UNIVERSIDADE FEDERAL DE OURO PRETO

ESCOLA DE MINAS

COLEGIADO DO CURSO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO - CECAU

LUDMILA PAOLA PEREIRA IWASAKI

DESENVOLVIMENTO E IMPLEMENTAÇÃO DE UM SISTEMA DE TANQUES ACOPLADOS

MONOGRAFIA DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Ouro Preto, Agosto de 2014

LUDMILA PAOLA PEREIRA IWASAKI

DESENVOLVIMENTO E IMPLEMETAÇÃO DE UM SISTEMA DE TANQUES ACOPLADOS

Monografia apresentada ao Curso de Engenharia de Controle e Automação da Universidade Federal de Ouro Preto como parte dos requisitos para a obtenção do Grau de Engenheiro de Controle e Automação.

Orientador: M.sc. José Alberto Naves Cocota Junior

Ouro Preto
Escola de Minas – UFOP
Agosto/2014

Monografia defendida e aprovada, em 21 de agosto de 2014, pela comissão avaliadora constituída pelos professores:

Profa. MSc. José Alberto Naves Cocota Júnior - Orientador

Prof. MSc. João Carlos Vilela de Castro – Professor Convidado

Prof. Prof. Dr. Alan Kardek Rêgo Segundo – Professor Convidado

AGRADECIMENTO

Agradeço a toda minha família pelo apoio, em especial minha mãe Célia por me ensinar a lutar e me apoiar em minhas decisões.

Aos meus amigos que me acompanharam e me motivaram. Em especial a República Virada pra Lua que me proporcionou um lar e uma família em Ouro Preto.

Ao orientador e mestre Cocota por esse ano de paciência e ensinamentos.

A Escola de Minas pelo ensino de qualidade, formando engenheiros de alto nível. A Fundação Gorceix pelo assistencialismo e subsidio ao projeto.

RESUMO

Nos últimos tempos o interesse no estudo de controle de sistemas vem crescendo. Isso se deve à vasta utilização deste tipo de sistema para estudos na academia ou em setores industriais, onde muitos dos sistemas apresentam diversas variáveis a controlar simultaneamente, tendo respostas aos estímulos de distintos sinais de entrada. Este trabalho apresenta o desenvolvimento de uma bancada didática com tanques acoplados para o estudo de teoria de controle através do controle dos níveis dos tanques. No qual trabalhos práticos são introduzidos ao discente para facilitar o aprendizado em controle de processos. Esses experimentos práticos irão complementar a formação como engenheiro, acrescentando habilidades no âmbito profissional. Para esse projeto foi elaborado um dispositivo utilizando um sensor de pressão para leitura do nível dos tanques. A motobomba é acionada através de um módulo de potência também desenvolvido para esse projeto, além disso, uma programação em LADDER para o controle utilizando um CLP Versamax® GE Fanuc foi implementada. No MATLAB® foram realizadas a modelagem do sistema e as simulações. Por fim, um sistema supervisório e especialista utilizando GUIDE do MATLAB® foi desenvolvido.

Palavras chaves: Tanques acoplados, Nível, Motor CC, MATLAB, OPC, CLP, Supervisório, Especialista.

ABSTRACT

Lately the interest in the study of control systems is growing. This is due to widespread use for studies in academia or in many industrial sectors, where many systems have several variables to be controlled simultaneously that responses to impulse of different input signals. This paper presents the development of a didactic bench with coupled tanks for the study of control theory by controlling the levels of the tanks. Pratical works are introduced to students to facilitate learning in control systems. These experiments will complement the practical training as an engineer, adding skills in a professional context. For this project a device was made using a pressure sensor for reading level of the tanks. The pump is driven through a power module also developed for this project, furthermore, a schedule in LADDER for control using a GE Fanuc Versamax® PLC was implemented. In MATLAB® system modeling and simulations were conducted.Finally, a supervisor and wizard system using the tool GUIDE in MATLAB® was developed.

Key word: coupled tanks, level, motor DC, MATLAB, OPC, PLC, Supervisory System, Wizard System.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIAÇÕES E SIGLAS

CA	Corrente Alternada
CC	Corrente Continua
CLP	Controlador Lógico Programável
DECAT	Departamento de Engenharia de Controle e Automação e Técnicas Fundamentais
GUIDE	*Graphical User Interface Development Environment*
IHM	Interface Homem-Máquina
IP	*Internet Protocol*
LCD	*Liquid Crystal Display*
MIMO	*Multiple input, Multiple Output*
OPC	*Object linking and embedding for Process Control*
OPC DA	*Data Access*
OPC HDA	*Historical Data Access*
PIC	*Programmable Interface Controller*
PCI	Placa de Circuito Impresso
PWM	*Pulse Widht Modulation*
RISC	*Reduced Instruction Set Computer*
SCADA	*Supervisory Control and Data Acquisition*
SISO	*Single Input, Single Output*
UFOP	Universidade Federal de Ouro Preto

LISTA DE SÍMBOLOS

δ	densidade do material
g	aceleração da gravidade
h	altura
k_i	ganho da bomba
k_{1j}	perdas de carga
Pa	Pascal
V	Volts
β	ganho de corrente
γ	fluxo da bomba
Ω	resistência

SUMÁRIO

1. INTRODUÇÃO

O controle do nível de líquido em tanque e de fluxo entre tanques é um problema básico nos processos industriais. Tais processos requerem líquidos para serem bombeados, armazenados em tanques e, então, bombeados para outro tanque (LAUBWALD, 2005).

Muitos pesquisadores têm investigado o problema de controle do fluxo de um líquido em um único tanque ou múltiplos tanques (KHAN e SPURGEN, 2006 apud ALMUTAIRI e ZRIBI, 2006). Esse interesse deve-se ao uso de tanques em diversos processos nas indústrias, tendo comumente como necessidade o controle do nível de líquidos. Principalmente nas indústrias petroquímicas, indústrias de papel e de tratamento de águas, as quais precisam controlar o nível de fluido (CHRISTY e KUMAR, 2014).

Dada essa demanda, é importante ressaltar a necessidade de uma complementação no ensino de sistemas de controle. Na área acadêmica, o ensino da engenharia de controle de sistemas se tornou uma importante e integral parte do currículo da engenharia nas disciplinas de elétrica, mecânica, química e aeronáutica (KHEIR, *et al.*, 1996).

A conexão entre teoria e prática é uma das características mais difíceis para ensinar engenharia. Esse problema é particularmente acentuado em controle automático, no qual geralmente há um alto nível de abstração (ASTRÖM e ÖSTBERG, 1986). Por mais que as empresas tenham cada vez mais um interesse na área de controle, ainda há essa defasagem no ensino da teoria de controle, que é a conciliação da teoria com a prática. Isto pode acarretar a evasão de diversos discentes diante dessa dificuldade em aprender.

Experimentos de laboratórios oferecem uma forma de introduzir mais realismo na educação de controle automático (ASTRÖM e ÖSTBERG, 1986). Uma maneira descrita por Aström (1986) para se diminuir esse vácuo no ensino, é a criação de um laboratório de ensino para controle de processos. Um sistema com dois tanques acoplados em cascata, uma bomba e um reservatório foi montado para a realização de trabalhos práticos. De forma que diversos projetos pudessem ser realizados, incluindo experimentos empíricos com controle PI e PID, modelagem e ajuste de parâmetros, projeto, implementação e afinação de controle PID, *antiwindup*, autoajuste, controle seletor, realimentação de estados, filtro de Kalman e realimentação da saída (ASTRÖM e ÖSTBERG, 1986). Esse sistema fez com que os discentes passassem mais tempo no laboratório realizando experimentos na área de controle.

A sincronização dos experimentos do laboratório com as aulas e os períodos de resolução de problemas é pedagogicamente boa (ASTRÖM e ÖSTBERG, 1986). Geralmente,

as práticas aumentam o interesse pela teoria, além de fazer com que o discente desenvolva a habilidade de resolver problemas, que no âmbito profissional terá que fazê-lo diariamente. Esta foi a principal motivação para o desenvolvimento deste trabalho, que foca a montagem de uma bancada didática para uso dos discentes na aprendizagem de teoria de controle.

1.1 Objetivos

1.1.1 Objetivos Gerais

Este trabalho propõe o desenvolvimento de uma bancada didática para ser utilizada nas disciplinas de Teoria de Controle oferecidas pelo Departamento de Engenharia de Controle e Automação e Técnicas Fundamentais (DECAT), De forma a complementar o ensino dessas disciplinas para os discentes do curso, além de completar a formação, incentivando o desenvolvimento educacional e profissional.

1.1.2 Objetivos Específicos

O desenvolvimento do projeto tem como objetivos específicos:

- Projetar os elementos mecânicos da bancada do sistema de quatro tanques acoplados;
- Projetar os circuitos eletrônicos para o acionamento das moto-bombas e amostragem dos níveis dos tanques;
- Desenvolver um sistema de tanques acoplados com arquitetura aberta.

1.2 Apresentação da Planta

A planta (Figura 1) é um sistema de tanques cilíndricos e um reservatório, desenvolvida no Laboratório de Máquinas Elétricas, Interfaceamento e Gestão da Qualidade de Energia, da Escola de Minas – UFOP. A finalidade do reservatório é armazenar a água utilizada de todo o processo.

Experimentos de controle SISO *(Single input – Single Output/* Única entrada - Única Saida) e MIMO *(Multiple input- Multiple output/* Multiplas entrada – Multiplas Saídas) são possíveis de serem realizados na planta. Para isso o usuário deve fazer o acoplamento ou não dos tanques da maneira desejada para o estudo, as Figura 2 e Figura 3 ilustram as duas configurações.

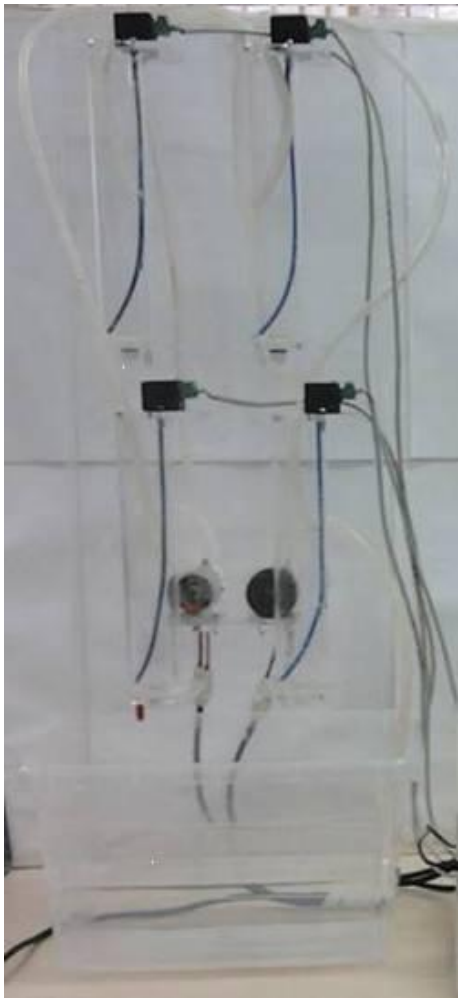


Figura 1 - Planta com 4 tanques

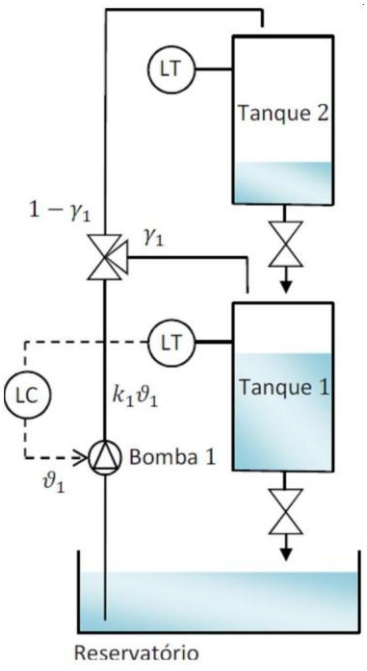


Figura 2 - Sistema com 2 tanques configurado em SISO

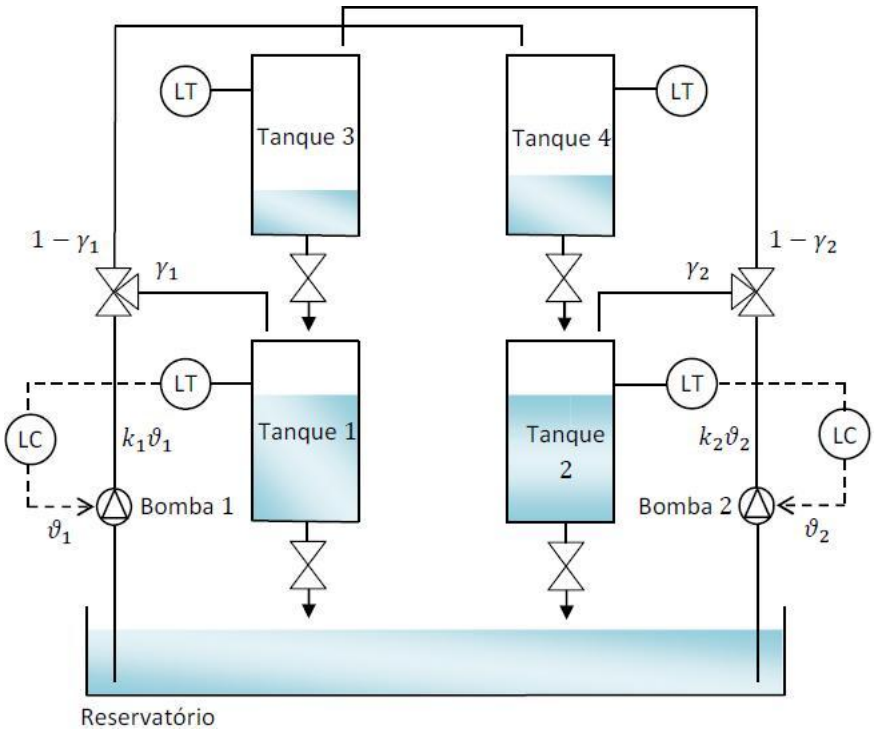


Figura 3 - Sistema com 4 tanques configurado em MIMO

Para transmitir a medida do nível do tanque foi desenvolvido um condicionador de sinais especificamente para este trabalho. O nível é medido através de um sensor de pressão diferencial cujo sinal é amplificado e transmitido.

O comportamento da planta pode ser caracterizado como um sistema de fase mínima ou não mínima, configurando as parcelas de fluxo para os tanques e o diâmetro do dreno de descarga. Um sistema de fase mínima possui função de transferência com zeros apenas no semipleno esquerdo do plano s, assim a função de transferência é estável. Em contrapartida um sistema de fase não mínima possui um zero ou mais no semiplano direito, apresentando um sistema instável.

Cada tanque (Figura 4) possui um tampão (Figura 5), que proporciona o fechamento do mesmo para que os ensaios possam ser realizados, ou ainda, trocar os diâmetros dos drenos de cada tanque pode ser feita, de forma a se ter uma variedade de experimentos.

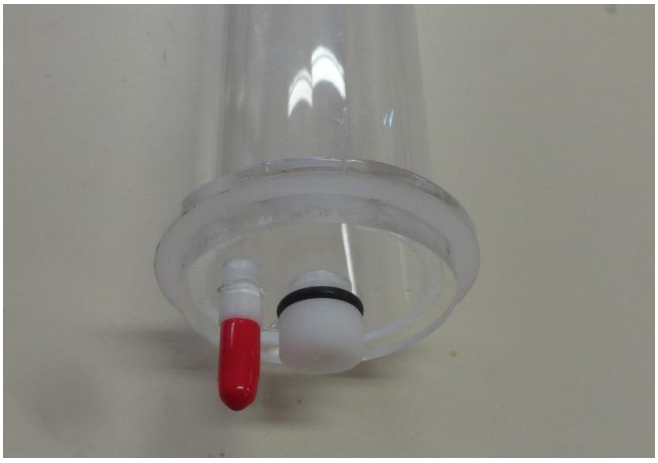


Figura 4 - Tanque

Figura 5 - Tampão Dreno

Uma bomba de engrenagem é utilizada para o controle do fluxo de água enviado aos tanques. Um módulo de potência aciona a bomba por meio de um sinal modulado por largura de pulso (PWM). O sinal de controle para esse acionamento provém de um controlador lógico programável (CLP) Versamax ®da GE Fanuc (Figura 6). O módulo de potência, o PLC e as fontes que alimentam os circuitos são fixados em um único painel de acrílico (Figura 7).

Figura 6 - PLC GE Fanuc

O PLC é conectado a um computador (PC) por meio de um cabo ethernet. Um computador com o software Proficy® Machine é utilizado para a programação do CLP

Versamax®. Além disso, são utilizados outros dois programas: o MATLAB® e o KEPServerEX®. O KEPServerEX® é o *software* utilizado para fazer a comunicação OPC do MATLAB® com o CLP. O MATLAB® é um *software* de alto nível que possui diversas ferramentas, entre elas o SIMULINK. O MATLAB® foi utilizado para o desenvolvimento do controlador, podendo plotar os gráficos desejados para as retiradas dos parâmetros necessários e o SIMULINK foi utilizado para a modelagem e simulação do sistema.

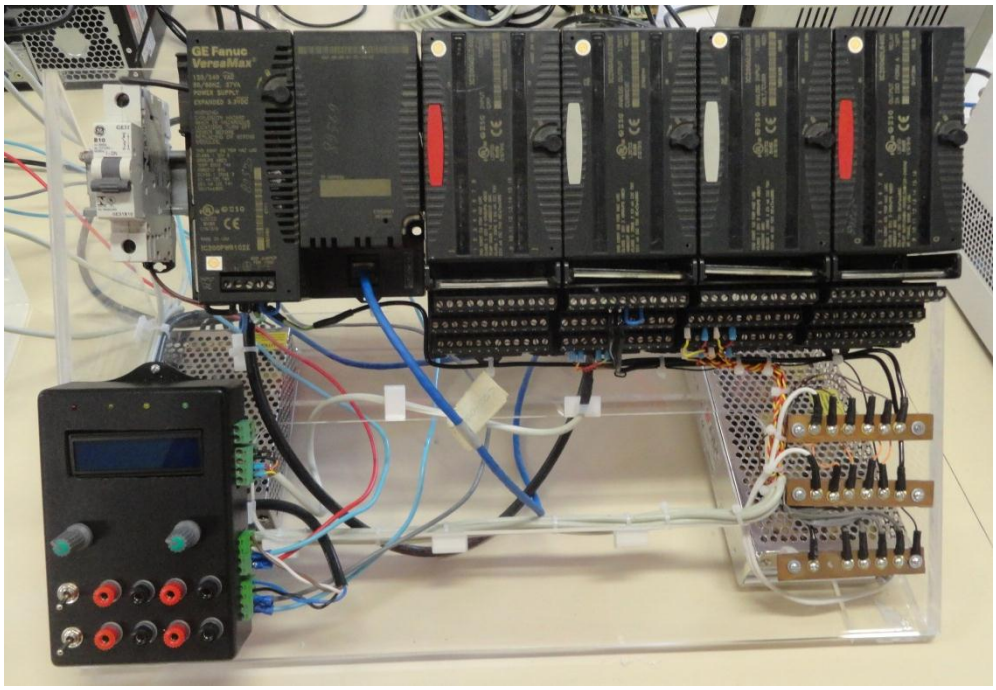


Figura 7 - Painel Acrílico para PLC, Módulo e Fontes.

1.3 Metodologia

Para o desenvolvimento deste trabalho foi adotada a seguinte metodologia: projeto para amostragem da coluna de água, projeto para acionamento do motor, programa CLP e interface com o usuário (em MATLAB®).

Para a realização da medição dos níveis dos tanques utilizando um sensor diferencial de pressão foi necessário o desenvolvimento de um amplificador e transdutor de pressão. Circuitos foram desenvolvidos para fazer a leitura dos tanques, sendo necessário fazer a calibração dos sensores, obtendo uma saída de 0 a 10V.

O acionamento do motor é feito através de um módulo de potência desenvolvido no laboratório. Primeiramente, desenvolveu-se a placa de circuito impresso, logo, a programação

em C com a técnica de modulação por largura de pulso é inserida no PIC16F867. Com isso, uma caixa foi moldada para proteger o módulo e melhor atender o usuário.

Utilizando o software *Proficy Machine*, uma programação em ladder foi feita viabilizando parametrizar o controle do sistema de tanques acoplador e criar uma programação que permita a elaboração de um sistema supervisório e especialista.

Após a montagem da bancada, foi desenvolvido um sistema especialista e supervisório com o intuito de promover a interação do usuário com o sistema. Esse sistema foi elaborado utilizando a ferramenta GUIDE (*Graphical User Interface Development Enviroment)* do *software* MATLAB ®. O usuário tem a opção de supervisionar o sistema, além de ter uma orientação para a obtenção dos parâmetros dos tanques através do sistema especialista.

1.4 Conteúdo dos capítulos

O capítulo 1 introduz o assunto do trabalho, juntamente com a justificativa, objetivos, descrição física da planta e a metodologia utilizada no desenvolvimento.

