dSPACE, representando a Estação Base da arquitetura abordada, e um dispositivo portátil realizando a função de configurador remoto do sistema de controle, representando a Estação Remota. A tecnologia Bluetooth é utilizada como padrão de comunicação sem fio.

Também foram apresentadas a estrutura da plataforma dSPACE e suas principais caracteríscticas, destacando-se o procedimento de geração automática de código e o conjunto de ferramentas que auxiliam o projeto de aplicações de controle. De fato, a plataforma dSPACE foi utilizada por possuir os recursos que viabilizaram a implementação da arquitetura do sistema de configuração remota. Além disso, o dSPACE compreende um ambiente para programação em linguagem Python que permite acesso aos recursos da plataforma de forma geral, tanto da placa controladora e das ferramentas de *software* quanto do próprio computador PC. Com isso, foi possível agregar conectividade sem fio ao sistema dSPACE.

Neste capítulo também foi apresentado e discutido o desenvolvimento de toda a estrutura cliente-servidor da plataforma experimental, detalhando-se a implementação das aplicações cliente e servidor, a integração com a plataforma dSPACE, bem como o funcionamento de toda a plataforma experimental construída.

Capítulo 4

Resultados Experimentais

4.1 Introdução

Este capítulo tem como objetivo descrever os testes realizados com a plataforma de desenvolvimento experimental deste trabalho, cuja finalidade é testar o funcionamento do configurador remoto e a implementação de toda a estrutura desenvolvida baseada na arquitetura abordada. Dois experimentos são realizados. O primeiro aborda um simples experimento de aquisição de dados utilizando os conversores A/D e D/A do *hardware* de controle da plataforma experimental, visando apresentar o funcionamento do processo de configuração remota de maneira detalhada. O segundo envolve o acionamento de um motor de indução alimentado por um inversor de tensão (cenário de aplicação discutido no Capítulo 2) onde é possível a implementação de diferentes estratégias de controle, sendo abordado o acionamento baseado no princípio Volts/Hertz.

4.2 Primeiro Exemplo: Aquisição de Dados

Aquisição de dados é uma atividade comumente utilizada em processos industriais e atividades laboratoriais. A facilidade proporcionada pela platforma dSPACE em tarefas de aquisição de dados influenciou a escolha de um exemplo envolvendo o uso de conversores A/D e D/A como primeiro exemplo para testar a plataforma experimental.

Este primeiro exemplo consiste em realizar um experimento utilizando um gerador de sinais e um osciloscópio para analisar a resposta de um sistema de segunda ordem. Inicialmente, um modelo utilizando blocos adequados para simulação tanto do gerador de sinais quanto do osciloscópio é elaborado na plataforma dSPACE através do Simulink, correspondendo ao modelo inicial presente na plataforma e que deve ser configurado. Em seguida, o modelo é alterado através do configurador remoto da plataforma experimental, utilizando-se os blocos que representam os conversores A/D e D/A da placa DS1103 PPC para realizar a aquisição do

sinal do gerador e enviar o sinal de resposta do sistema para visualização em um osciloscópio.

Configuração Inicial do Sistema

O modelo elaborado no Simulink corresponde a um sistema de segunda ordem simples, definido pelos seguintes parâmetros: coeficiente de amortecimento (ξ) igual a 0,7 e frequência natural igual a 10 Hz ($\omega_n = 2\pi f \cong 62,83 rad \setminus s$). A função de transferência que representa este modelo de segunda ordem pode ser descrita pela seguinte relação:

$$G(s) = \frac{\omega_n^2}{s^2 + 2\xi\omega_n + \omega_n^2} \tag{4.1}$$

A função de transferência obtida após a substituição dos parâmetros torna-se:

$$G(s) = \frac{3947,84}{s^2 + 87,966 + 3947,84} \tag{4.2}$$

Este primeiro modelo é construído utilizando blocos que simulam o gerador de sinais e o osciloscópio. Na Figura 4.1 é ilustrado o diagrama de blocos elaborado no Simulink e assumido como modelo inicial presente na plataforma experimental.

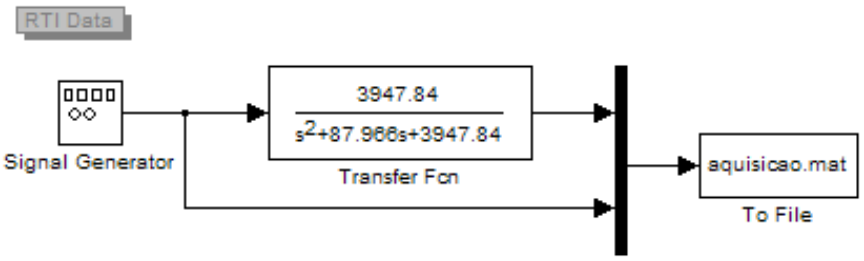


Figura 4.1: Modelo inicial para experimento com aquisição de dados.

O diagrama de blocos inclui os blocos: RTI DATA, Signal Generator, Transfer Fcn e To File. O bloco RTI DATA é necessário para indicar o uso da biblioteca RTI do dSPACE pelo modelo. Os dados referentes a resposta do sistema são armazenados no arquivo aquisicao.mat, tipo de arquivo próprio do MATLAB. O bloco Signal Generator está ajustado para uma onda quadrada com amplitude de um Volt e frequência de dois Hertz.

Após a elaboração do modelo inicial, a ferramenta RTW foi utilizada para a geração do código executável e carregamento deste na placa DS1103 da plataforma. Na Figura 4.2 é apresentado todo o processamento de geração automática de código realizado pelo RTW, incluindo geração do código-fonte, compilação, *linking* e carregamento na placa controladora DS1103.

A Figura 4.3 corresponde ao gráfico obtido com a simulação do modelo inicial, executado na própria placa controladora DS1103.

```
-------------------------------------------------------------------------
Starting build procedure with RTI 4.5 (RTI1103, 05-Aug-2003)
Model: "model" (C:\MATLAB6p5\work\WRC-Server\model.mdl)
-------------------------------------------------------------------------
*** Working directory: "C:\MATLAB6p5\work\WRC-Server"
*** Optional User System Description File model_usr.sdf not available
*** Initializing code generation
### Starting Real-Time Workshop build procedure for model: model
### Generating code into build directory: .\model_rti1103
### Invoking Target Language Compiler on model.rtw
*** Generating Variable Description File model.trc
*** Optional User Variable Description File model_usr.trc not available
*** Generating template: User-Code File model_usr.c
*** Generating template: User Makefile model_usr.mk
### Creating project marker file: rtw_proj.tmw
### Creating model.mk from c:\dspace40\matlab\rti1103\m\rti1103.tmf
### Building model: dsmake -f model.mk  WORKINGBOARD=ds1103

BUILDING APPLICATION (Single Timer Task Mode)

WORK  DIRECTORY "C:\MATLAB6p5\work\WRC-Server"
BUILD DIRECTORY "C:\MATLAB6p5\work\WRC-Server\model_rti1103"
TARGET COMPILER "C:\PPCTools20f"

COMPILING  model.c
COMPILING  model_data.c
COMPILING  C:\dSPACE40\MATLAB\RTI1103\C\rti_sim_engine.c
COMPILING  C:\dSPACE40\MATLAB\RTI1103\C\rti_init_c.c
COMPILING  C:\dSPACE40\MATLAB\RTI1103\C\rti_external_sim.c
COMPILING  C:\MATLAB6p5\rtw\c\src\rt_sim.c
COMPILING  C:\dSPACE40\MATLAB\RTI1103\C\rti_assert.c

USING LIBRARY "C:\dSPACE40\MATLAB\RTI1103\C\Lib\rtwlib_r13_0_ds1103.lib"

LINKING APPLICATION ...
LINKING FINISHED

LOADING APPLICATION "model.sdf" ...
[#1] ds1103 - RTI:      Initializing ... (720)
[#2] ds1103 - RTI:      Initialization completed (721)
[#3] ds1103 - RTI:      Simulation state: RUN (700)
LOADING FINISHED

MAKE PROCESS SUCCEEDED

### Successful completion of Real-Time Workshop build procedure for model: model
*** Finished RTI build procedure for model model
>>
```

Figura 4.2: Processamento de geração do código executável pelo RTW.

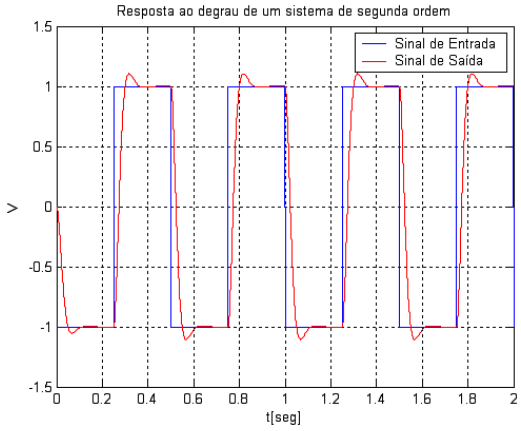


Figura 4.3: Resultado da simulação utilizando o modelo inicial - primeiro exemplo.

