UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CÂMPUS CORNÉLIO PROCÓPIO
CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

RODRIGO ALLAN SILVA

DESENVOLVIMENTO DE MÓDULO DE CONTROLE DE NÍVEL

TRABALHO DE CONCLUSÃO DE CURSO

CORNÉLIO PROCÓPIO

JANEIRO/2014

RODRIGO ALLAN SILVA

DESENVOLVIMENTO DE MÓDULO DE CONTROLE DE NÍVEL

Trabalho de Conclusão de Curso de graduação, apresentado à Universidade Tecnológica Federal do Paraná como requisito parcial para a obtenção do título de “Bacharel em Engenharia Elétrica”.
Orientador: Prof. Dr. Paulo Rogério Scalassara.

CORNÉLIO PROCÓPIO

JANEIRO/2014

RODRIGO ALLAN SILVA

DESENVOLVIMENTO DE MÓDULO DE CONTROLE DE NÍVEL

Trabalho de conclusão de curso apresentado às 14h do dia 27 de Janeiro de 2014 como requisito parcial para a obtenção do título de Engenheiro Eletricista no Programa de Graduação em Engenharia Industrial Elétrica da Universidade Tecnológica Federal do Paraná. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Prof. Dr. Paulo Rogério Scalassara
Professor(a) Orientador(a)
UTFPR/ Campus Cornélio Procópio

_____
Prof. Dr. Cristiano Marcos Agulhari
Professor(a) Convidado(a)
UTFPR/ Campus Cornélio Procópio

_____
Prof. Dr. Rodrigo Rodrigues Sumar
Professor(a) Convidado(a)
UTFPR/ Campus Cornélio Procópio

A Folha de Aprovação assinada encontra-se na Coordenação do Curso.

Aos meus pais e à minha irmã, meus alicerces para toda a vida.

AGRADECIMENTOS

Em primeiro lugar a Deus, a quem confio meus maiores segredos. O grande responsável por iluminar meu caminho e me abençoar com graças maravilhosas.

A minha mãe Tânia Regina Ferraz, ao meu pai Osmar Aparecido da Silva e à minha irmã Alyne Ferraz Silva, pessoas fundamentais para o meu desenvolvimento humano e grandes apoiadores das decisões que tomei ao longo de minha vida.

Ao meu orientador Prof. Dr. Paulo Rogério Scalassara pela dedicação e contribuição, com o grande conhecimento que possui, para a realização deste trabalho.

Ao Prof. Dr. Bruno Angélico pela colaboração em vários pontos importantes deste presente trabalho.

Aos membros da banca examinadora Prof. Dr. Rodrigo Rodrigues Sumar e Prof. Dr. Cristiano Marcus Agulhari pelas sugestões dadas que contribuíram fortemente no desenvolvimento deste projeto.

A Universidade Tecnológica Federal do Paraná e aos professores do curso de Engenharia Elétrica pelo ensino de qualidade que oferecem o qual foi base forte para aquisição de conhecimento nesta área profissional.

Aos amigos que fiz durante o curso de Engenharia Elétrica, pelos momentos de estudo e pelos momentos de descontração que tivemos dentro e fora desta Universidade. São pessoas de que sempre me lembrarei.

Ao amigo Clayton Luiz Graciola pela grande ajuda oferecida durante o processo de projeto do sistema de controle aplicado a este trabalho.

“O som da minha vida: A engenharia.
A engenharia da minha vida: A música.
Jesus Cristo, o meu engenheiro e
músico mentor.”

(Euclides Ferreira)

RESUMO

SILVA, Rodrigo Allan. DESENVOLVIMENTO DE MODULO DIDÁTICO DE CONTROLE DE NÍVEL. 72 f. Trabalho de Conclusão de Curso – Curso de Graduação em Engenharia Elétrica, Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2014.

Este trabalho propõe a construção de uma plataforma didática composta por reservatórios de água para aplicação de diversas estratégias de controle para nível, bem como o projeto de um de um controle PI de nível fazendo-se uso do *software* Matlab e do kit de desenvolvimento ARDUINO UNO, desenvolvendo um supervisório que, através de comunicação serial, envie e receba dados de comando da planta assim como permita a visualização gráfica da resposta do controle por parte do usuário através de uma interface gráfica. A plataforma construída ficará a disposição da UTFPR (Universidade Tecnológica Federal do Paraná) para ser utilizada em aulas laboratoriais de disciplinas que envolvam controle e automação dos cursos de engenharia elétrica, engenharia de controle e automação e engenharia de computação da instituição.

Palavras-chave: Controle de nível, Controle PI, Sistemas de controle, Plataforma didática, ARDUINO UNO.

ABSTRACT

SILVA, Rodrigo Allan, DIDACTIC MODULE DEVELOPMENT OF LEVEL CONTROL. 72 f. Undergraduate Final Project – Undergraduate Degree in Electrical Engineering. Federal Technological University of Paraná. Cornélio Procópio, 2014.

This project proposes the construction of a didactical platform built from water reservoirs aiming the application of several level control strategies, as well as designing a PI level control using the Matlab software and the ARDUINO UNO development kit, developing a supervisor that sends and receives command data from the plant through serial communication, as well as allow the graphical display of the control response by the user through a graphical interface. This platform will be available on the Federal Technological University of Paraná for use on laboratory classes related to control and automation of the Electrical Engineering Degrees, Control and Automation Engineering Degrees and Computer Engineering Degrees of the previously mentioned University.

Key words: Level Control, PI Control, Control Systems, Didactic Platform, ARDUINO UNO.

LISTA DE FIGURAS

FIGURA 1 - Estrutura de controle Proporcional e Integeral (PI).....18
FIGURA 2 - Estrutura do controle Proporcional e Derivativo (PD) 19
FIGURA 3 - Estrutura de controle Proporcional, Integral e Derivativo (PID) 21
FIGURA 4 - Resposta de um sistema de primeira ordem a uma entrada degrau 22
FIGURA 5 - Aplicação de degrau à planta em malha aberta.....23
FIGURA 6 - Aplicação do método de Ziegler e Nichols.....24
FIGURA 7 - Estrutura típica de sistemas de controle digitais.....25
FIGURA 8 - Sinais contínuos r(t) e y(t) e sinais discretos r(kts) e y(kts) obtidos da conversão D/A.....26
FIGURA 9 - Sinais r(Ts) e y(Ts) obtidos do circuito *sample-and-hold*.....26
FIGURA 10 - Método de Euler *Forward*. 28
FIGURA 11 - Método de Euler *Backward.*.....29
FIGURA 12 - Método Trapezoidal. 30
FIGURA 13 - Kit ARDUINO UNO 33
FIGURA 14 - Interface de programação IDE para ARDUINO UNO. 34
FIGURA 15 - Representação de sinal PWM. 36
FIGURA 16 - Sistema de nível de liquido do laboratório da UTFPR. 37
FIGURA 17 - Montagem final do sistema de nível de líquido.....38
FIGURA 18 - Identificação dos pinos do CI L298N. 39
FIGURA 19 - Arquitetura interna do CI L298N. 40
FIGURA 20 - Circuito de acionamento da bomba CC. 41
FIGURA 21 - Placa de acionamento da bomba CC composta pelo CI L298N. 41
FIGURA 22 - Sensor de nível ultrassônico HC-SR04. 42
FIGURA 23 - Representação gráfica de pulso de disparo e pulsos ultrassônicos. 43
FIGURA 24 - Representação da medição de nível por meio do sensor HC-SR04.....44
FIGURA 25 - Supervosório em funcionamento. 45
FIGURA 26 - Diagrama de funcionamento do sistema de controle de nível. 46
FIGURA 27 - Curva de resposta ao degrau do sistema de nivel em malha aberta. 49
FIGURA 28 - Tela inicial do *toolbox* IDENT do *Toolbox* de controle do Matlab.....50
FIGURA 29 - Tela de importação de dados. 50
FIGURA 30 - Opção para determinação do modelo da planta. 51
FIGURA 31 - Modelo da função transferência sem atraso de transporte. 52

FIGURA 32 - Comparação entre resposta em malha aberta e G(s) identificada por meio da função IDENT do Matlab. 53
FIGURA 33 - Resposta em malha fechada para um *set point* de 5 centímetros. 55
FIGURA 34 - Resposta em malha fechada para um *set point* de 10 centímetros. 55
FIGURA 35 - Resposta em malha fechada para um *set point* de 15 centímetros. 56

LISTA DE TABELAS

TABELA 1 – Método de Ziegler e Nichols. 24
TABELA 2 – Características do kit ARDUINO UNO. 33
TABELA 3 – Relação de custo dos componentes utilizados. 47

LISTA DE ABREVIATURAS E SIGLAS

A	Ampéres
A/D	Conversor analógico/digital
CC	Corrente contínua
CI	Circuito integrado
cm	Centímetros
D	Distância
D/A	Conversor digital/analógico
EEPROM	Electrically erasable programmable read-only memory
Eq.	Equação
GND	Ground
Htanque	Altura do tanque
IDE	Integrated development enviroment
K	Ganho
kB	Kilobytes
Kd	Ganho derivativo
Ki	Ganho integrativo
Kp	Ganho proporcional
L	Atraso de transporte
LTDA	Empresa limitada
m	Metros
m/s	Metros por segundo
mA	Miliampéres
MHz	Megahertz
mm	Milímetro
n	Número de bits
P	Proporcional
PC	Personal computer
PD	Proporcional e derivativo

PI	Proporcional e integrativo
PID	Proporcional, integrativa e derivativo
PR	Paraná
PWM	Modulação pod largura de pulso
Q	Resolução
SRAM	Static random access memory
t	Tempo
τ	Constante de tempo do sistema
Td	Tempo derivativo
Ti	Tempo integrativo
Tméd	Tempo médio
Tna	Tempo em nível lógico alto
Tnb	Tempo em nível lógico baixo
Ts	Período de amostragem
TTL	Transitor-transistor-logic signal
µs	Microsegundos
USB	Universal serial bus
V	Volts
Vcc	Tensão em corrente contínua
V_L	Tensão de saída do pwm
Vméd	Tensão média

SUMÁRIO

1. INTRODUÇÃO 15

1.1 OBJETIVOS 16
1.2 ORGANIZAÇÃO DO TEXTO 16

3. TEORIA DE CONTROLE 17

2.1 CONTROLADORES DE TEMPO CONTÍNUO 17
2.1.1 CONTROLE P 17
2.1.2 CONTROLE PI 18
2.1.3 CONTROLE PD 19
2.1.4 CONTROLE PID 20
2.2 PROCESSOS DE PRIMEIRA ORDEM 22
2.3 SINTONIA DE CONTROLADORES 23
2.3.1 MÉTODO DE ZIEGLER E NICHOLS 23
2.4 SISTEMAS DE CONTROLE DE TEMPO DISCRETO 25
2.4.1 MÉTODOS DE DISCRETIZAÇÃO 27
2.4.1.1 APROXIMAÇÃO POR MÉTODO DE EULER A FRENTE (*FORWARD)* 28
2.4.1.2 APROXIMAÇÃO POR MÉTODO DE EULER PARA TRÁS (*BACKWARD)* 28
2.4.1.3 APROXIMAÇÃO POR MÉTODO TRAPEZOIDAL 29
2.5 PID DIGITAL 30

3. MÓDULO DE CONTROLE DE NÍVEL 32

3.1 ARDUINO UNO 32
3.1.1 CARACTERÍSTICAS DO *HARDWARE* 32
3.1.2 LINGUAGEM DE PROGRAMAÇÃO 34
3.2 MODULAÇÃO POR LARGURA DE PULSO (PWM) 35
3.3 ESTRUTURA FÍSICA DO SISTEMA DE NÍVEL 37
3.4 CIRCUITO DE ACIONAMENTO 39
3.5 SENSOR DE NÍVEL 42
3.6 SISTEMA SUPERVISÓRIO DA PLANTA DE NÍVEL 45
3.7 ESTRUTURA DE FUNCIONAMENTO DA PLANTA DE NÍVEL 46
3.8 CUSTO TOTAL DOS COMPONENTES UTILIZADOS 46

4. RESULTADOS DO PROJETO 48

4.1 IDENTIFICAÇÃO DA PLANTA 48
4.2 SINTONIA DO CONTROLE PI 53
4.3 APLICAÇÃO DO CONTROLE PI 54

5 CONCLUSÕES 57

REFERÊNCIAS 58

APÊNDICE A-1: PROJETO DO TANQUE SUPERIOR DA PLANTA 60

APÊNDICE A-2: PROJETO DO TANQUE INFERIOR DA PLANTA 61

APÊNDICE B: CÓDIGO DA INTERFACE DE CONTROLE EM MATLAB..... 62

APÊNDICE C: CÓDIGODE INTERFACE MATLAB/PLANTA IMPLEMENTADO NO ARDUINO UNO..... 71

1 INTRODUÇÃO

Os estudos em sistemas de controle já eram desenvolvidos pelos gregos por volta do ano 300 antes de Cristo. Com o passar dos anos as técnicas de controle ficavam cada vez mais sofisticadas e eficientes, pois inovadoras técnicas surgiam, como exemplo a lei de controle PID, os meios de análise de sistemas por diagramas de Bode e Nyquist e o método de projeto baseado no lugar das raízes, sendo todas criadas no século XX e fundamentais até os dias atuais (NISE, 2002).

A possibilidade de se aplicar sistemas de controle a plantas industriais e a necessidade de sofisticar ainda mais os processos na indústria, contribuíram de forma significativa para o avanço da engenharia de controle e consequentemente para o crescimento industrial (OGATA, 2000).

O avanço em tecnologia computacional e de dispositivos eletrônicos de processamento como microcontroladores e microprocessadores, aliado a eficiência atual na elaboração de códigos de controle devido ao surgimento de algoritmos mais robustos, abriram espaço para a aplicação do controle digital, o qual é atualmente utilizado no desenvolvimento de projetos de controle em geral (KUO, 1992).

Sistemas digitais estão em destaque por possuírem vantagens singulares em relação a sistemas que não utilizam de tecnologia digital, são eles: componentes digitais são menos suscetíveis ao envelhecimento, a variações climáticas, são muito mais robustos em relação à sensibilidade de ruídos e distúrbios, circuitos digitais os quais são extremamente pequenos em relação aos analógicos, são mais baratos, e muito flexíveis quando se necessita realizar qualquer alteração, bastando apenas a modificações de *firmware* sem necessidade de alteração de circuitos (KUO, 1992).

Na indústria, uma das aplicações de sistemas digitais está ligada a processos que dependem do controle de nível de líquidos armazenados em tanques. Processos que envolvam a produção de petróleo, por exemplo, necessitam da aplicação de controladores de nível e dependem desses controladores para que a segurança do processo de fabricação seja mantida. Portanto o desenvolvimento de um controlador digital de nível eficiente é de grande importância para indústria, visto sua influência em processos de fabricação como um todo.

1.1 OBJETIVOS

Este trabalho tem como objetivo a implementação em *hardware* / *software* de um sistema de controle de nível aplicados a uma estrutura composta por dois tanques em acrílico que será completamente construída durante o desenvolvimento do trabalho e que posteriormente ficará a disposição dos professores que ministram cursos relacionados a sistemas de controle e sistemas microcontrolados como um equipamento didático, e desta forma, possibilitar aos alunos dos cursos de Engenharia Elétrica, Engenharia de Controle e Automação e Engenharia de Computação da Universidade Tecnológica Federal do Paraná (UTFPR) Campus Cornélio Procópio a realização de experimentos durante o período de graduação.

Como objetivos específicos podem ser citados:

- Construção de um protótipo para aplicação de estratégias de controle envolvendo nível;
- Identificação do protótipo, e projeto de um controlador PI embarcado no *software* Matlab.
- Utilização do *software* Matlab como interface usuário-planta por meio do desenvolvimento de um supervisório.

1.2 ORGANIZAÇÃO DO TEXTO

Após a introdução será apresentada, no capítulo 2, a revisão bibliográfica referente à de sistemas de controle e controle digital, necessários para o desenvolvimento deste projeto. A seguir, no capítulo 3, apresentam-se as ferramentas utilizadas, como sensores, microcontrolador, *softwares* e a planta desenvolvida. No capítulo 4 aos resultados obtidos a partir da realização do presente trabalho serão mostrados e discutidos. O capítulo 5 refere-se às conclusões sobre o presente trabalho.

2 TEORIA DE CONTROLE

Este capítulo dedica-se a um breve resumo dos conceitos abordados durante o desenvolvimento deste projeto tais como controle em tempo contínuo, controladores, métodos de sintonia de controladores, discretização de funções de transferência de controladores de tempo contínuo, taxa de amostragem para controladores de tempo discreto e implementação computacional de controladores digitais.

2.1 CONTROLADORES DE TEMPO CONTÍNUO

Grande parte dos sistemas de controle industriais é baseada em plantas que podem ser descritas por um sistema de primeira ou segunda ordem. Tais plantas podem ser representadas por modelos que descrevem seu comportamento estabelecendo relações entre variáveis de entrada e saída. Por meio destes modelos é possível desenvolver uma estratégia de controle que permite coordenar variáveis de saída tendo como referência, variáveis de entrada (D'AZZO, 1978).

2.1.1 CONTROLE P

O controle proporcional é a ação de controle mais simples aplicada a sistemas de controles em geral. Com ela, os sinais de saída U(s) e de entrada E(s) do sistema são relacionados entre si por meio de um ganho proporcional Kp, como mostra a Eq. 1.

$$\frac{\mathrm{U(s)}}{\mathrm{E(s)}} = \mathrm{K_p} \tag{1}$$

A aplicação da ação de controle proporcional possui algumas vantagens assim como algumas desvantagens. A ação de controle proporcional tem a vantagem ser muito simples e possibilita que sistemas de controle respondam mais rapidamente com o aumento do ganho Kp, porém, deve-se tomar muito cuidado com o ganho proporcional, pois, se estabelecido um ganho muito alto, o sistema pode deixar de ser

estável e ruídos podem ser amplificados. Quando o assunto é erro em regime, o controle proporcional pode apresentar uma resposta com um erro considerável em regime permanente, pelo fato de tal controlador não rejeitar distúrbios em regime estacionário (OGATA, 2000).