O capítulo 2 expõe conceitos sobre medida de nível e algumas tecnologias utilizadas para fazer a medição de nível. Citam-se os critérios para a escolha do dispositivo adequado e explana sobre o circuito desenvolvido para fazer a medida do nível dos tanques.

O capítulo 3 trata do acionamento de motor de corrente contínua (CC), iniciando com conceitos e funcionamento de motores CC. Descreve o circuito desenvolvido para o acionamento do motor, e o código da programação do microcontrolador utilizado.

O capítulo 4 relata o funcionamento da rede, descrevendo cada *software* utilizado e os laços de comunicação entre eles.

Por fim, o capítulo 5 apresenta o sistema supervisório e especialista desenvolvido em conjunto com o discente Pedro Henrique Lopes Faria.

2. MEDIÇÃO DE NÍVEL

A medição de nível, embora tenha conceituação simples, requer por vezes artifícios e técnicas apuradas (VIANA, 1999). O nível contido dentro de um recipiente é uma variável que tem grande interesse industrial e que, por conseguinte, geralmente é muito habitual em plantas e processos de fabricação e armazenamento (GARCIA *et al.*, 2004).

Para facilitar a compreensão, costuma-se definir nível como sendo a altura do conteúdo de um reservatório, que poderá ser um líquido ou um sólido (VIANA, 1999). A medida de nível nas indústrias é feita tanto de líquidos como de sólidos. A informação sobre a quantidade de material que há no interior de um recipiente é o propósito da medida de nível.

O nível não é a variável direta de saída quando o interesse é medir o volume ou a massa de um material dentro de um recipiente. Os parâmetros geométricos do recipiente e as características de densidade do material precisam ser considerados.

A característica do material para fazer a medição do nível pode ser tanto líquida quanto sólida. A diferença mais importante entre ambas as situações está em que a superfície livre do material pode ter formas diferentes (GARCIA *et al.*, 2004). A Figura 8 mostra três situações de superficies livres em um depósito. A situação da Figura 8 (a) trata-se de líquidos estáticos, ou seja, o líquido está em repouso e sua superficie é plana em relação a superfície da terra. Quando o tipo de material utilizado é massa sólida, observa-se na Figura 8 (b) que pode haver uma distribuição na qual algum ponto fique mais alto dependendo de onde é feita a entrada ou saída do material. Além disso, pode-se ter uma superficie variante quando o fluido que se encontra em agitação, como ilustrado na Figura 8 (c). Esses casos precisam ser avaliados para se fazer a melhor escolha do sensor de nível e onde será instalado.

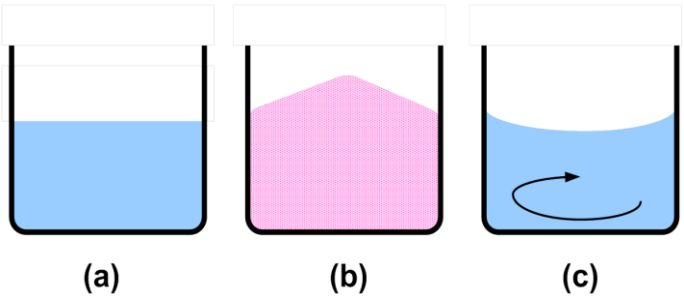


Figura 8 - Superfícies livres em um depósito

Fonte: GARCIA *et al.*, 2004, p. 513.

2.1 Tecnologias para medida de nível

Existem dois métodos básicos que classificam os sistemas de medição de nível, a medida indireta e a direta. Esses sistemas possuem uma grande variedade, desde simples visores para leituras locais até indicação remota, registro ou controle automático.

De acordo com Viana (1999), a medida indireta é o tipo de medição que se faz para determinar o nível em função de uma segunda variável. Exemplos desse tipo de medidor são: *displacer* (empuxo), pressão diferencial, borbulhador, capacitância, ultrassônico, por pesagem e por raio gama.

A medida direta é a medição que se faz tendo como referência a posição do plano superior da substância medida (VIANA, 1999). Ou seja, a medida pode ser feita medindo diretamente a diferença entre o nível no qual a substância se encontra e o um referencial definido previamente. As réguas ou gabaritos, os visores de nível e a boia são alguns dos medidores diretos mais utilizados na indústria.

Além disso, a medida de nível pode ser classificada como discreta ou continua. Na medida discreta, os níveis são relatados quando os valores predeterminados são atingidos. No geral, a medida discreta é empregada quando se pretende apenas incluir um alarme (Figura 9), podendo ser um alarme de sobrecarga, de tanque vazio ou de um determinado nível. A medida continua (Figura 10) proporciona uma saída proporcional ao nível do tanque (GARCIA *et al.*, 2004). Geralmente, esse tipo de medida é necessário quando o nível faz parte de controle de processo.

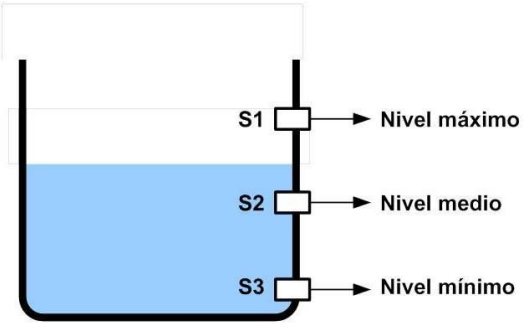


Figura 9 - Medida Discreta

Fonte: GARCIA et al., 2004, p.515

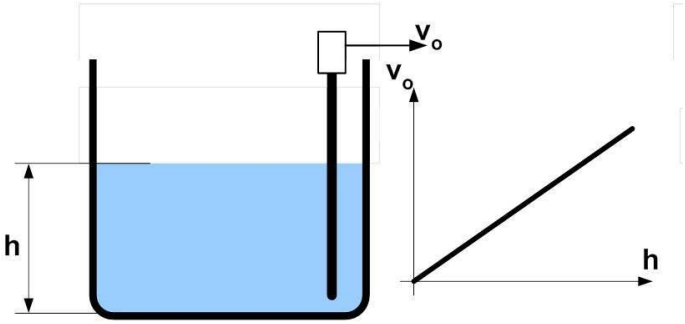


Figura 10 - Medida Continua

Fonte: GARCIA et al., 2004, p.515

É possível encontrar uma diversidade de instrumentos para fazer a medição de nível, sendo mais utilizados os interruptores e os medidores de nível. Os interruptores de nível funcionam a partir de um determinado nível predeterminado. E eles proporcionam uma saída digital ou um determinado valor de tensão assim que o nível requerido é alcançado. Os medidores de nível informam o valor do nível do material em um depósito, realizam essa medição através da saída de tensão ou corrente que informam o nível real do material.

2.1.1 Medidor de nível tipo régua ou gabarito

Este medidor realiza medição direta e possui um funcionamento simples. O instrumento é uma régua graduada com comprimento suficiente para se colocar dentro do recipiente (Figura 11). A medida é realizada através da leitura direta do material na régua. É um sensor de baixo custo e que permite leituras instantâneas

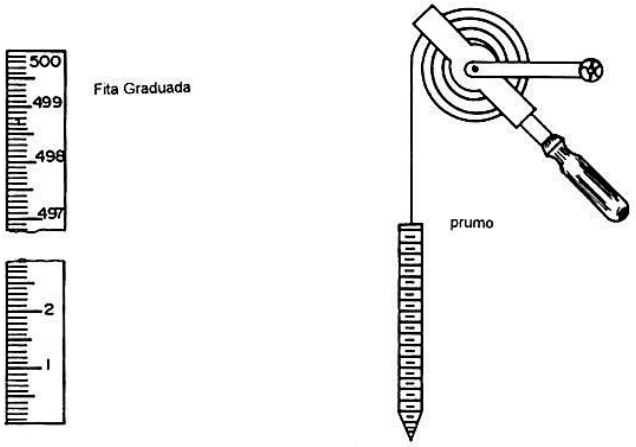


Figura 11 - Medidor tipo Régua ou Gabarito

Fonte: VIANA, 1999, p. 83.

2.1.2 Sensor de nível tipo Flutuador

Esse tipo de medidor é mais comumente conhecido como boia. Na Figura 12, dois exemplos de boias comercializadas são ilustrados. Esse tipo de boia é muito utilizado como chave de nível em sistemas de automação industrial.

Figura 12 - Flutuador
Fonte: Página da Internet[1]

A Figura 13 mostra o funcionamento básico de um flutuador. O dispositivo é colocado dentro do tanque, devido a composição do seu material, ele fica flutuando na superfície do fluido. O movimento vertical da alavanca da boia produz uma saída em forma de tensão que será a referência para a indicação.

Esse tipo de medidor é utilizado geralmente em tanques abertos ou tanques fechados não pressurizados. Além disso, possuem uma vida útil limitada, o que faz com que haja necessidade de uma constante manutenção.

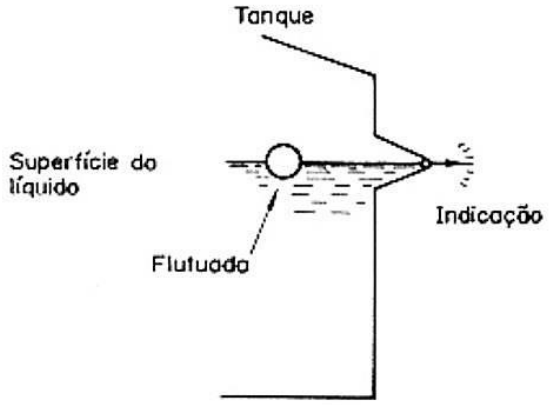


Figura 13 - Atuação de um flutuador
Fonte: VIANA, 1999, p.92.

[1] Disponivel em: <http://portuguese.alibaba.com/product-gs-img/merc-rio-interruptor-de-b-ia-518470505.html> Acesso em 13 jul 2014.

2.1.3 Sensor tipo Capacitivo

Um capacitor consiste de dois condutores, denominados placas, separados por um material dielétrico. Este componente, muito utilizado em circuitos elétricos, tem como principal característica a propriedade de armazenar cargas elétricas. A grandeza que caracteriza um capacitor é a capacitância, expressa em farad (VIANA, 1999).

O sensor do tipo capacitivo utiliza o princípio de um capacitor, ou seja, a medição é feita através da permissividade dielétrica entre o fluido e a atmosfera. A Figura 14 mostra um sensor capacitivo, que consiste em um condensador cilíndrico inserido verticalmente no tanque. O dispositivo funciona como uma das placas do capacitor, a outra placa é formada pelas paredes do depósito e o fluido se comporta como o material dielétrico. Ao variar o nível no líquido, as proporções entre o líquido e o vapor são alteradas. Dada que a constante dielétrica da maioria dos líquidos é maior que a dos vapores, as variações de nível são proporcionais à variação da capacitância.

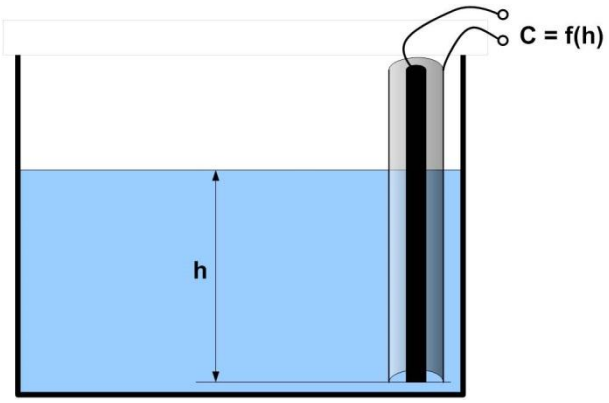


Figura 14 - Sensor de nível tipo capacitivo

Fonte: GARCIA *et al.*, 2004, p. 518.

Este tipo de sensor é baseado no valor da permissividade dielétrica, caso esse valor seja constante, não é possível fazer a medição. É necessário que a medida seja feita em líquidos limpos e com baixas tensões superficiais para que se tenham valores precisos e corretos.

2.1.4 Medição de nível por pressão diferencial

Conceitualmente, o Teorema de *Stevin*, desenvolvido pelo engenheiro, físico e matemático Simon Stevin, que nasceu na Bélgica, descreve que a pressão desenvolvida por

um fluido depende exclusivamente da sua altura, dando assim, uma explicação para o chamado paradoxo hidrostático (ou pressão hidrostática) (SILVA, 2008). A pressão hidrostática (Figura 15) é dada pelo produto do peso específico pela gravidade e altura da coluna desse líquido ($P = \delta . g . h$). A pressão não é a mesma em todos os pontos de um fluido qualquer, porém se o fluido for homogêneo e estiver em repouso, a pressão será a mesma para qualquer ponto.

A medição de nível utilizando sensor de pressão segue o mesmo princípio básico descrito anteriormente, ou seja, o instrumento mede o nível através da diferença das pressões ocasionada pela coluna do líquido dentro de um recipiente. A utilização desse tipo de sensor só pode ser feita em fluidos não compressíveis.

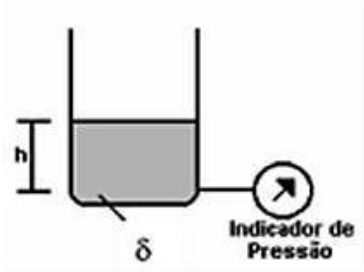


Figura 15 - Definição pressão hidrostática

Fonte: SILVA, 2008, p.77.

Esse tipo de sensor apresenta características vantajosas quanto a robustez e compatibilidade com diversos meios, já que os sensores de pressão podem suportar diversas condições de trabalho sem problema (GARCIA *et al.*, 2004). A Figura 16 mostra um caso simples do cálculo do nível de um liquido mediante sensores de pressão em um tanque, no qual a densidade do líquido e a pressão da atmosfera do recipiente são conhecidas.

Observa-se na Figura 17 a relação entre a pressão hidrostática e o nível do tanque, ou seja, a relação entre o nível e a pressão na atmosfera do tanque não tem que ser constante ou igual à pressão atmosférica. Logo, inicialmente é necessário medir a pressão na parte superior do tanque. Já na Figura 18 a pressão em um ponto inferior do tanque depende do nível da coluna do líquido e da densidade do líquido dentro do tanque. Caso a densidade não seja constante, adiciona-se um terceiro sensor (S3) a uma altura predeterminada do sensor inferior (S1). A diferença de pressão entre esses sensores (S1e S3), quando estão submersos no líquido, permite que o cálculo da densidade da substância seja feito. Obtendo a densidade e,

com a pressão diferencial entre o sensor localizado na atmosfera do tanque (S2) e o sensor da parte inferior (S1), é possível calcular o nível do tanque.

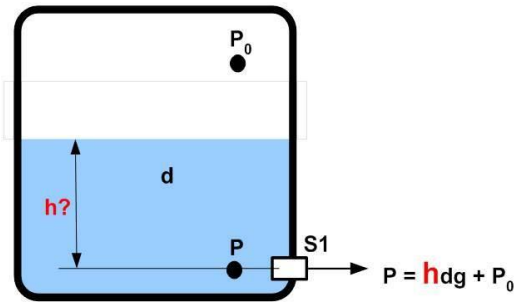


Figura 16 - Calculo do nível com sensor de pressão conhecendo a densidade

Fonte: GARCIA *et al.*, 2004, p.517.

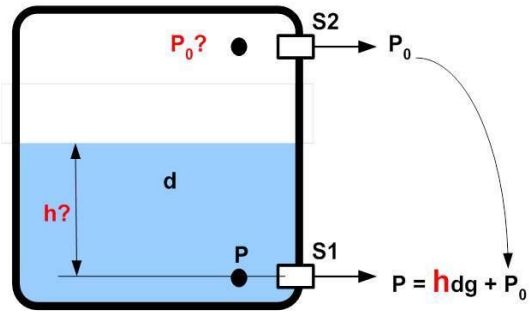


Figura 17 - Calculo do nível sem conhecimento da pressão da atmosfera

Fonte: GARCIA *et al*., 2004, p.517

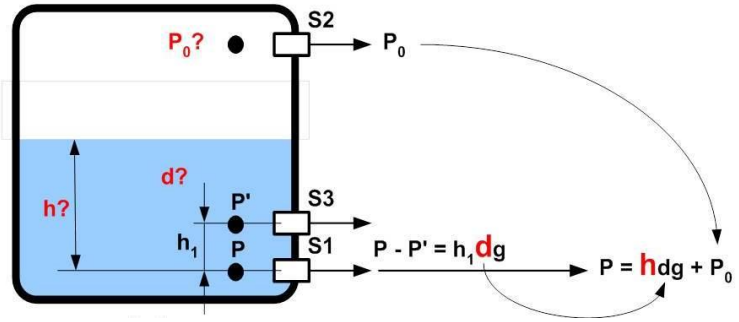


Figura 18 - Calculo do nível com densidade não constante

Fonte: GARCIA et al., 2004, p.517

Um dos princípios de funcionamento dos transmissores de pressão diferencial é por diafragma (Figura 19). Este princípio é baseado no equilíbrio de forças, as pressões que definem um dado diferencial são aplicadas através das conexões de entrada do instrumento a

duas câmaras situadas em lados opostos, estanques entre si e separadas por um elemento sensível (diafragma). Estas pressões, atuando sobre o elemento com uma superfície determinada, produzem forças de mesma direção e sentidos opostos, fazendo originar uma força resultante (VIANA, 1999). Esta força resultante provoca uma variação proporcional a pressão diferencial que será a referência para um sinal de saída em corrente.

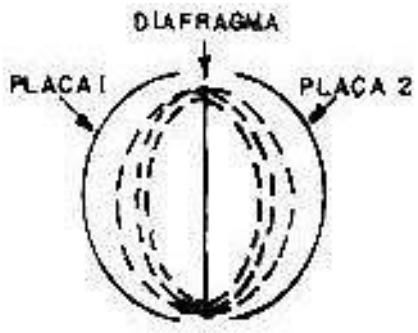


Figura 19 - Esquema do funcionamento por diafragma

Fonte: VIANA,1999, p.99.

2.1.5 Seleção do sensor de nível

No processo para selecionar o medidor de nível adequado deve-se levar em consideração algumas características. De acordo com Garcia (2004), os critérios básicos de seleção de um medidor são:

1. A possibilidade de introduzir o sensor dentro do tanque; caso exista essa possibilidade, precisa-se levar em consideração quando houver necessidade de retirá-lo para manutenção, substituição ou recalibração;
2. A possibilidade de colocar o sensor mergulhado no material ou na atmosfera em que está situada, e determinar a compatibilidade química e física do produto;
3. O material a ser medido mantém a densidade constante; isso pode alterar a pressão hidrostática, variável utilizada em alguns sensores;
4. E, qual tipo de medida realizar: continua ou discreta.

Na escolha do dispositivo para medição de nível é importante observar todas as circunstancias, tais como o estado da matéria, a temperatura de trabalho, a pressão, a presença de agitadores no interior do tanque, a manutenção, ou até mesmo o custo.

O sensor para este trabalho foi selecionado de acordo com os critérios descritos, tal que sensor por pressão diferencial satisfaz as necessidades do projeto. A água é o fluido que foi inserido no tanque, logo, a densidade do fluido é conhecida e constante, isso possibilita a utilização de um único sensor de pressão. No desenvolvimento de um controlador o volume do fluido precisa ser levado em consideração, caso um sensor seja introduzido dentro do tanque, poderia causar alguma alteração na leitura do volume do liquido, assim o sensor não pode ser inserido dentro do tanque, logo não pode submergi-lo no fluido. Dessa forma, necessita-se de um uma medida contínua, para obter uma relação entre o nível do tanque e a tensão de saída, e adquirir a valor do nível. O sensor de nível de pressão diferencial além de atender as exigências não possui um custo elevado, tornando-o viável.

2.2 Circuito para amostragem indireta de nível por pressão

Para este projeto foi desenvolvido um circuito para amostrar o nível dos tanques. Para medir o nível foi utilizado o sensor de pressão MPXM2010GS, que é um sensor piezo-resistivo capaz de proporcionar uma saída de tensão diretamente proporcional a pressão com alta precisão e de forma linear. Sua saída linear varia de 0 a 10kPa.

Esse sensor possui um diafragma de silício e um medidor de tensão, juntamente com uma rede de resistência de película fina integrada ao chip. O sensor de silício piezo-resistivo é o mais apropriado em aplicações com picos de pressão. A faixa dos transmissores baseados na tecnologia de filme fino garante precisão excepcional, tempo de resposta rápido e estabilidade de longo prazo. Outros fatores relevantes são seu baixo custo e dimensões reduzidas (COCOTA, 2009).

A altura do nível de cada tanque varia de 0 a 24 cm, que proporciona uma saída de 0 a 10V e uma variação de 0 a 4kPa. O sensor possui um encapsulamento MPAK, possibilitando que o um tubo seja colocado em sua extremidade, isolando o sensor de contato com a água.

A amostragem no nível no CLP é feita através da amplificação da saída do sensor. Um circuito foi desenvolvido para amplificar essa saída. A Figura 20 ilustra a PCI (Placa de circuito impresso) do transdutor de pressão e a Figura 21, o circuito do mesmo, no qual o amplificador INA126PA faz o condicionamento da saída e os potenciômetros são responsáveis pela calibração do sensor. O transdutor é alimentado por uma fonte de 24V (Figura 22).