Processo de Configuração Remota

Verificado o processo de geração do código executável e o resultado da execução do algoritmo (modelo), o passo seguinte consiste na conexão dos equipamentos que serão utilizados neste experimento: o osciloscópio e o gerador de sinais. Na Figura 4.4 é apresentada a bancada com os instrumentos conectados a plataforma.

Em seguida, utilizando-se o configurador remoto, o usuário da plataforma pode configurar o

Figura 4.4: Montagem experimental - primeiro exemplo.

modelo presente na mesma. Para isso, é necessário que o usuário inicie antes o aplicativo servidor que anuncia e disponibiliza o serviço de configuração remota. Na Figura 4.5 é apresentado o processamento referente ao anúncio de serviço pelo servidor, executado a partir do ambiente de desenvolvimento Eclipse [45].

Figura 4.5: Anúncio do serviço de configuração remota - primeiro exemplo.

```
~/MyDocs/Yuri/RCW_tool $ python RCW-Client.py
Trying connection...
Starting connection ...
Services:  dSPACE Platform Service
Connecting to "dSPACE Platform Service" on 00:10:60:AD:B2:25
Connected.
Uploading model...
Number of packages: 109
package  1  >> <Model><Name>"model"</Name><Version>5.0</Version><SaveDefaultBlockParams>on</SaveDefaultBlockParams> •••
 •
 •
 •
package  108  >> /Branch></Line><Line><SrcBlock>"Transfer Fcn"</SrcBlock><SrcPort>1</SrcPort><DstBlock>"Mux"</DstBlock> •••
package  109  >> </DstBlock><DstPort>1</DstPort></Line></System></Model>
XML file received and saved: file_received.xml
File received.
File (*.mdl) saved!
```

Figura 4.6: Visualização das mensagens referentes ao processamento das operações de conexão com o servidor e *Upload* do modelo inicial - primeiro exemplo.

Então, o usuário utiliza o configurador remoto para executar o aplicativo cliente, realizar a

busca pelo serviço de configuração, conectar-se ao servidor e realizar uma operação de *Upload* do modelo presente na plataforma. Na Figura 4.6 é apresentado o processamento referente a estas operações através do terminal de linha de comando do configurador remoto.

Na Figura 4.7 é ilustrado o arquivo XML referente ao modelo inicial enviado pelo servidor, enquanto que na Figura 4.8 é apresentado o arquivo MDL correspondente, obtido pelo cliente.

Figura 4.7: Arquivo XML referente ao modelo inicial enviado pelo servidor (*Upload*) - primeiro exemplo.

Neste experimento, a idéia é poder alterar os blocos do modelo de forma a configurar um

```
Model {
  Name "model"
  Version 5.0
  SaveDefaultBlockParams on
  •
  •
  •
  BlockDefaults {
    Orientation "right"
    ForegroundColor "black"
    •
    •
    •
  }
  BlockParameterDefaults {
    Block {
      BlockType Gain
      Gain "1"
      •
      •
      •
    }
    Block {
      BlockType SubSystem
      ShowPortLabels on
      •
      •
      •
    }
    •
    •
    •
  }
  AnnotationDefaults {
    HorizontalAlignment "center"
    VerticalAlignment "middle"
    •
    •
    •
  }
  LineDefaults {
    FontName "Helvetica"
    FontSize 9
    •
    •
    •
  }
  System {
    Name "model"
    Location [362, 141, 942, 434]
    Open on
    •
    •
    •
    Block {
      BlockType SubSystem
      Name "RTI Data"
      •
      •
      •
    •
    •
    •
    Line {
      SrcBlock "Transfer Fcn"
      SrcPort 1
      •
      •
      •
    }
    •
    •
    •
  }
}
```

Figura 4.8: Arquivo MDL recebido pelo cliente após a operação de *Upload* - primeiro exemplo.

novo funcionamento da plataforma. A opção adotada consiste, respectivamente, na substituição dos blocos Signal Generator e To File pelos blocos DS1103ADC_C17 e DS1103DAC_C1, que utilizam conversores A/D e D/A da placa controladora DS1103. A adição de dois blocos Gain é necessária para compensar o condicionamento do sinal realizado pelos conversores A/D (que divide o valor do sinal de entrada por um fator de dez) e D/A (que multiplica o valor do sinal de saída por um fator de dez) [19]. Assim, o bloco Gain após o bloco DS1103ADC_C17 deve ter um valor de ganho igual a dez, enquanto que bloco Gain1 antes do bloco DS1103DAC_C1 deve ter um valor de ganho igual a um décimo, conforme a Figura 4.9. A função de transferência é

mantida para comparação dos resultados.

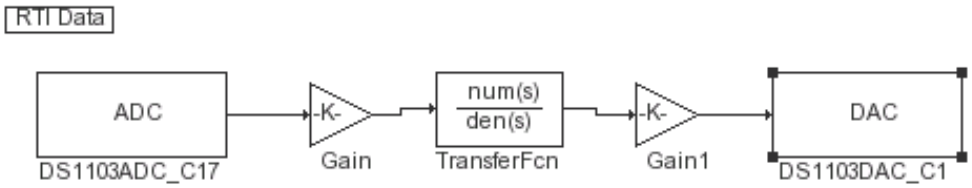


Figura 4.9: Modelo modificado pelo configurador remoto para experimento com aquisição de dados.

O ajuste dos parâmetros dos blocos do modelo é realizado através de janelas de configuração para cada tipo de bloco, como por exemplo para o bloco Gain, conforme apresentada na Figura 4.10.

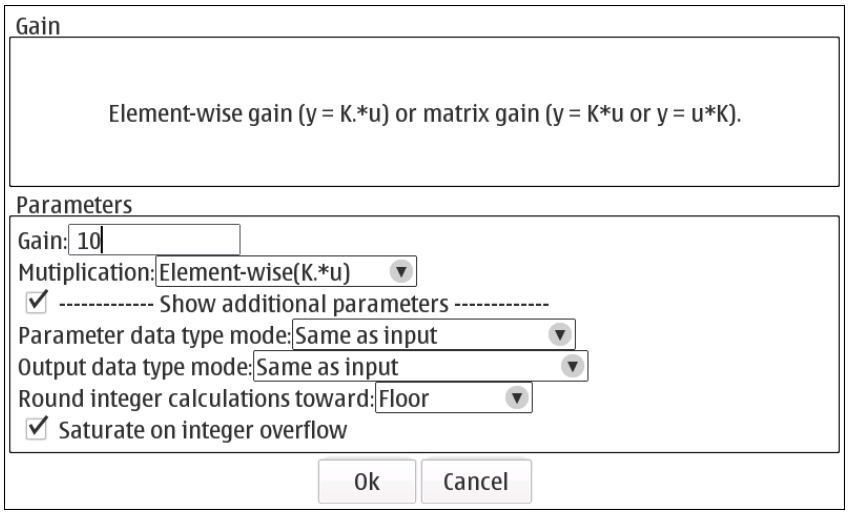


Figura 4.10: Janela de configuração de parâmetros do modelo para o bloco Gain (em tela cheia).

Após a configuração do modelo, o usuário realiza uma operação de *Download* para envio do modelo modificado ao servidor. Na Figura 4.11 é apresentado o processamento executado pelo servidor referente a esta operação.

Concluído o envio do modelo modificado, o processo de geração automática de código é executado pela invocação do *software* MATLAB e uso da ferramenta RTW. O processamento de geração do código executável pelo RTW é exatamente igual ao apresentado na Figura 4.2 em virtude de ser mantido um único arquivo MDL no sistema de arquivos da plataforma, o único usado pelo RTW, sendo sempre sobrescrito após uma operação de *Download*. Vale ressaltar que este arquivo MDL também é o único utilizado nas operações de *Upload*. A execução do algoritmo de controle é iniciada imediatamente após o carregamento do código executável na placa controladora. A Figura 4.12 corresponde ao gráfico obtido com o experimento do modelo modificado pelo configurador remoto.