2.1.2 CONTROLE PI

A Figura 1 descreve a estrutura da ação de controle PI que é composta pelas ações de controle proporcional e pela integral.

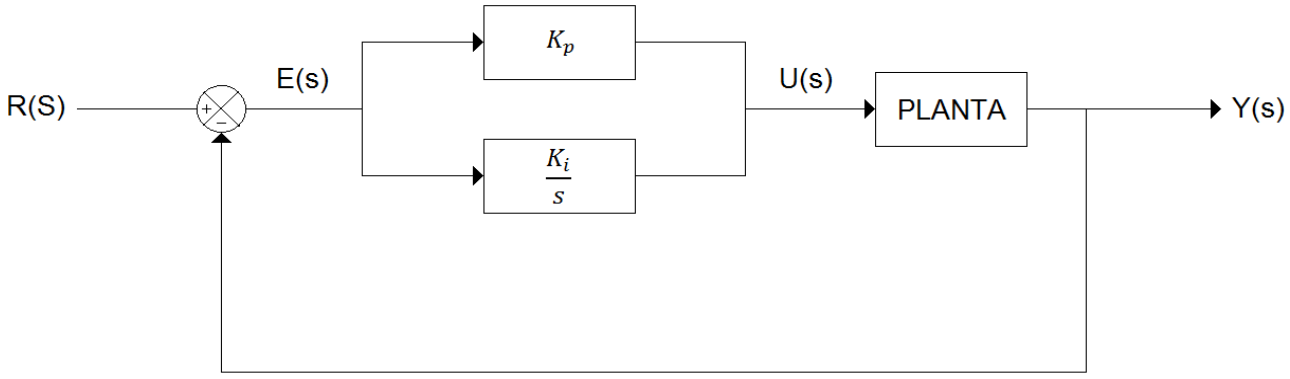


Figura 1 - Estrutura de controle Proporcional e Integral (PI).
Fonte: Autoria própria.

A ação de controle PI gera o sinal de saída u(t) do controlador a partir da variação da integral do erro e(t) mais a parte proporcional do erro e(t) do sistema (OGATA, 2000).

A ação de controle integral é definida pela Eq. 2.

$$u(t) = K_i \int_0^t e(\tau)d\tau \quad (2)$$

Ao submeter a Eq. 2 a uma transformada de Laplace obtém-se a Eq. 3 que demonstra a função transferência da ação integral no plano ‘s’.

$$\frac{\mathrm{U(s)}}{\mathrm{E(s)}} = \frac{\mathrm{K_i}}{\mathrm{s}} \tag{3}$$

O ganho integral Ki da Eq. 3 é composto por um ganho proporcional e pelo tempo integral Ti, o qual define a taxa de restabelecimento da ação proporcional (NISE, 2002).

Este tipo de ação de controle permite com que o erro em regime permanente seja diminuído ou até mesmo eliminado, porém se aumentado, o ganho integral Ki pode proporcionar à resposta transitória um alto sobre sinal assim como tornar o sistema instável (DORF; BISHOP, 2001).

2.1.3 CONTROLE PD

Ações de controle PD, de acordo com a Figura 2, atuam sobre o sistema por meio das ações proporcional e derivativa.

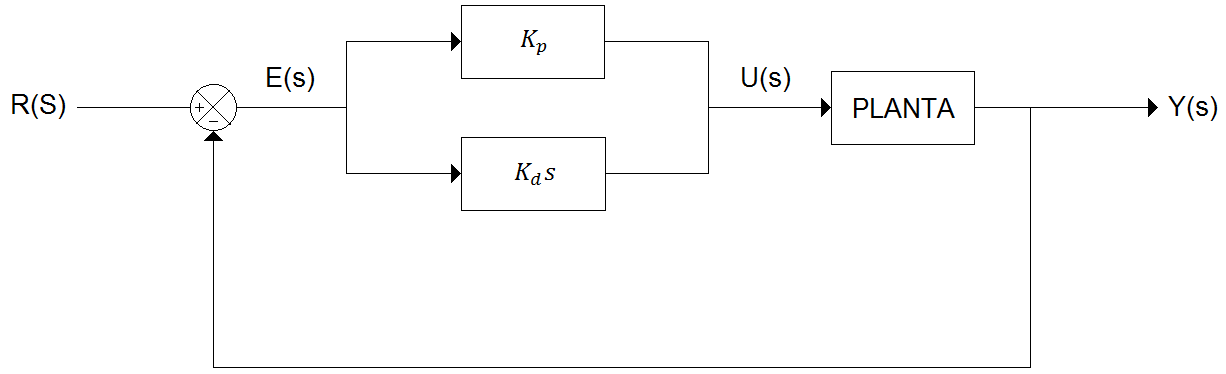


Figura 2 - Estrutura do controle Proporcional e Derivativo (PD).
Fonte: Autoria própria.

O comportamento do sinal de saída u(t) do controlador que possui a ação derivativa depende diretamente da taxa de variação do sinal de erro e(t) conforme Eq. 4 (NISE, 2002).

$$u(t) = K_d \frac{de(t)}{dt} \quad (4)$$

Passando a Eq. 4 para o plano ‘s’ através da transformada inversa de Laplace tem-se a função de transferência do termo derivativo do controle de acordo com a Eq. 5.

$$\frac{U(s)}{E(s)} = K_d s \quad (5)$$

O ganho derivativo é resultado do produto de um ganho proporcional Kp e do tempo derivativo Td, que é tempo em que a ação proporcional é avançada pela derivativa (OGATA, 2000).

Um dos grandes problemas atribuídos à ação de controle derivativa refere-se a ruídos de alta frequência. O termo derivativo, agora atribuído ao controlador, é muito influenciado por ruídos de frequência elevada e isso torna, por consequência, o sistema também sensível a tais ruídos (DORF; BISHOP, 2011).

Contudo, o controle PD possui vantagens que o torna muito útil. Com ele pode-se conseguir melhoras significativas no amortecimento, no sobre sinal e no tempo de assentamento da resposta do sistema (NISE, 2002).

A função de transferência para o controlador PD assume a forma da Eq. 6.

$$\frac{U(s)}{E(s)} = K_p(1 + T_d s) \quad (6)$$

2.1.4 CONTROLE PID

A ação de controle PID é resultado da junção das ações de controle proporcional, integral e derivativa, como representado pela Figura 3.

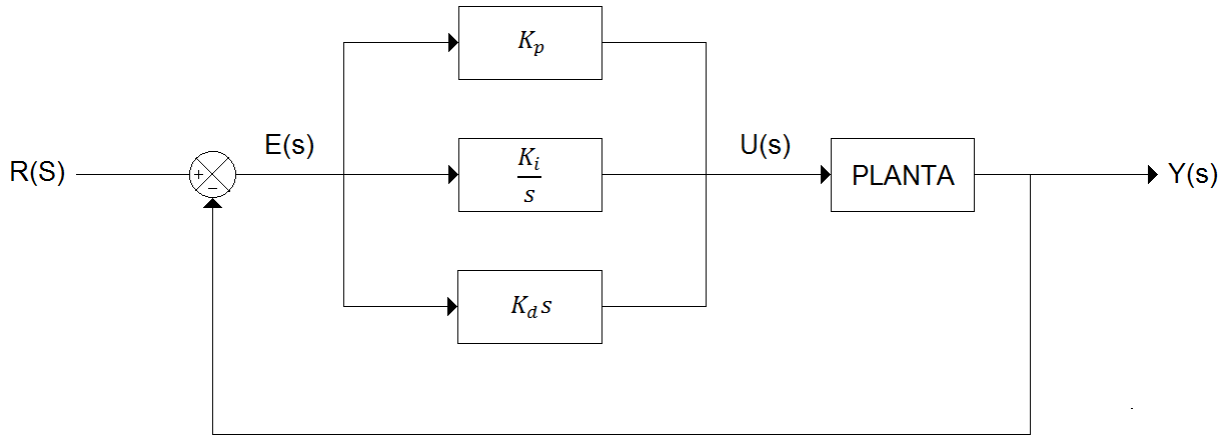


Figura 3 - Estrutura de controle Proporcional, Integral e Derivativo (PID).
Fonte: Autoria própria.

Esta ação de controle envolve todas as características das ações discutidas anteriormente e sua função de transferência e dada pela Eq. 7.

$$\frac{\mathrm{U(s)}}{\mathrm{E(s)}} = \mathrm{K_p}\left[1 + \frac{1}{\mathrm{T_i s}} + \mathrm{T_d s}\right] \tag{7}$$

O controle PID também pode ser considerado como uma forma de um compensador por avanço e atraso de fase, pelo fato de possuir um polo na origem e um polo no infinito (LI; ANG; CHONG, 2006).

Separadamente as ações integrativas e derivativas possuem desvantagens que podem comprometer o desempenho do controle, porém juntas contribuem muito para este desempenho, visto que o erro em regime pode ser reduzido ou completamente eliminado por conta da ação integrativa e o comportamento do sistema pode ser previsto por meio da ação derivativa (ASTROM; HAGGLUND, 1995).

2.2 PROCESSOS DE PRIMEIRA ORDEM

Processos industriais os quais envolvem o controle de nível em tanques são descritos como sistemas de controle de primeira ordem, o que torna possível a elaboração de projetos por meios empíricos tal qual o abordado pela seção 2.3 deste capítulo.

A função transferência de um sistema de primeira ordem dada pela Eq. 8 onde K é o ganho, τ é a constante de tempo de sistema, L é o tempo de atraso e termo exponencial é denominado atraso de transporte.

$$\frac{Y(s)}{R(s)} = \frac{Ke^{-Ls}}{(\tau s+1)} \tag{8}$$

Sistemas com esta característica, quando submetidos a uma entrada degrau R(s) = 1/s, responde, em malha aberta, com uma curva característica de forma semelhante a da Figura 4.

Nota-se que para um tempo τ = t, o sistema atinge 63,2% de U, assim quanto menor a constante de tempo τ mais rápido o sistema atinge o valor do degrau aplicado a sua entrada (NISE, 2002).

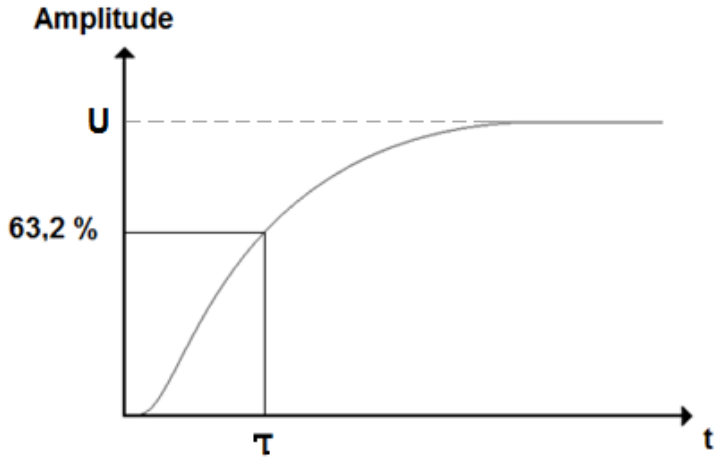


Figura 4 - Resposta de um sistema de primeira ordem a uma entrada degrau.
Fonte: Autoria própria.

2.3 SINTONIA DE CONTROLADORES

A realização de projetos de sistemas de controle pode se tornar muito trabalhoso, pois existem casos em que a planta a ser controlada possui muitas variáveis e o processo de cálculo se torna muito extenso. Em se tratando de projetos de envolvam controladores PID, existem métodos empíricos pelo qual a definição dos valores de ganho Kp, tempo integral Ti e tempo derivativo Td podem ser obtidos de forma relativamente simples e a sintonia do controlador pode ser realizada, minimizando esforços e tornado o processo de projeto mais ágil (DORF; BISHOP, 2001).

Um método muito usual que se baseia na curva de reação do sistema é o método de Ziegler e Nichols.

2.3.1 MÉTODO DE ZIEGLER E NICHOLS

O método de Ziegler e Nichols é baseado na resposta do sistema em malha aberta quando submetido a uma entrada degrau, Figura 5, porém para que o método possa ser de fato aplicado para a sintonia do PID, a curva de reação da planta deve possuir a forma de semelhante a um ‘s’ e o sistema também não deve possuir nenhum polo complexo-conjugado e nenhum integrador (OGATA, 2000).

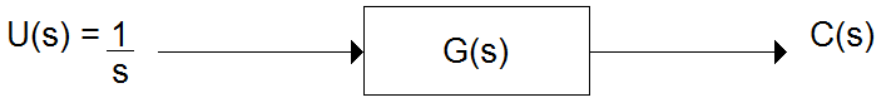


Figura 5 - Aplicação de degrau à planta em malha aberta.
Fonte: Autoria Própria.

Tal método consiste em traçar uma reta tangente ao ponto de inflexão da curva de resposta da planta. Feito isto se analisam os pontos de intersecção entre a reta tangente e o eixo do tempo e entre a reta tangente e a reta onde c(t) = K, de acordo

com a Figura 6. Por meio dos pontos de intersecção é possível determinar os valores de L e de τ (NISE, 2002).

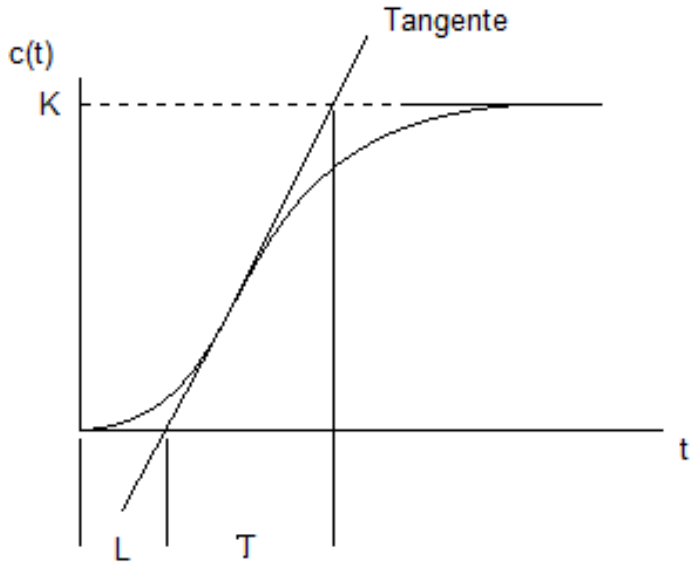


Figura 6 - Aplicação do método de Ziegler e Nichols.
Fonte: Autoria própria.

Utilizando os valores de L e τ encontrados a partir da aplicação da reta tangente a resposta da planta, pode se definir os valores de Kp, Ti e Td fazendo-se uso da Tabela 1. As equações presentes na Tabela 1 compõem o método de Ziegler e Nichols.

Tabela 1 - Método de Ziegler e Nichols.

Controlador	Kp	Ti	Td
P	T/L	∞	0
PI	0,9T/L	L/0,3	0
PID	1,2T/L	2L	0,5L

Fonte: Autoria própria.

2.4 SISTEMAS DE CONTROLE DE TEMPO DISCRETO

Diferentemente do controle em tempo contínuo, o controle digital trabalha com sinais de tempo discreto que são obtidos a partir de um processo de amostragem, em certa frequência, de um sinal contínuo, a fim de adequar este sinal ao controlador digital (KUO, 1992).

A estrutura básica de funcionamento de um sistema de controle em tempo discreto em malha fechada é detalhada pela Figura 7.

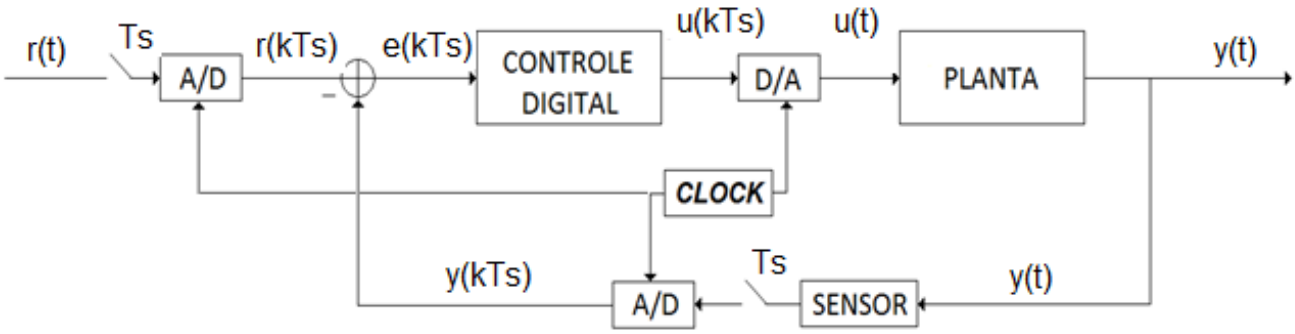


Figura 7 - Estrutura típica de sistemas de controle digitais.
Fonte: Autoria própria.

Os sinais de entrada r(t) e de saída y(t) referem-se aos sinais contínuos a serem convertidos para sinais digitais. Para isso, conversores analógicos-digitais (A/D) atuam como dispositivos de discretização e quantização destes sinal,

A discretização do sinal ocorre a uma determinada periodo Ts de amostragem e o resultado deste processo são os sinais de tempo discreto r(kTs) e y(kTs), que representam a amplitude dos sinais contínuos no exato momento de sua amostragem de acordo com o representado pela Figura 8 (KUO, 1992).