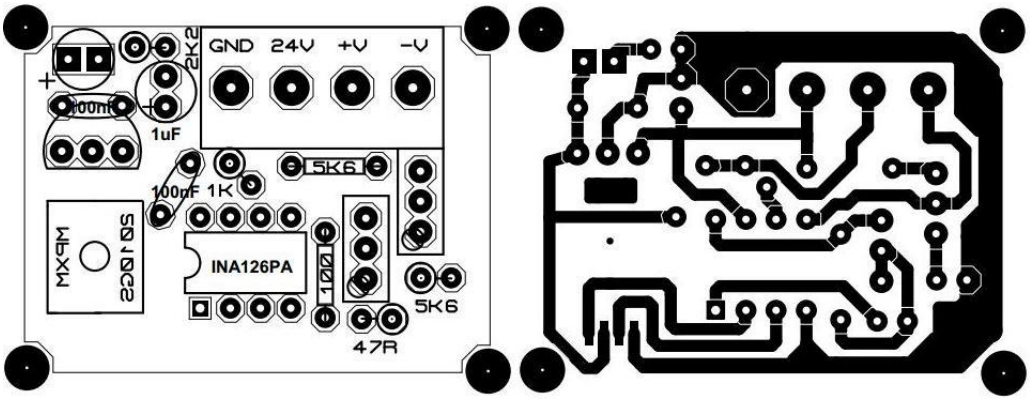


Figura 20 - PCI do Transdutor de Pressão
Fonte: Elaborada pelo autor.

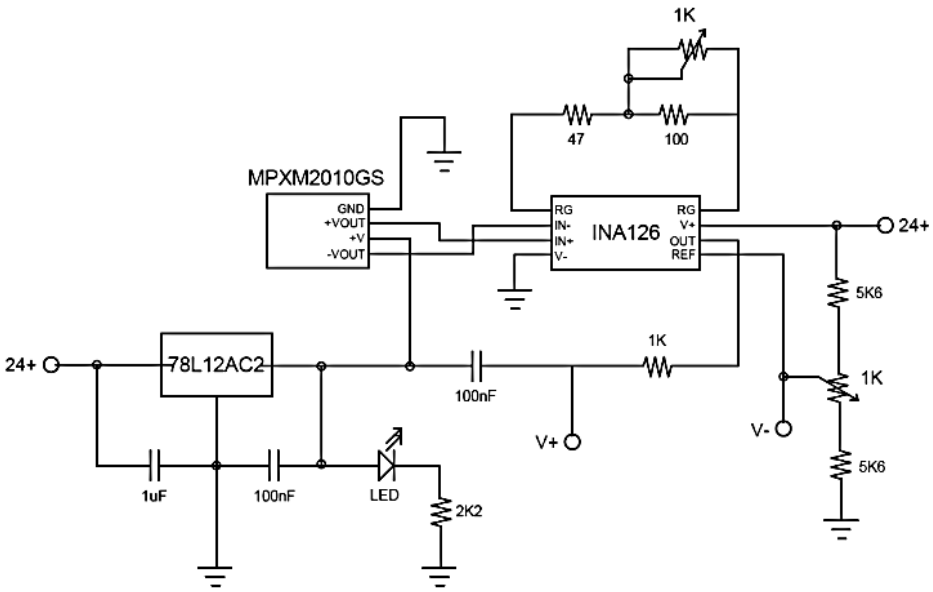


Figura 21 - Circuito do Transdutor de Pressão
Fonte: COCOTA, 2009, p. 89.

Finalizando a PCI, uma caixa de plástico foi adequada para a placa ser alocada nela (Figura 23). Dessa forma, o transdutor ficou protegido, podendo ser fixado em uma base acima do tanque.

Figura 22 - Fonte de Alimentação do Transdutor de pressão

Fonte: Elaborada pelo autor.

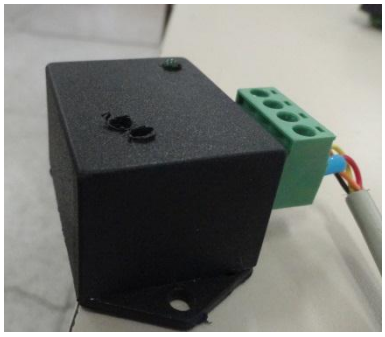


Figura 23 - Transdutor de pressão

Fonte: Elaborada pelo autor.

A calibração do sensor é necessária para adequar a saída (0 a 10 V) ao nível do líquido do tanque (0 a 24 cm), de forma que a saída e a altura sejam lineares e proporcionais. Para realizar essa calibração o circuito possui dois potenciômetros de 1kΩ, sendo um para se calibrar o zero do tanque e o outro o ganho. Ou seja, primeiramente, coloca-se o tanque sem fluido, e ajusta o potenciômetro até que a saída seja 0V, depois carrega o tanque até o máximo (nesse caso, 24cm), e posiciona o outro potenciômetro até se obter uma saída de 10V.

A Figura 24 e a Figura 25 são os gráficos que representam a calibração dos sensores. Em ambos os gráficos, os dados em vermelho representam a tensão média na carga de água, e, em azul na descarga de água. Nota-se a equação que representa a reta de carga na parte superior da área de plotagem. Na parte inferior, tem-se a equação que representa a reta de descarga do tanque. Os resultados obtidos nesses sensores foram satisfatórios, as histereses não possuem significância para a leitura dos dados.

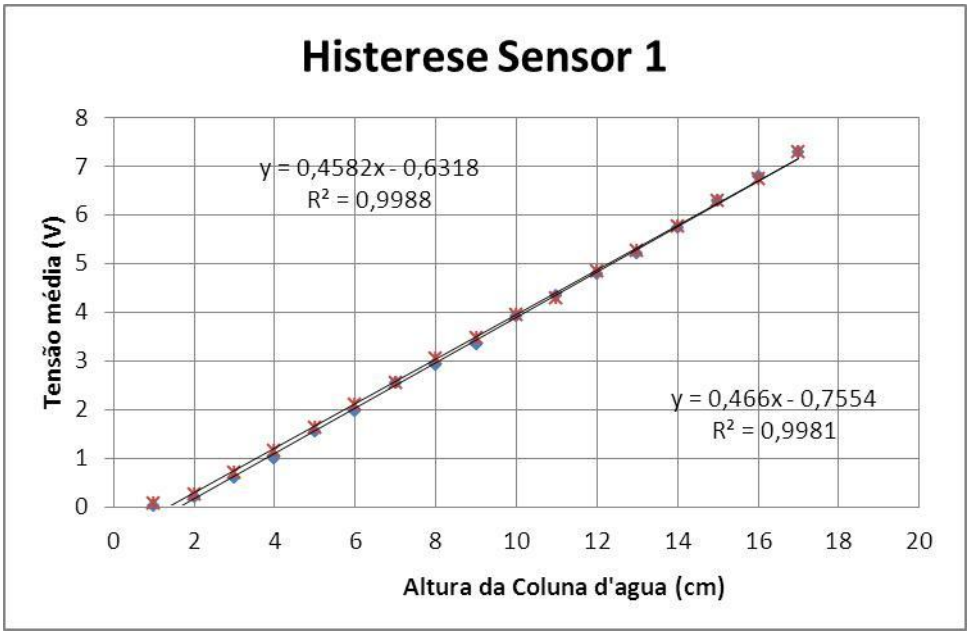


Figura 24 - Histerese do Sensor 1

Fonte: Elaborada pelo autor.

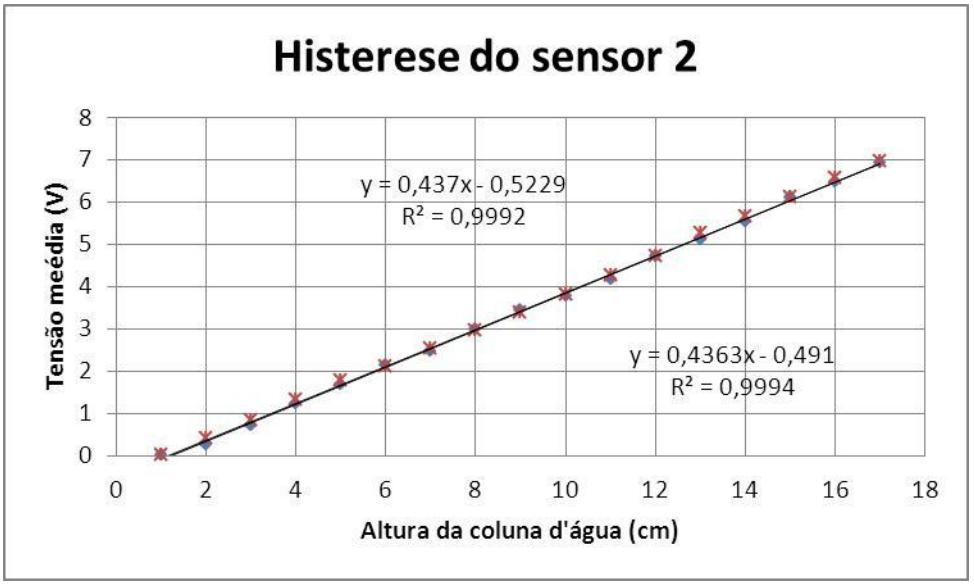


Figura 25 - Histerese do Sensor 2

Fonte: Elaborada pelo autor.

3. ACIONAMENTO DE MOTOR CC

Nos motores de corrente contínua (CC) a energia elétrica é transformada em energia mecânica, ou seja, energia elétrica será entregue ao motor e ele entregará energia mecânica na forma de um movimento rotacional.

O motor CC serve ao ajuste de velocidade muito mais facilmente que o motor CA o faz, de maneira que o desenvolvimento inicial de sistemas de controle de velocidade utilizando dispositivos semicondutores concentrou-se nas máquinas CC (LANDER, 1996). Devido a facilidade com que podem ser controlados, os motores de CC são usuais em aplicações nas quais são necessários mais de uma velocidade ou controles mais precisos.

Basicamente, o motor CC consiste em um circuito para geração de fluxo magnético, conhecido como circuito de excitação e um circuito onde circulará a corrente que criará o torque, conhecido como circuito de armadura. O fluxo magnético é gerado, em geral, pela passagem de corrente em enrolamentos fixo à carcaça, também chamados de bobina de campo ou de excitação. Observa-se na Figura 26 que esse fluxo enlaça a armadura. O funcionamento de um motor de corrente contínua está baseado nas forças produzidas entre o campo magnético e a corrente de armadura no rotor, que fazem o condutor mover num sentido que depende do sentido do campo e da corrente na armadura. A Figura 27 mostra o diagrama elétrico de um motor CC.

Esta capacidade de transformar energia elétrica em mecânica é aproveitada em diversos equipamentos, entre eles: bombas de água, ventiladores, compressores, prensas, tornos, entre outros.

Motores CC têm sido comumente utilizados para acionamentos de bombas, devido a sua capacidade de controle de velocidade variável, especialmente em baixas velocidades, sistemas de controle simples, alto torque de partida e uma boa resposta transitória (OHIO, 2012). O conjunto composto por um motor acoplado a uma bomba mecânica pode ser entendido como motobomba.

A principal função de uma bomba hidráulica é a conversão de energia mecânica em energia hidráulica. As bombas hidráulicas podem ser classificadas como bombas de deslocamento positivo ou de deslocamento não positivo. Nas bombas de deslocamento não positivo a movimentação do fluído ocorre pela ação de forças que se desenvolvem na massa do mesmo, em consequência da rotação de um eixo no qual é acoplado um disco dotado de pás o qual recebe o fluído pelo seu centro e o expulsa pela periferia, pela ação da força

centrífuga. Já as bombas de deslocamento positivo fornecem determinada quantidade de fluido a cada rotação ou ciclo. A movimentação do fluído é causada diretamente pela ação do órgão de impulsão da bomba que obriga o fluído a executar o mesmo movimento a que está sujeito este impulsor (BRASIL, 2013).

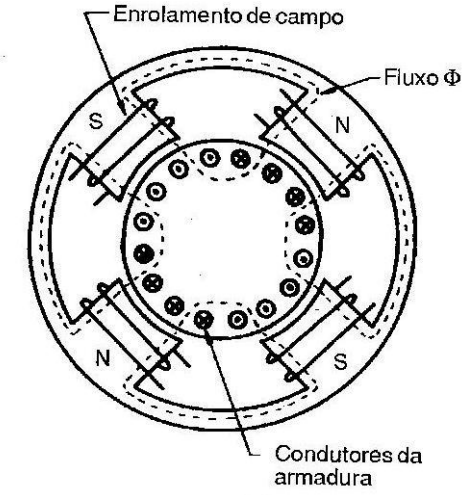


Figura 26 - Máquina CC

Fonte: LANDER, 1996, p.445.

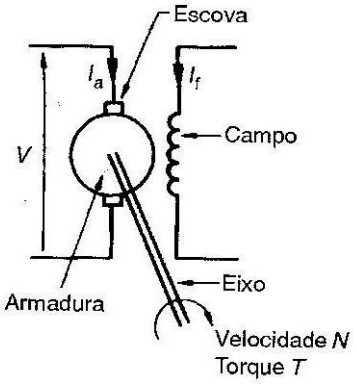


Figura 27 - Representação do Circuito

Fonte: LANDER, 1996, p.445.

Um tipo de bomba muito comum em sistemas hidráulicos são as bombas de engrenagem (Figura 28). A bomba de engrenagem consiste basicamente de uma carcaça com orifícios de entrada e de saída, e de um mecanismo de bombeamento composto de duas engrenagens. Uma das engrenagens, a engrenagem motora, é ligada a um eixo que é conectado a um elemento acionador principal. A outra engrenagem é a engrenagem movida. No lado da entrada, os dentes das engrenagens desengrenam, o fluido entra na bomba, sendo

conduzido pelo espaço existente entre os dentes e a carcaça, para o lado da saída onde os dentes das engrenagens engrenam e forçam o fluido para fora do sistema (BRASIL, 2013).

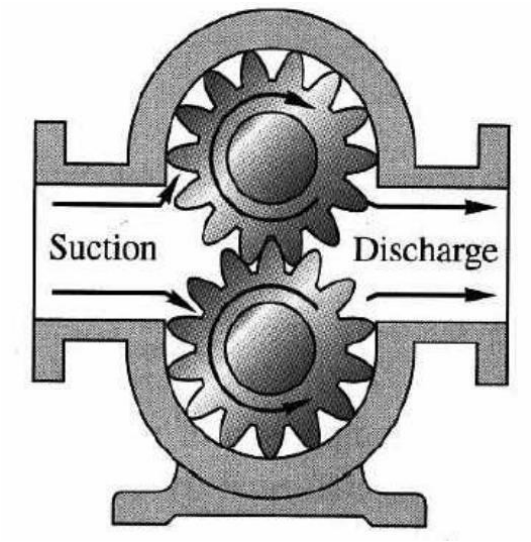


Figura 28 - Bomba de engrenagem
Fonte: BRASIL, 2013, p. 72.

3.1 Motobomba utilizada na planta

A motobomba utilizada para fazer o controle do fluxo de fluido na planta foi uma JM-PUMP, fabricada pela Jersey Modeler, uma empresa fabricante de sistemas de combustível para aeromodelagem. Esse equipamento é constituído de um motor de corrente contínua (CC) de 12 V, um dissipador de calor e uma bomba de engrenagens acoplada no eixo do motor (Figura 29). A bomba pesa em torno de 300 gramas e possui um desempenho comparado a bombas maiores. Além disso, é resistente a produtos químico tais como gás ou querosene.

Na Tabela 1, as características de desempenho da JM-PUMP são mostradas, sendo P correspondente a pressão, H_{man} a altura manométrica, Q a vazão e I a corrente.

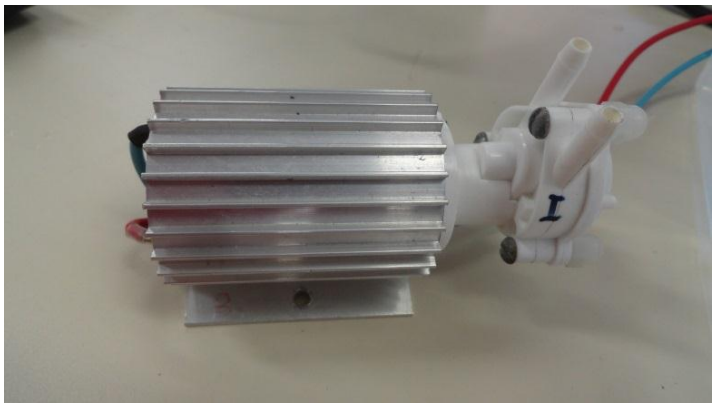


Figura 29 - JM-PUMP
Fonte: Elaborado pelo autor.

Tabela 1- Dados da bomba

P [psi]	H_{man}[m]	Q [ml/min]	I [A]
0	0	2200	0,5
5	3,7	2100	0,9
10	7	1750	1,3
15	10,7	1250	1,6
20	14	350	1,8

Fonte: Página da internet[2]

3.2 Circuito para acionamento da bomba

O acionamento e o controle da vazão da bomba são feitos através de um módulo de potência (Figura 30) que foi desenvolvido no laboratório para esse projeto. O módulo aciona o motor de corrente contínua de 12 V utilizando Modulação por Largura de Pulso (PWM) que, através da variação do ciclo de trabalho, produz uma variação na tensão média aplicada ao motor, ou mais especificamente, na armadura do motor.

[2] Disponível em: <www.jerseymodeler.com/id29.html>. Acesso em 24 jul. 2014.

Figura 30 - Modulo de Potência
Fonte: Elaborada pelo Autor

Figura 31 - Fonte Módulo de Potencia
Fonte: Elaborada pelo Autor

O módulo é alimentado por uma fonte (Figura 31) de 10V e, através de um regulador de tensão, essa tensão é regulada para 5V, que irá alimentar o microcontrolador *PIC16F876.* Esse microntrolador é responsável por gerar o sinal PWM que fará o controle da vazão da bomba, ainda, proporciona a visualização em um LCD (*Liquid Crystal Display)* do valor de referência do PWM e a tensão de acionamento da bomba. A Figura 32 e Figura 33 ilustram, respectivamente, o circuito do módulo de potência e sua placa de circuito impresso.

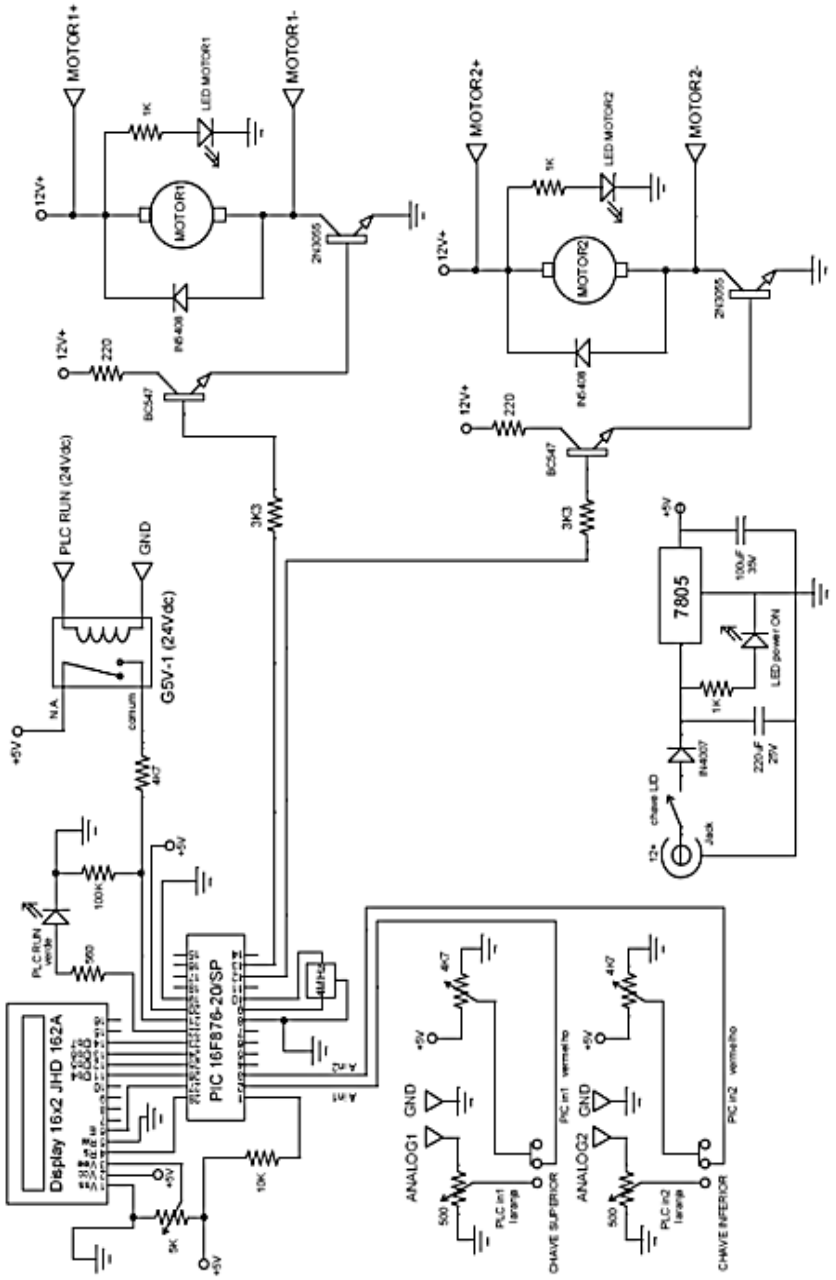


Figura 32 - Circuito Módulo de Potência

Fonte: COCOTA, 2009, p.