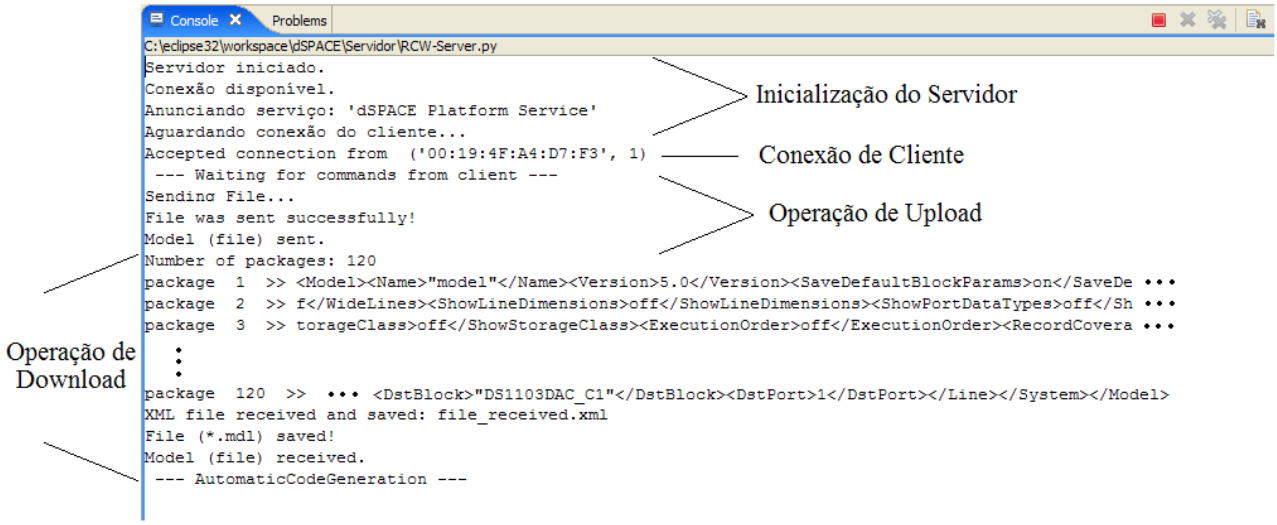


Figura 4.11: Visualização das mensagens referentes ao processamento da operação de *Download* do modelo modificado - primeiro exemplo.

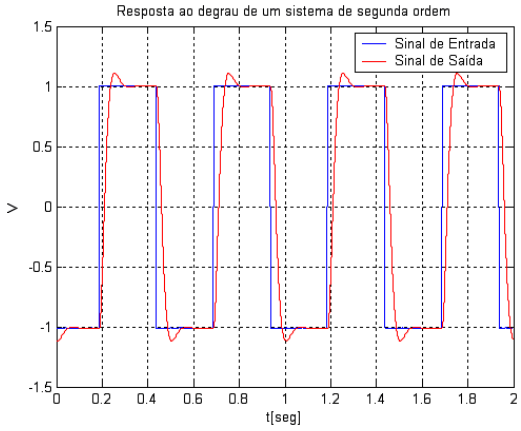


Figura 4.12: Resultado experimental utilizando o modelo modificado - primeiro exemplo.

Na Figura 4.13 é ilustrado o gráfico referente ao experimento com o modelo modificado, visualizado no osciloscópio.

Através do ControlDesk também foi obtido o gráfico (Figura 4.14) referente ao experimento com o modelo modificado, servindo para comparação com o gráfico visualizado no osciloscópio.

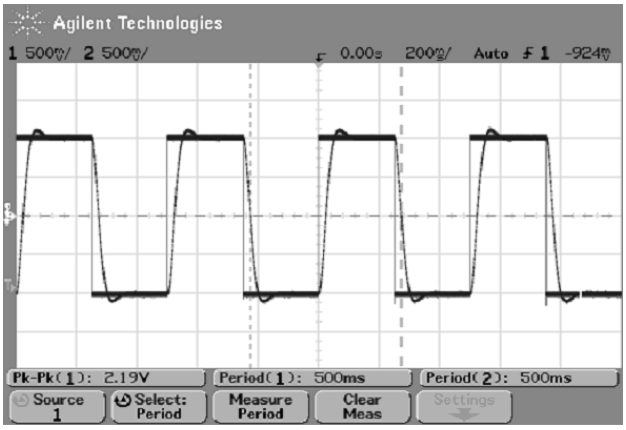


Figura 4.13: Resultado experimental utilizando o modelo modificado (Osciloscópio) - primeiro exemplo.

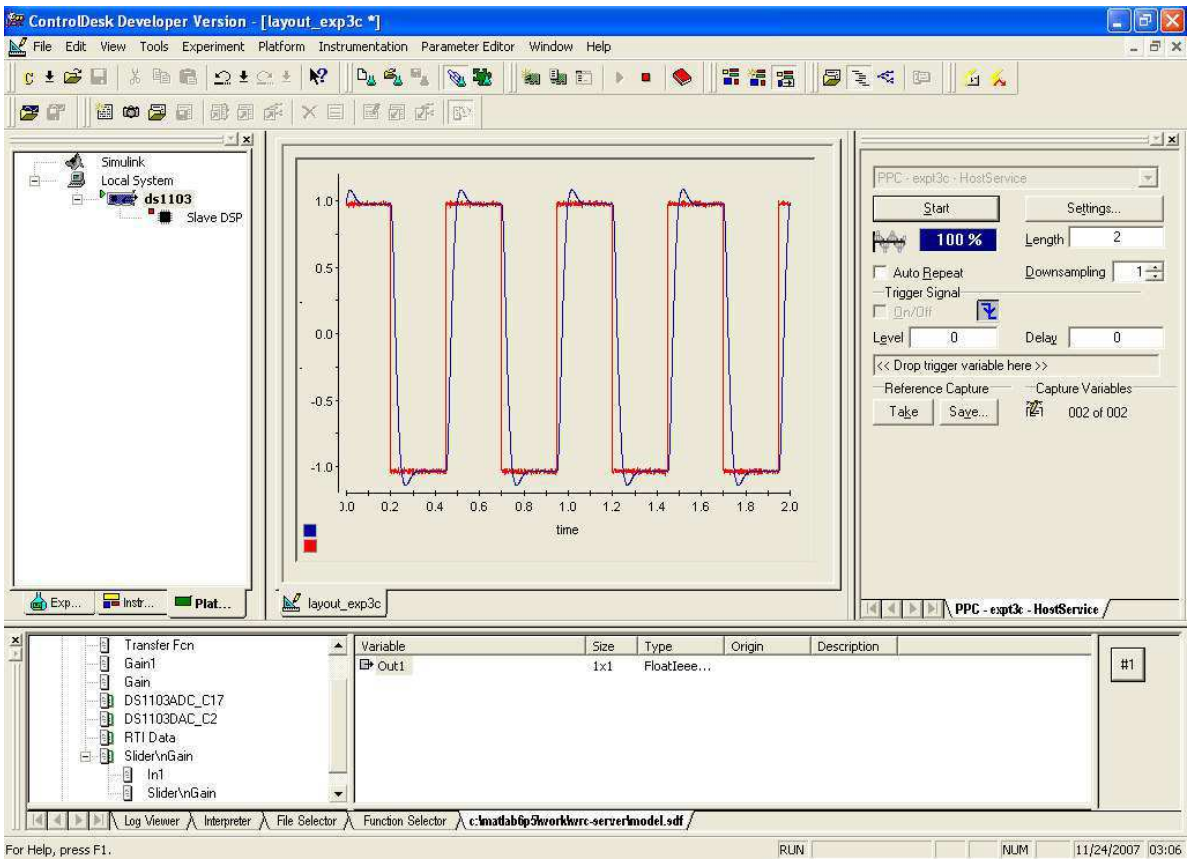


Figura 4.14: Resultado experimental utilizando o modelo modificado (ControlDesk) - primeiro exemplo.

4.3 Segundo Exemplo: Acionamento de um Motor de Indução

Na Figura 4.15 é apresentada a planta disponível em laboratório e acoplada à plataforma experimental para o segundo experimento. A descrição dessa estrutura é apresentada no apêndice

A desta dissertação.

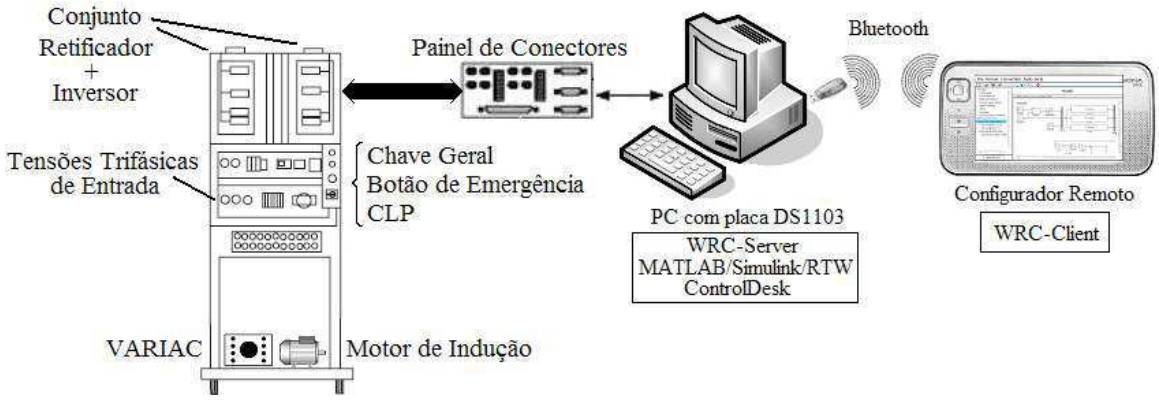


Figura 4.15: Plataforma experimental para o segundo exemplo - acionamento do motor de indução.

Configuração Inicial do Sistema

Com o intuito de implementar, primeiramente, uma estratégia de controle simples para o motor de indução alimentado por um inversor de tensão, foi adotado um controle em malha aberta utilizando o princípio Volts-Hertz (V/f), conhecido também como tensão/frequência, para acionamento da máquina.

