UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

LUCAS BROLLO ALPI

Desenvolvimento de uma Planta Didática para o Controle de Nível de Tanques Acoplados

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação.

Orientador: Prof. Dr. Jeferson Vieira Flores

Porto Alegre
2016

AGRADECIMENTOS

Meus sinceros agradecimentos a todos que colaboraram comigo para que fosse possível o desenvolvimento e conclusão deste curso de Engenharia de Computação na tão prestigiada Universidade Federal do Rio Grande do Sul, um sonho pessoal que eu tinha e que neste momento se torna realidade.

Agradeço à Universidade pela infraestrutura e interesse em formar grandes alunos. Reconheço o enorme incentivo e dedicação para que seus alunos sejam os mais atualizados e ambientados com o cenário atual. Fico muito feliz e realizado por ter nesta caminhada professores tão dedicados e interessados na qualidade de ensino durante a graduação, alguns ensinando não somente conhecimentos técnicos, mas interessados em trabalhar o lado pessoal e humano de todos os alunos.

Entre todos os professores que tive, faço um agradecimento especial ao meu orientador, o prof. Dr. Jeferson Vieira Flores, que sempre se mostrou disponível e envolvido na resolução e conclusão de muitas dúvidas que tive durante o desenvolvimento deste trabalho de conclusão. A sua personalidade atenciosa e preocupada trouxe empatia no decorrer do trabalho que gerou entusiasmo e dedicação para concluirmos esta caminhada juntos e com sucesso.

Agradeço também a minha família que me deu a base durante muitos anos, repleta de paciência e ensino, para ser formado caráter pessoal capaz de fazer escolhas acertadas durante minha vida. Sempre me incentivaram e apoiaram nos caminhos que percorri, sinto que todos desejam meu sucesso e felicidade de forma altruísta, e torcem muito por mim. Aprendi muita coisa com meus pais e o agradecimento por isso não cabe neste pequeno parágrafo, não posso deixar de agradecer toda minha família por esta conquista, em especial aos meus pais, a eles meu reconhecimento e muito obrigado por tudo.

Finalmente agradeço a Deus por ter me acompanhado em toda a vida. Creio que temos um relacionamento amigável e que Ele anda comigo. Reconheço que muitas coisas na minha vida só foram possíveis através dEle, assim como o ingresso nesta Universidade e conclusão do curso. A capacidade e dedicação para cumprir toda esta jornada que representa desenvolver o curso de Engenharia não vem só de mim, mas creio que tenho ganhado ajuda especial desde que escolhi andar mais perto de Deus, a Ele o meu sincero agradecimento.

RESUMO

O controle de tanques acoplados é de grande interesse, pois este normalmente representa partes de processos industriais como, por exemplo, em empresas petroquímicas, de celulose ou tratamento de água. Mesmo com uma estrutura relativamente simples, os controladores PID são os mais utilizados na indústria para controle de malhas de processos industriais, estão presentes em rotinas programadas de vários equipamentos comerciais que executam controle. Este trabalho tem por objetivo a construção de uma planta didática de tanques acoplados visando demonstrar as características de funcionamento e a aplicação dos controladores PID nestes sistemas. O funcionamento prático serve para compreensão de processos industriais que possuem as mesmas características, assim como para complemento dos métodos de aprendizagem teóricos vistos em sala de aula.

Palavras-chave: Controle PID. Sistemas de Controle. Tanques acoplados. Teoria de Controle. Controle Clássico.

Developing of a Didactic Plant for Coupled Tank Level Control

ABSTRACT

The coupled tanks control is of great interest because they usually represent parts of industrial processes such as, for example, in petrochemical companies, cellulose or water treatment. Even with a relatively simple structure, PID controllers are the most used in industry for industrial process mesh control, they are present in programmed routines of various commercial equipment that perform control. This work aims at the construction of a didactic plant coupled tanks seeking to demonstrate the operating characteristics and the application of PID controllers in these systems. The practical operation serves to understand of industrial processes that have the same characteristics as well as to complement the theoretical learning methods seen in class.