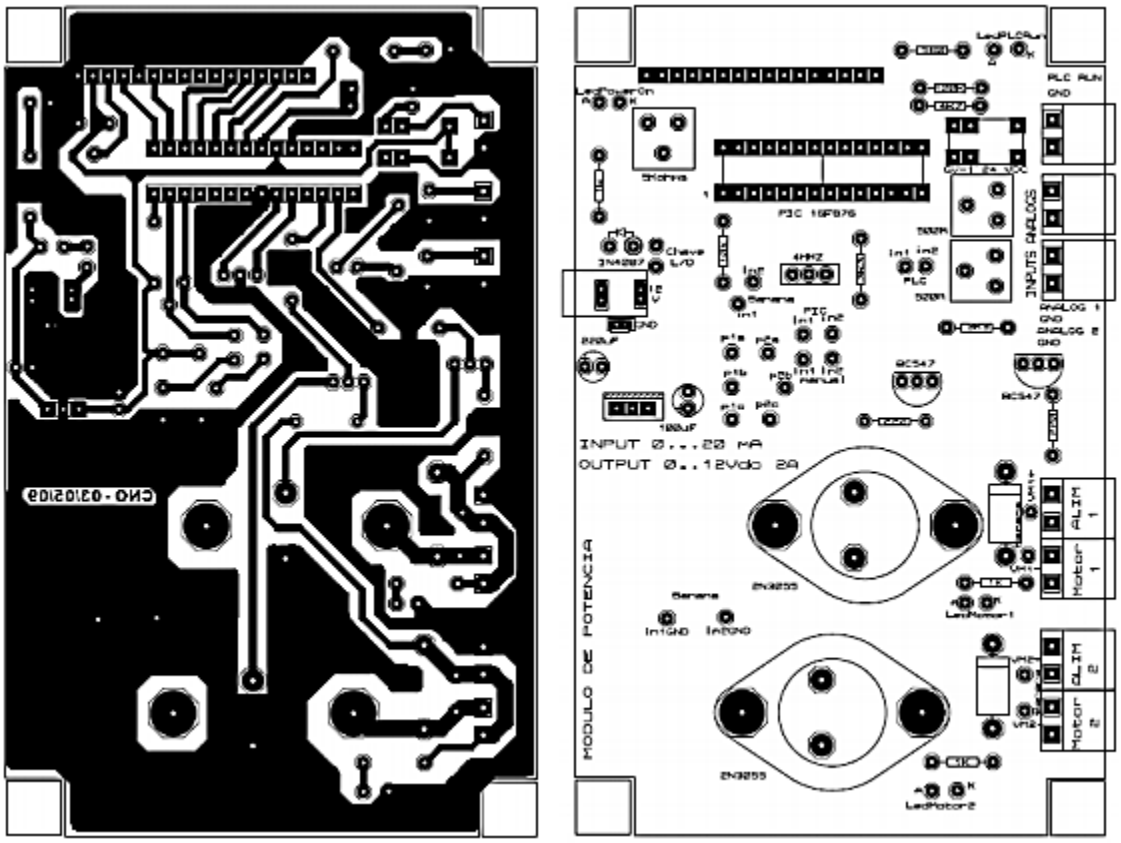


Figura 33 - PCI Módulo de Potência

Fonte: COCOTA, 2009, p 93.

O sinal PWM controla a velocidade do motor através do ciclo de trabalho ou *duty cycle*. A Figura 34 ilustra dois PWM, um com ciclo de trabalho de 50% e outro com 25%. No primeiro caso, a forma de onda retangular possui o ciclo ativo por 50% do tempo e inativo no restante dos 50% do tempo, logo, tem-se metade do tempo com corrente e metade sem corrente, consequentemente, a potência e a tensão média são 50% do valor de entrada. O mesmo ocorre no segundo caso, porém com 25% de ciclo de trabalho, obtendo a potência e tensão média 25% do valor de entrada.

Observa-se na Figura 32 que o sinal de PWM sai do pino 12 e 13 do PIC16F876 e entra em um circuito de potência para acionar o motor. Esse dispositivo de potência está configurado com os transistores BC547 e 2N3055 em uma conexão *Darlington* (Figura 35).

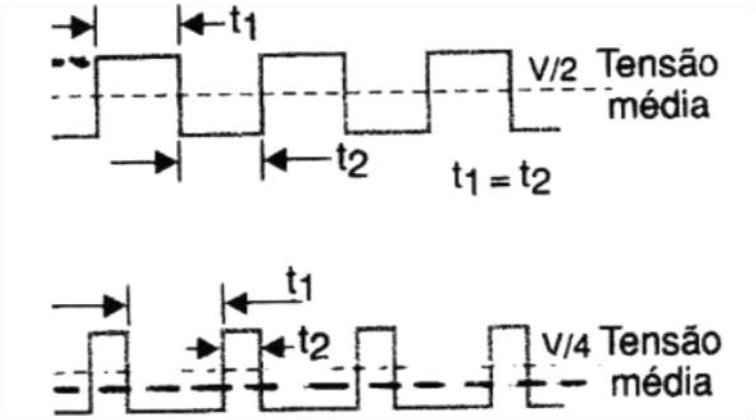


Figura 34 – PWM

Fonte: Página da internet[3]

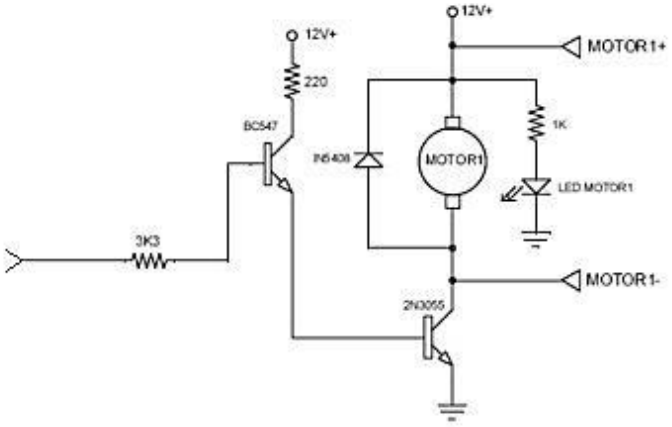


Figura 35 - Conexão *Darlington*
Fonte: Elaborada pelo autor

A conexão *Darlington* é uma ligação em cascata de dois transistores, nesse caso, um transistor BC547 e um 2N3055. O emissor do BC547 é ligado à base do 2N3055. A principal finalidade dessa ligação é aumentar o ganho da corrente (β), sendo que o ganho total dos dois transistores é o produto do ganho individual de cada transistor (Equação 1).

$$\beta = \beta_1 \beta_2 \quad (1)$$

Onde:

β_1 = é o ganho do BC547 e;

β_2 = é o ganho do 2N3055.

[3] Disponível em <www.newtoncbraga.com.br/index.php/5534-mec139>. Acesso em 25 jul 2014.

3.3 Programação do PIC para acionamento do motor

O acionamento da bomba é feito através de um microcontrolador PIC16F876, fabricado pela Microchip. Esse microcontrolador possui arquitetura interna de 8 bits, baseado em uma arquitetura HARVARD RISC, modificada com apenas 35 instruções. Ainda possui memória FLASH, para armazenamento do programa e memória EEPROM integrada. Esse microcontrolador é amplamente utilizado em diversos projetos, desde projetos simples como contadores, até projetos mais complexos como controle.

A programação do PIC, chamada de *firmware*, foi feita no CCS utilizando linguagem C. O CCS é uma ferramenta utilizada para programar e compilar *firmware* em C. Esse software possui uma extensa biblioteca de funções integrada, comandos específicos pré-processados para PIC e programas prontos para executar de forma rápida alguns projetos.

A Figura 36 representa o fluxograma da programação feita em C para o acionamento da motobomba, vide Apêndice A. A programação é iniciada com a definição e inicialização das variáveis que serão necessárias para a implementação do *firmware.*. Logo, o display é iniciado, no qual os valores da referência do PWM e a tensão aplicada nas bombas podem ser visualizados. Os sinais para o acionamento da bomba podem ser feitos manualmente, através de um potenciômetro, ou pelo CLP, a opção depende da necessidade do usuário, a escolha é feita no hardware do módulo de potência. O PIC faz a leitura desse sinal e a rotina faz uma conversão do valor do PWM de forma a se ter o valor correto para a visualização no display. Assim, o sinal de comando aciona as bombas de acordo com o valor de tensão enviado pelo potenciômetro ou pelo CLP. Na medida em que os valores são alterados, os mesmo são atualizados no display.

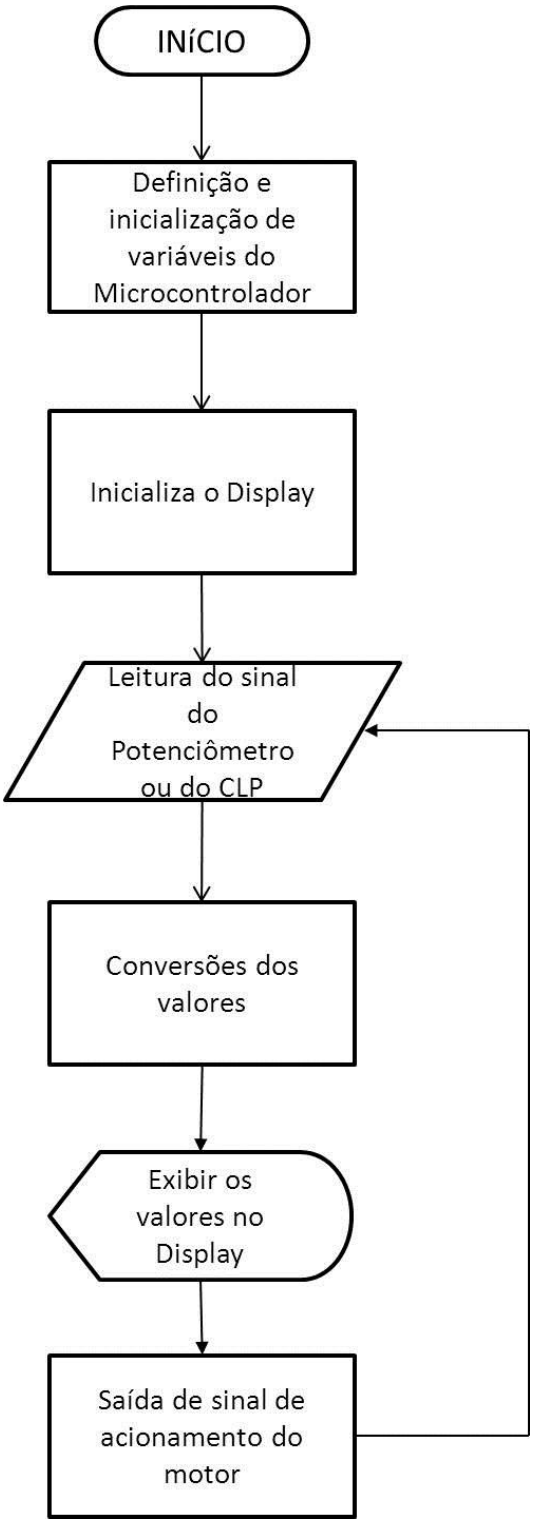


Figura 36 - Fluxograma da programação do PIC

4. REDE

O controle dos tanques é feito através do software MATLAB® e do CLP Versamax® da GE Fanuc. Basicamente, o CLP faz o controle e o MATLAB® faz a modelagem e a simulação. Entretanto, o protocolo de comunicação utilizado pelo CLP não é compatível com o do MATLAB®, assim, é necessário a utilização de um padrão de comunicação conhecido como OPC.

A sigla OPC significa *OLE (Object Linking and Embedding/Objeto Vinculado e embutido)* para Controle de Processos, e foi desenvolvido pela OPC Foundation. OPC é o padrão de interoperabilidade para o intercâmbio seguro e confiável de dados no espaço de automação industrial e em outras indústrias (OPCFOUNDATION, 2014). O objetivo é permitir a comunicação de protocolos específicos de CLP (Modbus, Profibus, etc) com sistemas IHM (Interface Homem-Máquina)/SCADA, de forma que haja uma padronização dessa comunicação.

O OPC unifica o padrão de comunicação de dados de controle de processo e permite que diferentes produtos sejam interfaceados com uma única tecnologia, promovendo interações dos sistemas de operação e integração de vários processos em um só sistema, isso com custo e tempo de implementação reduzidos (FARIA, 2013). A Figura 37 representa um diagrama do OPC, onde OPC *server* é o *software* que converte o protocolo de comunicação do CLP e o OPC *client software* é qualquer programa que necessite se comunicar com o *hardware*, como por exemplo uma IHM.

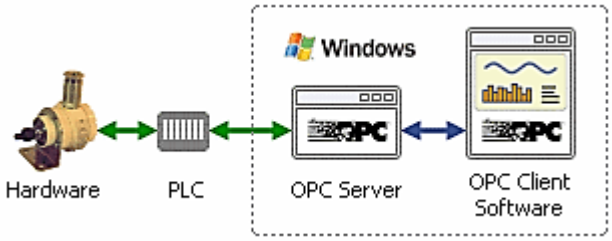


Figura 37 - Diagrama básico do OPC

Fonte: Página da Internet[4]

[4]Disponivel em: <www.opcdatahub.com/WhatIsOPC.html>. Acesso em: 30 de julho de 2014.

O *software* utilizado para fazer a comunicação OPC nesse trabalho foi o KEPServerEX®, produzido pela empresa Kepware Technologies, que foi o responsável por proporcionar o interfaceamento entre o MATLAB® e o CLP. A Figura 38 representa o diagrama da rede utilizada no trabalho.

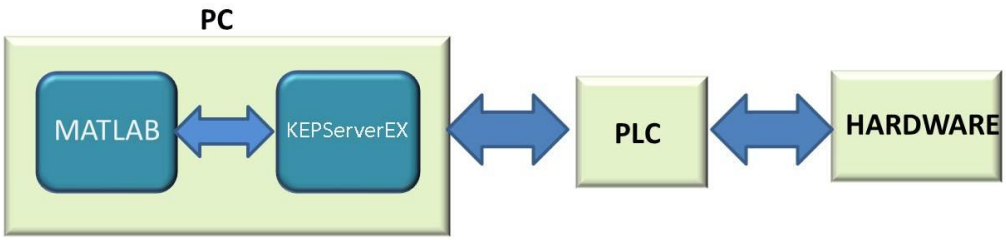


Figura 38 - Diagrama da Rede.
Fonte: Elaborada pelo autor.

4.1 MATLAB® e KEPServerEX

O MATLAB® é um ambiente interativo que utiliza linguagem de alto nível, permitindo a execução de tarefas complexas, tais como, processamento de imagens, comunicações, aquisições de dados, sistemas de controle, robótica, entre outras.

Esse *software* foi utilizado neste projeto para aquisição dos dados, modelagem do sistema e simulação do controlador. Como o MATLAB® possui comunicação OPC, utilizou-se KEPServerEX® para fazer a padronização e adquirir os dados vindo do CLP. Para obter esse laço entre o KEPServerEx e o MATLAB® é necessário que o OPC Toolbox™ seja carregado no MATLAB®.

O OPC Toolbox™ é uma ferramenta do MATLAB® que permite a conexão com os servidores OPC DA *(Data Access)* e OPC HDA *(Historical Data Access)*. Cria-se um objeto OPC DA para se comunicar com a um servidor OPC, assim, utiliza-se uma estrutura hierárquica intuitiva para ajudar a gerenciar as conexões OPC e os itens do servidor (tags). Essa ferramenta possibilita configurar e controlar todos os "clientes", grupos e objetos do item, podendo modificar suas propriedades.

O objeto OPC HDA é criado para se conectar a um servidor OPC HDA. Esse cliente permite que se navegue nos dados do servidor e recupere as IDs completas de cada item armazenado, essas IDs são utilizadas para solicitar dados históricos a partir do servidor. Esses dados podem ser recuperados especificando, quais são as IDs, o período de tempo para o qual deseja se recuperar os dados e os parâmetros opcionais.

Dessa forma, os servidores OPC DA e OPC HDA dão acesso aos dados em tempo real e históricos diretamente do MATLAB® e do Simulink®, possibilitando a leitura e o registro dos dados vindos do CLP, ainda, a escrita de dados OPC no mesmo.

4.2 CLP e KEPServerEX®

O *software* utilizado pelas linhas de controlador GE Fanuc é o Proficy Machine Edition. Através deste *software* foram feitas as configurações do CLP, além da programação, utilizando linguagem LADDER (Apêndice B), para gerar os sinais de controle e a leitura dos valores para fazer a aquisição de dados.

No Proficy Machine Edition foi implementada a rotina principal e duas sub-rotinas, de forma a organizar a lógica utilizada no projeto. Uma sub-rotina é utilizada para obter as conversões de escalas dos instrumentos e demais cálculos auxiliares. Outra sub-rotina recebe as informações dos sensores e atua na bomba controlando sua velocidade, consequentemente controlando os níveis dos tanques. O controle é realizado utilizando um bloco PID, no qual os parâmetros do controlador podem ser inseridos.

A comunicação do CLP com o PC é feita através de um cabo Ethernet, muito comum em comunicação de redes domesticas. Esse cabo é de par trançado e consiste em oito fios, formando quatro pares de fios de cobre trançados. Para esse trabalho ele foi crimpado na forma *crossover* (Figura 39).

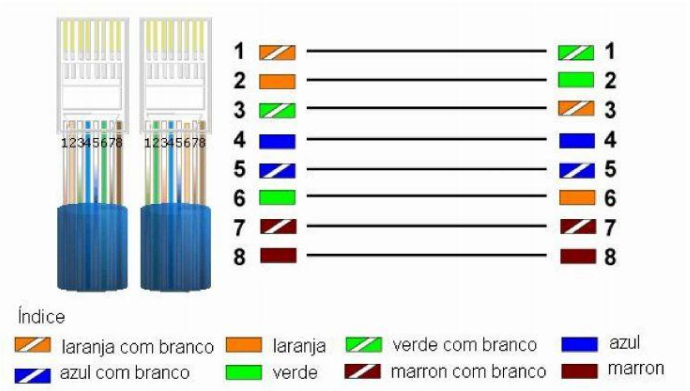


Figura 39 - Cabo *crossover*

Fonte: Página da Internet[5]

[5] Disponivel em : <pt.slideshare.net/titaoyamamoto/como-fazer-cabo-crossover-e-direto>. Acesso em 30 jul 2014.

A troca de dados utilizando Ethernet é feita através do envio de pacotes. Logo, para ter a comunicação entre o CLP Versamax® e o PC, o endereço IP do PC precisar estar configurado em suas conexões de rede um endereço IP da mesma classe do CLP. Da mesma forma, o IP do KEPServerEX® precisa ser o mesmo utilizado no CLP.

O KEPServerEX® permite a criação de *tags*, através desses *tags* são feitas a leitura e escrita no CLP. Os *tags* são a forma de fazer um link com os endereços de memória do CLP. Quando um *tag* é criado, podendo ser de leitura ou leitura/escrita, e associado a um endereço de memória, através dos *drivers* de comunicação, o valor do endereço que é lido ou escrito no CLP, pode ser lido ou escrito por qualquer *software* que utilize o mesmo padrão de comunicação. A Figura 40 ilustra *tags* criadas para um projeto, onde o campo *address* é o que faz o laço com o CLP.

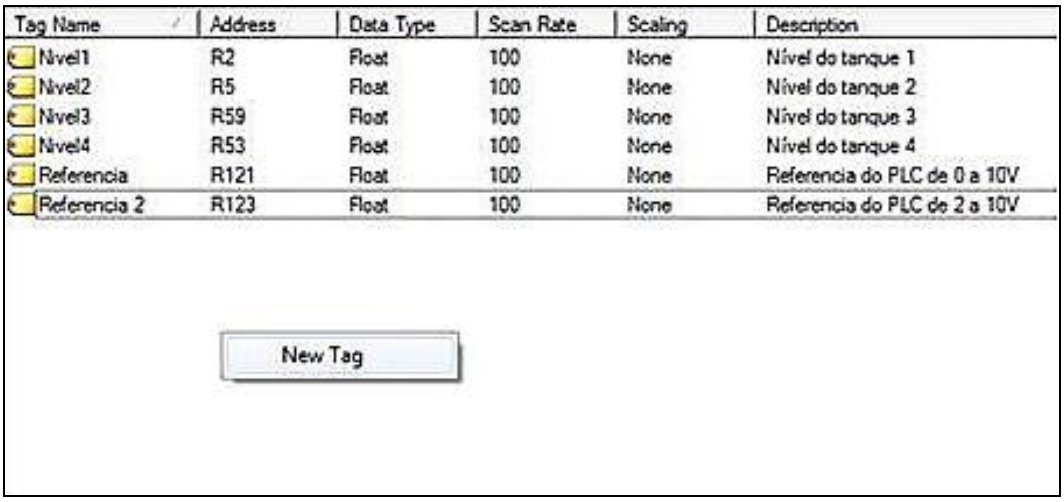


Figura 40 - Tags em KEPServerEX

Fonte: Elaborado pelo autor.