O princípio V/f é uma técnica para manter o fluxo no entreferro da máquina de indução constante, através do controle da tensão (V) e da frequência (f) do estator, de forma que a relação V/f é mantida constante. Utilizando este método de controle, o torque e a velocidade da máquina podem ser controlados variando-se tanto a tensão quanto a freqüência.

Nos inversores em geral, o intervalo de tempo em que um dispositivo semicondutor (interruptor) permanece em seu estado de condução somado ao intervalo de tempo em que ele permanece no seu estado de bloqueio é chamado de pulso de comando do interruptor. Os pulsos de comando para os interruptores de um inversor de dois níveis (Figura 4.16) podem ser gerados quando três tensões senoidais (modulantes) defasadas de 120° uma da outra (v_a, v_b e v_c) são comparadas com um sinal triangular em alta frequência, chamada de portadora triangular (Figura 4.16). A frequência dos sinais modulantes estabelece a freqüência desejada da componente fundamental do sinal de saída do inversor. Esta técnica é chamada de *Carrier Based Pulse Width Modulation*(CB-PWM) [47].

Conforme a Figura 4.17, baseado no valor médio que o sinal de saída modulado deve ter em um período de modulação (T_s) a partir do sinal de referência de entrada (v_x), os tempos em que os interruptores permanecem em condução (T_1, T_2 e T_3) são dados por:

$$T_y = (\frac{v_x}{2E} + \frac{1}{2})T_S \tag{4.3}$$

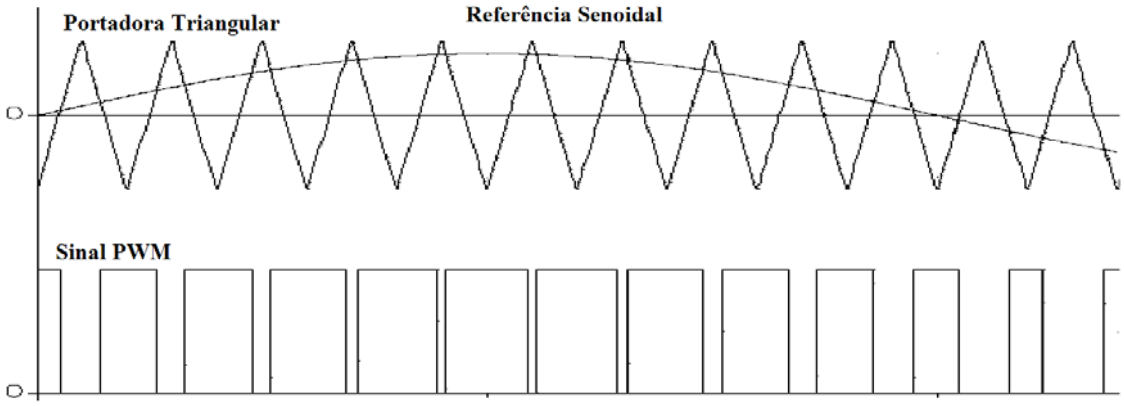


Figura 4.16: Modulação por comparação com portadora triangular.

onde $x \epsilon$ {a,b,c}, $y \epsilon$ {1,2,3} e E corresponde à tensão do barramento CC do inversor.

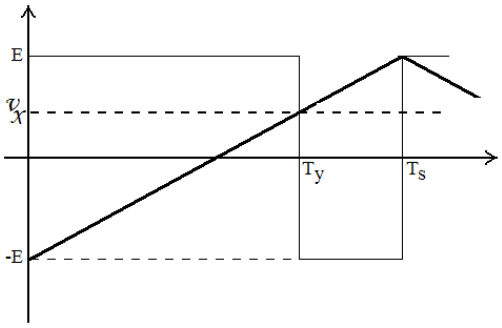


Figura 4.17: Pulso de comando dos interruptores do inversor de 2 níveis com modulação por portadora triangular.

Neste exemplo, como trata-se de um controle em malha aberta, a variável E corresponde à amplitude do sinal da portadora triangular, utilizada para a geração dos pulsos de comando dos interruptores do inversor de tensão.

A técnica de acionamento utilizando o princípio V/f consiste na elavação da amplitude das tensões de referência (v_x) de forma proporcional ao aumento gradual de zero até o valor da frequência destas tensões, de modo que o acionamento da máquina seja contínuo e suave. Assim, os tempos de condução dos interruptores também são proporcionais ao aumento gradativo da amplitude das tensões de referência.

O modelo em diagrama de blocos que implementa o acionamento do motor de indução pelo princípio V/f é apresentado na Figura 4.18 Primeiramente, o modelo construído foi utilizado para analisar a geração dos sinais PWM. O diagrama de blocos foi elaborado no Simulink e é assumido como modelo inicial presente na plataforma experimental para este segundo exemplo.

O diagrama de blocos inclui os blocos: RTI DATA, Ramp, Constant, Product, Saturation, Mux, Demux, Fcn, To File, Gain, Sum e DS1103SL_DSP_PWM3. O diagrama pode ser dividido em três partes funcionais: geração das tensões de referência seguindo o princípio V/f

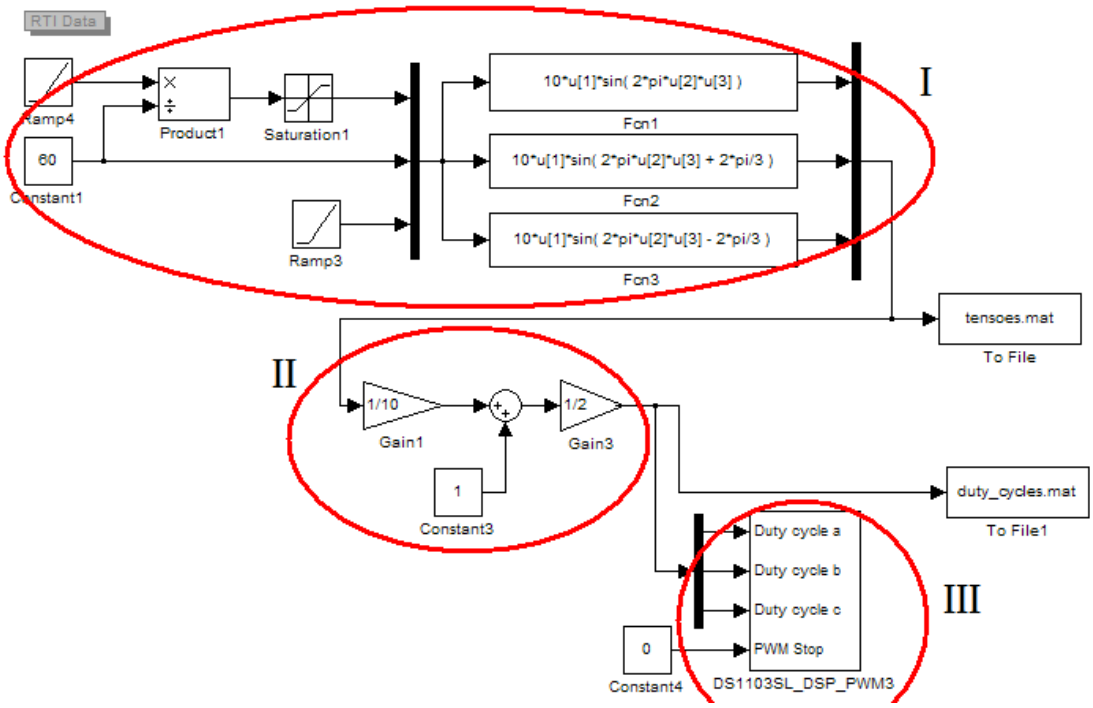


Figura 4.18: Modelo inicial para geração de sinais PWM.

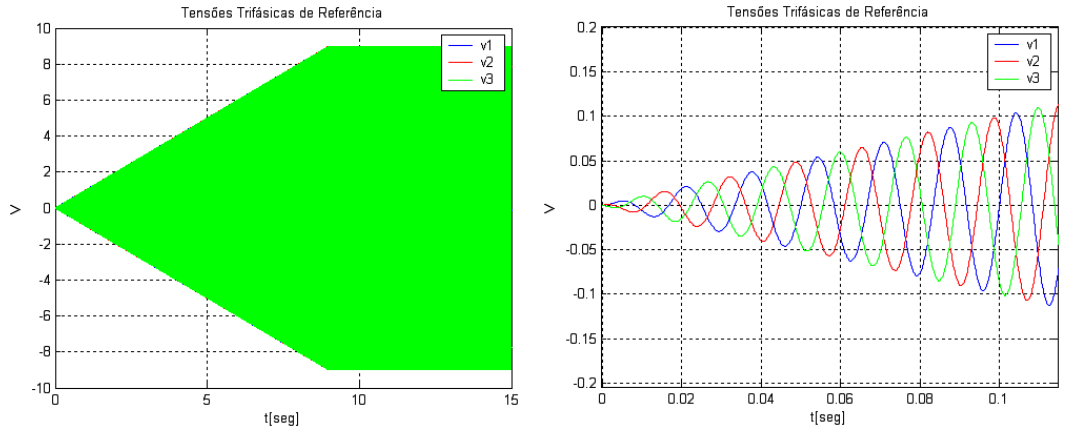
(I), cálculo dos tempos de condução dos interruptores segundo a equação 4.3 (II), e geração dos sinais PWM para o inversor (III), conforme a Figura 4.18.