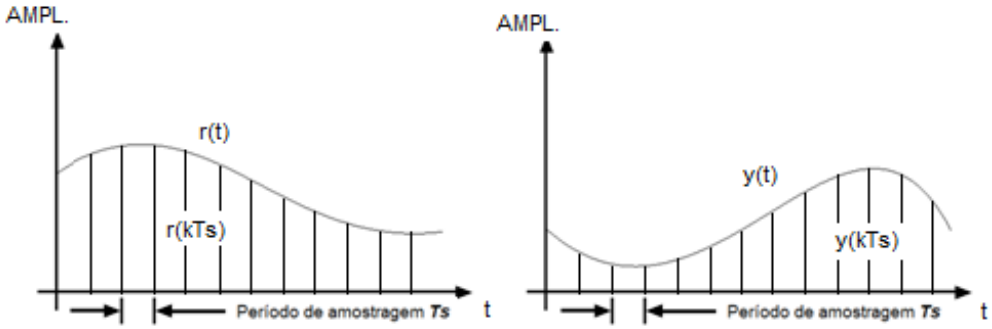


Figura 8 - Sinais contínuos r(t) e y(t) e sinais discretos r(kTs) e y(kTs) obtidos da conversão D/A.
Fonte: Autoria própria.

Em se tratando de conversores A/D, o circuito chamado de *sample-and-hold* realiza a etapa de discretização (*sample*), porém é somado a este sinal o efeito do segurador (*hold*) que mantém constante o valor de amplitude amostrado até que a próxima amostra aconteça. Logo os sinais r(kTs) e y(kTs) podem ser chamados de r(Ts) e y(Ts), ilustrados pela Figura 9, devido a não serem mais sinais compostos simplesmente por impulsos (OGATA, 2000).

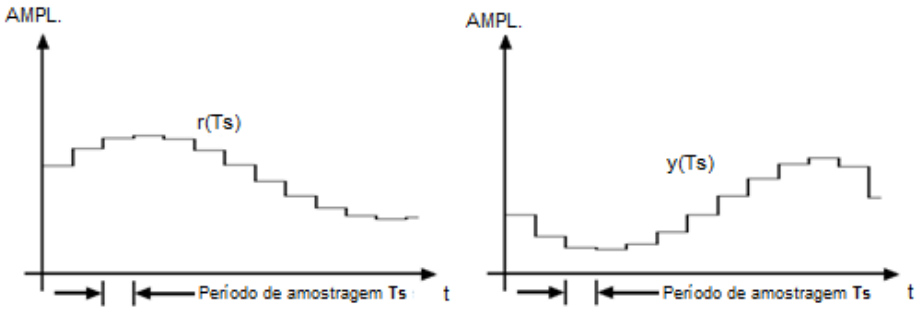


Figura 9 - Sinais r(Ts) e y(Ts) obtidos do circuito *sample-and-hold*.
Fonte: Autoria própria.

Já a quantificação dos sinais r(kTs) e r(kTs) refere-se a conversão dos valores das amplitudes amostradas do sinal contínuo em valores binários, ou seja, a transformação do sinal contínuo em um sinal digital propriamente dito.

O processo de quantificação depende de uma característica muito importante dos conversores A/D, sua resolução.

A resolução destes conversores determina a magnitude do erro produzido durante o processo de conversão e esta ligado diretamente ao número de bits do conversor, de 1 a n bits, limitando o erro entre 0 a 0,5% da tensão de entrada. Esta resolução pode ser calculada através da equação Eq. 9 onde Q é a resolução e n é numero de bits do conversor.

$$Q = \frac{1}{2^n} \tag{9}$$

Obtidos os sinais de R(kTs) e Y(kTs), o controle implementado digitalmente atua de forma a compensar o sinal de erro E(kTs) resultante da subtração de R(kTs) por Y(kTs). O sinal de resposta à entrada aplicada ao sistema U(kTs) é então convertido a um sinal contínuo U(t) por meio de um conversor digital-analógico (D/A) e transferido para os atuadores da planta a ser controlada (OGATA, 2000).

2.4.1 MÉTODOS DE DISCRETIZAÇÃO

O desenvolvimento de sistemas de controle são os mesmos modelos matemáticos baseados em equações diferenciais aplicados aos sistemas de controle analógicos. Contudo estes modelos não podem ser simplesmente implementados em *hardware* digital pelo fato de representarem sistemas completamente contínuos.

Porém, existem métodos numéricos aplicados à solução de equações diferenciais que permitem a aproximação de uma derivada ou uma integral em determinado ponto a uma equação de diferenças que depende da taxa de amostragem da conversão, fazendo com que o sistema de controle digital se comporte de forma muito próxima a sistemas contínuos, permitindo assim o embarque de modelos de controladores em dispositivos digitais (FRANKLING, POWELL E WORKMAN, 1998).

A garantia de linearidade do sistema também deve ser mantida, para isso, tais métodos devem possuir passos fixos para que o processo de discretização não seja comprometido (KUO,1992).

2.4.1.1 APROXIMAÇÃO POR MÉTODO DE EULER A FRENTE (*FORWARD)*

O método de aproximação de Euler pela equação de diferenças para frente, ou como é chamado equação de Euler *Forward*, aproxima a derivada de um sinal proveniente de uma fonte contínua em um tempo de amostragem Ts de uma amostra a frente, u(k+1), e amostra inicial, u(k), conforme Figura 10 (KUO, 1992).

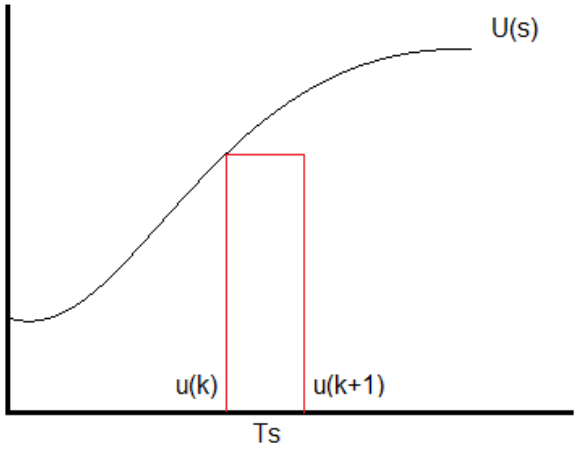


Figura 10 - Método de Euler *Forward*.
Fonte: Autoria própria.

A Eq. 10 representa a forma matemática do método de Euler *Forward.*

$$\dot{u}(k) \cong \frac{u(k+1) - u(k)}{T_s} \tag{10}$$

2.4.1.2 APROXIMAÇÃO POR MÉTODO DE EULER PARA TRÁS (*BACKWARD)*

De forma muito semelhante ao método de Euler *Forward* apresentado, o método de Euler por aproximação para trás, ou método de Euler *Backward*, é utilizado para realizar aproximações de derivadas de sinais contínuos a partir de amostragem em tempo fixo Ts de uma amostra anterior, u(k-1), e a primeira amostra, u(k), de acordo com a Figura 11 (KUO, 1992).

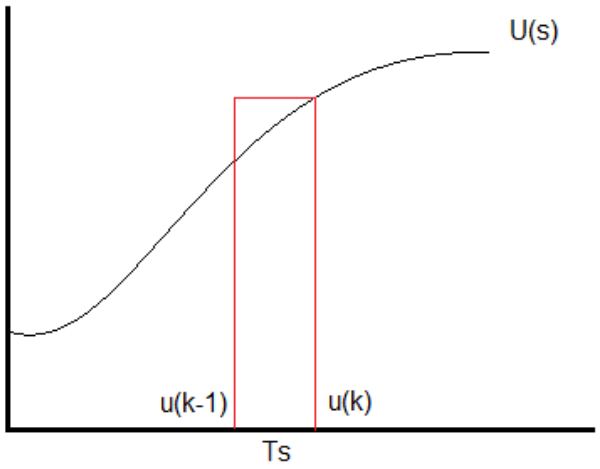


Figura 11 - Método de Euler *Backward.*
Fonte: Autoria própria.

A forma matemática da equação de diferenças pelo método Euler *Backward* é mostrada pela Eq. 11 a seguir.

$$\dot{u}(k) \cong \frac{u(k) - u(k-1)}{T_s} \tag{11}$$

2.4.1.3 APROXIMAÇÃO POR MÉTODO TRAPEZOIDAL

Um dos métodos mais eficientes em se tratando de aproximações por métodos numéricos de equações diferenciais é o método de Euler trapezoidal, comumente chamado também de método de *Tustin* ou método Bilinear.

A aproximação de um sinal discretizado para um sinal analógico por este método garante maior precisão, visto que, diferentemente dos métodos de Euler *Forward* e *Backward*, os pontos amostrados durante o período Ts são conectados a partir uma reta inclinada, formando um trapézio entre estes dois pontos, que acompanha, de forma mais eficaz, as variações de amplitude do sinal amostrado. Assim, quanto menor o tempo de amostragem Ts, mais próximo do sinal analógico o sinal discreto estará.

A Figura 12 mostra a forma de conversão do método Trapezoidal.

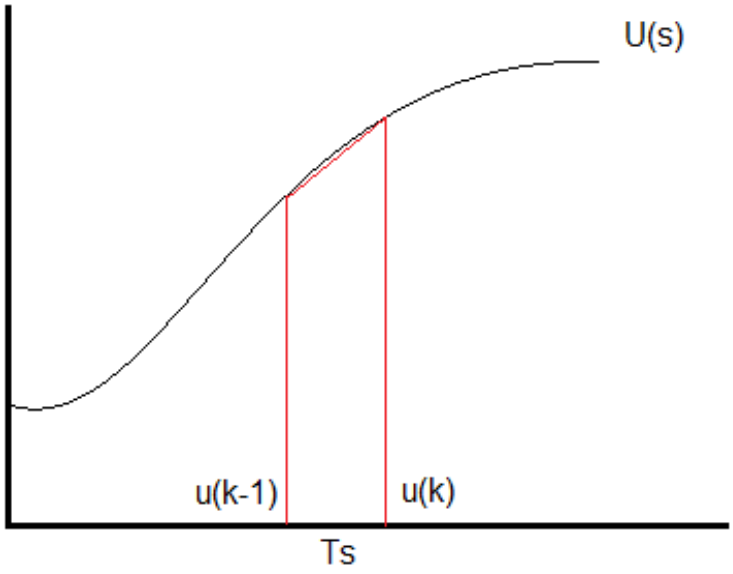


Figura 12 - Método Trapezoidal.
Fonte: Autoria própria.

Tal método de discretização pode ser representado pela Eq. 12 a seguir.

$$\int_{(k-1)T}^{kT} u(\tau)\, d\tau \cong \frac{T_s}{2}[u(k) + u(k-1)] \tag{12}$$

2.5 PID DIGITAL

O controle PID é composto pelas partes proporcional, integrativa e derivativa. Em se tratando de implementação de um controle PID digital, o grande problema para discretização da sua função de transferência de tempo contínuo está ligado às partes integrativas e derivativas do modelo. (OGATA, 2000).

Visto, então, que o controle PID assume uma forma de equação diferencial, é fato que os métodos de discretização são totalmente aplicáveis a esta lei de controle, pois, aproximando os termos P(s), I(s) e D(s) para equações discretas obtemos o PID

discreto que pode ser implementado em *hardware* digital (FRANKLING, POWELL E WORKMAN, 1998).

Para o presente trabalho, a equação discreta para o PID digital foi definida aplicando-se dois métodos de aproximação, estes métodos são trapezoidal (ou método bilinear) para a parte integrativa, pois tal método é aplicável à aproximação de integrais e por manter a as características de estabilidade do sistema, e o método *Backward* para a parte derivativa, pois, para aproximação de derivadas, é o método que não tendências a levar o sistema a instabilidade. Portanto fazendo uso da Eq. 7, é possível determinar a forma discreta do controle PID no plano como mostrado pela Eq. 13. Esta equação foi implementada no *software* Matlab para realização do controle.

$$u[n] = u[n-1] + K_p\left[e[n] - e[n-1]\right] + \frac{K_i T_s}{2}\left[e[n] + e[n-1]\right] + \frac{K_d}{T_s}\left[e[n] - 2e[n-1] + e_{n-2}\right] \quad (13)$$

3 MÓDULO DE CONTROLE DE NÍVEL

Neste capítulo são abordados os métodos de estudo e a proposta do sistema de controle de nível de líquido. O estudo envolve a pesquisa referente ao uso da plataforma de desenvolvimento escolhida para implementação do trabalho, bem como linguagem de programação e métodos utilizados para comunicação entre *hardware* e o *software* Matlab. Este estudo engloba também as ferramentas utilizadas como circuitos de acionamento, elementos sensores e atuadores do sistema além da proposta de bancada para o sistema de controle de nível de líquidos.

3.1 ARDUINO UNO

Atualmente, o setor de eletrônicos disponibiliza uma gama de tipos de microcontroladores no mercado, e muitos fabricantes utilizam estes componentes para a construção de plataformas de desenvolvimento de sistemas e kits didáticos para o uso em laboratórios de universidades ou até mesmo para uso pessoal.

Um desses fabricantes é o ARDUINO, a qual fabrica tais plataformas incluindo a plataforma de desenvolvimento aplicada à implementação do projeto deste trabalho, o ARDUINO UNO.

3.1.1 CARACTERÍSTICAS DO HARDWARE

O kit ARDUINO UNO tem sua construção baseada em microcontroladores do modelo ATmega328 de 8 bits, produzido pelo fabricante ATMEL®, que possui características próprias.

A Tabela 2 mostra de forma detalhada essas características e, a seguir, a Figura 13 apresenta disposição dos pinos e periféricos e localização do microcontrolador no kit.

Tabela 2 – Características do KIT Arduino Uno

Tensão de operação	**5 Vcc**
Tensão de entrada	**7 a 12 Vcc (limitados entre 6 e 20 Vcc)**
Entradas e saídas digitais	**14 pinos (incluindo 6 pinos PWM)**
Entradas analógicas	**6 pinos (A0 ao A5)**
Corrente CC por pino de entrada e saída	**40 mA**
Saídas de tensão	**3,3 Vcc e 5 Vcc**
Memória flash	**32 kB (0,5 kB dedicados ao *bootloader*)**
SRAM	**2 kB**
EEPROM	**1 kB**
Velocidade de Clock	**16 MHz**
Conexão com PC	**Via USB**

Fonte: Autoria Própria.

Figura 13 - Kit ARDUINO UNO.
Fonte: Autoria própria.

3.1.2 LINGUAGEM DE PROGRAMAÇÃO

O kit Arduino UNO traz consigo algumas facilidades que o tornam extremamente prático de se trabalhar, dentre estas facilidades está a linguagem de programação e o *software* usado para a criação dos códigos das rotinas.

Baseado em uma linguagem muito próxima da linguagem C, o kit é programado de forma intuitiva e suas configurações de portas são feitas de forma rápida, além de possuir funções e bibliotecas destinadas a vários tipos de aplicações.

A estrutura básica de programação do kit é composta por quatro etapas fundamentais: inclusão de bibliotecas, declaração de variáveis, laço *void setup()* para determinação das condições de operação das portas digitais e analógicas e laço *void loop()* onde a rotina principal do código é implementada.

Em se tratando de ambiente de programação, o ARDUINO UNO, e demais kits disponibilizados pela fabricante ARDUINO, possui uma IDE (*Integrated Development Environment*) executável destinada à elaboração e compilação dos programas escritos para o equipamento, Figura 14.

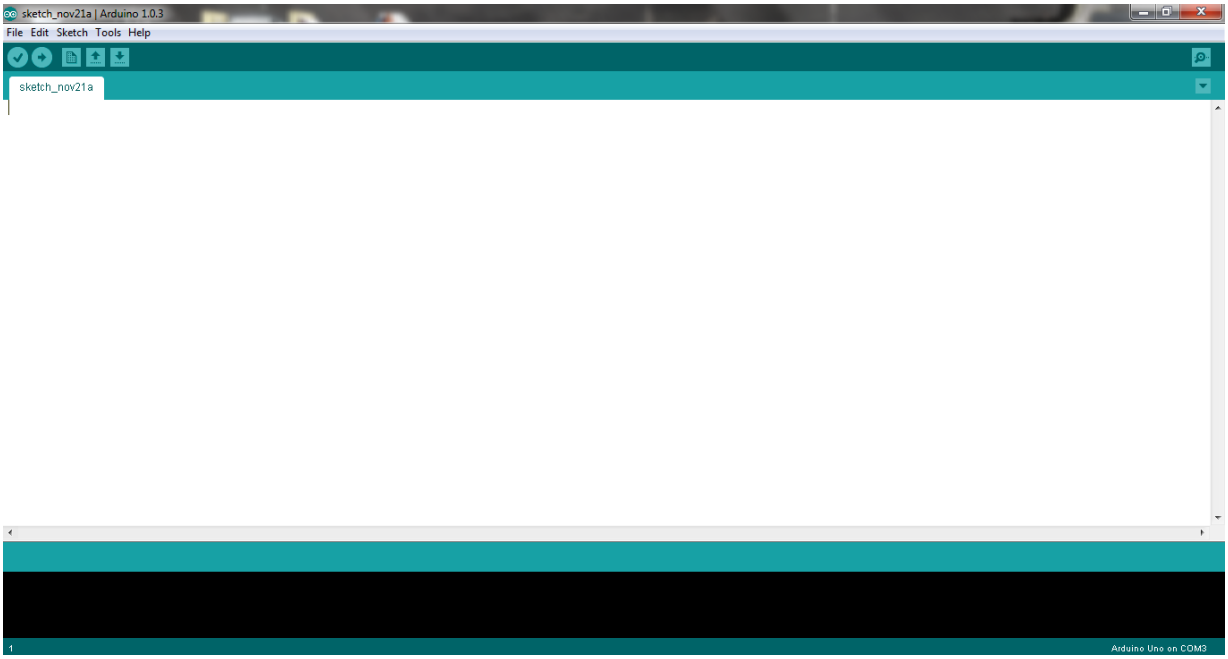


Figura 14 - Interface de programação IDE para ARDUINO UNO.
Fonte: Autoria própria.

Outra vantagem presente no kit se refere à transferência do código compilado. Internamente o Arduino UNO possui um *firmware* chamado *bootloader* dedicado exclusivamente a transferência dos códigos para o ATmega328, subtraindo assim a necessidade de remoção do microcontrolador da placa e o uso de circuitos de gravação.