5. SISTEMA SUPERVISÓRIO E ESPECIALISTA

Ao fim do desenvolvimento desse trabalho, em conjunto com discente Pedro Henrique Lopes Faria (FARIA, 2013), foi desenvolvido um sistema supervisório (Apêndice C) e especialista (Apêndice D) utilizando a ferramenta GUIDE do MATLAB®.

Os supervisórios são sistemas de supervisão e aquisição de dados Que permitem que sejam monitoradas e rastreadas informações de um processo produtivo ou instalação física (LOPES, 2009). Tais informações são coletadas através de equipamentos de aquisição de dados e, em seguida, manipuladas, analisadas, armazenadas e posteriormente, apresentadas ao usuário. Estes sistemas também são chamados de SCADA (*Supervisory Control and Data Acquisition*) (FARIA, 2013).

Os sistemas especialistas solucionam problemas que normalmente são resolvidos por especialistas humanos que detenham grande conhecimento no assunto (LESSA *et al.*, 2010). Logo, esses programas solucionam problema em campos específicos e devem ter um desempenho comparado ao dos especialistas humanos. Além disso, para o desenvolvimento desse sistema é necessário que um profissional da área a ser tratada faça um acompanhamento, para que as informações necessárias ao sistema sejam passadas ao profissional que irá desenvolvê-lo.

O GUIDE é uma ferramenta do MATLAB® usada para criação de interfaces com o usuário. Sua sigla significa *Graphic User Interface Development Enviroment* (Ambiente de Desenvolvimento de Interface Gráfica para o Usuário). É possível criar uma ou mais telas, com comandos e tarefas interativas. A escolha dessa ferramenta para a criação desses sistemas deve-se a posterior utilização de outras ferramentas do MATLAB® no decorrer do desenvolvimento do projeto, de forma que algumas rotinas necessárias para o supervisório já haviam sido implementadas para outras etapas do trabalho.

O sistema supervisório foi desenvolvido previamente ao sistema especialista. O sistema supervisório é utilizado unicamente para o monitoramento do processo. O sistema especialista permite a inserção e a leitura de dados para que as atividades especificadas possam ser realizadas. O sistema especialista nesse caso foi implementado para o auxílio do usuário na bancada didática quanto à retirada dos parâmetros para a elaboração de um controlador.

A Figura 41 mostra a tela inicial do sistema, na qual há duas possibilidades para o usuário, visualizar o supervisório através do botão “Supervisão dinâmica de níveis e tensões” ou realizar o ensaio para a identificação os parâmetros, sendo esse o sistema especialista.

Figura 41 - Tela inicial do Sistema.
Fonte: Elaborada pelo autor.

5.1 Sistema Supervisório

Ao entrar no sistema supervisório, o usuário visualiza a tela ilustrada na Figura 42. Observa-se que ao lado direito a tela são mostrados os gráficos dos níveis do tanque em tempo real, além do valor da tensão aplicada à bomba.

Os parâmetros do sistema tais como área dos tanques, perda de carga, ganho da bomba, são visualizado na parte esquerda da tela. Caso o usuário deseje voltar a tela inicial, basta clicar em “VOLTAR”. Isso permite que o usuário retorne e faça a escolha do sistema especialista ou simplesmente abstenha-se a outra operação.

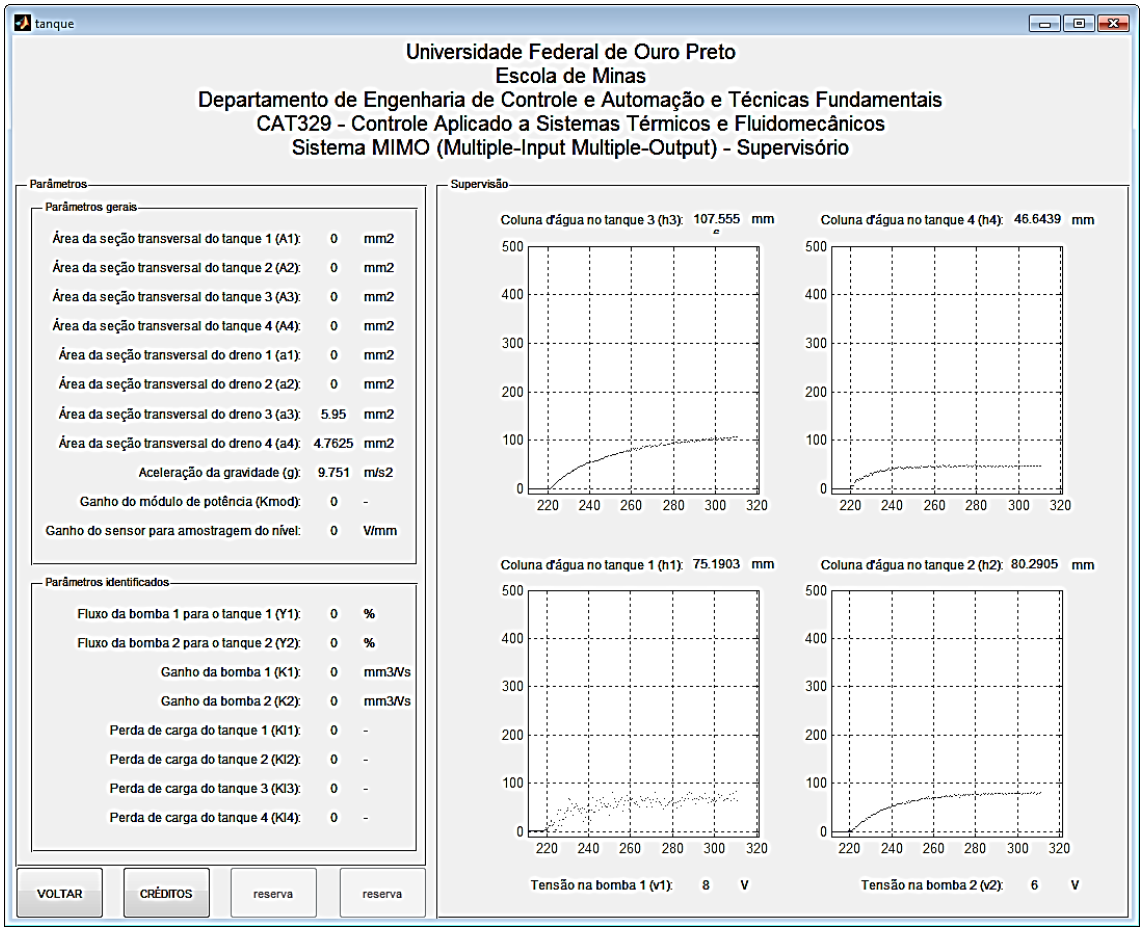


Figura 42 - Sistema Supervisório

5.2 Sistema especialista

O sistema especialista foi desenvolvido com o intuito de introduzir ao usuário uma sequência de tarefas para atingir o objetivo proposto, nesse caso é encontrar os parâmetros para posteriormente implementar um controlador.

O sistema especialista opera em conjunto com o supervisório criado em ambiente MATLAB®, a simulação do modelo através do SIMULINK® e a programação LADDER embarcada no PLC. A tela principal do sistema especialista (Figura 43) possui dois botões, que são os dois passos em ordem necessários para se obter os parâmetros. O primeiro passo é o “Ensaio para fluxos e ganhos da bomba” e depois o “Ensaio para as perdas de cargas dos tanques”.

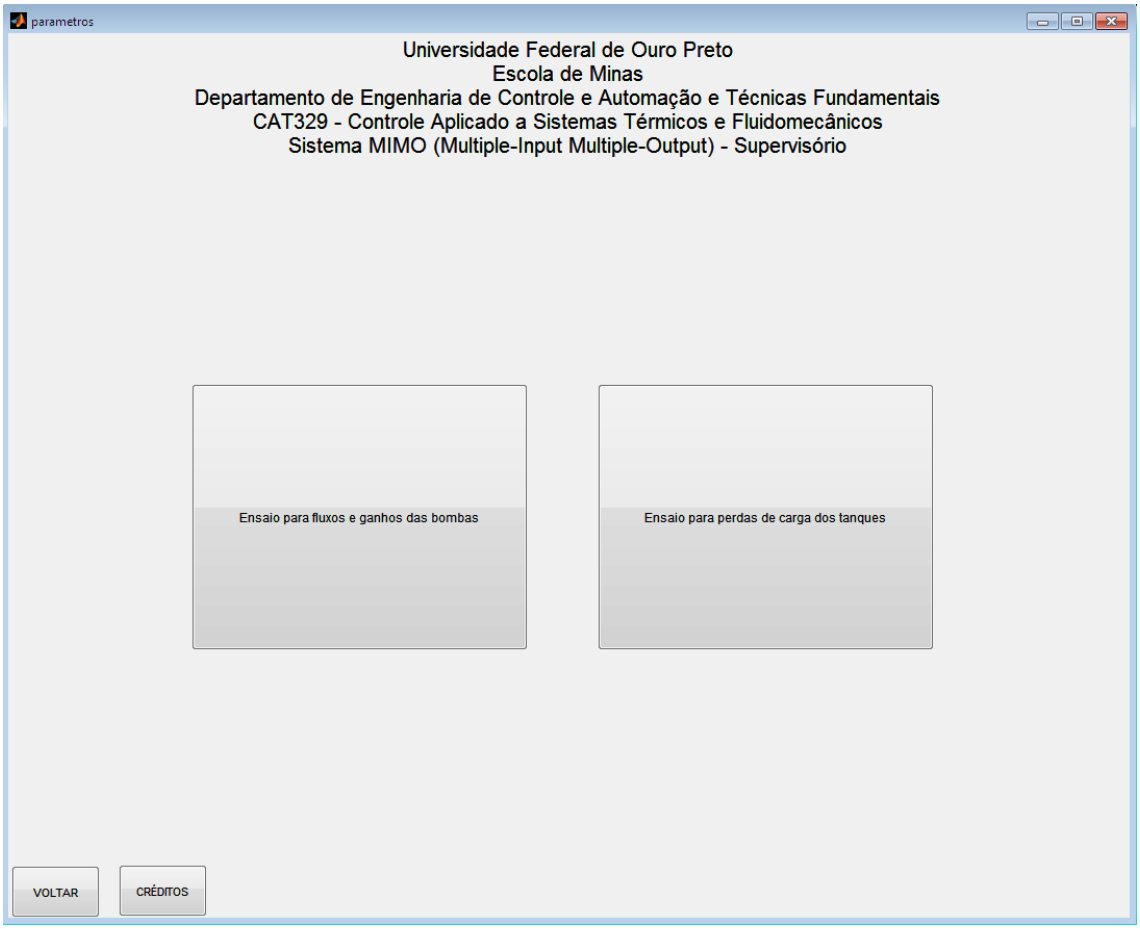


Figura 43 - Tela inicial do Sistema Especialista

5.2.1 Ensaio para fluxos *(γ's)* e ganhos da bomba (k_i's)

A Figura 44 ilustra a tela para que o primeiro ensaio seja realizado. Uma breve descrição do que será realizado neste ensaio vem na parte superior da tela. As instruções do passo-a-passo vêm descritas no lado esquerdo da tela.

Para realizar o ensaio, devem-se aplicar as tensões das bombas 1 e 2, para que se mantenham os níveis de água dos tanques em alturas desejadas com o sistema em malha aberta, sendo essas alturas as dos pontos de operação do sistema. Para isso, inserem-se os valores dessas tensões nos campos específicos da tela, clicando em "TESTE", em seguida os tanques começam a encher e os níveis da água nos tanques entram em regime permanente. Caso não sejam satisfatórios, escolhem-se novas tensões das bombas, repetindo-se a operação. Após ser atingido o regime permanente satisfatório, clica-se em "FIM".

Após o esvaziamento dos tanques, vedam-se os orifícios dos drenos dos tanques. Inserem-se os valores das tensões para o regime permanente, clica-se em "INÍCIO", e em "FIM" quando os níveis estiverem satisfatórios. Após um tempo pré-determinado na

programação do MATLAB®, o gráfico da altura da coluna de água dos tanques em função do tempo é desenhado, e os dados do ensaio são gravados no arquivo “ensaio01.mat”, para que possam ser posteriormente acessados pelo usuário.

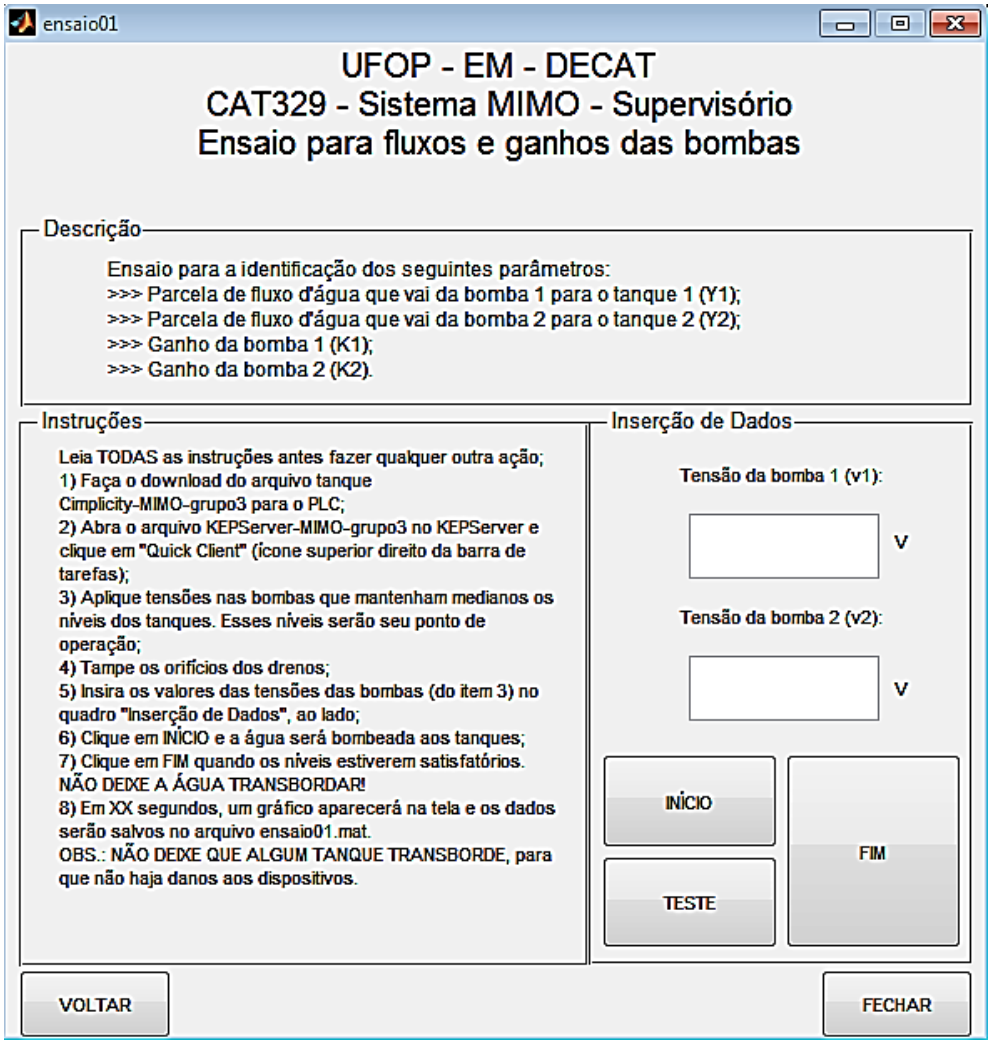


Figura 44 - Ensaio para fluxo e ganho da bomba

Uma vez finalizado o ensaio de carga de água, os discentes calculam os parâmetros dos ganhos das bombas e as parcelas de fluxo de água com base nos resultados experimentais. Como as seções transversais dos tanques são iguais, a parcela γ_1 é obtida pela razão do nível do tanque 1, ao final do processo de carga, em relação ao somatório dos níveis dos tanques 1 e 4. De forma análoga o parâmetro γ_2 é encontrado a partir dos níveis dos tanques 2 e 3. Assim, o ganho da bomba i (k_i) é obtido dividindo o valor da vazão da bomba pela tensão média que foi aplicada nela durante o ensaio de carga de água.

5.2.2 Ensaio para perda de carga (k_{1j}'s)

A Figura 45 ilustra a interface com o usuário para a realização do ensaio para a obtenção das perdas de cargas. O *layout* da tela é semelhante ao da descrita no ensaio anterior.

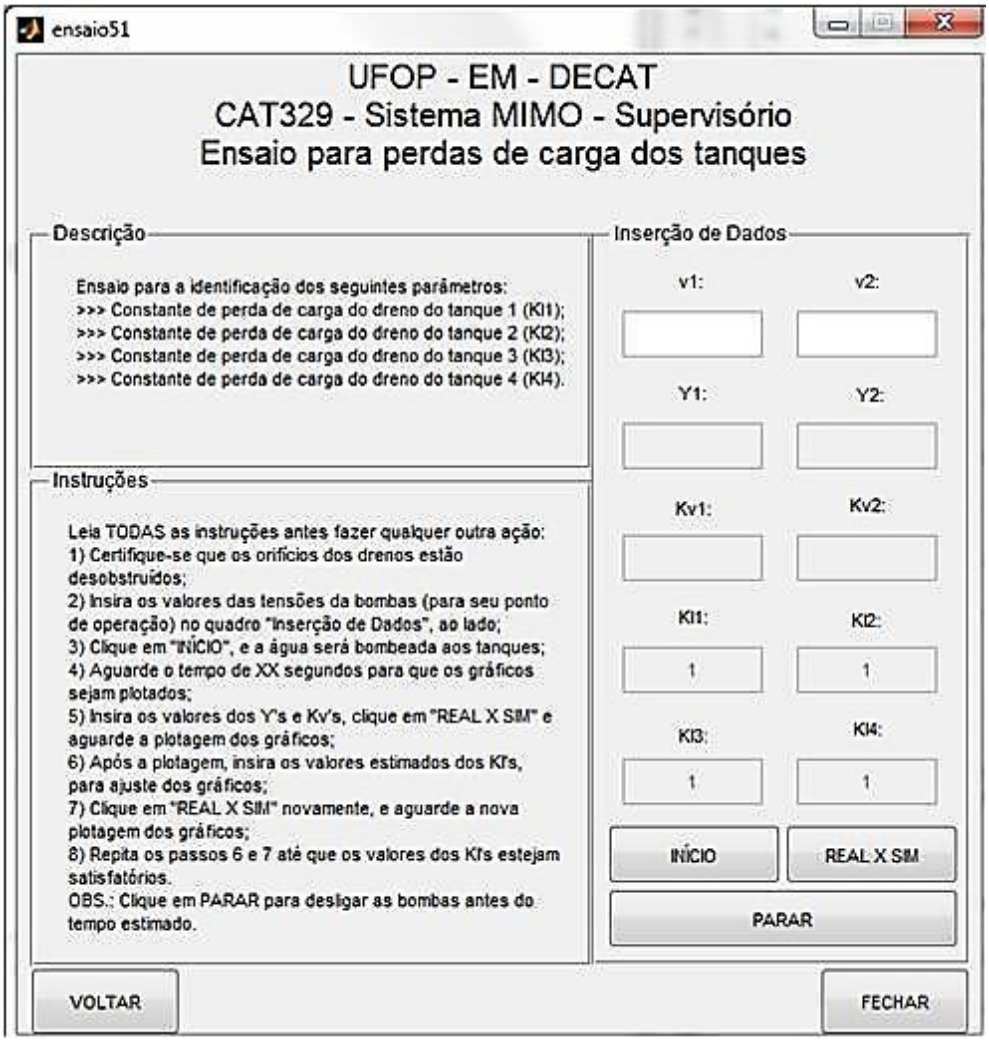


Figura 45 - Ensaio para perda de cargas
Fonte: Elaborado pelo autor

Para esse ensaio é necessário que os drenos sejam desobstruídos. Logo, os valores das tensões do ponto de operação escolhidos no ensaio anterior são inseridos nos campos correspondentes; apenas esses campos estarão habilitados a receber valores.

Clica-se em "INÍCIO" e água começa a ser bombeada aos tanques. Após certo tempo, definido na programação do MATLAB® para que os níveis dos tanques entrem em regime permanente, um gráfico com a altura dos níveis nos tanques em função do tempo é plotado, apenas para mostrar ao usuário que os dados foram adquiridos com sucesso. A Figura 46

ilustra esse passo. O botão “PARAR”, permite que o experimento real seja interrompido a qualquer momento.