A amplitude do sinal da portadora triangular é igual ao valor da amplitude máxima definida para as tensões de referência, no caso 10V. Na primeira parte (I), a amplitude das tensões de referência aumenta de 0 até 9V gradualmente e de forma proporcional ao aumento do valor da frequência representada pelo bloco Ramp4 em virtude do bloco Saturation1 definir os limites de saturação superior e inferior como 0,9 e 0, respectivamente. Nota-se, portanto, que o limite superior representa o índice de modulação em amplitude do sinal PWM. A segunda parte (II) envolve a modelagem matemática da equação 4.3, definindo-se os tempos de condução dos interruptores do inversor, e a terceira parte (III) é baseada em um bloco próprio da biblioteca RTI do dSPACE, DS1103SL_DSP_PWM3. A biblioteca RTI do dSPACE disponibiliza este bloco para geração de sinais PWM. O bloco DS1103SL_DSP_PWM3 recebe como entradas os tempos de condução dos interruptores do inversor e um sinal para controle da geração dos sinais PWM (PWM Stop). Este bloco engloba um gerador de onda triangular e um comparador para gerar cada sinal de comando do inversor trifásico, incluindo os sinais dos interruptores complementares.

Os dados referentes aos tempos de condução dos interruptores e tensões de referência são armazenados nos arquivos duty_cycles.mat e tensoes.mat, respectivamente.

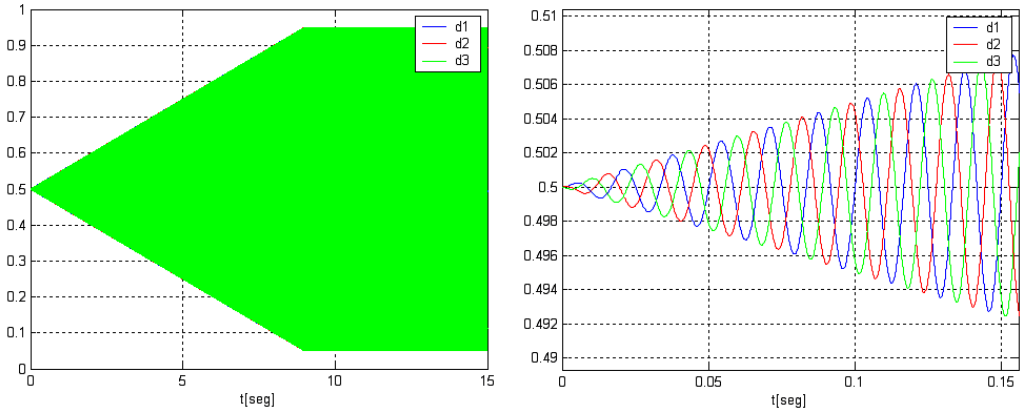
As figuras 4.19 e 4.20 correspondem aos gráficos obtidos com a simulação do modelo inicial, executado na própria placa controladora DS1103, para geração das tensões de referência e dos

sinais PWM.



(a) (b)

Figura 4.19: Tensões trifásicas de referência utilizando o modelo inicial - segundo exemplo.



(a) (b)

Figura 4.20: Sinais PWM gerados utilizando o modelo inicial - segundo exemplo.

Processo de Configuração Remota

Através do configurador remoto, o usuário da plataforma pode configurar o modelo corrente realizando os mesmos passos descritos no primeiro exemplo, isto é, realizando uma operação de *Upload* do modelo, configurando-o e reenviando-o para a plataforma por meio de uma operação de *Download* . A configuração a ser realizada consiste, respectivamente, na adição dos blocos DS1103ADC_C17 e DS1103ADC_C18, que utilizam conversores A/D da placa controladora DS1103 (canais 17 e 18, respectivamente) para a medição de tensão e corrente do motor de indução. Conforme a Figura 4.21, a adição de quatro blocos Gain é necessária, dois para compensar o condicionamento do sinal realizado pelos conversores A/D, e dois para compensar os fatores de escala dos instrumentos de medição de tensão e corrente utilizados. O sensor de tensão foi ajustado para o fator de escala de 200:1, enquanto que o sensor de corrente teve seu

fator de escala ajustado para 100:1. Assim, os blocos Gain usados para compensação dos sinais de tensão e corrente devem ter valores de ganho iguais a 200 e 100, respectivamente. Além disso, mais dois blocos To File foram adicionados para capturar os valores dos sinais de corrente e tensão do motor de indução. Os valores são armazenados nos arquivos tensao_MI.mat e corrente_MI.mat.

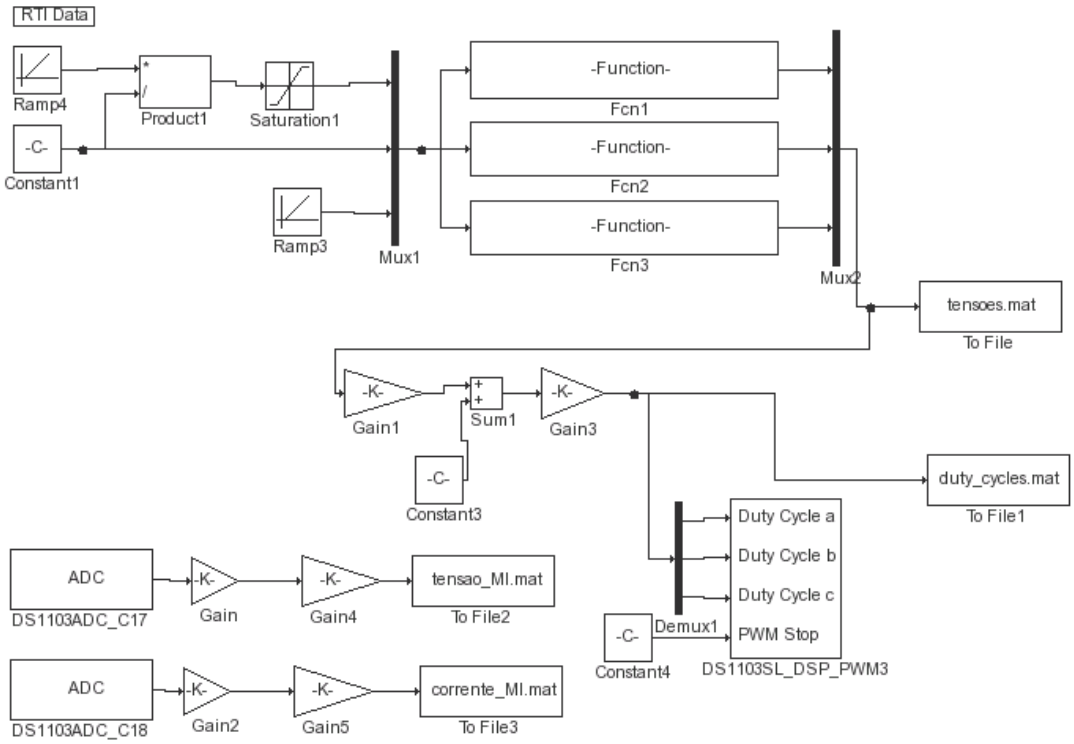
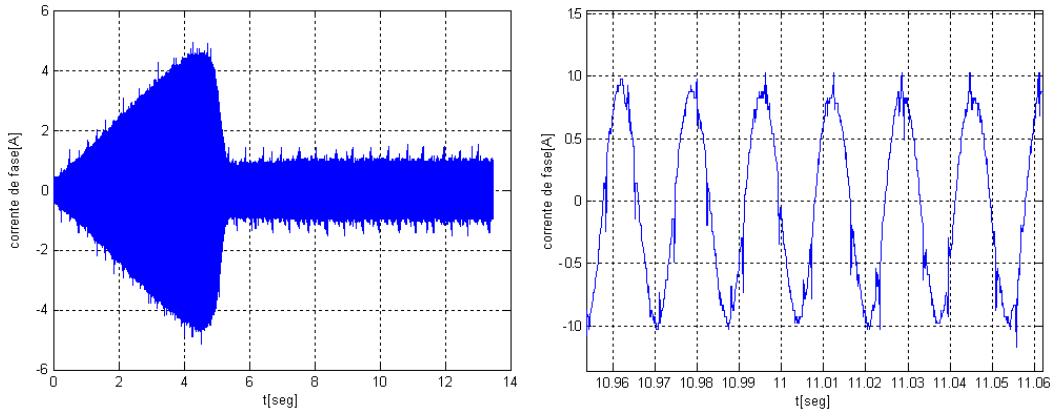


Figura 4.21: Modelo modificado pelo configurador remoto para acionamento do motor de indução.

Na Figura 4.22 é apresentado o gráfico da corrente de fase do motor de indução, referente ao experimento com o modelo modificado pelo configurador remoto para o acionamento da máquina, utilizando uma tensão de barramento CC de $167V$.