3.2 MODULAÇÃO POR LARGURA DE PULSO (PWM)

Motores de corrente contínua (motores CC de corrente de armadura) podem ter sua velocidade controlada de diversas formas, como por exemplo, através de um potenciômetro linear que controla a potência do motor de acordo com a tensão e corrente fornecida ao mesmo (ZHAN, 2007)

Mas, um fator importante quando se trata de controle de velocidade de máquinas CC é o torque. O fundamental é que o motor possua um torque constante independente da tensão aplicada nos terminais da máquina e isso não acontece em muitas formas de controle incluindo a exemplificada logo acima.

Para evitar esse tipo de problema e tornar o controle mais robusto, um método relativamente simples pode usado para realizar o controle de velocidade de um motor CC, este método é chamado de modulação por largura de pulso (PWM - *Pulse Width Modulation*) que mantém o torque da máquina constante além de ser um sinal muito menos sensível a ruídos, além de ser muito mais barato de outras formas de controle de tensão (ZHAN, 2007)

O controle por PWM é muito utilizado em sistemas de controle implementados digitalmente, pois na realidade o sinal PWM é um sinal completamente digital desde sua fonte até seu destino, no caso os circuitos os quais se deseja realizar o controle (BARR, 2001).

A modulação PWM consiste em gerar um sinal que possua amplitude constante, entretanto, com larguras de pulso que variam de 0 a 100% do período do sinal. A variação da largura de pulso é denominada ciclo de trabalho (*Duty Cicle),* que corresponde à relação entre o tempo em que o sinal se encontra em nível lógico alto e o período, e está relacionada diretamente com a tensão entregue a carga. Neste caso:

Duty Cicle = 0% equivale a V_L = 0 Vcc e *Duty Cicle = 100%* equivale a V_L = 12 Vcc, podendo *Duty Cicle* e V_L assumir qualquer valor entre seus respectivos limites (ZHAN, 2007).

E equação Eq. 14 representa o calculo do *Duty Cicle.*

$$\text{Duty Cicle} = \frac{T_{na}}{T_{na}+T_{nb}} = \frac{T_{na}}{T} \tag{14}$$

Onde:

- Tna: Tempo em que o sinal se encontra em nível lógico alto;
- Tnb: Tempo em que o sinal se encontra em nível lógico baixo;
- T: Período do sinal PWM;

O motor CC, por outro lado, interpreta o sinal PWM gerado como sendo um sinal completamente analógico, ou seja, mesmo havendo um “chaveamento” no sinal de PWM o motor funciona como se estivesse recebendo uma tensão analógica constante que é exatamente a tensão média resultante do ciclo de trabalho do sinal PWM. Logo, para controlar a velocidade de motores CC basta apenas controlar o valor de *Duty Cicle* do sinal. A Figura 15 mostra a relação entre tensão média (Vméd) aplicada ao motor e o ciclo de trabalho *Duty Cicle* (BARR, 2001).

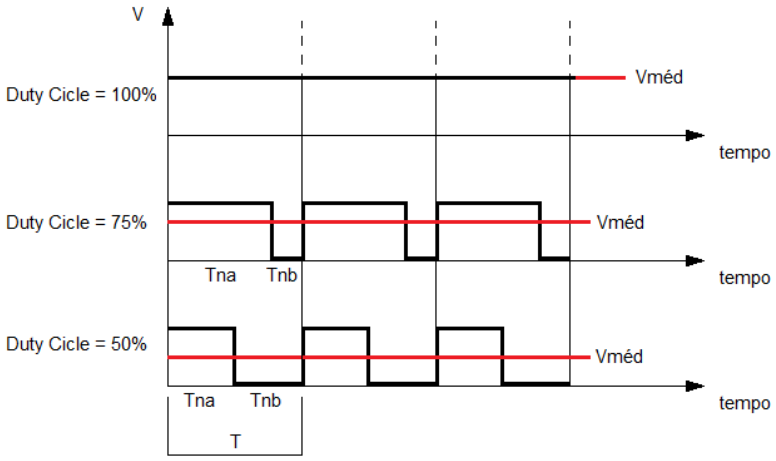


Figura 15 - Representação de sinal PWM.
Fonte: Autoria própria.

Em se tratando do microcontrolador ATMEGA utilizado no ARDUINO UNO®, o sinal PWM é gerado em uma porta digital e seu *Duty Cicle* é calculado automaticamente, subtraindo a necessidade de implementação em forma de linhas de códigos da Eq. 14.

Para as plataformas ARDUINO UNO o sinal por modulação de pulso assume valores na faixa de 0 a 255, por se tratar de um microcontrolador de 8 bits de resolução, e tensões entre 0 e 5V.

3.3 ESTRUTURA FÍSICA DO SISTEMA DE NÍVEL.

A construção dos reservatórios de água para a implementação do sistema de controle projetado para este trabalho foi baseado em reservatórios já existentes no laboratório de controle de automação da UTFPR campus Cornélio Procópio, conforme Figura 16. A planta é composta por dois tanques dispostos um sobre o outro onde o controle de nível é aplicado exclusivamente no tanque de cima, ficando o tanque de baixo apenas como reservatório da água que é utilizada no processo.

Figura 16 - Sistema de nível de líquido do laboratório da UTFPR.
Fonte: Autoria própria.

Porém, algumas melhorias foram acrescentadas, sendo elas, o aumento do volume de água admissível por cada reservatório e a confecção dos mesmos utilizando materiais com custos menores.

Os reservatórios foram projetados para serem construídos em chapas de acrílico de espessura de aproximadamente 3 mm e cada um deles com 300 mm de altura, 150 mm de comprimento e 150 mm de largura. Os desenhos dos projetos, Apêndice A, foram encaminhados para a fábrica Valorize Placas LTDA localizada em Londrina PR. Para cada tanque foram usados também válvulas globo e adaptadores niquelados, fabricados para que não enferrujem em contato direto com a água, para encaixe de mangueiras de ¼ de polegada, medida exata da entrada e saída de água da bomba utilizada, e uma régua transparente fixada ao reservatório onde o controle de nível foi aplicado.

Para que fosse possível encaixar os reservatórios um sobreposto ao outro, bases em madeira foram montadas e parafusadas ao fundo de cada tanque, de maneira a garantir mais firmeza quando estiverem completos com água.

O resultado final da montagem dos tanques para o sistema de nível de líquido é mostrado na Figura 17.

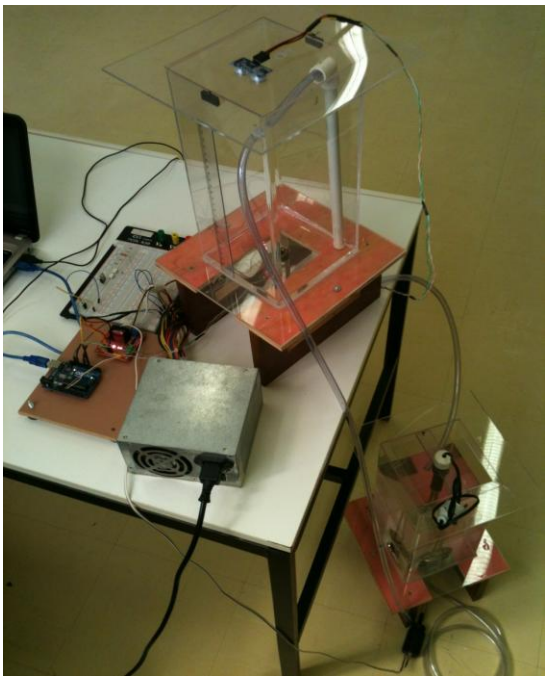


Figura 17 - Montagem final do sistema de nível de líquido.
Fonte: Autoria própria.

3.4 CIRCUITO DE ACIONAMENTO

Para realizar o papel de atuador do sistema optou-se pelo uso de uma bomba CC de 12 V e 2 A, semelhante às utilizadas em para-brisas de automóveis.

Devido à magnitude da corrente e da tensão solicitadas pela bomba em relação ás correntes e tensões do circuito de controle, foi necessário o uso de um circuito de acionamento capaz de alimentar a bomba e também isolar eletricamente circuitos onde circulam altas correntes de circuitos de comando.

Como componente do circuito de acionamento foi definido o uso do CI L298N, do fabricante *ST Electronics*, composto por dois circuitos ponte H independentes que são comandados externamente através de seus respectivos pinos. As Figuras 18 e 19 detalham a disposição dos pinos do L298N e sua arquitetura interna, respectivamente.

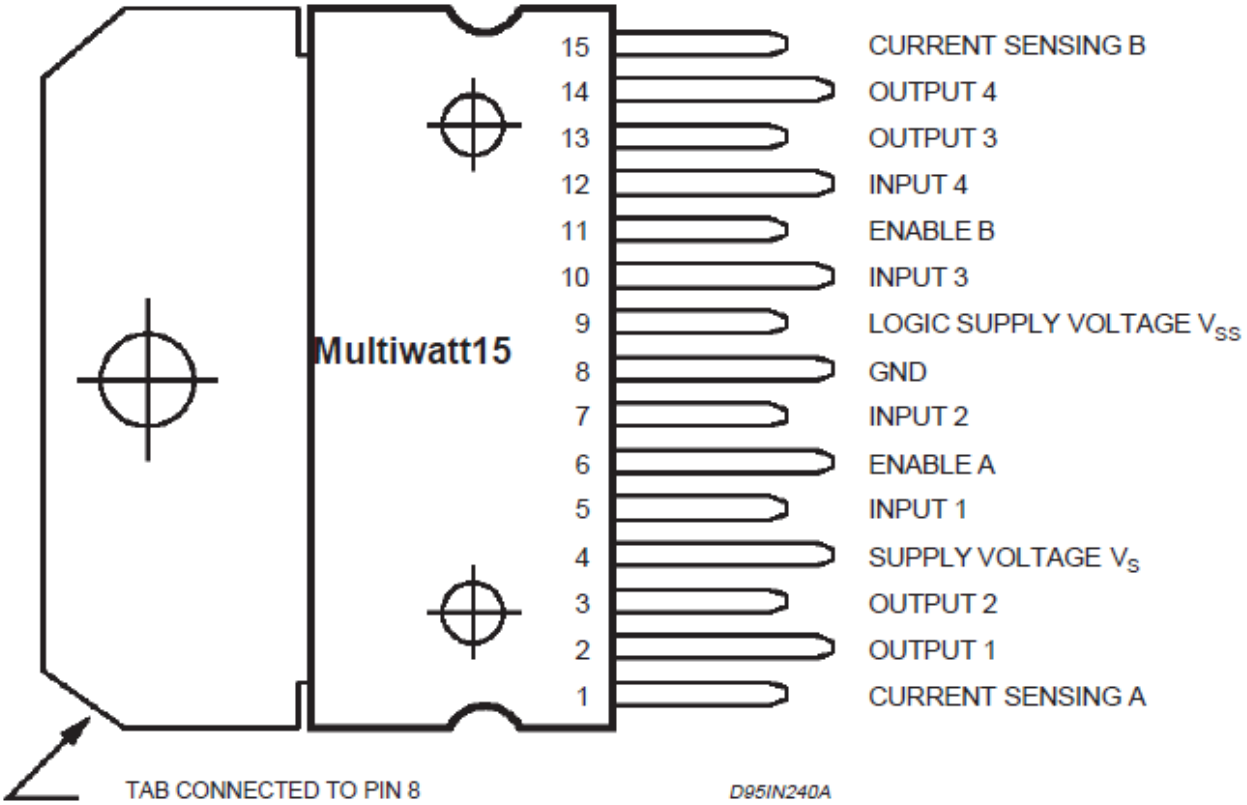


Figura 18 - Identificação dos pinos do CI L298N.
Fonte: Datasheet L298N.

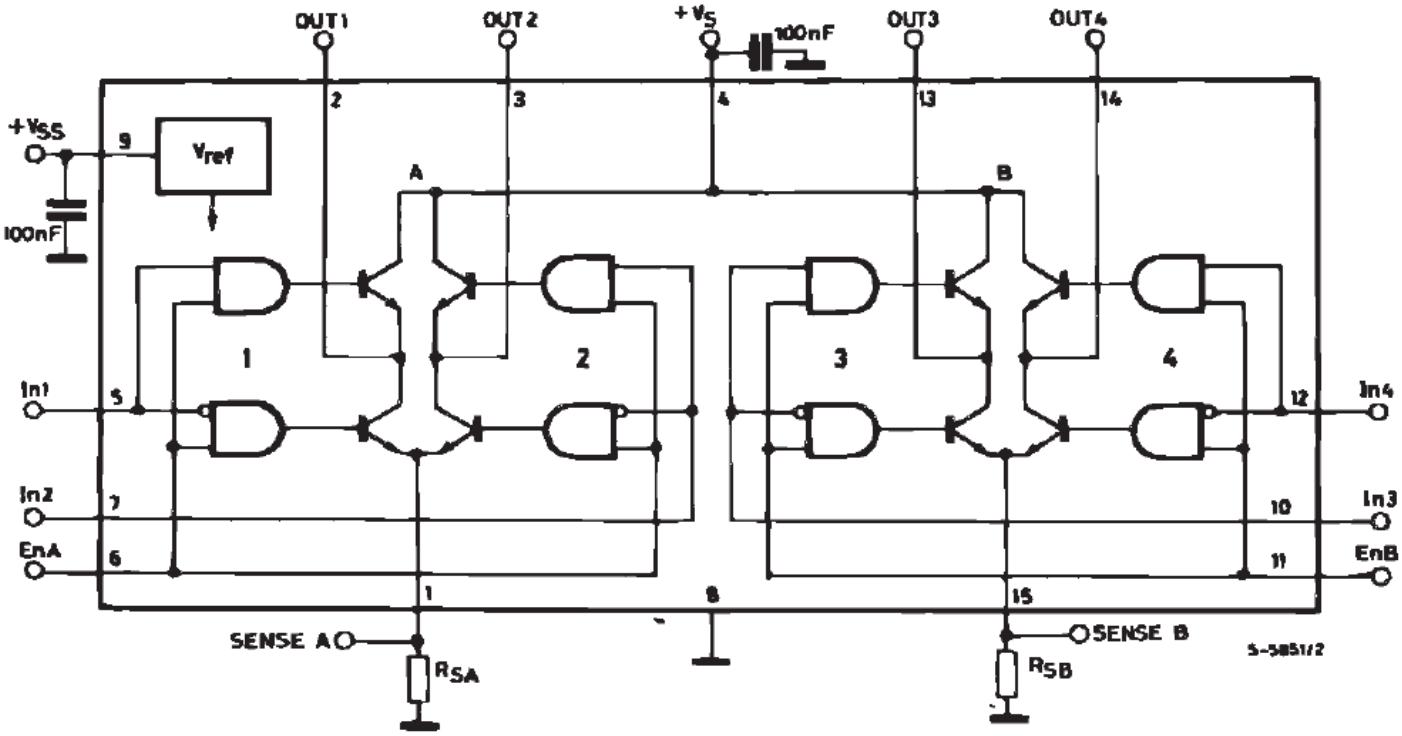


Figura 19 - Arquitetura interna do CI L298N.
Fonte: Datasheet L298N.

Circuitos em ponte H são circuitos compostos por transistores dispostos de maneira que possibilitem a inversão da rotação de motores de corrente contínua simplesmente invertendo o sentido da corrente que percorre os enrolamentos da máquina. O CI L298N dispõe de dois destes circuitos, porém, para este trabalho, foi necessário utilizar apenas uma das duas pontes H existentes pelo fato de existir apenas um motor, ficando o esquemático do circuito conforme a Figura 20.

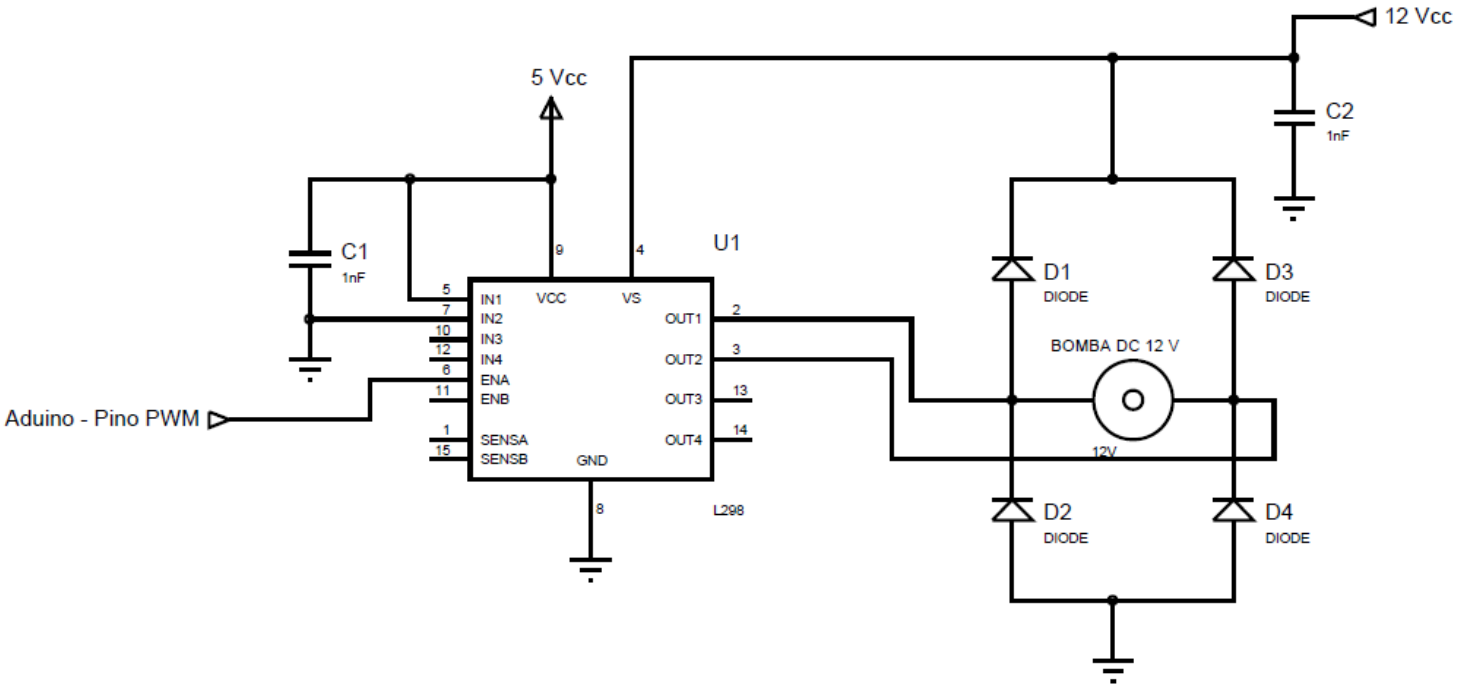


Figura 20 - Circuito de acionamento da bomba CC.
Fonte: Autoria própria.