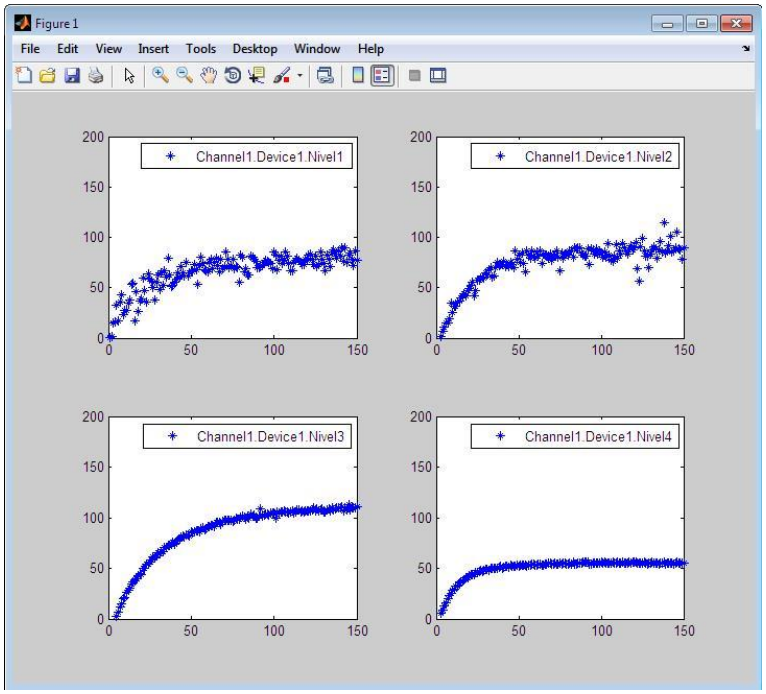


Figura 46 - Aquisição de dados para ensaio
Fonte: Elaborada pelo autor.

Após a exibição do gráfico, os campos de γ_1, γ_2, k_1 e k_2 são habilitados para inserção de dados, como mostrado na Figura 47.

Nesses campos são inseridos os valores dos parâmetros identificados no ensaio anterior. Clica-se, então, no botão "REAL X SIM", e os gráficos dos resultados experimentais e da simulação do modelo do processo são plotados, finalizando o segundo passo, como mostra a Figura 48.

Conseguinte a execução dos passos citados, os campos para inserção dos k_{1j}’s são habilitados. O usuário deve estimar os valores para k_{11}, k_{12}, k_{13} e k_{14} e clicar novamente em “REAL X SIM”, obtendo novos gráficos. Esse passo deve ser repetido até que se tenham os gráficos da simulação ajustados aos do experimento real. A Figura 49 ilustra a tela com os k_{1i}’s habilitados. A Figura 50 ilustra os resultados da simulação com os parâmetros ajustado de maneira que apresenta comportamento semelhante ao do processo real.

Os resultados são meramente ilustrativos, as Figura 48 e Figura 50 não representam fielmente o resultado desejado para o ensaio, sendo que a intensão é exibir as telas que aparecerão na utilização do sistema.

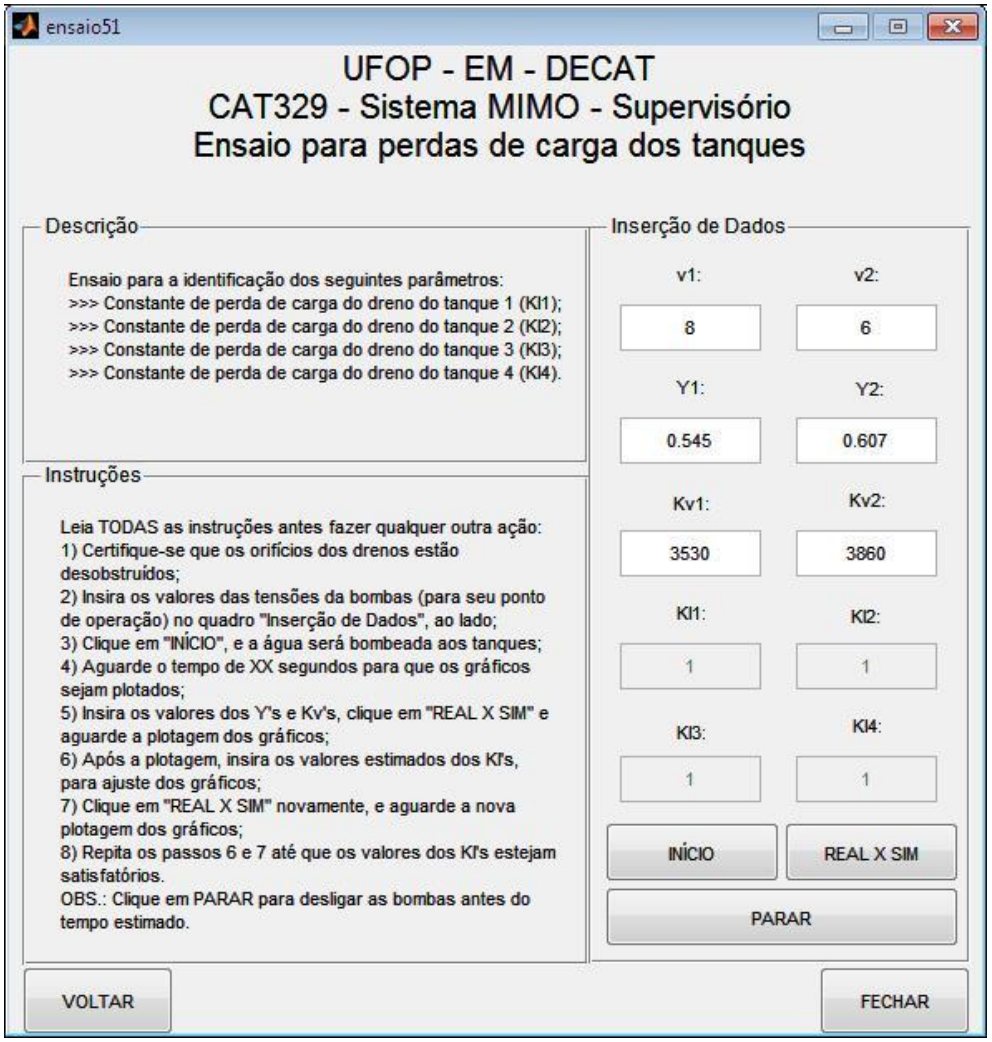


Figura 47 - Tela apos aquisição de dados
Fonte: Elaborada pelo autor.

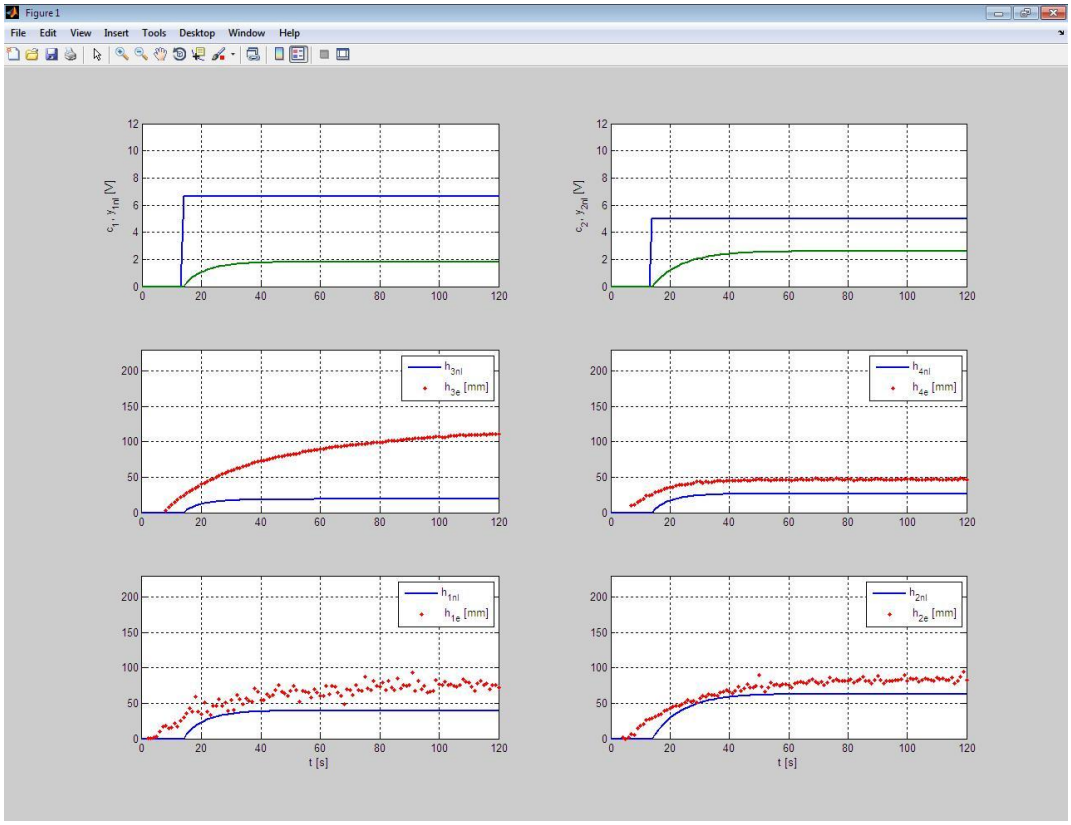


Figura 48 - Dados experimentais e de simulação sem perda de carga
Fonte: Elaborada pelo autor.

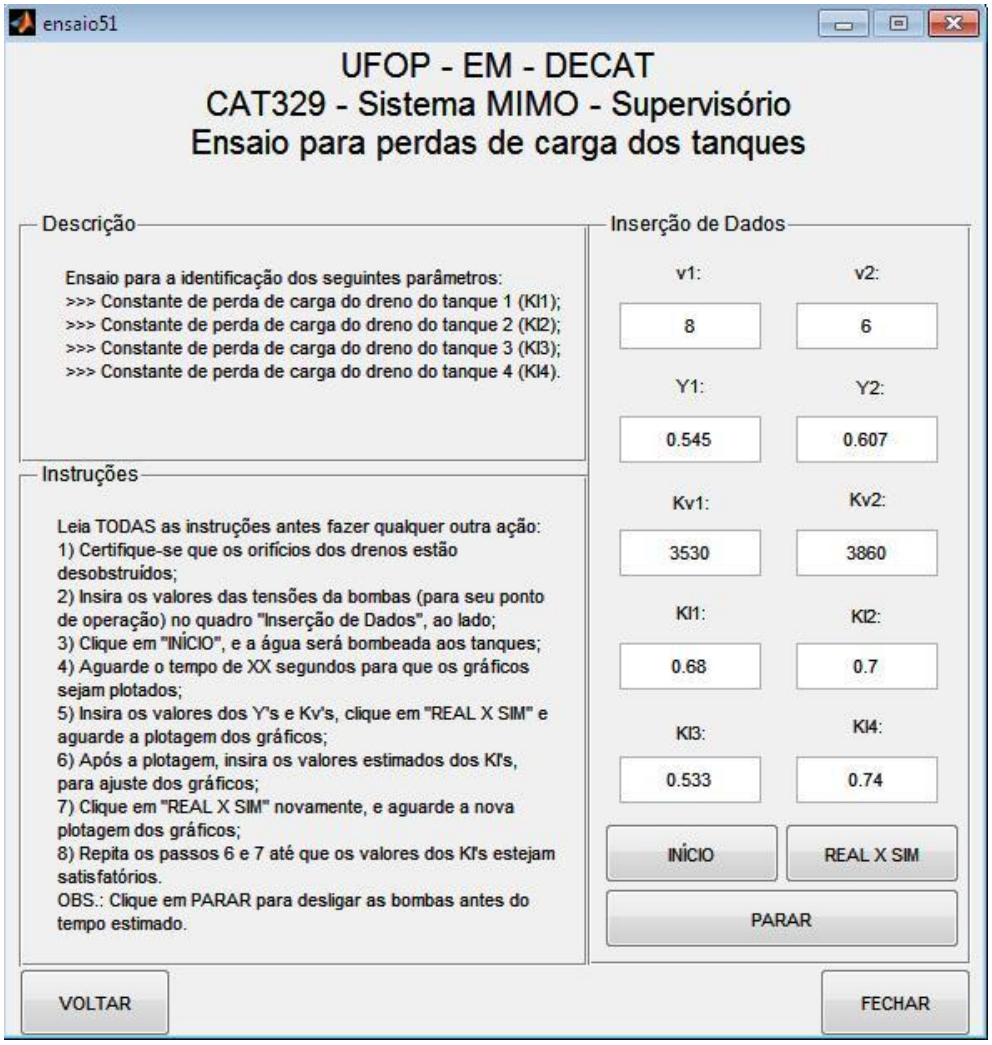


Figura 49 - Tela com K1j's habilitados
Fonte: Elaborada pelo autor.

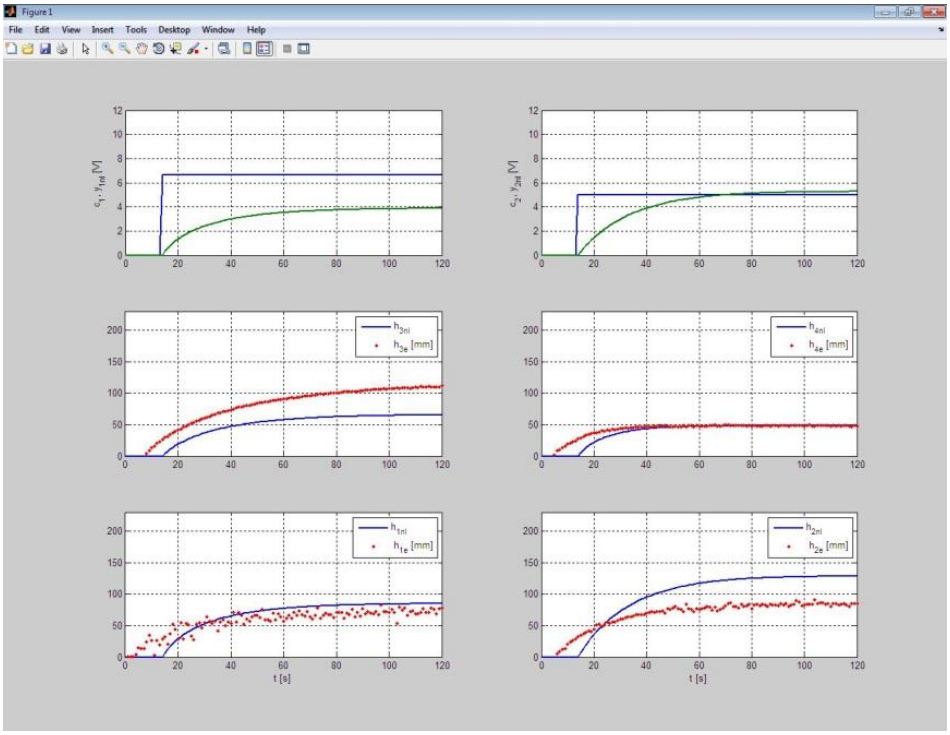


Figura 50 - Dados experimentais e simulação com a perda de carga
Fonte: Elaborada pelo autor.

6. CONSIDERAÇÕES FINAIS

O intuito desse trabalho foi de contribuir no desenvolvimento de bancadas educacionais para prática de controle de processos contínuos com os discentes do curso de Engenharia de Controle e Automação da Escola de Minas – UFOP. O desenvolvimento consistiu na montagem mecânica da bancada, assim como, a confecção dos circuitos de amostragem de nível e de acionamento da bomba, e, um sistema supervisório e especialista.

Após a montagem da bancada, observou-se um aumento do fluxo de discentes no laboratório de Laboratório de Máquinas Elétricas, Interfaceamento e Gestão da Qualidade de Energia, onde está localizada. Diversos trabalhos vêm sendo realizados utilizando o sistema, desde a utilização por docentes para lecionar algumas disciplinas até trabalhos de monografia de discentes. A bancada proporciona uma diversidade de configurações e possibilidade de estudos, tendo como alguns trabalhos realizados pelos discentes o controle do nível tanque utilizando lógica *Fuzzy*, controle de nível com dois tanques ou quatro tanques acoplados em configurações *SISO* ou *MIMO*, entre outros.

Sendo assim o principal objetivo desse trabalho foi alcançado, que era fazer o discente realizar trabalhos práticos na área de teoria de controle, obtendo resultados reais, a fim de compará-los com o a teoria aprendida em sala de aula, além de inserir o aluno em um ambiente de resolução de problemas práticos, incitando essa habilidade que será utilizada na vida profissional do mesmo.

Sugestões de trabalhos futuros:

- Desenvolvimento de mais bancadas, para que se tenha uma gama maior de alunos realizando trabalhos;
- Alterar o encapsulamento do transdutor de pressão para que se possa fazer a calibração do sensor sem que seja necessário abrir a caixa;
- Utilização de um potenciômetro digital no módulo de potência;
- Correção do código do PIC, para melhor visualização no LCD;
- Aperfeiçoamento do Sistema supervisório, o sistema supervisório apresenta um conflito no comando *figure*, quando se retorna a tela anterior, algumas vezes o gráfico em tempo real, continua a ser desenhado, e;
- Desenvolvimento de um hardware para acesso remoto a planta.

REFERÊNCIA BIBLIOGRÁFICA

ALMUTAIRI, N. B.; ZRIBI, M. Sliding mode control of coupled tanks. **Mechatronics**, 16, n. 7, 30 setembro 2006. 427-441.

ASTRÖM, K. J.; ÖSTBERG, A.-B. A teaching Laboratory for process control. **IEEE Control System Magazine**, Boston, Outubro 1986. 37-42.

BRANDÃO, D. P. L. **Electrotecnica Geral**. Lisboa: Fundação Calouste Gulbenkian, v. 1, 1987.

BRASIL, A. N. **Alex Nogueira Basil**. Disponivel em: <http://www.alexbrasil.com.br/_upload/51a0bcc7437fce368e92a7c952d32204.pdf>. Acesso em: 16 junho 2014.

CHRISTY, Y.; KUMAR, D. D. Modeling and Design of Controllers for Interacting Two Tank Hybrid System (ITTHS). **International Journal of Engineering and Innovative Technology**, 3, n. 7, janeiro 2014. 88-91.

FARIA, P. H. L. **Sistema Supervisório e Sistema Especialista para o Controle de Nível de um Processo de Quatro Tanques.** Monografia de Graduação, Universidade Federal de Ouro Preto, Ouro Preto, 2013.

GARCIA, M. A. P.; ANTÓN, J. C.;RODRÍGUEZ, J. C. C; MARTÍN, F. J. F.; ORTEGA, G. J. G. Instrumentación Eletrónica. Madrid: Thomson Editores Spain, 2004. Cap. Criterios para la selecctión de sensores, p. 513-519.

KHEIR, N. A; ASTRÖM, P. K. J.; AUSLANDER, D.; CHECK, K. C.; FRANKLIN, O. F.; MASTEN, M.; RABINS, M. Control Systems Engineering Education. **Automatica**, 32, 1996. 147-166.

LANDER, C. W. **Eletrônica Insdustrial - Aplicações e teorias**. 2. ed. São Paulo: Makron Books, 1996.

LAUBWALD, E. Coupled Tank Systems 1. **Control Systems Principles**, 2005. Disponivel em: <www.control-systems-principles.co.uk>. Acesso em: 30 abril 2014.

LESSA, G. O.; RIBEIRO, V. G.; SILVEIRA, S. R. **Intelivet: Sistema Especialista para Auxílio no Diagnótico Diferencial de Doenças em Animais de Pequeno Porte.** Revista INESC, 2010.

LOPES, M. A. M.; **A importância dos Sitemas Supervisórios no Controle de Processos Industriais.** Monografia de Graduação da Universidade Federal de Ouro Preto, Ouro Preto, 2009.

OHIO, E. M. DC motors in hydraulic pump applications, 27 março 2012. Disponivel em: <www.ohioelectricmotors. com/dc-motors-in-hydraulic-pump-applications-1029>. Acesso em: 30 julho 2014.

OPCFOUNDATION. **Site da OPCFoundation**, 2014. Disponivel em: <http://opcfoundation.org/about/what-is-opc>. Acesso em: 30 julho 2014.

SILVA, H. Medição de nível em tanques pela pressão diferencial. **Controle e Instrumentação**, São Paulo, n. 137, 2008.

VIANA, U. B. **Instrumentação Básica - Pressão e Nivel - Instrumentação**. Serviço Naciona de Aprendizagem Industrial - SENAI. Espirito Santo, p. 123. 1999.