Através do ControlDesk também foram obtidos gráficos (Figura 4.23) referentes ao experimento com o modelo modificado, incluindo a geração dos sinais PWM, o sinal de tensão entre fase e neutro e o de corrente de fase do motor de indução, utilizando neste caso uma tensão de barramento CC de $100V$..

Outras estratégias de controle podem ser implementadas, como por exemplo um controlador de corrente PI, PI modificado ou PID, ou ainda um controle por histerese. A implementação destes controladores são propostos como trabalhos futuros.



(a) (b)

Figura 4.22: Resultado experimental utilizando o modelo modificado (corrente de fase do motor de indução) - segundo exemplo.

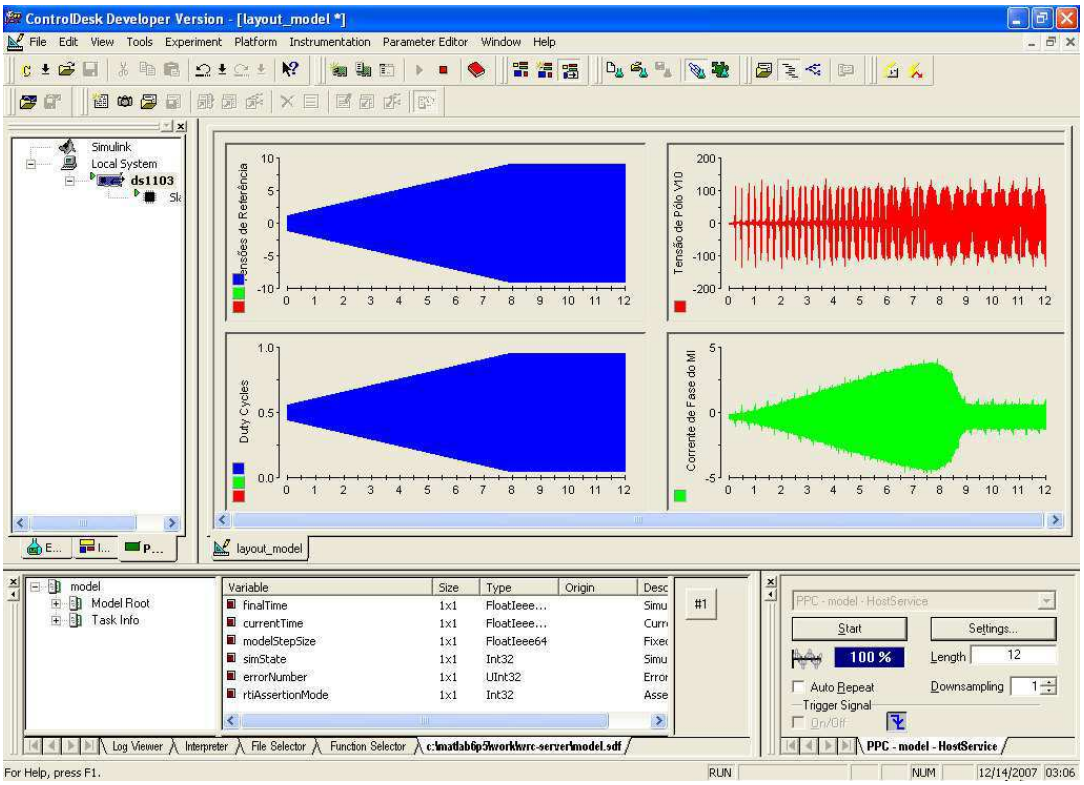


Figura 4.23: Resultado experimental utilizando o modelo modificado (ControlDesk) - segundo exemplo.

4.4 Sumário

Este capítulo apresentou os resultados dos testes realizados com a plataforma experimental desenvolvida neste trabalho. Dois sistemas foram implementados: um sistema simples de aquisição de dados e um sistema de acionamento de um motor de indução alimentado por um inversor de tensão.

Em todas as etapas dos experimentos, o desempenho da plataforma demonstrou ser satisfatório, visto que os resultados obtidos estão de acordo com o esperado. Com isso, foi possível testar o funcionamento do configurador remoto e a implementação de toda a estrutura desenvolvida baseada na arquitetura abordada.

Capítulo 5

Conclusão

Neste trabalho, uma ferramenta de *software* para configuração remota de algoritmos de controle para aplicações industriais a partir de dispositivos móveis sem fio foi apresentada. Tal configurador remoto é compatível com um sistema com conectividade sem fio, capaz de realizar geração automática de código a partir de modelos em diagrama de blocos, e que pode ser acoplado a um sistema de controle industrial. No entanto, um sistema com tais características ainda não existe, e por isso uma arquitetura que torna possível a configuração de sistemas de controle industriais baseada em dispositivos móveis sem fio e geração automática de código foi definida, e uma plataforma de desenvolvimento experiemental para testar o funcionamento do configurador remoto e a implementação da arquitetura abordada foi utilizada.

Inicialmente, com o objetivo de fundamentar de maneira consistente o trabalho desenvolvido e de contextualizar os principais tópicos abordados nesta dissertação, uma revisão bibliográfica foi elaborada.

Ao longo deste trabalho foram discutidos temas relacionados ao projeto e desenvolvimento da ferramenta de configuração e da arquitetura para um sistema de configuração remota. A definição da estrutura, organização e funcionamento da arquitetura foram baseados na infraestrutura típica de plataformas para prototipagem rápida de sistemas de controle encontradas na literatura.

A arquitetura do sistema consiste em uma estrutura cliente-servidor na qual são definidas duas entidades, Estação Base e Estação Remota, que se comunicam segundo um padrão de tecnologia sem fio. A Estação Remota constitui a parte cliente da arquitetura, sendo responsável pela configuração do algoritmo de controle da aplicação industrial. A Estação Base representa a parte servidor e consiste em um sistema com conectividade sem fio, capaz de realizar geração automática de código e que pode ser acoplado ao sistema de controle.

A plataforma de desenvolvimento experimental elaborada é constituída por um sistema dSPACE, representando a Estação Base da arquitetura, e um dispositivo portátil realizando a função de configurador remoto do sistema de controle, i.e., representando a Estação Remota. A

tecnologia Bluetooth foi escolhida como padrão para a comunicação sem fio entre as estações. De fato, a plataforma dSPACE foi utilizada por possuir os recursos que viabilizaram a implementação de módulos da Estação Base da arquitetura do sistema. Para o desenvolvimento da estrutura cliente-servidor da plataforma experimental foram desenvolvidos dois aplicativos de *software*: um é o cliente (Estação Remota), responsável pela configuração do modelo do algoritmo de controle da aplicação, e o outro servidor (Estação Base), responsável pela integração entre os demais módulos da Estação.

O desenvolvimento de infra-estruturas de rede sem fio em ambientes industriais está ocorrendo de forma gradual, e neste cenário aspectos como interoperabilidade e extensibilidade estão entre os principais requisitos para soluções futuras. No entanto, toda a potencialidade da tecnologia sem fio ainda precisa ser explorada. Também há uma contínua e crescente necessidade por parte das indústrias por ferramentas para configuração, análise e diagnóstico de processos industriais que proporcionem redução do tempo envolvendo atividades de configuração e programação de sistemas de controle, sem comprometer a qualidade dos mesmos e ainda proporcionar flexibilidade, portabilidade, confiabilidade e praticidade. Entretanto, até o presente momento não foi encontrada na literatura uma linha de pesquisa abordando um sistema de configuração remota de algoritmos de controle para aplicações industriais que combine os benefícios da comunicação sem fio e da geração automática de código. A arquitetura abordada neste trabalho busca agregar estes dois aspectos de modo a satisfazer as características exigidas neste contexto.

Em geral, aplicações industriais de controle onde sensores e atuadores são empregados para controlar parâmetros e/ou o estado do sistema, são favoráveis para a implementação da arquitetura do sistema de configuração remota. Um sistema baseado na arquitetura abordada é bastante adequado e versátil para sistemas de controle em ambientes industriais, oferecendo conectividade e integração de forma prática e simples com estações remotas para configuração de sistemas de controle.

O sistema de configuração remota discutido pode ser considerado uma aplicação em potencial para o setor industrial, porém não com os mesmos recursos utilizados para a implementação da plataforma experimental. O desenvolvimento de um sistema embarcado para implementação da Estação Base da arquitetura pode ser considerado ideal, no entanto o desenvolvimento de tal sistema demanda tempo e complexidades de projeto que inviabilizaram sua implementação neste trabalho. Um cenário de aplicação foi discutido, considerando um ambiente industrial onde é possível a implementação do sistema de configuração. A aplicação de controle consiste em um motor de indução alimentado por um inversor de tensão, possibilitando o uso de diferentes estratégias de controle para o acionamento e controle da máquina.

Os resultados experimentais obtidos com a plataforma demonstram que a estrutura desenvolvida é consistente e a arquitetura é adequada para a finalidade que se propõe. Dois ex-

perimentos distintos foram realizados e comprovaram que a plataforma experimental funciona satisfatoriamente, visto que os resultados obtidos estão de acordo com o esperado.