Para o desenvolvimento do projeto, utlizou-se a placa de acionamento, composta pelo CI L298N, de acordo com a Figura 21.

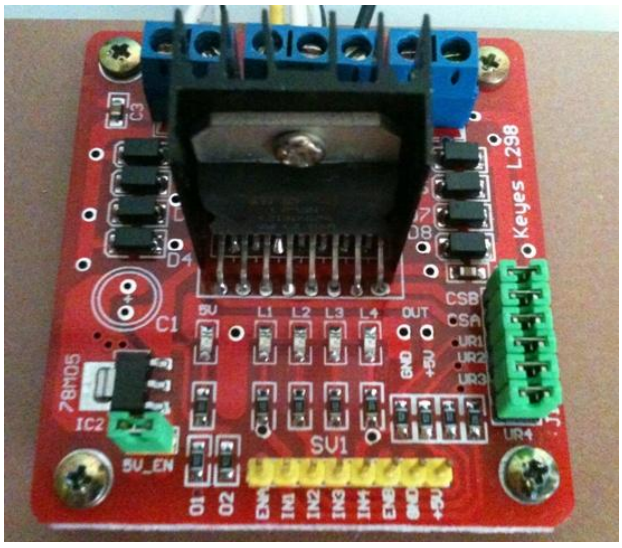


Figura 21 -Placa de acionamento da bomba CC composta pelo CI L298N.
Fonte: Autoria própria.

3.5 SENSOR DE NÍVEL

Dentre as diversas formas de se obter medidas de nível em reservatórios de líquidos em geral, optou-se para este projeto usar o sensor medidor de distância por ultrassom HC-SR04, Figura 22, também fornecido pela ARDUINO.

Figura 22 - Sensor de nível ultrassônico HC-SR04.
Fonte: Autoria Própria.

O sensor possui uma resolução de 3 mm e realiza medidas de distancia entre 2,5 cm à 4,3 m. É composto por quatro pinos, nos quais Vcc é para alimentação dos circuitos (tensão não superior à 5 Vcc) e GND para referência de terra, já os pinos ECHO e TRIGGER são responsáveis pela operação do dispositivo.

O pino TRIGGER recebe um pulso TTL de 10 us enviado pelo microcontrolador, em seguida, o transmissor do sensor (localizado a esquerda) envia uma sequência de oito pulsos ultrassônicos que ao chegarem até o objeto posicionado à sua frente são refletidos e captados pelo receptor do sensor (localizado a direita). A Figura 23 representa os pulsos TTL aplicados ao pino TRIGGER e os pulsos ultrassônicos enviados pelo sensor para medição da distância.

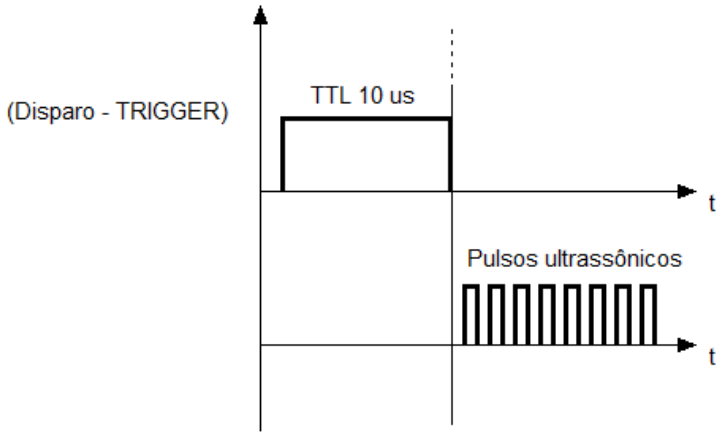


Figura 23 - Representação gráfica de pulso de disparo e pulsos ultrassônicos.
Fonte: Autoria própria.

Internamente, o HC-SR04 calcula o tempo através da média dos tempos que cada um dos oito pulsos leva para ir e voltar até o sensor. Obtido o valor de tempo médio, o pino ECHO é levado à condição de nível alto, mantendo-se assim por um tempo igual ao tempo médio calculado. Este pulso é por fim enviado para uma porta digital do microcontrolador, que mensura este tempo e converte este sinal para uma medida de nível de água em centímetros por meio da Eq. 15, deduzida levando-se em consideração que a velocidade do ultrassom é de cerca de 340 m/s e que o sinal ultrassônico atinge o objeto exatamente na metade do tempo que leva para ir e voltar ao sensor.

$$D = H_{tan\ que} - \frac{T_{médio}}{58} \tag{15}$$

A Figura 24 descreve como a medição de distância é realizada a partir da colocação do sensor ultrassônico na tampa do tanque superior da planta onde o controle de nível é realizado.

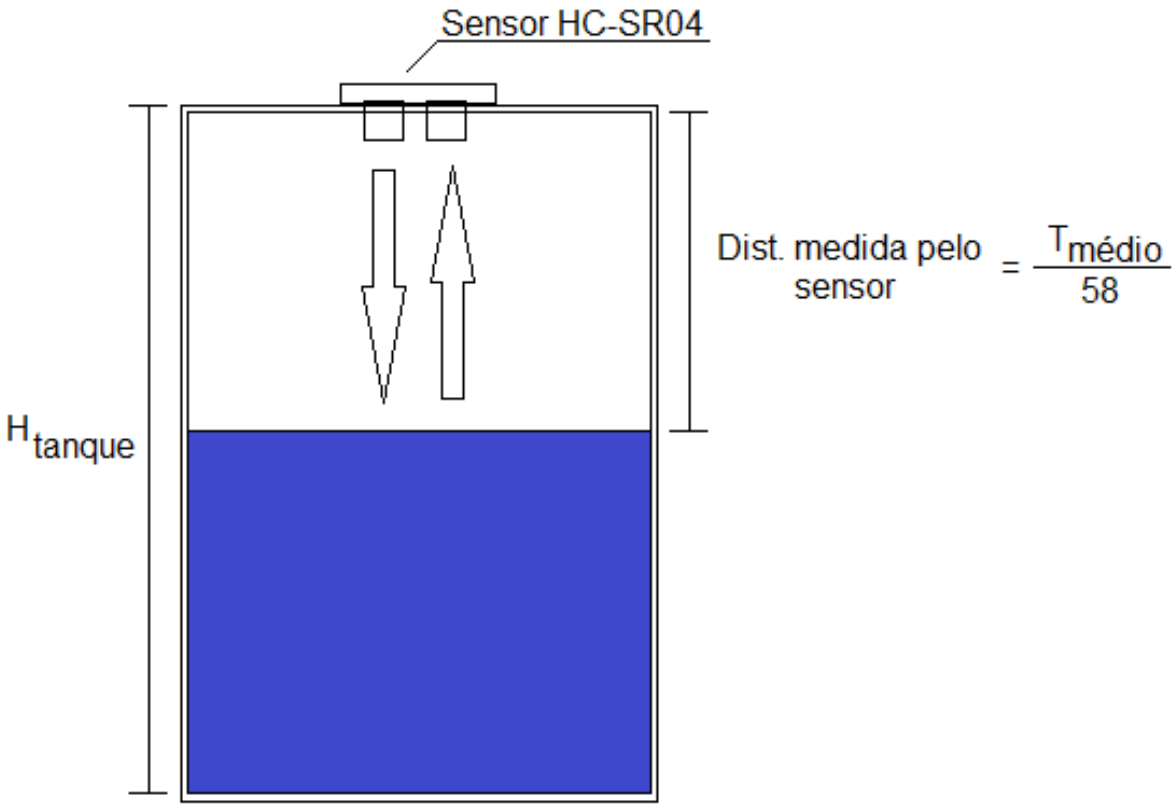


Figura 24 - Representação da medição de nível por meio do sensor HC-SR04.
Fonte: Autoria própria.

3.6 SISTEMA SUPERVISÓRIO DA PLANTA DE NÍVEL

O presente trabalho também envolve a elaboração de uma interface usuário-planta. Esta interface foi elaborada usando o *software* Matlab, mais precisamente uma ferramenta específica do *software* para desenvolvimento de interfaces chamado *Guide*. A tela do supervisório é a interface entre o usuário e os comandos existentes na planta. Este supervisório também comanda toda a rotinha dos códigos implementados em Matlab para ensaios em malha aberta ou ensaios em malha fechada.

Na tela do supervisório existem comandos os quais são adicionados diretamente pelo usuário. Para iniciar o experimento, o usuário primeiramente deve selecionar qual o tipo de ensaio deseja realizar, se é em malha aberta ou em malha fechada. Quando

selecionado o ensaio em malha aberta, o usuário deve inserir os valores de degrau a serem aplicados à planta e o número de amostras do ensaio, em seguida, basta pressionar o botão INICIAR/PARAR para dar início ao ensaio.

Já, quando o ensaio a ser realizado pelo usuário é em malha fechada, o mesmo deve indicar os valores de ganhos Kp, Ki, e Kd para o controlador, o número de amostras para o ensaio e o valor de *setpoint* e, em seguida, deve pressionar novamente o botão INICIAR/PARAR para começar o ensaio, todos os valores das variáveis utilizadas no ensaio podem assumir novos valores durante o ensaio sem a necessidade de que o mesmo seja interrompido. Para todos os ensaios realizados fazendo uso deste supervisório a taxa de amostragem assume um valor fixo onde Ts = 250 ms. O gráfico ao lado da tela é responsável por mostrar, em tempo real, a resposta de nível do sistema tanto em malha aberta quanto em malha fechada durante todo o período de ensaio. A Figura 25 representa a tela do supervisório em funcionamento.

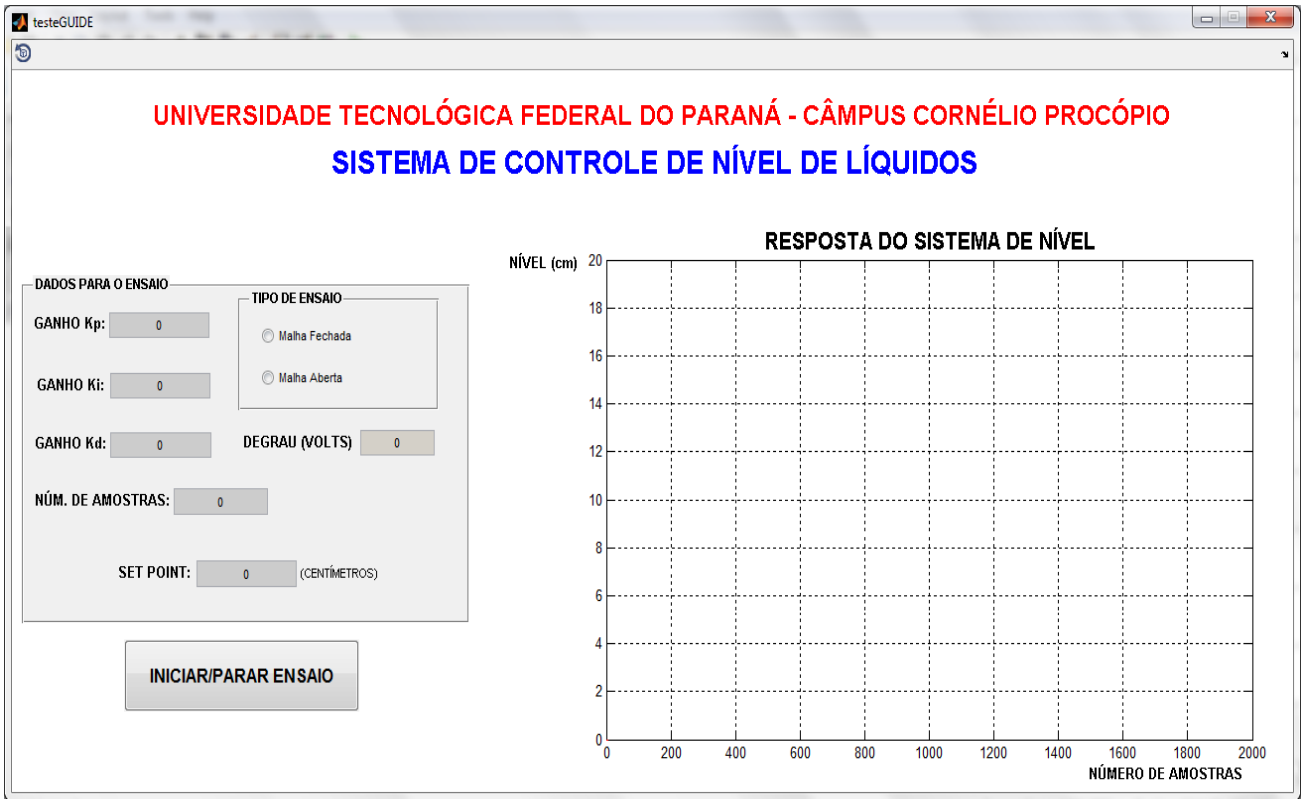


Figura 25 - Supervisório em funcionamento.
Fonte: Autoria própria.

3.7 ESTRUTURA DE FUNCIONAMENTO DA PLANTA DE NÍVEL

O funcionamento geral do sistema de controle de nível de líquidos desenvolvido pode ser representado pelo diagrama da Figura 26. Todos os elementos atuadores e sensores do sistema estão diretamente acoplados ao ARDUINO UNO que desempenha a função de aquisição dados e comando do atuador. O ARDUINO recebe os sinais de *Duty Cilcle* do PWM, provenientes do código em Matlab, e envia sinais de nível, recebidos do sensor, para o Matlab por meio de uma comunicação serial estabelecida entre os mesmos, assim toda a planta pode ser comandada pelo supervisório montado.

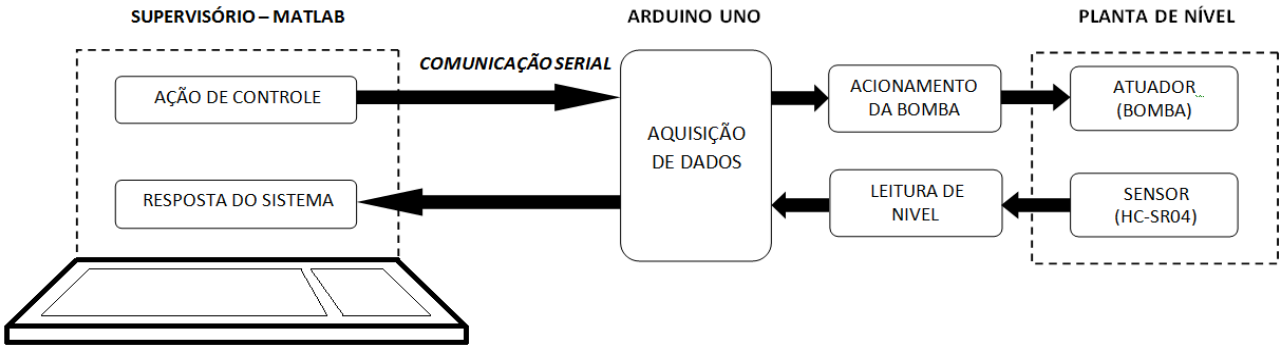


Figura 26 - Diagrama de funcionamento do sistema de controle de nível.
Fonte: Autoria própria.

3.8 CUSTO TOTAL DOS COMPONENTES UTILIZADOS

A Tabela 3 relaciona o custo de cada componente utilizado na construção física da planta bem como componentes relacionados à automação da mesma. O custo de construção do módulo é muito inferior ao custo dos módulos de controle de nível adquiridos para os laboratórios de controle da Universidade Tecnológica Federal do Paraná, assim a construção de módulos como o proposto por este trabalho torna-se uma opção viável.

Tabela 3 - Relação de custo de componentes utilizados.

Descrição do componente	Valor
Tanque superior em acrílico	R$ 75,00
Tanque inferior em acrílico	R$ 75,00
Duas válvulas globo niqueladas	R$ 20,00
Dois metros de mangueira ¼”	R$ 2,00
1 metro de cano tigre 1,5 mm	R$ 6,70
Três adaptadores de bronze ¼”	R$ 13,50
Régua em acrílico	R$ 2,00
Bomba DC 12V	R$ 12,00
Kit ARDUINO UNO	R$ 69,00
Shield L298N	R$ 25,00
Sensor ultrassônico HC-SR04	R$ 16,00
TOTAL INVESTIDO	**R$ 316,20**

Fonte: Autoria própria.

4 RESULTADOS DO PROJETO

Este capítulo tem por objetivo a apresentação dos resultados atingidos durante o desenvolvimento e conclusão do projeto. Serão apresentados resultados ligados à resposta em malha aberta do sistema de nível, identificação da planta de nível, implementação do controle PID embarcado no kit de desenvolvimento ARDUINO UNO, sintonia do controle PID e resultados obtidos com a planta em funcionamento.

4.1 IDENTIFICAÇÃO DA PLANTA

O processo de identificação da planta didática de nível desenvolvida foi realisado por meio do *software* Matlab, mais precisamente por meio da função *ident* do *toolbox* de controle, dedicada para diversas aplicações incluindo identificação de sistemas em geral.