APÊNDICE A – PROGRAMAÇÃO PIC PARA ACIONAMENTO DO MOTOR

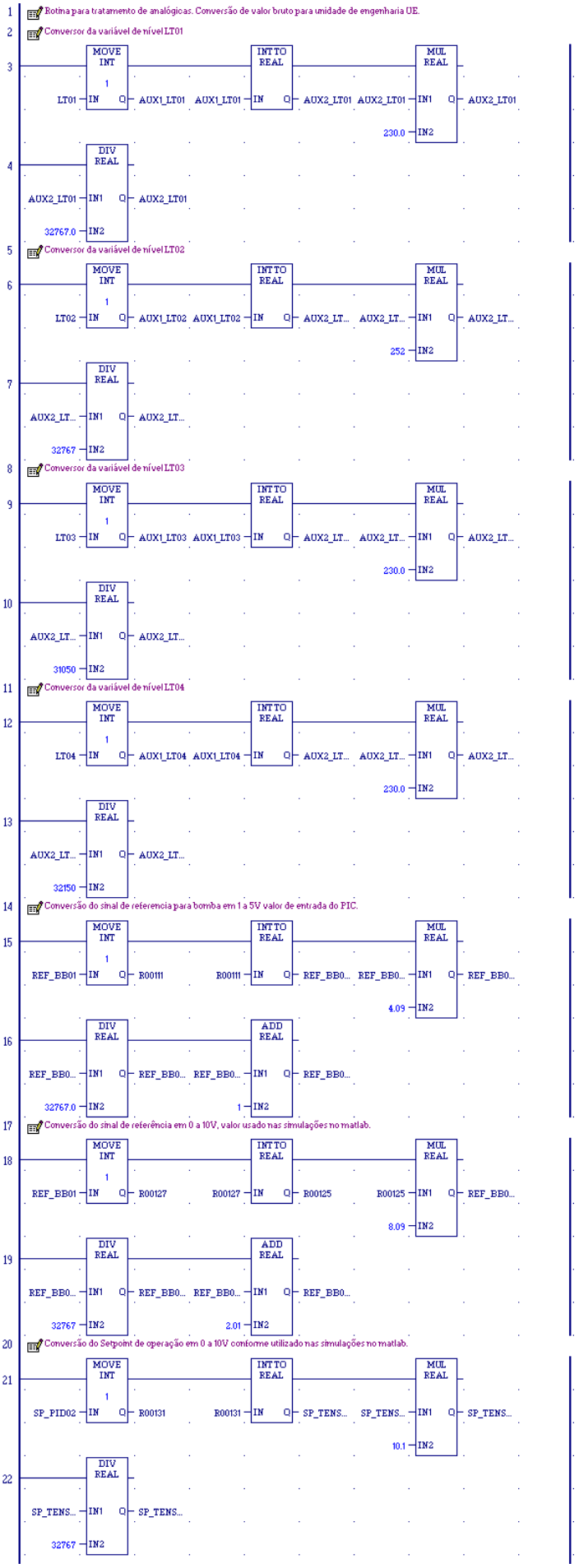
```
#include <16F876A.h>
#device adc = 10
#fuses XT,NOWDT,NOPROTECT,PUT,BROWNOUT,NOLVP,NOCPD,NOWRT
#use delay(clock=4000000)

//--->>Constantes Internas
#define lcd_enable     pin_b6      // pino enable do LCD
#define lcd_rs         pin_b7      // pino rs do LCD
#define lcd_rw         pin_b2      // pino rs do LCD
#define lcd_d4         pin_b5      // pino de dados d4 do LCD
#define lcd_d5         pin_b4      // pino de dados d5 do LCD
#define lcd_d6         pin_b3      // pino de dados d6 do LCD
#define lcd_d7         pin_b2      // pino de dados d7 do LCD

//--->>Definição e inicialização dos Ports
// Estas são as definições dos pinos que o LCD utiliza.
// Definem quais pinos do PIC controlarão os pinos do LCD

#use   fast_io(a)
#use   fast_io(b)
#use   fast_io(c)

#byte porta = 0x05
#byte portb = 0x06
#byte portc = 0x07

#define led_teste pin_b1

#include "lcd4b.c"
```

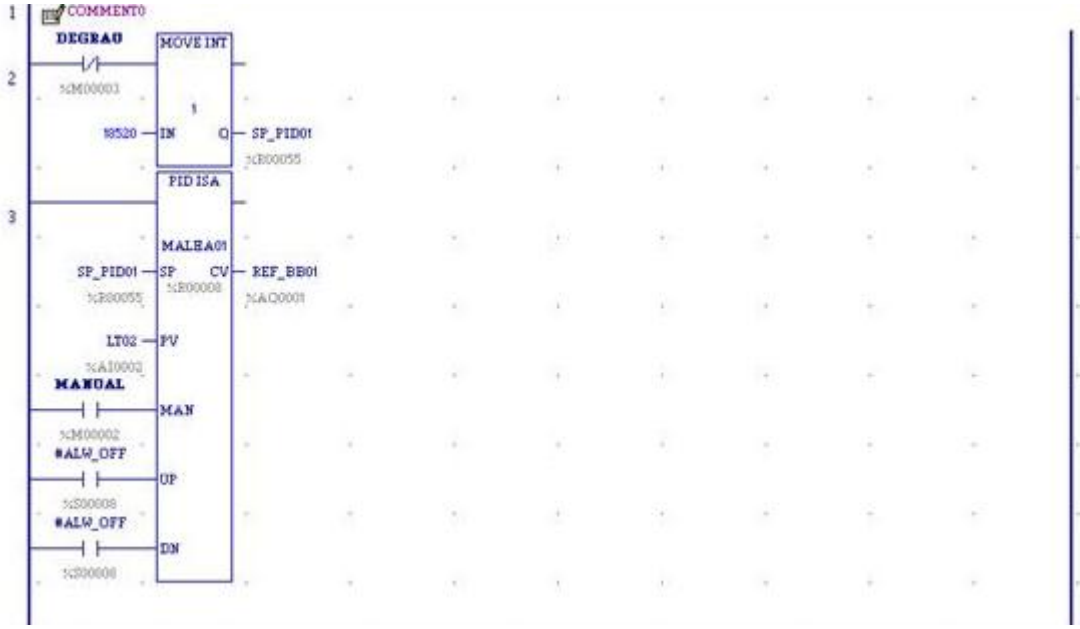
```
//--->>Rotina do PIC
void main()
{

  // Variáveis
unsigned long duty1 = 0;
unsigned long duty2 = 0;
unsigned long PWM1= 0;
unsigned long PWM2= 0;
float        ubomba1 = 0;
float        ubomba2 = 0;

  // Reseta portas
porta = 0;
portb = 0;
portc = 0;

  // configura os tris
  // configuração da direção dos pinos de I/O 0 = saida 1 = entrada
set_tris_b(0b00000000);
  set_tris_c(0b00000000);

  output_high(led_teste);

  // Inicializa o LCD
  lcd_init();
printf(lcd_putc,"\fIniciando...");                delay_ms(1000);
printf(lcd_putc,"\f   PROJ 4 TQS \n  Prof. Cocota");  delay_ms(1000);

  //setup_adc_ports( AN0_TO_AN1 );
setup_adc_ports( AN0_AN1_AN3 ); // define as portas RA0, RA1 como portas analógicas, aparentemente não tem como configurar apenas 2 portas
```

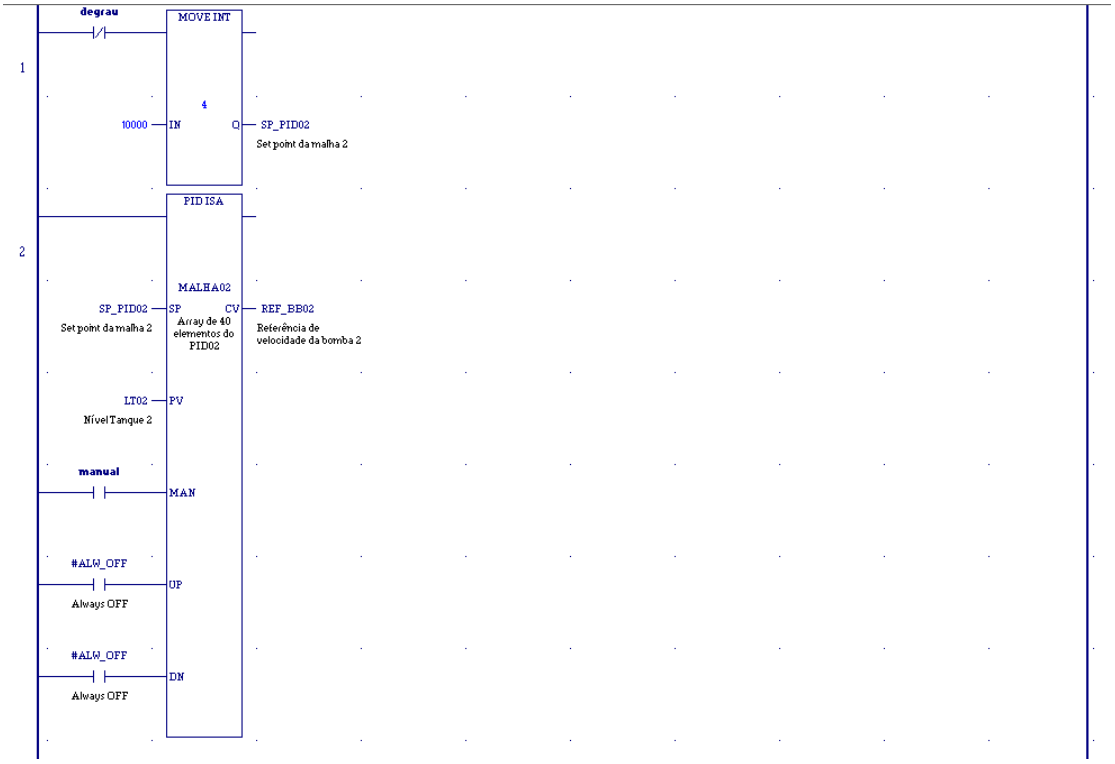
```
setup_adc(adc_clock_div_8 ); // tabela 11-1 pag 117 datasheet do PIC16F876, aparentemente para o maximo de 5 MHz esta deve ser o setup correto
/////// PWM
  setup_timer_2(T2_DIV_BY_1,255,1);   //3921.56 Hz para 4MHz
  setup_ccp1(CCP_PWM);
  setup_ccp2(CCP_PWM);
  set_pwm1_duty(duty1);
  set_pwm2_duty(duty2);

   output_low(led_teste);

while(true)
  {
     output_high(led_teste);
    set_adc_channel(0);        // seleciona o canal AN0 para leitura
delay_us(300);            // tempo necessário para carregar o capacitor do ADC , segundo pág 323 do livro do Fabio Pereira. Tem que confirmar o tempo
duty1 = read_adc();
delay_ms(20);               // atraso para evitar a influência de uma porta analogica em outra ( Tem que confirmar)
     PWM1 = (duty1-205)*1.25;   // Converte para intervalo de leitura entre 1V a 5V
set_pwm1_duty(PWM1);

     delay_ms(20);

set_adc_channel(1);        // seleciona o canal AN1 para leitura
delay_us(300);            // tempo necessário para carregar o capacitor do ADC , segundo pág 323 do livro do Fabio Pereira. Tem que confirmar o tempo
duty2 = read_adc();         // lê o canal 1 do ADC e atribui o valor a variavel si2 ( para calibração )
delay_ms(20);
     PWM2=(duty2-205)*1.25;     // Converte para intervalo de leitura entre 1V a 5V
set_pwm2_duty(PWM2);
```

```
      delay_ms(20);

      output_low(led_teste);

      ubomba1 = ((float)PWM1/1023)*12;
      ubomba2 = ((float)PWM2/1023)*12;

printf(lcd_putc,"\fPW1:%lu", duty1);
      lcd_gotoxy(10,1);
printf(lcd_putc,"U1:%.1f", ubomba1);
      lcd_gotoxy(1,2);
printf(lcd_putc,"PW2:%lu", duty2);
      lcd_gotoxy(10,2);
printf(lcd_putc,"U2:%.1f", ubomba2);

delay_ms(50);
   }

}
```

APÊNDICE B – PROGRAMAÇÃO CLP


1 Rotina para tratamento de analógicas. Conversão de valor bruto para unidade de engenharia UE.
2 Conversor da variável de nível LT01
3 MOVE INT 1 IN Q LT01 AUX1_LT01 AUX1_LT01
INT TO REAL IN Q AUX2_LT01 AUX2_LT01
MUL REAL IN1 IN2 230.0 Q AUX2_LT01
4 DIV REAL IN1 IN2 AUX2_LT01 32767.0 Q AUX2_LT01
5 Conversor da variável de nível LT02
6 MOVE INT 1 IN Q LT02 AUX1_LT02 AUX1_LT02
INT TO REAL IN Q AUX2_LT... AUX2_LT...
MUL REAL IN1 IN2 252 Q AUX2_LT...
7 DIV REAL IN1 IN2 AUX2_LT... 32767 Q AUX2_LT...
8 Conversor da variável de nível LT03
9 MOVE INT 1 IN Q LT03 AUX1_LT03 AUX1_LT03
INT TO REAL IN Q AUX2_LT... AUX2_LT...
MUL REAL IN1 IN2 230.0 Q AUX2_LT...
10 DIV REAL IN1 IN2 AUX2_LT... 31050 Q AUX2_LT...
11 Conversor da variável de nível LT04
12 MOVE INT 1 IN Q LT04 AUX1_LT04 AUX1_LT04
INT TO REAL IN Q AUX2_LT... AUX2_LT...
MUL REAL IN1 IN2 230.0 Q AUX2_LT...
13 DIV REAL IN1 IN2 AUX2_LT... 32150 Q AUX2_LT...
14 Conversão do sinal de referencia para bomba em 1 a 5V valor de entrada do PIC.
15 MOVE INT 1 IN Q REF_BB01 R00111 R00111
INT TO REAL IN Q REF_BB0... REF_BB0...
MUL REAL IN1 IN2 4.09 Q REF_BB0...
16 DIV REAL IN1 IN2 REF_BB0... 32767.0 Q REF_BB0...
ADD REAL IN1 IN2 REF_BB0... 1 REF_BB0... REF_BB0...
17 Conversão do sinal de referência em 0 a 10V, valor usado nas simulações no matlab.
18 MOVE INT 1 IN Q REF_BB01 R00127 R00127
INT TO REAL IN Q R00125 R00125
MUL REAL IN1 IN2 8.09 Q REF_BB0...
19 DIV REAL IN1 IN2 REF_BB0... 32767 Q REF_BB0...
ADD REAL IN1 IN2 REF_BB0... 2.01 Q REF_BB0... REF_BB0...
20 Conversão do Setpoint de operação em 0 a 10V conforme utilizado nas simulações no matlab.
21 MOVE INT 1 IN Q SP_PID02 R00131 R00131
INT TO REAL IN Q SP_TENS... SP_TENS...
MUL REAL IN1 IN2 10.1 Q SP_TENS...
22 DIV REAL IN1 IN2 SP_TENS... 32767 Q SP_TENS...


1
COMMENT0
DEGRAU
%M00003
MOVE INT
1
19520
IN
Q
SP_PID01
%R00055
PID ISA
2
3
MALHA01
SP_PID01
%R00055
SP
CV
%R00008
REF_BB01
%AQ0001
LT02
%AI0002
PV
MANUAL
%M00002
MAN
#ALW_OFF
%S00009
UP
#ALW_OFF
%S00008
DN


degrau
MOVE INT
1
4
10000
IN
Q
SP_PID02
Set point da malha 2
PID ISA
2
MALHA02
SP_PID02
Set point da malha 2
SP
CV
Array de 40 elementos do PID02
REF_BB02
Referência de velocidade da bomba 2
LT02
Nível Tanque 2
PV
manual
MAN
#ALW_OFF
Always OFF
UP
#ALW_OFF
Always OFF
DN

APÊNDICE C- PROGRAMAÇÃO DO SUPERVISÓRIO EM MATLAB®

```
function varargout = tanque(varargin)
% TANQUE MATLAB code for tanque.fig
%      TANQUE, by itself, creates a new TANQUE or raises the existing
%      singleton*.
%
%      H = TANQUE returns the handle to a new TANQUE or the handle to
%      the existing singleton*.
%
%      TANQUE('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in TANQUE.M with the given input arguments.
%
%      TANQUE('Property','Value',...) creates a new TANQUE or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before tanque_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to tanque_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help tanque

% Last Modified by GUIDE v2.5 28-Aug-2013 20:31:34

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
'gui_Singleton',  gui_Singleton, ...
'gui_OpeningFcn', @tanque_OpeningFcn, ...
'gui_OutputFcn',  @tanque_OutputFcn, ...
'gui_LayoutFcn',  [] , ...
'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before tanque is made visible.
function tanque_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to tanque (see VARARGIN)
```

```
% Choose default command line output for tanque
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes tanque wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = tanque_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

set(handles.text25,'string',000);
set(handles.text37,'string',000);
set(handles.text39,'string',000);
set(handles.text40,'string',000);
set(handles.text41,'string',000);
set(handles.text42,'string',000);
set(handles.text43,'string',5.95);
set(handles.text44,'string',4.7625);
set(handles.text45,'string',9.751);
set(handles.text46,'string',000);
set(handles.text47,'string',000);

set(handles.text24,'string',000);
set(handles.text14,'string',000);
set(handles.text12,'string',000);
set(handles.text11,'string',000);
set(handles.text10,'string',000);
set(handles.text9,'string',000);
set(handles.text8,'string',000);
set(handles.text7,'string',000);

% Gerando e lendo dados do KEPServerEX
hostInfo = opcserverinfo('localhost');
da = opcda ('localhost', 'Kepware.KEPServerEX.V5');
connect(da);
grupo=addgroup(da,'dados');
itens={'Channel1.Device1.Nivel1','Channel1.Device1.Nivel2','Channel1.Device
1.Nivel3','Channel1.Device1.Nivel4','Channel1.Device1.RefBomba1','Channel1.
Device1.RefBomba2'};
item=additem(grupo, itens);

while 1

% Gerando graficos de niveis
handles.axes1=subplot(2,2,1);
plot(cputime,item(3).Value)
gridon
axis([cputime-100 cputime+10 -10 500])
```

```
drawnow
holdon;

    handles.axes1=subplot(2,2,2);
plot(cputime,item(4).Value)
gridon
axis([cputime-100 cputime+10 -10 500])
drawnow
holdon;

    handles.axes1=subplot(2,2,3);
plot(cputime,item(1).Value)
gridon
axis([cputime-100 cputime+10 -10 500])
drawnow
holdon;

    handles.axes1=subplot(2,2,4);
plot(cputime,item(2).Value)
gridon
axis([cputime-100 cputime+10 -10 500])
drawnow
holdon;

% Tensoes nas bombas

bomba1_plc=item(5).Value/3276.7;
bomba1_volts=bomba1_plc/0.833;
auxbomba1=num2str(bomba1_volts);
set(handles.text62,'string',auxbomba1);

bomba2_plc=item(6).Value/3276.7;
bomba2_volts=bomba2_plc/0.833;
auxbomba2=num2str(bomba2_volts);
set(handles.text63,'string',auxbomba2);

% Supervisão dos níveis:
nivel1_aux=num2str(item(1).Value);
set(handles.campo_nivel1,'string',nivel1_aux);
nivel2_aux=num2str(item(2).Value);
set(handles.campo_nivel2,'string',nivel2_aux);
nivel3_aux=num2str(item(3).Value);
set(handles.campo_nivel3,'string',nivel3_aux);
nivel4_aux=num2str(item(4).Value);
set(handles.campo_nivel4,'string',nivel4_aux);

end

% --- Executes on button press in voltar.
function voltar_Callback(hObject, eventdata, handles)
closeall;
clearall;
inicial;
% hObject    handle to voltar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton6.
```

```
function pushbutton6_Callback(hObject, eventdata, handles)
creditos;
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton8.
function pushbutton8_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in gerar3.
function gerar3_Callback(hObject, eventdata, handles)
% hObject    handle to gerar3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in gerar4.
function gerar4_Callback(hObject, eventdata, handles)
% hObject    handle to gerar4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in gerar1.
function gerar1_Callback(hObject, eventdata, handles)
% hObject    handle to gerar1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on button press in gerar2.
function gerar2_Callback(hObject, eventdata, handles)
% hObject    handle to gerar2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

APÊNDICE D – PROGRAMA SISTEMA ESPECIALISTA EM MATLAB®

- Ensaio para (γ's) e (k's)

```
function varargout = ensaio01(varargin)
% ENSAIO01 MATLAB code for ensaio01.fig
%      ENSAIO01, by itself, creates a new ENSAIO01 or raises the existing
%      singleton*.
%
%      H = ENSAIO01 returns the handle to a new ENSAIO01 or the handle to
%      the existing singleton*.
%
%      ENSAIO01('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in ENSAIO01.M with the given input
arguments.
%
%      ENSAIO01('Property','Value',...) creates a new ENSAIO01 or raises
the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before ensaio01_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to ensaio01_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help ensaio01

% Last Modified by GUIDE v2.5 03-Sep-2013 17:01:20

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
'gui_Singleton',  gui_Singleton, ...
'gui_OpeningFcn', @ensaio01_OpeningFcn, ...
'gui_OutputFcn',  @ensaio01_OutputFcn, ...
'gui_LayoutFcn',  [] , ...
'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before ensaio01 is made visible.
function ensaio01_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to ensaio01 (see VARARGIN)

% Choose default command line output for ensaio01
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes ensaio01 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = ensaio01_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in voltar.
function voltar_Callback(hObject, eventdata, handles)
closeall;
parametros;
% hObject    handle to voltar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in fim.
function fim_Callback(hObject, eventdata, handles)