5.1 Perspectivas para Trabalhos Futuros

Os trabalhos desenvolvidos nesta dissertação tornam possível a realização de outros estudos e análises, além dos que foram apresentados ao longo do texto. A seguir são enumerados alguns temas e atividades que poderão ser investigados e desenvolvidos em futuros trabalhos:

1. Projeto e desenvolvimento de um sistema embarcado com capacidade de geração automática de código e conectividade sem fio para configuração, controle e monitoramento de sistemas de controle em tempo real em ambientes industriais;

2. Aprimoramento da integração da estrutura cliente-servidor com os módulos da arquitetura abordada, isto é, da integração dos aplicativos cliente e servidor com as ferramentas de *software* responsáveis pela geração automática de código e pelo monitoramento do sistema de controle;

3. Aperfeiçoamento do aplicativo cliente através da adição/implementação de novos blocos para configuração dos modelos e para monitoramento remoto, de modo a utilizar o sistema de configuração remota com outros sistemas de controle;

4. Adaptar o aplicativo cliente para funcionamento em outros dispositivos portáteis;

5. Tornar a estrutura cliente-servidor adotada compatível com outros padrões de tecnologia sem fio;

6. Aprimoramento da troca de dados e mensagens da estrutura cliente-servidor, de modo a oferecer maior confiabilidade aos procedimentos remotos;

7. Implementação de outras estratégias de controle para a planta industrial utilizada neste trabalho.

Apêndice A

Motor de Indução alimentado por um Inversor de Tensão Trifásico

Na área de eletrônica industrial, o inversor é uma estrutura que possibilita a conversão de energia da forma contínua (CC) para alternada (CA), dando origem a conversão CC-CA entre o elemento gerador e o elemento consumidor de energia. Atualmente, as estruturas dos inversores utilizam dispositivos semicondutores (interruptores) para controle do fluxo de energia. Os conversores CC-CA que possuem o estágio de entrada do tipo fonte de tensão são denominados de inversores de tensão (VSI, do inglês *Voltage Source Inverter*). O controle do tempo de condução e de bloqueio dos dispositivos semicondutores pode ser feito de forma que o inversor controle a amplitude e frequência da tensão CA de saída. Esse controle pode ser realizado utilizando diferentes estratégias de modulação [48].

Os inversores de tensão podem ser utilizados em diversas aplicações. Além das aplicações industriais, no acionamento de motores síncronos e de indução, eles podem ser encontrados em sistemas ininterruptos de fornecimento de energia (UPS, do inglês *Uninterruptible Power Supply*), compensação de harmônicos (filtros ativos) e controle do fator de potência, entre outros.

A figura A.1 apresenta a configuração do inversor convencional de dois níveis e três braços utilizado neste experimento. O inversor é constituído por dois dispositivos semicondutores com seus respectivos diodos em antiparalelo, por braço.

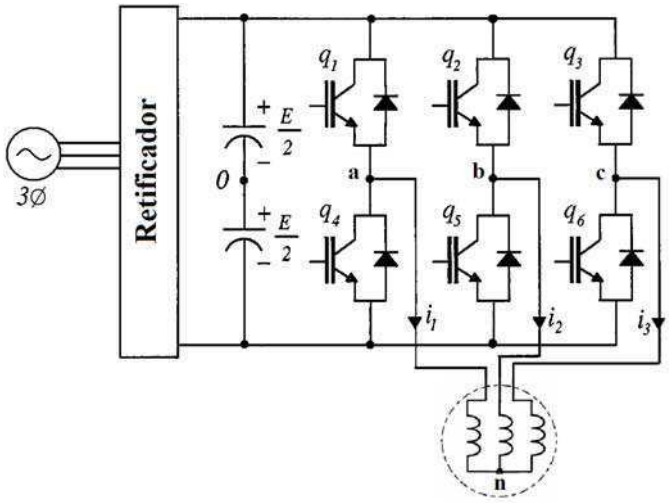


Figura A.1: Inversor de tensão trifásico de dois níveis.

Nos inversores de dois níveis a tensão em um terminal de saída, dita tensão de pólo, que é a tensão entre o terminal a, b ou c e um terminal terminal virtual 0 que divide a tensão do barramento E em duas de valor $E/2$, pode assumir apenas dois valores (figura A.2).

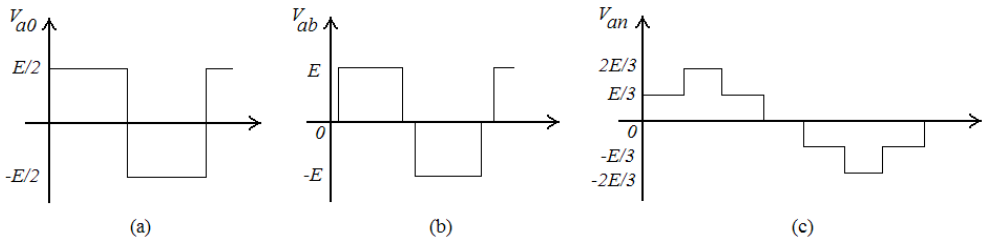


Figura A.2: Inversor de tensão trifásico de dois níveis: (a)Tensão de pólo, (b)tensão entre fases e (c)tensão entre fase e neutro da carga.

A.1 Estrutura da Plataforma Experimental Montada em Laboratório

A estrutura da plataforma experimental utilizada para a obtenção dos resultados experimentais apresentados neste trabalho encontra-se no Laboratotio de Eletrônica Industrial e Acionamento de Máquinas (LEIAM-DEE-UFCG). Na figura A.3 é apresentada a estrutura montada.

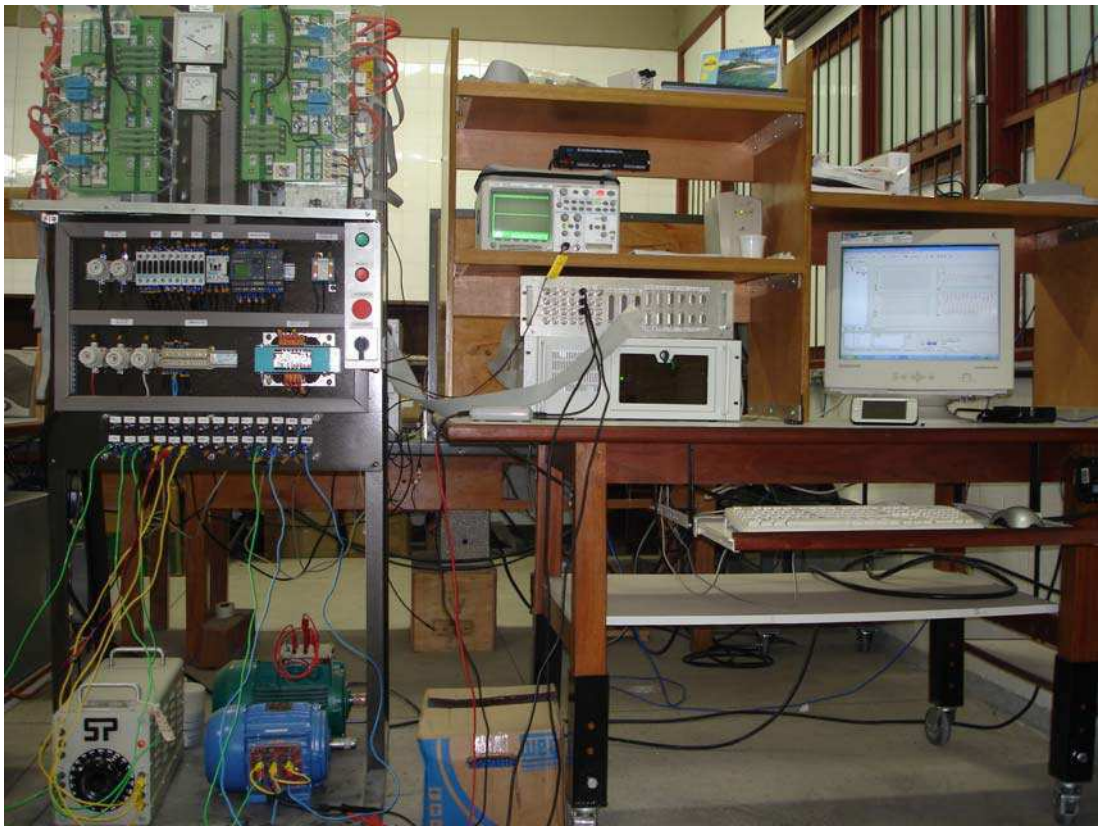


Figura A.3: Plataforma experimental.

A figura A.4 ilustra o configurador remoto da plataforma experimental e o adaptador Bluetooth-USB conectado ao PC.

Figura A.4: Configurador remoto.