Para dar inicio a identificação do sistema de nível por meio do *ident*, primeiramente é necessário realizar um ensaio em malha aberta com a planta e levantar a curva de reação da mesma quando submetida a um degrau de entrada. Para realizar o ensaio em malha aberta basta selecionar o item “malha aberta” no supervisório da planta e definir os valores de degrau e número de amostras.

Assim para o ensaio em malha aberta foi definido um degrau de entrada a uma tensão constante de 7 V aplicada na bomba e uma quantidade de 700 amostras, resultando em um nível estável em aproximadamente 3 cm. A Figura 27 mostra a curva obtida do ensaio.

Considerou-se que o motor não está saturado com esse nível de tensão, portanto operando em condições lineares do sistema.

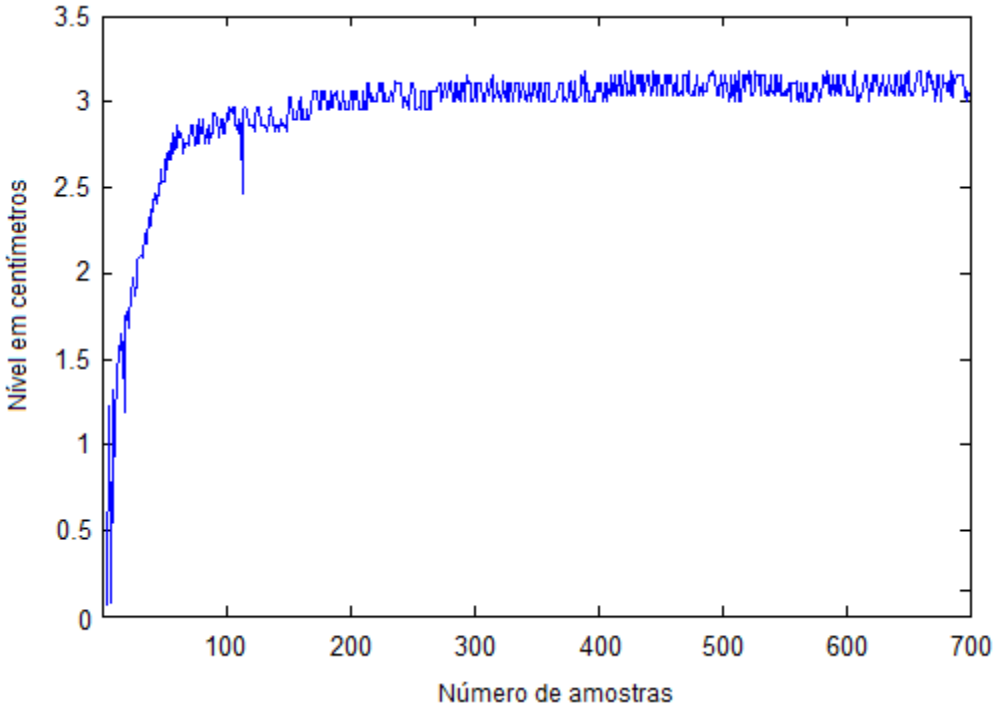


Figura 27 - Curva de resposta ao degrau do sistema de nível em malha aberta.
Fonte: Autoria própria.

A partir da resposta em malha aberta e com os dados do gráfico da resposta armazenados em uma matriz, a determinação da função transferência G(s) que representa o comportamento dinâmico do sistema de nível pode então ser determinada fazendo uso da função *ident,* Figura 28.

A curva apresenta muito ruído, o qual, provavelmente, se deve a dois fatores: primeiro, o motor CC apresenta resposta com oscilações de carga; segundo, o sensor de nível possui baixa resolução e é sensível a turbulência do líquido, a qual ocorre devido a grande espessura da válvula de drenagem.

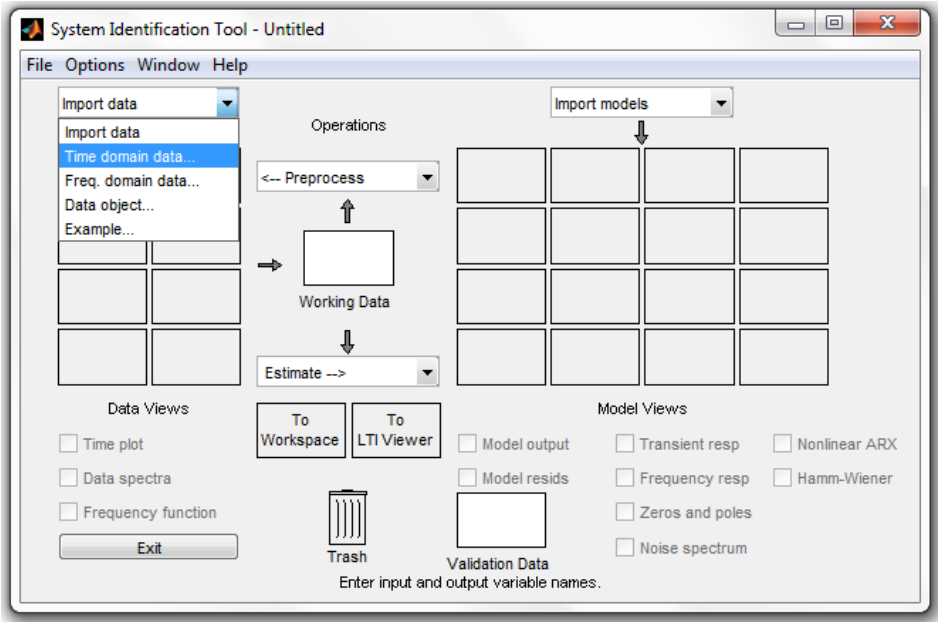


Figura 28 - Tela inicial da função IDENT do *Toolbox* de controle do Matlab.
Fonte: Autoria própria.

Os dados de entrada, ou seja, o valor do degrau aplicado na bomba CC e os valores das leituras de nível armazenados em matriz são inseridos no IDENT como dados no domínio do tempo (*Time Domain Data)*, conforme mostrado na Figura 29.

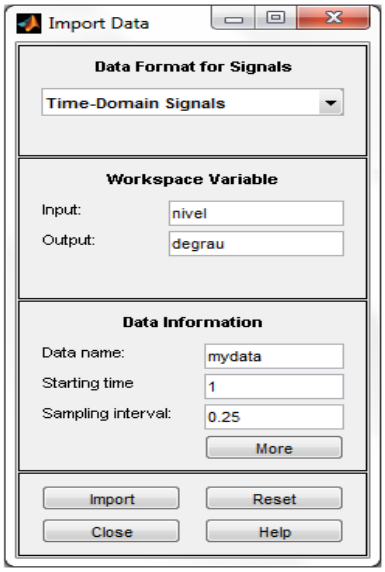


Figura 29 - Tela de importação de dados.
Fonte: Autoria própria.

Com os dados carregados faz-se então a modelagem do sistema por meio do ícone *process model* conforme Figura 30.

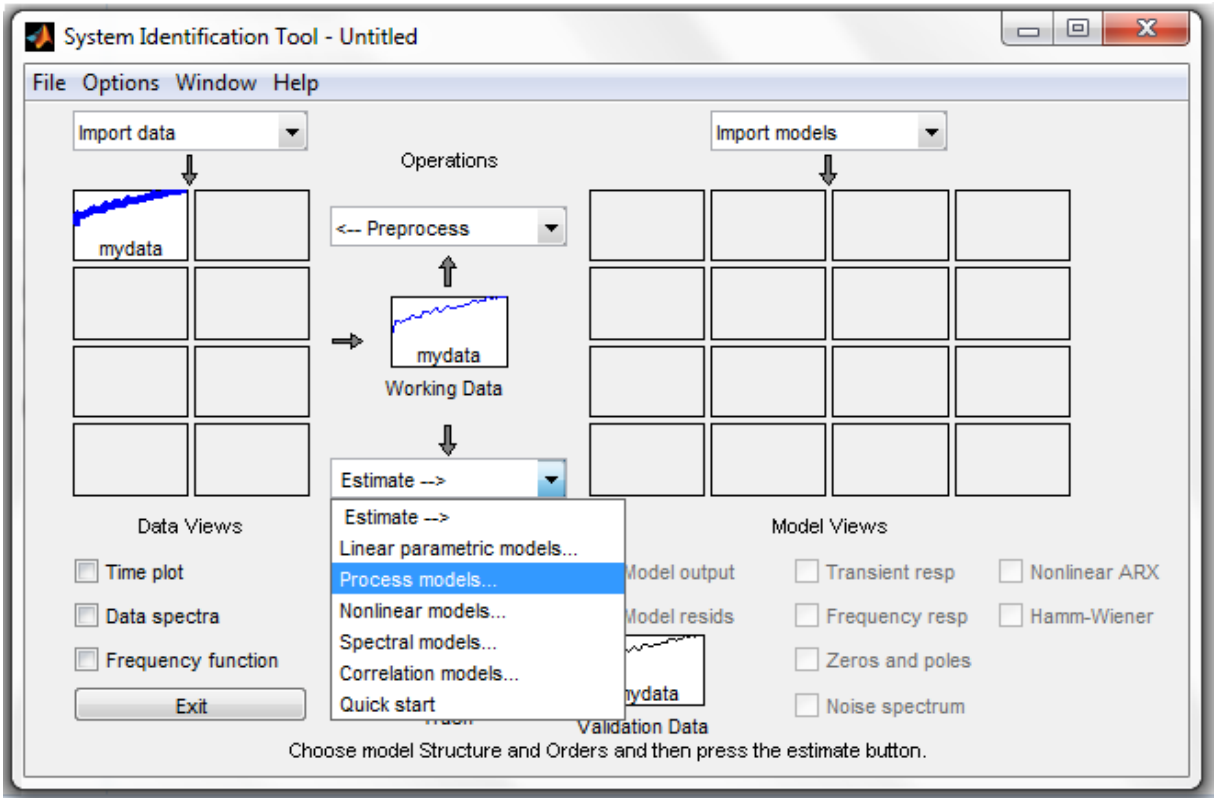


Figura 30 - Opção para determinação do modelo da planta.
Fonte: Autoria própria.

O sistema de nível construído possui um atraso de transporte mínimo, assim para a modelagem este atraso foi desconsiderado e equação a ser determinada para G(s) assume a forma mostrada na Figura 31.

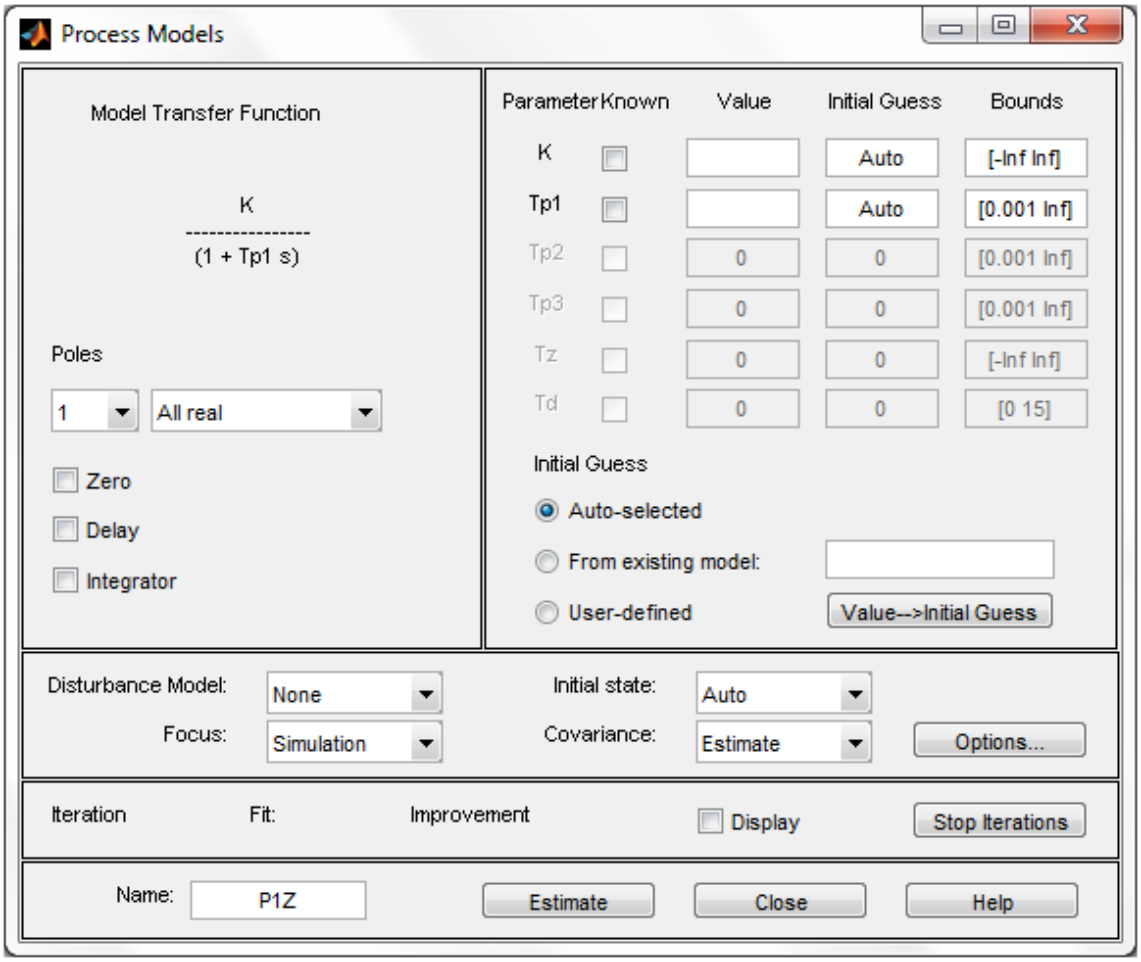


Figura 31 - Modelo da função de transferência sem atraso de transporte.
Fonte: Autoria própria.

A função G(s) determinada pela estimativa gerada pelo *ident* é uma função de primeira ordem que é descrita pela Eq. 16 abaixo.

$$G(s) = \frac{0.4357}{6.436s+1} \tag{16}$$

Levando em consideração o mínimo atraso de transporte presente na resposta em malha aberta da planta, foi inserido na Eq. 16 um valor para de L = 0.4, obtido empiricamente, para representar o atraso de transporte e melhorar a identificação do

sistema. A Eq. 17 representa a G(s) que mais se aproxima do comportamento real da planta.

$$G(s) = \frac{0.4357 \times e^{-0.4s}}{6.436s+1} \quad (17)$$

A comparação entre a resposta real do sistema obtida pelo ensaio em malha aberta e a resposta da G(s) identificada, ambas com frente ao mesmo valor de degrau de entrada é demonstrada no gráfico da Figura 32.

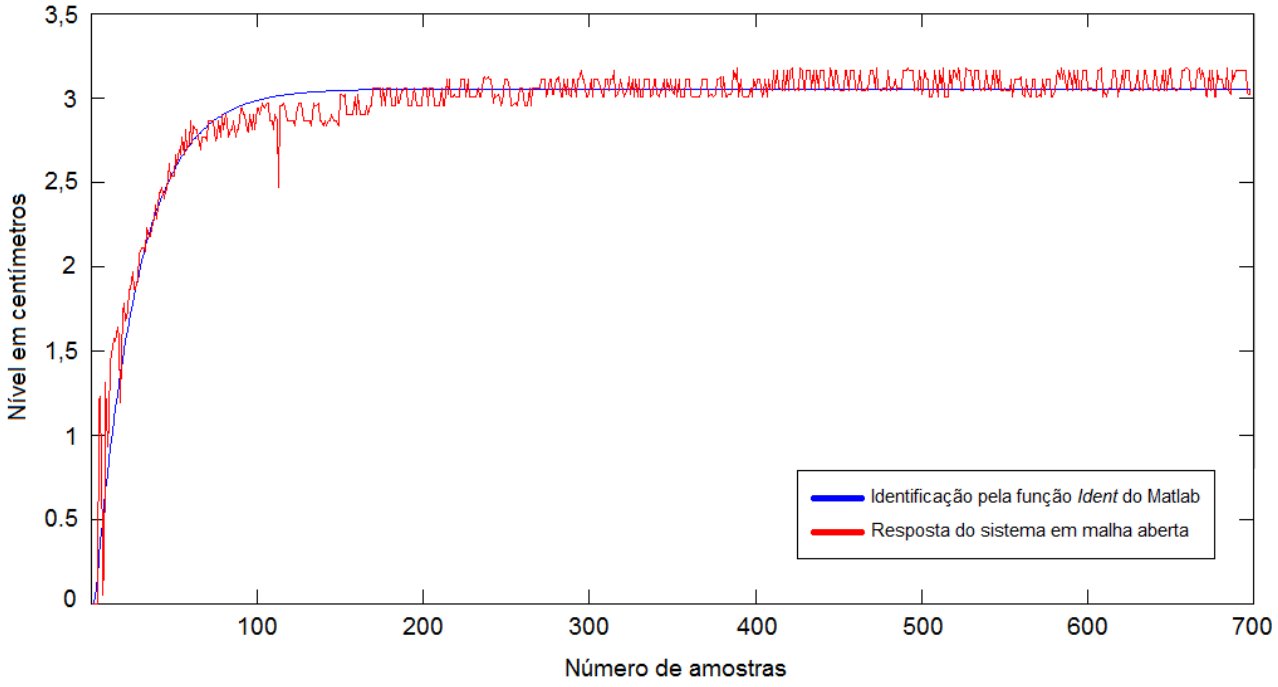


Figura 32 - Comparação entre resposta em malha aberta e G(s) identificada por meio da função IDENT do Matlab.

Fonte: Autoria própria

4.2 SINTONIA DO CONTROLE PI

Com a identificação da planta satisfatoriamente concluída, a sintonia do controle PI pode ser realizada. Utilizou-se, neste projeto, o método de sintonia de

Ziegler e *Nichols*, onde o valor do ganho proporcional, tempo integrativo e tempo derivativo podem ser diretamente definidos fazendo-se uso das equações da Tabela 1.

Conforme as Eq. 16 e Eq.17, pode-se determinar os valores das variáveis T, K e L necessárias para aplicar o método de sintonia. Assim define-se T = 0.636, K = 0.4357 e L = 0.4.