% Conexão com o KEPServer:
hostInfo = opcserverinfo('localhost');
da = opcda ('localhost', 'Kepware.KEPServerEX.V5');
connect(da);
grupo=addgroup(da,'dados');
itens={'Channel1.Device1.Nivel1','Channel1.Device1.Nivel2','Channel1.Device
1.Nivel3','Channel1.Device1.Nivel4','Channel1.Device1.RefBomba1','Channel1.
Device1.RefBomba2'};
item=additem(grupo, itens);
```

```
% Envio das variáveis para o PLC (através do KEPServer):
write(item(5), 0)
write(item(6), 0)
% hObject    handle to fim (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
creditos;
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executa quando o botão INÍCIO é pressionado:
function inicio_Callback(hObject, eventdata, handles)

% Conexão com o KEPServer:
hostInfo = opcserverinfo('localhost');
da = opcda ('localhost', 'Kepware.KEPServerEX.V5');
connect(da);
grupo=addgroup(da);
itens={'Channel1.Device1.Nivel1','Channel1.Device1.Nivel2','Channel1.Device
1.Nivel3','Channel1.Device1.Nivel4','Channel1.Device1.RefBomba1','Channel1.
Device1.RefBomba2'};
item=additem(grupo, itens);

% Pausa antes de envio do sinal para a bomba,
% utilizada para melhor visualização dos dados no gráfico:
%pause(1);

% Aquisição dos dados inseridos:
v1 = get(handles.campo_v1,'string');
v2 = get(handles.campo_v2,'string');

% Transformando v1 e v2 em número:
v1_aux1 = str2double(v1);
v2_aux1 = str2double(v2);

% Inserção de dados:

if v1_aux1<0||v1_aux1>9||v2_aux1<0||v2_aux1>9
uiwait(msgbox('ERRO!!!! Insira uma tensão entre 0V e 12V!', 'Erro'));

else
% Transformação da variável v1 para o PLC:
v1_aux2 = v1_aux1*0.833;
v1_plc = v1_aux2*3276.7;

% Transformação da variável v2 para o PLC:
v2_aux2 = v2_aux1*0.833;
v2_plc = v2_aux2*3276.7;

% Envio das variáveis para o PLC (através do KEPServer):
write(item(5), v1_plc)
write(item(6), v2_plc)
```

```
end

%Definição do tempo do gráfico
logDuration = 120;
logRate = 1.0;
numrecords=ceil(logDuration/logRate);
tempo = 0:1:119;
tempo_grafico = tempo';
set(grupo, 'UpdateRate', logRate, 'RecordsToAcquire', numrecords);
start(grupo)
wait(grupo)
s = getdata(grupo);
[logIDs, logVal, logQual, logTime, logEvtTime] = opcstruct2array(s
,'double');

%Plotando ensaio
figure
for i = 1:1:4
subplot(2,2,i);
plot(tempo_grafico, logVal(:,i), '*');
legend(logIDs(i));
axis ([0 120 0 200]);
end

%Salvando os dados em ensaio01.mat
saveensaio01.mat

% hObject    handle to inicio (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


function campo_v1_Callback(hObject, eventdata, handles)
% hObject    handle to campo_v1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of campo_v1 as text
%        str2double(get(hObject,'String')) returns contents of campo_v1 as
a double


% --- Executes during object creation, after setting all properties.
function campo_v1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to campo_v1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end


function campo_v2_Callback(hObject, eventdata, handles)
% hObject    handle to campo_v2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of campo_v2 as text
%        str2double(get(hObject,'String')) returns contents of campo_v2 as
a double


% --- Executes during object creation, after setting all properties.
function campo_v2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to campo_v2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in fechar.
function fechar_Callback(hObject, eventdata, handles)
closeall;
% hObject    handle to fechar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes button press in inicio.
function teste_Callback(hObject, eventdata, handles)
%Aquisição de dados
hostInfo = opcserverinfo('localhost');
da = opcda ('localhost', 'Kepware.KEPServerEX.V5');
connect(da);
grupo = addgroup(da);
itemIDs = {'Channel1.Device1.Nivel1',
'Channel1.Device1.Nivel2','Channel1.Device1.Nivel3','Channel1.Device1.Nivel
4','Channel1.Device1.RefBomba1','Channel1.Device1.RefBomba2'};
item = additem (grupo, itemIDs);

% Aquisição dos dados inseridos:
v1 = get(handles.campo_v1,'string');
v2 = get(handles.campo_v2,'string');

% Transformando v1 e v2 em número:
v1_aux1 = str2double(v1);
v2_aux1 = str2double(v2);

% Inserção de dados:

if v1_aux1<0||v1_aux1>9||v2_aux1<0||v2_aux1>9
uiwait(msgbox('ERRO!!!! Insira uma tensão entre 0V e 12V!', 'Erro'));

else
% Transformação da variável v1 para o PLC:
v1_aux2 = v1_aux1*0.833;
v1_plc = v1_aux2*3276.7;

% Transformação da variável v2 para o PLC:
```

```
v2_aux2 = v2_aux1*0.833;
v2_plc = v2_aux2*3276.7;

% Envio das variáveis para o PLC (através do KEPServer):
write(item(5), v1_plc)
write(item(6), v2_plc)
end


% hObject    handle to inicio (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

- Ensaio para k_{1j}'s

```
function varargout = ensaio51(varargin)
% ENSAIO51 MATLAB code for ensaio51.fig
%      ENSAIO51, by itself, creates a new ENSAIO51 or raises the existing
%      singleton*.
%
%      H = ENSAIO51 returns the handle to a new ENSAIO51 or the handle to
%      the existing singleton*.
%
%      ENSAIO51('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in ENSAIO51.M with the given input
arguments.
%
%      ENSAIO51('Property','Value',...) creates a new ENSAIO51 or raises
the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before ensaio51_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to ensaio51_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help ensaio51

% Last Modified by GUIDE v2.5 05-Sep-2013 11:11:30

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
'gui_Singleton',  gui_Singleton, ...
'gui_OpeningFcn', @ensaio51_OpeningFcn, ...
'gui_OutputFcn',  @ensaio51_OutputFcn, ...
'gui_LayoutFcn',  [] , ...
'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before ensaio51 is made visible.
function ensaio51_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to ensaio51 (see VARARGIN)

% Choose default command line output for ensaio51
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes ensaio51 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = ensaio51_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in inicio.
function inicio_Callback(hObject, eventdata, handles)

set(handles.campo_Y1, 'Enable',' on');
set(handles.campo_Y2, 'Enable',' on');
set(handles.campo_K1, 'Enable',' on');
set(handles.campo_K2, 'Enable',' on');

% Conexão com o KEPServer:
hostInfo = opcserverinfo('localhost');
da = opcda ('localhost', 'Kepware.KEPServerEX.V5');
connect(da);
grupo=addgroup(da);
itens={'Channel1.Device1.Nivel1','Channel1.Device1.Nivel2','Channel1.Device
1.Nivel3','Channel1.Device1.Nivel4','Channel1.Device1.RefBomba1','Channel1.
Device1.RefBomba2'};
item=additem(grupo, itens);


% Aquisição dos dados inseridos:
v1_aux00 = get(handles.campo_v1,'string');
v2_aux00 = get(handles.campo_v2,'string');
```

```
% Transformando v1 e v2 em número:
v1_aux01 = str2double(v1_aux00);
v2_aux01 = str2double(v2_aux00);

% Inserção de dados:

if v1_aux01<0||v1_aux01>12||v2_aux01<0||v2_aux01>12
uiwait(msgbox('ERRO!!!! Insira uma tensão entre 0V e 12V!', 'Erro'));

else
% Transformação da variável v1 para o PLC:
v1_aux02 = v1_aux01*0.833;
v1_plc = v1_aux02*3276.7;

% Transformação da variável v2 para o PLC:
v2_aux02 = v2_aux01*0.833;
v2_plc = v2_aux02*3276.7;

% Envio das variáveis para o PLC (através do KEPServer):
write(item(5), v1_plc)
write(item(6), v2_plc)
end

%Definição do tempo do gráfico
logDuration = 120;
logRate = 1.0;
numrecords=ceil(logDuration/logRate);
tempo = 0:1:119;
tempo_grafico = tempo';
set(grupo, 'UpdateRate', logRate, 'RecordsToAcquire', numrecords);
start(grupo)
wait(grupo)
s = getdata(grupo);
[logIDs, logVal, logQual, logTime, logEvtTime] = opcstruct2array(s
,'double');

%Plotando ensaio
figure(2)
for i = 1:1:4
subplot(2,2,i);
plot(tempo_grafico, logVal(:,i), '*');
legend(logIDs(i));
axis ([0 120 0 200]);
end

%Salvando os dados em perda_de_carga.mat
saveperda_de_carga.mat

% hObject    handle to inicio (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in REAL X SIM
function realxsim_Callback(hObject, eventdata, handles)

set(handles.campo_kl1, 'Enable',' on');
set(handles.campo_kl2, 'Enable',' on');
```

```
set(handles.campo_kl3, 'Enable',' on');
set(handles.campo_kl4, 'Enable',' on');

% Aquisição dos dados inseridos:
v1_aux00 = get(handles.campo_v1,'string');
v2_aux00 = get(handles.campo_v2,'string');
Y1 = get(handles.campo_Y1,'string');
Y2 = get(handles.campo_Y2,'string');
K1 = get(handles.campo_K1,'string');
K2 = get(handles.campo_K2,'string');
Kl1_aux0 = get(handles.campo_kl1,'string');
Kl2_aux0 = get(handles.campo_kl2,'string');
Kl3_aux0 = get(handles.campo_kl3,'string');
Kl4_aux0 = get(handles.campo_kl4,'string');

% Transformando em número:
v1_aux01 = str2double(v1_aux00);
v2_aux01 = str2double(v2_aux00);
Y1_aux1 = str2double(Y1);
Y2_aux1 = str2double(Y2);
K1_aux1 = str2double(K1);
K2_aux1 = str2double(K2);
Kl1_aux1 = str2double(Kl1_aux0);
Kl2_aux1 = str2double(Kl2_aux0);
Kl3_aux1 = str2double(Kl3_aux0);
Kl4_aux1 = str2double(Kl4_aux0);

% OBS.: Abaixo, haverá viariáveis definidas de duas formas:
%       1) normalmente, para ser usada dentro do próprio GUI;
%       2) em "assignin", para ser enviada ao workspace e ser
%       utilizada pelo simulink.
%       Algumas, como são usadas nos dois casos, são definidas
%       das duas formas.

% Inicialização
tfinal = 120;                   % Tempo de simulação
assignin('base','tfinal',120)   % Tempo de simulação
assignin('base','dt',1)         % Intervalo de amostragem para o workspace

% Parâmetros universais
assignin('base','g',9787.9)     % Valor da aceleração da gravidade [mm/s^2]


% Sinal de referência 1 - tanques 1 e 4
assignin('base','v10_ref',0)            % tensão de referência 1 inicial
[V]
assignin('base','v1_ref',v1_aux01/1.2)   % tensão de referência 1 [V] 4.93
assignin('base','t1_ref',14)            % Instante do degrau
% Sinal de referência 2 - tanques 2 e 3
assignin('base','v20_ref',0)            % tensão de referência 2 inicial
[V]
assignin('base','v2_ref',v2_aux01/1.2)   % tensão de referência 2 [V] 5.1
assignin('base','t2_ref',14)            % Instante do degrau

assignin('base','r_min',0)  % Tensão mínima da saída A do PLC
assignin('base','r_max',10) % Tensão máxima da saída A do PLC [V]

%Na verdade o PLC envia 0 a 20 mA, e na entrada do modulo um divisor
%de tensão converte este sinal para 0 a 5 V, e em sua saída PWM
%a tensão varia de 0 a 12 Vdc nos polos do motor da bomba
```

```
% Parâmetros do tanque
assignin('base','rd1',2.975)    % raio do dreno 1 [mm]
assignin('base','rd2',2.775)    % raio do dreno 2 [mm]
assignin('base','rd3',2.185)    % raio do dreno 3 [mm]
assignin('base','rd4',2.38125)  % raio do dreno 4 [mm]

assignin('base','kl1',Kl1_aux1)    % cte de perda de carga pelo dreno para
o tq 1
assignin('base','kl2',Kl2_aux1)    % cte de perda de carga pelo dreno para
o tq 2
assignin('base','kl3',Kl3_aux1)    % cte de perda de carga pelo dreno para
o tq 3
assignin('base','kl4',Kl4_aux1)    % cte de perda de carga pelo dreno para
o tq 4
kl1 = Kl1_aux1;
kl2 = Kl2_aux1;
kl3 = Kl3_aux1;
kl4 = Kl4_aux1;

assignin('base','A1',pi*566.44) % Área do tanque 1 [mm^2]
assignin('base','A2',pi*566.44) % Área do tanque 2 [mm^2]
assignin('base','A3',pi*566.44) % Área do tanque 3 [mm^2]
assignin('base','A4',pi*566.44) % Área do tanque 4 [mm^2]

assignin('base','a1',kl1*pi*8.850625)   % Área do orificio de saida do
tanque 1 [cm^2]
assignin('base','a2',kl2*pi*7.700625)   % Área do orificio de saida do
tanque 2 [cm^2]
assignin('base','a3',kl3*pi*4.774225)   % Área do orificio de saida do
tanque 3 [cm^2]
assignin('base','a4',kl4*pi*5.67035156) % Área do orificio de saida do
tanque 4 [cm^2]

assignin('base','h_min',0)      % Nível mínimo
assignin('base','h_max',230)    % Nível máximo [mm]
assignin('base','h10',0)        %Altura h1 inicial
assignin('base','h20',0)        %Altura h2 inicial
assignin('base','h30',0)        %Altura h3 inicial
assignin('base','h40',0)        %Altura h4 inicial

% Parâmetros do modulo
assignin('base','kmod',1.2) % Ganho do módulo de potencia
assignin('base','v_min',0)  % tensão mínima de saída
assignin('base','v_max',12) % tensão máxima de saída

% Parâmetros da bomba
assignin('base','k1',K1_aux1)   % Ganho da bomba 1 (mm^3)/(V*s)
assignin('base','k2',K2_aux1)   % Ganho da bomba 2
k1 = K1_aux1;                   % Ganho da bomba 1 (mm^3)/(V*s)
k2 = K2_aux1;                   % Ganho da bomba 2
assignin('base','q1_min',0)     % Vazão mínima
assignin('base','q1_max',k1*12) % Vazão máxima (cm^3/s)
assignin('base','q2_min',0)     % Vazão mínima
assignin('base','q2_max',k2*12) % Vazão máxima (cm^3/s)

% Parâmetros da valvula Y1 e Y2
assignin('base','gamma1',Y1_aux1)   % Relacionado com o estrangulamento do
fluxo da bomba 1
```

```
assignin('base','gamma2',Y2_aux1)    % Relacionado com o estrangulamento do
fluxo da bomba 2

% Parâmetros do sensor
assignin('base','kc1',10.51/230)     % Ganho do sensor - faixa de 0 a 30 mm
assignin('base','kc2',9.43/230)
assignin('base','kc3',9.79/230)
assignin('base','kc4',10.14/230)
assignin('base','ymax',10.6)         % Tensão máxima da saída do sensor de
pressão [V]
assignin('base','ymin',0)            % Tensão mínima da saída do sensor de
pressão [V]


% Opções de impressão
graficos   = 'OF';
diagramas  ='OF';
parametros = 'OF';

% Simulação

sim('mod_NL_Lucas',tfinal);

load('perda_de_carga');

% Plotagem dos gráficos

figure(1)

subplot(3,2,1);
plot(t1,r1,t1,y1,'Linew',2);
axis([0,tfinal,0,12]);
ylabel('c_1, y_{1nl} [V]')
gridon;
drawnow;

subplot(3,2,2);
plot(t2,r2,t2,y2,'Linew',2);
axis([0,tfinal,0,12]);
ylabel('c_2, y_{2nl} [V]')
gridon;
drawnow;

subplot(3,2,3);
plot(t2,h3,'Linew',2);
holdon;
plot(logVal(:,3),'red .');
holdon;
axis([0,tfinal,0,23*10]);
legend('h_{3nl}','h_{3e} [mm]')
gridon
drawnow;

subplot(3,2,4);
plot(t1,h4,'Linew',2);
holdon;
plot(logVal(:,4),'red .');
holdon;
axis([0,tfinal,0,23*10]);
```

```
legend('h_{4nl}','h_{4e} [mm]')
gridon
drawnow;

subplot(3,2,5);
plot(t1,h1,'Linew',2);
holdon;
plot(logVal(:,1),'red .');
holdon;
axis([0,tfinal,0,23*10]);
legend('h_{1nl}',' h_{1e} [mm]')
xlabel('t [s]')
gridon;
drawnow;

subplot(3,2,6);
plot(t2,h2,'Linew',2);
holdon;
plot(logVal(:,2),'red .');
holdon;
axis([0,tfinal,0,23*10]);
legend('h_{2nl}',' h_{2e} [mm]')
xlabel('t [s]')
gridon;
drawnow;

shg

% hObject    handle to realxsim (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)



% --- Executes on button press in parar.
function parar_Callback(hObject, eventdata, handles)
hostInfo = opcserverinfo('localhost');
da = opcda ('localhost', 'Kepware.KEPServerEX.V5');
connect(da);
grupo=addgroup(da,'dados');
itens={'Channel1.Device1.Nivel1','Channel1.Device1.Nivel2','Channel1.Device
1.Nivel3','Channel1.Device1.Nivel4','Channel1.Device1.RefBomba1','Channel1.
Device1.RefBomba2'};
item=additem(grupo, itens);

% Envio das variáveis para o PLC (através do KEPServer):
write(item(5), 0)
write(item(6), 0)
% hObject    handle to parar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on button press in voltar.
function voltar_Callback(hObject, eventdata, handles)
closeall;
parametros;

% hObject    handle to voltar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on button press in fechar.
function fechar_Callback(hObject, eventdata, handles)
close all;
% hObject    handle to fechar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


function campo_v1_Callback(hObject, eventdata, handles)
% hObject    handle to campo_v1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of campo_v1 as text
%        str2double(get(hObject,'String')) returns contents of campo_v1 as
a double


% --- Executes during object creation, after setting all properties.
function campo_v1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to campo_v1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end

function campo_v2_Callback(hObject, eventdata, handles)
% hObject    handle to campo_v2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of campo_v2 as text
%        str2double(get(hObject,'String')) returns contents of campo_v2 as
a double

% --- Executes during object creation, after setting all properties.
function campo_v2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to campo_v2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end


function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to campo_v2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of campo_v2 as text
%        str2double(get(hObject,'String')) returns contents of campo_v2 as
a double


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to campo_v2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end



function campo_Y1_Callback(hObject, eventdata, handles)
% hObject    handle to campo_Y1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of campo_Y1 as text
%        str2double(get(hObject,'String')) returns contents of campo_Y1 as
a double


% --- Executes during object creation, after setting all properties.
function campo_Y1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to campo_Y1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end



function campo_Y2_Callback(hObject, eventdata, handles)
% hObject    handle to campo_Y2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of campo_Y2 as text
%        str2double(get(hObject,'String')) returns contents of campo_Y2 as
a double


% --- Executes during object creation, after setting all properties.
```

```
function campo_Y2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to campo_Y2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end



function campo_K1_Callback(hObject, eventdata, handles)
% hObject    handle to campo_K1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of campo_K1 as text
%        str2double(get(hObject,'String')) returns contents of campo_K1 as
a double


% --- Executes during object creation, after setting all properties.
function campo_K1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to campo_K1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end

function campo_K2_Callback(hObject, eventdata, handles)
% hObject    handle to campo_K2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of campo_K2 as text
%        str2double(get(hObject,'String')) returns contents of campo_K2 as
a double

% --- Executes during object creation, after setting all properties.
function campo_K2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to campo_K2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end
```

```
function campo_kl1_Callback(hObject, eventdata, handles)
% hObject    handle to campo_kl1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of campo_kl1 as text
%        str2double(get(hObject,'String')) returns contents of campo_kl1 as
a double

% --- Executes during object creation, after setting all properties.
function campo_kl1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to campo_kl1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end

function campo_kl3_Callback(hObject, eventdata, handles)
% hObject    handle to campo_kl3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of campo_kl3 as text
%        str2double(get(hObject,'String')) returns contents of campo_kl3 as
a double

% --- Executes during object creation, after setting all properties.
function campo_kl3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to campo_kl3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end

function campo_kl2_Callback(hObject, eventdata, handles)
% hObject    handle to campo_kl2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of campo_kl2 as text
%        str2double(get(hObject,'String')) returns contents of campo_kl2 as
a double

% --- Executes during object creation, after setting all properties.
function campo_kl2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to campo_kl2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end

function campo_kl4_Callback(hObject, eventdata, handles)
% hObject    handle to campo_kl4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of campo_kl4 as text
%        str2double(get(hObject,'String')) returns contents of campo_kl4 as
a double

% --- Executes during object creation, after setting all properties.
function campo_kl4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to campo_kl4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end
```