A estrutura apresentada na figura A.3 é constituida pelos seguintes itens:

- Um microcomputador equipado com uma placa controladora DS1103 PPC e um pacote de *software* (MATLAB/Simulink/RTW/ControlDesk) - sistema dSPACE [19];
- Um painel de conectores da dSPACE GmbH [19];

- Três sensores, sendo dois de corrente e um sensor de tensão;
- Dois conversores estáticos de quatro braços cada um, e cada braço composto por dois interruptores IGBT e um circuito de acionamento dos interruptores (*driver* SKHI23 - Semikron);
- Placas de interface entre o painel de conectores e os circuitos de acionamento dos interruptores dos conversores;
- Um motor de indução trifásico de $1,5HP$;
- Um variador de tensão de $4,5KVA$.

A figura 2.7 apresentada no capítulo 2 ilustra as ligações entre as partes que formam a plataforma desenvolvida para o presente trabalho. A descrição detalhada de cada item mencionado acima é enumerado em seguida:

1. A placa controladora DS1103 PPC possui todos os recursos necessários para o controle da aplicação, entre eles os conversores A/D e D/A e o módulo de geração de sinais PWM (DSP escravo da placa). Estes recursos são utilizados através do painel de conectores.
2. As placas de interface entre o microcomputador e os *drivers* dos conversores foram desenvolvidas especificamente para esta plataforma. A placa de interface recebe os sinais PWM do painel de conectores.
3. É utilizado apenas um dos dois conversores estáticos, cada um com quatro capacitores de $2200uF$ que constituem o barramento capacitivo. Os circuitos de acionamento dos interruptores recebem os sinais de comando, a partir das placas de interface.

Referências Bibliográficas

[1] W. Wolf. *Computers as Components: Principles of Embedded Computing System Design*. Morgan Kaufmann, 2005.

[2] P. Marwedel. *Embedded System Design*. Kluwer Academic Publishers, 2003.

[3] B. Arnold. *Embedded System Design*. CMP Books, 2002.

[4] Semiconductor Industry Blue Book. World semiconductor trade statistics, 2003.

[5] The MathWorks Inc. Real-time workshop: User's guide, version 6, 2007.

[6] The MathWorks Inc. Matlab/simulink user's guide, Natick, MA, 1998.

[7] H. Hanselmann. Automotive control: from concept to experiment to product. *Conf. Rec. IEEE/CACSD*, pages 129–134, Sep. 1996.

[8] H. Ramamurthy, B. S. Prabhu, R. Gadh, and A. M. Madni. Wireless industrial monitoring and control using a smart sensor platform. *IEEE Sensors Journal*, 7(5):611–618, 2007.

[9] H. Ramamurthy, B. S. Prabhu, R. Gadh, and A. M. Madni. Smart sensor platform for industrial monitoring and control. *Conf. Rec. 4th IEEE Sensors*, 2005.

[10] H. Ramamurthy, D. Lal, B. S. Prabhu, and R. Gadh. Rewins: A distributed multi-rf sensor control network for industrial automation. *Conf. Rec. IEEE/CACSD*, pages 24–33, 2005.

[11] H. Ramamurthy, B. S. Prabhu, and R. Gadh. Reconfigurable wireless interface for networking sensors (rewins). *Conf. Rec. PWC*, 2004.

[12] J. Hamby. The revolution in drive configuration & diagnostic tools. *Conf. Rec. Cement Industry Technical Conference*, pages 115–123, May 2005.

[13] M. Kesler, M. Uçar, and E. Özdemir. Rapid prototyping of real time dsp based shunt active power filter using automatic embedded code generation. *Proceedings of the PCIM Europe Conference*, pages 763–767, 2006.

[14] M. Glesner, A. Kirschbaum, F. M. Renner, and B. Voss. State-of-the-art in rapid prototyping for mechatronic systems. *Proc. of 1st IFAC-Conference on Mechatronics Systems*, 2000.

[15] W. S. Gan, Y. K. Chong, W. Gong, and W. T. Tan. Rapid prototyping system for teaching real-time digital signal processing. *IEEE Trans. Educ.*, 43:19–24, Feb. 2000.

[16] S. Rebeschieβ. Mircos - microcontroller-based real time control system toolbox for use with matlab/simulink. *Conf. Rec. IEEE/CACSD*, pages 267–272, August 1999.

[17] D. Hercog and K. Jezernik. Rapid control prototyping using matlab/simulink and dsp-based motor controller. *International Journal of Engineering Education (IJEE)*, 21(4), 2005.

[18] K. H. Hong, W. S. Gan, Y. K. Chong, K. K. Chew, C. M. Lee, and T. Y. Koh. An integrated environment for rapid prototyping of dsp algorithms using and texas instruments' tms320c30. *Microprocessors and Microsystems*, 24:349–363, Nov. 2000.

[19] dSPACE GmbH. dspace user's guide, Germany, 2003.

[20] R. Safaric, M. Truntic, D. Hercog, and G. Pacnik. Control and robotics remote laboratory for engineering education. *International Journal of Online Engineering (iJOE) [Online]*, June 2005.

[21] D. Hercog, B. Gergic, and V. Matko. Remote lab for electric drives. *Conf. Rec. IEEE/ISIE*, 4:1685–1690, June 2005.

[22] D. Hercog, B. Gergiè, S. Uran, and K. Jezernik. A dsp-based remote control laboratory. *IEEE*, 54(6), December 2007.

[23] R. U. Garbus, I. J. O. Aguirre, R. C. Sánchez, and O. R. Pureco. Virtual remote lab for control practice. *Conf. Rec. CERMA)*, pages 361–366, 2006.

[24] A. Laksmikanth and M. M. Morcos. A power quality monitoring system: a case study in dsp-based solutions for power electronics. *IEEE Transactions on Instrumentation and Measurements*, 50:724–731, 2003.

[25] J. Jacobs, D. Detjen, C. U. Karipidis, and R. W. De Doncker. Rapid prototyping tools for power electronic systems: Demonstation with shunt active power filters. *IEEE Trans. on Power Electronics*, 19(2), March 2004.

[26] R. A. M. Braga, F. B. Líbano, and F. A. B. Lemos. Development environment for control strategies of hybrid active power filters using matlab and dspace dsp. *IEEE Porto Power Tech'2001*, 2001.

[27] W. Lee, M. Shin, and M. Sunwoo. Target-identical rapid control prototyping platform for model-based engine control. *Proceedings of Institution of Mechanical Engineers*, 218:755–765, July 2004.

[28] A. Rubaai, A. Ofoli, and M. Castro. dspace dsp-based rapid prototyping of fuzzy pid controls for high performance brushless servo drives. *Proc. of the 41st IAS Annual Meeting*, 3:1360–1364, Oct 2006.

[29] L. Qiao and W. Jie. Implementation of a new nonlinear controller for dc-dc converter using matlab and dspace dsp. *Conf. Rec. IEEE/ISIE*, 2:621–625, June 2005.

[30] W. Grega, K. Kolek, and A. Turnau. Rapid prototyping environment for real-time control education. *Conf. Rec. IEEE/RTEW*, pages 85–92, Nov. 1998.

[31] D. Hercog, M. Curkovic, and K. Jezernik. Dsp based rapid control prototyping systems for engineering education and research. *Conf. Rec. IEEE/ISIC*, pages 2292–2297, Oct. 2006.

[32] R. Saco, E. Pires, and C. Godfrid. Real-time controlled laboratory plant for control education. *Conf. Rec. FIE'02*, 1:T2D12–16, 2002.

[33] Bluetooth Special Interet Group. Specification of the bluetooth system - version 1.2, Nov. 2003. Disponível em http://www.bluetooth.com.

[34] dSPACE GmbH. Ds1103 instalation and configuration guide release 4.0, Germany, August 2003.

[35] dSPACE GmbH. Ds1103 hardware reference release 4.0, Germany, August 2003.

[36] dSPACE GmbH. Controldesk experiment guide release 4.0, Germany, August 2003.

[37] dSPACE GmbH. Real-time interface (rti) implementation guide release 4.0, Germany, August 2003.

[38] dSPACE GmbH. Real-time library (rtlib) reference release 4.0, Germany, August 2003.

[39] G. van Rossum. Python tutorial, May 1995. Disponível em: http://www.python.org.

[40] A. Berson. *Client-server architecture*. McGraw-Hill, 1994.

[41] Nokia Corporation. Nokia n800 internet tablet, features. Disponível em: http://www.nokiausa.com/N800.

[42] Object Management Group OMG. Unified modeling language specification, version 1.5, March 2003. Disponível em: http://www.rational.com/.

[43] The MathWorks Inc. Simulink, dynamic system simulation for matlab - using simulink, version 3, January 1999.

[44] W3C World Wide Web Consortium. Extensible markup language - xml 1.0, 3^a edição, Fevereiro 2004.

[45] The Eclipse Foundation. Eclipse ide, version 3.2, 2007. Disponível em: http://www.eclipse.org/.

[46] A. Huang. The use of bluetooth in linux and location aware computing. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, 2005.

[47] J. Holtz. Pulse width modulation for electronic power conversion. *IEEE*, 82(8):1194–1214, 1994.

[48] J. A. Pomílio. Técnicas de modulação de potência, 2007. Disponível em: http://www.dsce.fee.unicamp.br/ antenor/elpot.html.