Portanto os ganhos Kp e Ki, encontrados a partir da aplicação do método de sintonia de *Ziegler e Nichols,* são: Kp = 15 e Ki = 11.

4.3 APLICAÇÃO DO CONTROLE PI

Os valores dos ganhos calculados para a sintonia do controlador PI foram aplicados na rotina de controle por meio do supervisório da planta. Três ensaios em malha fechada foram executados, estes ensaios possuem como *set point* os valores de nível equivalentes a 5 centímetros, 10 centímetros e 15 centímetros. Também foi inserido, em cada um dos ensaios em malha fechada, uma perturbação ao sistema causado pelo fechamento e abertura da válvula de drenagem do tanque superior da planta. As Figuras 33, 34 e 35 mostram os resultados dos ensaios para os respectivos valores de *set point* de nível acima citados.

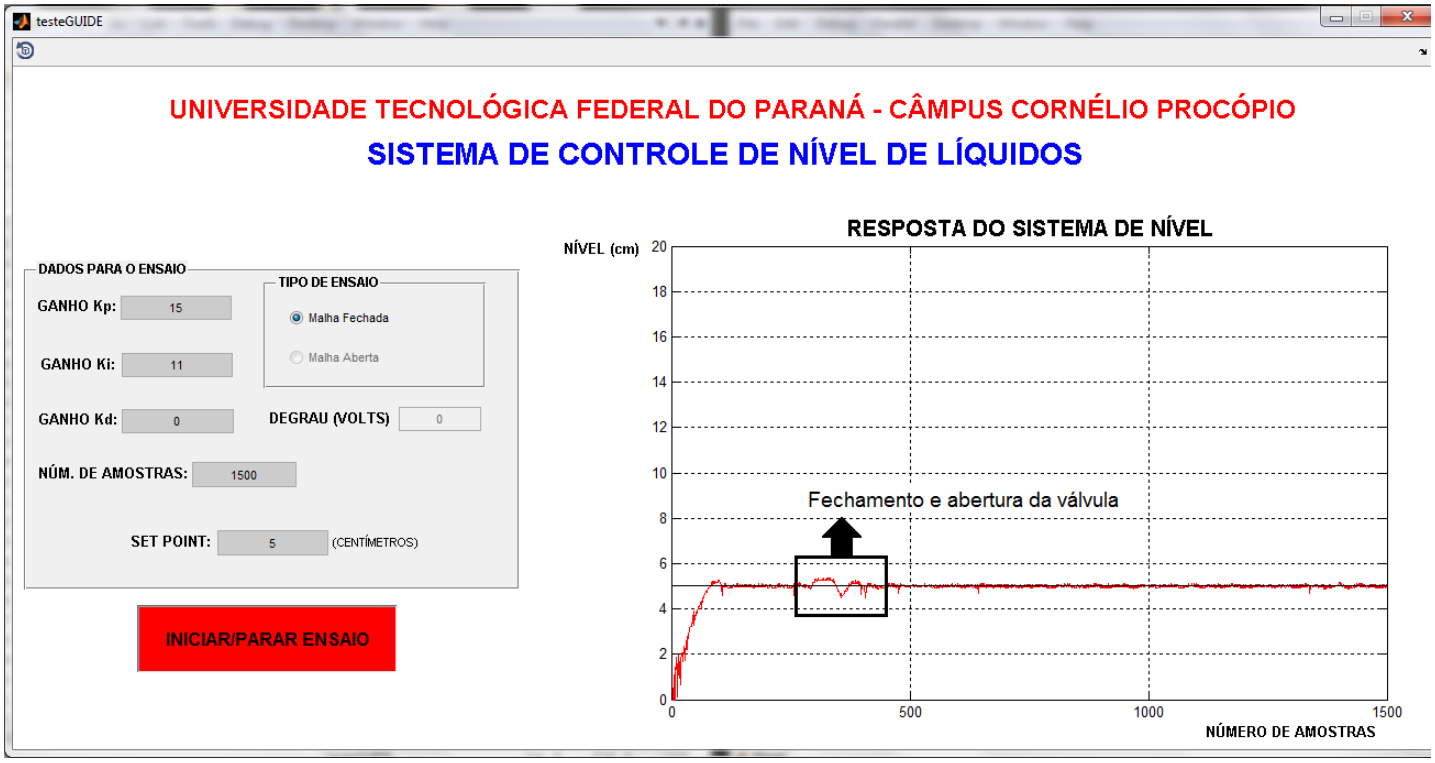


Figura 33 - Resposta em malha fechada para um *set point* de 5 centímetros.
Fonte: Autoria própria.

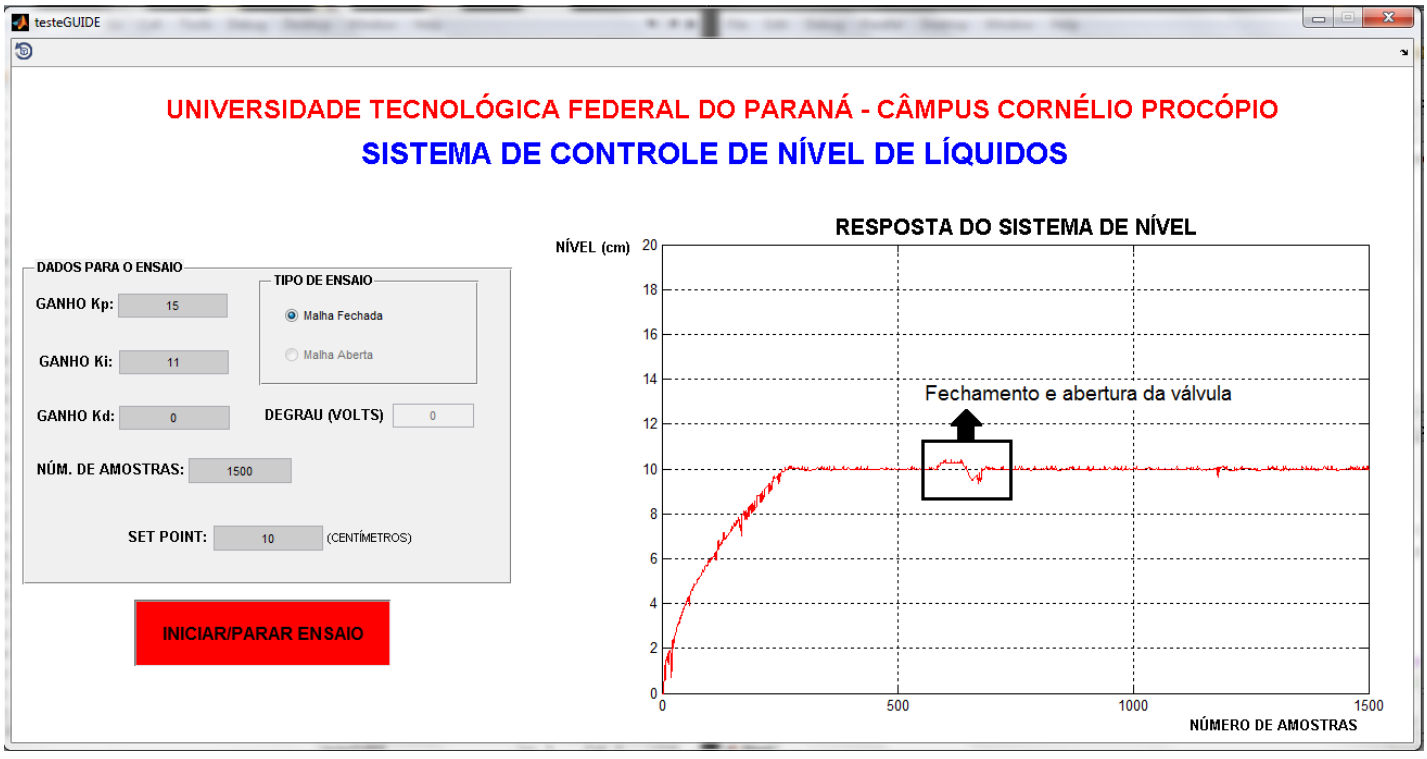


Figura 34 - Resposta em malha fechada para um *set point* de 10 centímetros.
Fonte: Autoria própria.

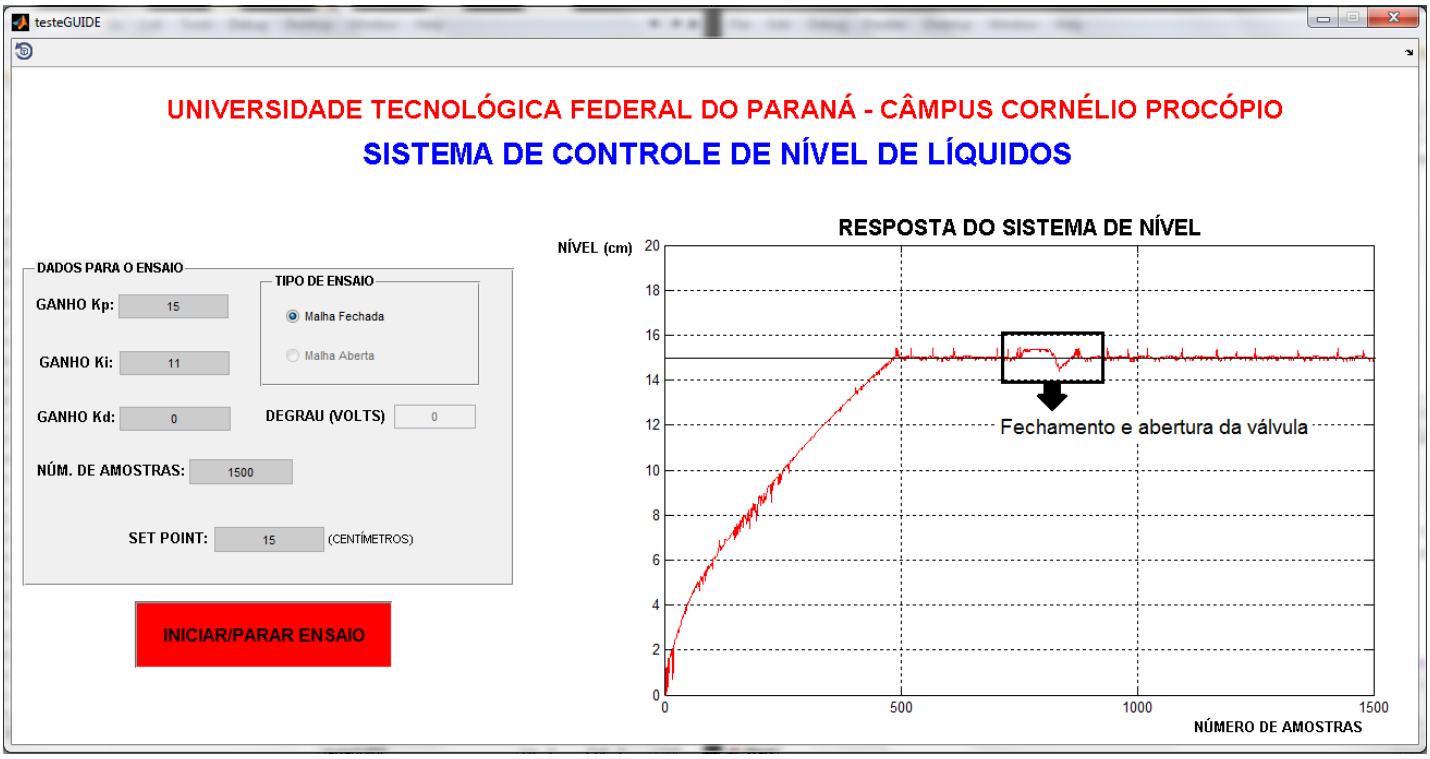


Figura 35 - Resposta em malha fechada para um *set point* de 15 centímetros.
Fonte: Autoria própria.

De acordo com o resultado dos obtidos já com o controle PI sintonizado, verificou-se que o controle PI atuou de forma eficaz, mantendo o nível do tanque superior da planta sempre na referência. Durante o período no qual houve a aplicação da perturbação, o controle PI se mostrou robusto, fazendo que a bomba DC se mantivesse desligada enquanto a válvula de drenagem se mantinha completamente fechada e retornando o valor de nível à referência quando a válvula retornava a condição de vazão máxima de água. Vale ressaltar também, que para o ensaio representado pela Figura 35, a bomba DC operava próxima a sua região de saturação, porém mesmo operando nestas condições, o controle do nível manteve sua eficácia mesmo com a presença da perturbação. Portanto a sintonia do controle PI projetado é satisfatória e a atuação do controle sobre a planta pode ser considerada estável.

5 CONCLUSÃO

O trabalho proposto apresentou resultados satisfatórios, os objetivos foram atingidos e a construção física da planta de nível junto com sua automação resultou em um módulo didático de controle de nível eficaz. Sua construção como um todo envolve materiais e componentes de baixo custo o que o torna uma opção mais viável e de boa qualidade para módulos com a mesma finalidade.

Em relação aos resultados obtidos a partir do projeto do controlador PI, pode-se concluir que a ação de controle desenvolvida também apresentou resultados satisfatórios desde o processo de identificação do sistema até a aplicação do controle projetado, permitindo um controle eficiente da planta e mantendo o nível na referência com um erro em regime permanente praticamente nulo. O sistema também mostrou-se robusto frente às perturbações causadas pelo fechamento e abertura da válvula de drenagem do tanque superior, elevando o nível em alguns milímetros acima da referência quando se fecha a válvula e alguns milímetros abaixo da referência quando se abre a válvula, porém após a aplicação da perturbação, o controle se recupera e novamente se mantém no *set point*. Os milímetros a mais e a menos provenientes da aplicação da perturbação ao controle são devido a dinâmica da planta. Observou-se então que, mesmo com o motor CC trabalhando perto da saturação, Figura 35, o controle PI implementado com a identificação sem restrições se mostrou eficiente.

A comprovação do funcionamento da planta didática proposta por meio dos resultados obtidos validou a eficiência do kit como um todo, portanto da planta propriamente dita como também sua automação em geral, assim pode-se obter uma relevante contribuição para os laboratórios da Universidade Tecnológica Federal do Paraná, pois o kit ficará a disposição de alunos e professores dos cursos de engenharia para que possa ser utilizado na aprendizagem e também na elaboração de trabalhos futuros os quais envolvam controle de nível.

REFERÊNCIAS

ASTRON, K. J.; HAGGLUND, T. **PID Controllers: Theory, Design and Tuning.** 2 ed. Research Triangle Park: Instrument Society of America, 1995.

ARDUINO, **Arduino UNO Rev 3**, 2013. Disponível em: <http://arduino.cc/en/Main/ArduinoBoardUno>. Acesso em: 21 fev. 2013.

BARR, Michael. **Pulse Width Modulation.** Embedded Systems Programing, September 2001, p 103-104. Disponível em: <http://homepage.cem.itesm.mx/carbajal/Microcontrollers/ASSIGNMENTS/readings/ARTICLES/barr01_pwm.pdf>. Acesso em: 23 mai. 2013.

CYTRON TECHNOLOGIES, **Product User's Manual – HC-SR04 Ultrasonic Sensor.** 2003. Disponível em: <https://docs.google.com/document/d/1Y-yZnNhMYy7rwhAgyL_pfa39RsB-x2qR4vP8saG73rE/edit?pli=1>. Acesso em: 15 mar. 2013.

D'AZZO, John J. **Analise e Projeto de Sistemas de Controle Lineares.** 2 ed. Rio de Janeiro: Editora Guanabara Dois S.A., 1978.

DORF, Richards C.; BISHOP, Robert H. **Sistemas de Controle Modernos.** 8 ed. Rio de Janeiro: LTC, 2001.

FRANKLIN, Gene F.; POWELL, J. David; WORKMAN, Michael L. **Digital control of dynamic systems.** 3 ed. Addison-Wesley, 1998.

KUO, Benjamin C. **Digital control systems.** New York: Oxford University Press, 1992.

LI, Y.; ANG, K.; CHONG, G. **PID control system analysis and design**. IEEE Control Systems Magazine, IEEE, v. 26, n. 1, Feb. 2006.

NISE, Norman S. **Engenharia de Sistemas de Controle.** 3. ed. Rio de Janeiro: LTC, 2002.

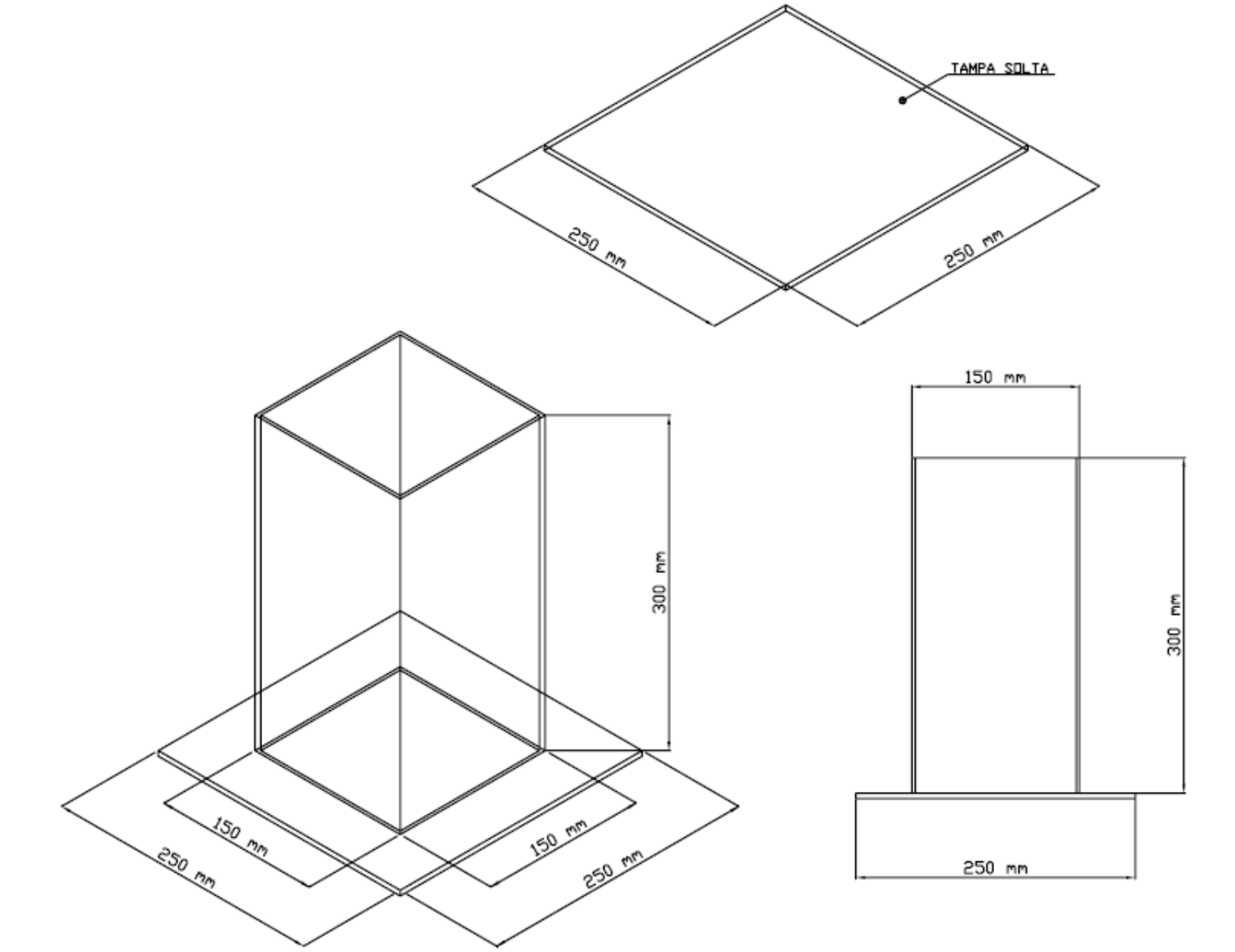
OGATA, Katsuhiko. **Engenharia de controle moderno.** 3. ed. Rio de Janeiro: LTC, 2000.

_____. **Discrete-time control systems.** 2. ed. New Jersey: Prentice Hall, 1995.

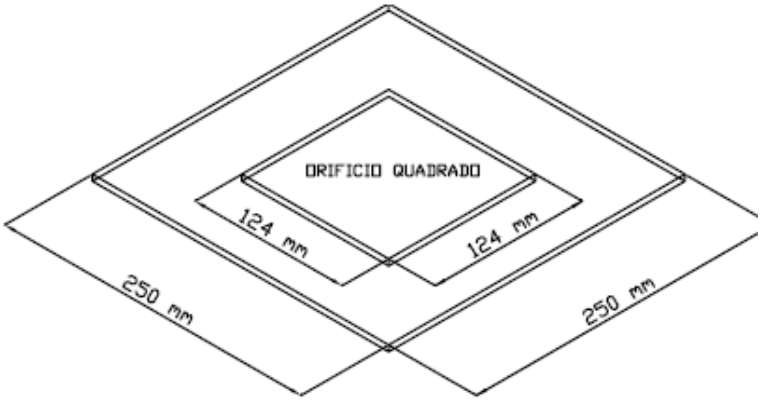
ST MICROELETRONICS, **L298N Dual Full-Bridge Driver Datasheet**. 2000. Disponível em: <https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf>. Acesso em: 15 mar 2013.

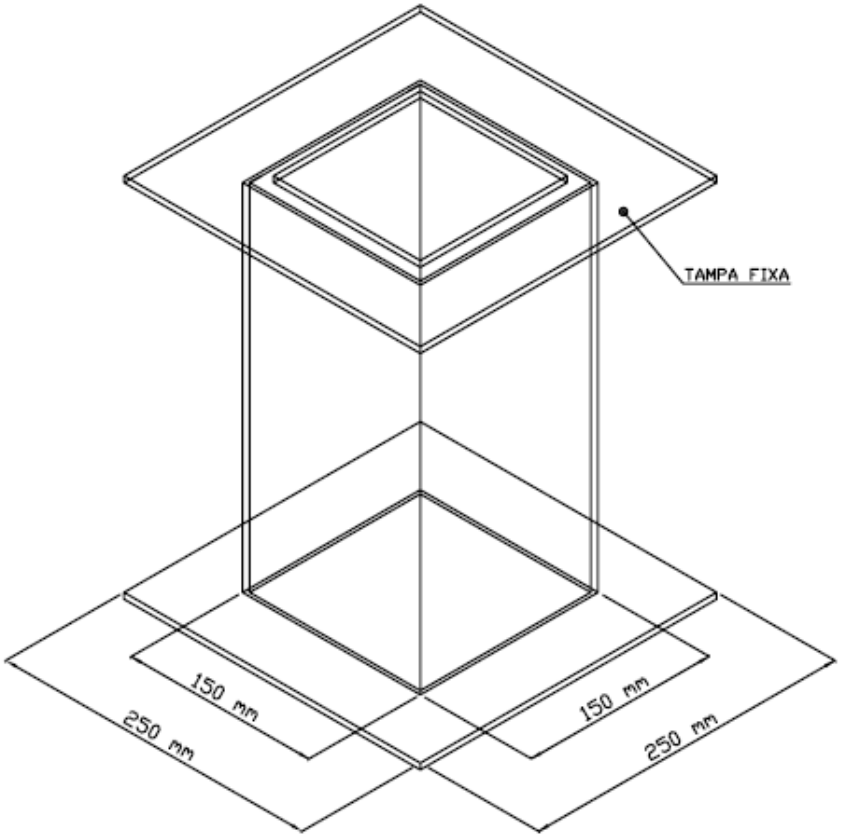
ZHAN, Wei. **Robustness Analysis of Pulse Width Modulation Control of Motor Speed.** World Congress of Engineering and Computer Science 2007, San Francisco, October 2007.

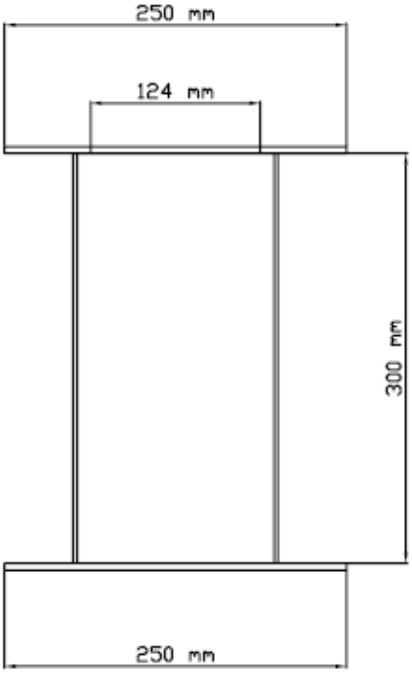
APÊNDICE A-1: PROJETO DO TANQUE SUPERIOR DA PLANTA.



APÊNDICE A-2: PROJETO DO TANQUE INFERIOR DA PLANTA.







APÊNDICE B: CÓDIGO DA INTERFACE DE CONTROLE EM MATLAB.

```
function varargout = testeGUIDE(varargin)
% TESTEGUIDE M-file for testeGUIDE.fig
%      TESTEGUIDE, by itself, creates a new TESTEGUIDE or raises the existing
%      singleton*.
%
%      H = TESTEGUIDE returns the handle to a new TESTEGUIDE or the handle to
%      the existing singleton*.
%
%      TESTEGUIDE('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in TESTEGUIDE.M with the given input arguments.
%
%      TESTEGUIDE('Property','Value',...) creates a new TESTEGUIDE or raises
the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before testeGUIDE_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to testeGUIDE_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 testeGUIDE

% Last Modified by GUIDE v2.5 06-Nov-2013 10:57:38

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @testeGUIDE_OpeningFcn, ...
                   'gui_OutputFcn',  @testeGUIDE_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 testeGUIDE is made visible.
function testeGUIDE_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 testeGUIDE (see VARARGIN)

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

% Update handles structure
guidata(hObject, handles);

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



%% INICIA COM '0' CADA LACUNA DO SUPERVISÓRIO

set(handles.k,'String','0');
set(handles.ki,'String','0');
set(handles.kd,'String','0');
set(handles.amostras,'String','0');
set(handles.sp,'String','0');
set(handles.degrau,'String','0');

%% INICIALIZAÇÃO DO GRAFICO " RESPOSTA DO SISTEMA "

axes(handles.resp_sistema);


% --- Outputs from this function are returned to the command line.
function varargout = testeGUIDE_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 togglebutton1.
function togglebutton1_Callback(hObject, eventdata, handles)
% hObject    handle to togglebutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of togglebutton1

%% VARIAVEIS DO SISTEMA
global n_amostras s_point tensao kp ki kd nivel
iniciar_ensaio = get(handles.togglebutton1,'Value');
pwm = '0';
den = 0;
nivel1 = 0.0;
saida = 0;
```

```
nivel = zeros(1,n_amostras);
erro_k = 0;
erro_k1 = 0;
erro_k2 = 0;
parc1 = 0;
parc2 = 0;
parc3 = 0;
T_samp = 0.25; % TEMPO DE AMOSTRAGEM

%% ROTINA PRINCIPAL DE CONTROLE

if iniciar_ensaio==1
    set(handles.togglebutton1,'BackgroundColor','Green');

    handle(handles.resp_sistema);
    graf=plot(nan,nan,'r-');
    grid on;
    axis([0 n_amostras 0 20]);


    % INICIALIZAÇÃO DA COMUNICAÇÃO SERIAL
    delete(instrfind({'Port'},{'COM3'}));
    %CRIA CONEXÃO COM A SERIAL COM3
    s = serial('COM3','BaudRate',9600);
    warning('off','MATLAB:serial:fscanf:unsuccessfulRead');
    %ABRE PORTA
    fopen(s);
    fprintf(s,pwm);

   for k=1 : n_amostras

       a = get(handles.ma,'Value');

       if (get(s, 'BytesAvailable') == 0)
      else
          if (get(s, 'BytesAvailable') >0.00)
              nivel1 = fscanf(s,'%f');
          end
       end

       nivel(k) = nivel1;
       if a==1 % SELECIONA ENSAIO EM MALHA ABERTA
           den = 12;
           saida = tensao;
       else
           if a==0 % SELECIONA ENSAIO EM MALHA FECHADA
               den = 20;
               erro_k = s_point - nivel(k); % CÁLCULO DO ERRO

               % AÇÃO DE CONTROLE PID

               parc1 = kp*(erro_k - erro_k1); % AÇÃO PROPORCIONAL
               parc2 = ((ki*T_samp)/2)*(erro_k + erro_k1); % AÇÃO INTEGRAL
               parc3 = (kd/T_samp)*(erro_k - 2*erro_k1 + erro_k2); % AÇÃO
DERIVATIVA
```

```
                saida = saida + parc1 + parc2 + parc3;

                erro_k2 = erro_k1;
                erro_k1 = erro_k;

                if saida>20
                    saida = 20;
                else
                    if saida<0
                        saida = 0;
                    end
                end

            end
        end

        pwm =(saida*255)/den;
        pwm = round(pwm);
        pwm = num2str(pwm);
        fprintf(s,pwm);
        set(graf,'xdata',[1:k],'ydata',[nivel([1:k])]);
        pause(0.25);

        if get(handles.togglebutton1,'Value')==0
            iniciar_ensaio = 0;
            set(handles.togglebutton1,'BackgroundColor','Red');
            msgbox('ENSAIO CANCELADO');
            break
        end

    end
        fprintf(s,'0'); % seta a bomba DC com 0 Volts
        delete(instrfind({'Port'},{'COM3'})); % Deleta porta COM3 e finaliza a
comunicação
        clear s;
        inicia_ensaio = 0;
        set(handles.togglebutton1,'BackgroundColor','Red');
        uisave({'nivel'}); %SALVA DADOS DO ENSAIO
end
guidata(hObject,handles)


% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over
togglebutton1.
function togglebutton1_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to togglebutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes during object creation, after setting all properties.
function togglebutton1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to togglebutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns called



function k_Callback(hObject, eventdata, handles)
% hObject    handle to k (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 k as text
%        str2double(get(hObject,'String')) returns contents of k as a double
global kp

kp_1 = get(handles.k,'String');
kp = str2num(kp_1);
if kp<0
    msgbox('NÃO É POSSÍVEL UTILIZAR NÚMEROS NEGATIVOS. DEFINA UM NOVO
NÚMERO');
    set(handles.k,'String',num2str(0));
    kp = 0;
end
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function k_CreateFcn(hObject, eventdata, handles)
% hObject    handle to k (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 ki_Callback(hObject, eventdata, handles)
% hObject    handle to ki (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 ki as text
%        str2double(get(hObject,'String')) returns contents of ki as a double
global ki

ki_1 = get(handles.ki,'String');
ki = str2num(ki_1);
if ki<0
    msgbox('NÃO É POSSÍVEL UTILIZAR NÚMEROS NEGATIVOS. DEFINA UM NOVO
NÚMERO');
    set(handles.k,'String',num2str(0));
    ki = 0;
end
```

```
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function ki_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ki (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 kd_Callback(hObject, eventdata, handles)
% hObject    handle to kd (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 kd as text
%        str2double(get(hObject,'String')) returns contents of kd as a double
global kd

kd_1 = get(handles.kd,'String');
kd = str2num(kd_1);
if kd<0
    msgbox('NÃO É POSSÍVEL UTILIZAR NÚMEROS NEGATIVOS. DEFINA UM NOVO
NÚMERO');
    set(handles.k,'String',num2str(0));
    kd = 0;
end
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function kd_CreateFcn(hObject, eventdata, handles)
% hObject    handle to kd (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 amostras_Callback(hObject, eventdata, handles)
% hObject    handle to amostras (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 amostras as text
%        str2double(get(hObject,'String')) returns contents of amostras as a
double
global n_amostras

amostra_1 = get(handles.amostras,'String');
n_amostras = str2num(amostra_1);
if n_amostras<1000
    msgbox('UTILIZAR NÚMERO DE AMOSTRAS >= 1000');
    set(handles.amostras,'String',num2str(0));
    n_amostras = 0;
end
guidata(hObject,handles);


% --- Executes during object creation, after setting all properties.
function amostras_CreateFcn(hObject, eventdata, handles)
% hObject    handle to amostras (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 sp_Callback(hObject, eventdata, handles)
% hObject    handle to sp (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 sp as text
%        str2double(get(hObject,'String')) returns contents of sp as a double
global s_point

sp_1 = get(handles.sp,'String');
s_point = str2num(sp_1);
if s_point<0
    msgbox('NÃO É POSSÍVEL UTILIZAR NÚMEROS NEGATIVOS. DEFINA UM NOVO
NÚMERO.');
    set(handles.sp,'String',num2str(0));
    s_point = 0;
end
guidata(hObject,handles)

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

% Hint: place code in OpeningFcn to populate resp_sistema


% --- Executes during object creation, after setting all properties.
function uipanel1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to uipanel1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called


% --- Executes when selected object is changed in uipanel1.
function uipanel1_SelectionChangeFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in uipanel1
% eventdata  structure with the following fields (see UIBUTTONGROUP)
%   EventName: string 'SelectionChanged' (read only)
%   OldValue: handle of the previously selected object or empty if none was
selected
%   NewValue: handle of the currently selected object
% handles    structure with handles and user data (see GUIDATA)

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

% Hint: get(hObject,'Value') returns toggle state of mf
set(handles.mf,'Value',1);
set(handles.ma,'Value',0);
set(handles.ma,'Enable','off');
set(handles.degrau,'Enable','off');

guidata(hObject,handles)


% --- Executes on button press in ma.
function ma_Callback(hObject, eventdata, handles)
% hObject    handle to ma (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

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

% Hint: get(hObject,'Value') returns toggle state of ma
set(handles.mf,'Value',0);
set(handles.ma,'Value',1);
set(handles.mf,'Enable','off');
set(handles.k,'Enable','off');
set(handles.ki,'Enable','off');
set(handles.kd,'Enable','off');
set(handles.sp,'Enable','off');

guidata(hObject,handles)



function degrau_Callback(hObject, eventdata, handles)
% hObject    handle to degrau (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 degrau as text
%        str2double(get(hObject,'String')) returns contents of degrau as a
double
global tensao

degrau_1 = get(handles.degrau,'String');
tensao = str2num(degrau_1);
if tensao<0
    msgbox('NÃO É POSSÍVEL UTILIZAR TENSÃO MENOR A 0V, DEFINA UM NOVO
VALOR.');
    set(handles.degrau,'String',num2str(0));
    tensao = 0;
else
    if tensao>12
        msgbox('NÃO É POSSÍVEL UTILIZAR TENSÃO MAIOR A 12V, DEFINA UM NOVO
VALOR');
        set(handles.degrau,'String',num2str(0));
        tensao = 0;
    end
end
guidata(hObject,handles);



% --- Executes during object creation, after setting all properties.
function degrau_CreateFcn(hObject, eventdata, handles)
% hObject    handle to degrau (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
```

APÊNDICE C: CÓDIGO DE INTERFACE MATLAB/PLANTA IMPLEMENTADO NO ARDUINO UNO.

```
#include "Ultrasonic.h" / Biblioteca do sensor de nível

#define pwm 6


int k=1;

int duty = 0;


Ultrasonic ultrasonic(8,12);


void setup()

{

  Serial.begin(9600); / Inicialização da comunicação serial

  pinMode(pwm, OUTPUT);

}


void loop()

{

  if (Serial.available() && k==1)

  {

    duty = Serial.parseInt();

    k = 0;

  }


  if (k==0)
```

```
  {
  analogWrite(pwm, duty);
  float dist  =  ultrasonic.Timing(); / Leitura do nivel
  delay(10);
  float nivel = 28.78 -(dist/58); / Conversão da leitura do nivel
  Serial.println(nivel,4);
  k=1;
  }
}
```