Keywords: PID Control. Control Systems. Coupled Tank. Control Theory. Control Classic.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

PID	Proporcional-Integral-Derivativo
K_p	Proporcional
T_i	Integral
T_d	Derivativo
LGR	Lugar Geométrico das Raízes
CLP	Controlador Lógico Programável
SDCD	Sistema Digital de Controle Distribuído
EDO	Equação Diferencial de Primeira Ordem
SPO	Sistema de Primeira Ordem
SSO	Sistema de Segunda Ordem
R.P.	Regime Permanente
cm	centímetros
m	metros
mm	milímetros
IHM	Interface Homem-Máquina
PIC	Programmable Interface Controller (Controlador de Interface Programável)
PCB	Printed Circuit Board (Placa de Circuito Impresso)
PWM	Pulse Width Modulation (Modulação de Largura de Pulso)
RAM	Random Access Memory (Memória de Acesso Aleatório)
ROM	Read-Only Memory

SUMÁRIO

1 INTRODUÇÃO

O sucesso econômico na indústria é resultado de uma administração empresarial eficiente, e principalmente, do uso dos recursos com máxima relação custo-benefício. Quando recursos são subutilizados perde-se tempo de produção e lucro, quando são superutilizados, perde-se tempo de vida útil, e em longo prazo, o lucro de produção em função da necessidade de reposição. Para alcançar a máxima eficiência na produção usualmente são empregadas na indústria malhas e algoritmos de controle, isto para garantir que o sistema opere explorando seus limites dentro de um padrão de utilização aceitável. Frequentemente a *planta* industrial de uma empresa possui tanques de armazenamento acoplados em outros tanques que são de grande importância para o *processo* e também precisam de *controle* quando em operação.

Processos industriais usam com frequência tanques acoplados para várias finalidades, como por exemplo, armazenamento e transporte de líquidos. Indústrias petroquímicas, de celulose ou de tratamento de líquidos, costumam possuir processamento de líquidos por química ou tratamento de misturas. Este processamento sempre precisa de um controle criterioso sobre o nível do fluído, assim como regular o fluxo entre os tanques. O *controle do nível* de líquidos é um problema comum, pois para realizar qualquer ação precisa-se saber o nível dos tanques envolvidos, e no caso de transporte, o fluxo entre os tanques. Usualmente os tanques são acoplados em conjunto de modo que os níveis de interação também devem ser controlados na planta (LAUBWALD, 2015).

Mesmo com uma estrutura relativamente simples, os *controladores PID* (Proporcional-Integral-Derivativo) mostram-se suficientes para o controle adequado de muitos processos. Pode-se dizer que ainda hoje, apesar da existência de técnicas de controle mais avançadas, é o controlador mais utilizado na indústria (BAZANELLA, 2005). Segundo Aström; Hägglund (1995), estimativas apontam que cerca de 90% das *malhas de controle* em *processos industriais* operam com este controlador. Na indústria são encontrados como equipamentos dedicados à execução de um algoritmo *PID* em uma *malha de controle*. Alguns exemplos são funções programadas em *CLPs* (controlador lógico programável), blocos funcionais em *SDCDs* (sistemas digitais de controle distribuído), e barramentos industriais (BAZANELLA, 2005).

O estudo e modelagem destes dois elementos unidos, a *planta de tanques acoplados* e o *controlador PID*, são de extrema importância, pois abrangem o funcionamento prático didático e proporcionam entendimento do cenário atual de muitas empresas. A proposta deste trabalho é a construção de uma planta didática de tanques acoplados para demonstrar as

características de funcionamento e a aplicação dos *controladores PID* neste sistema através do controle de nível de um dos tanques. Será montada uma estrutura com dois tanques e um reservatório em cascata, onde se deseja controlar o nível do segundo tanque conforme parâmetros configurados pelo operador.

A organização deste relatório foi dividida em partes para facilitar o entendimento, começando pelo segundo capítulo com a revisão bibliográfica, que tem objetivo de fundamentar os conceitos que são explorados no projeto, seguido pelo terceiro e maior capítulo deste relatório, onde é apresentada a descrição detalhada de todo o projeto da bancada experimental, software e hardware. No quarto capítulo, é feita a análise e apresentação dos resultados, assim como a identificação do modelo da planta e o projeto do controlador. No quinto é apresentada a conclusão do projeto, depois, são apresentadas as referências utilizadas, e por fim a seção de apêndice.

2 REVISÃO BIBLIOGRÁFICA

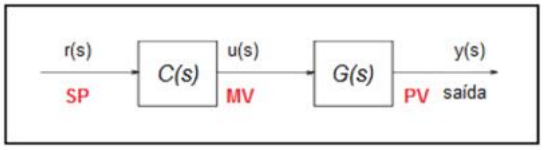
Neste capítulo, é apresentada uma contextualização dos conceitos que foram explorados ao longo do projeto. A *Seção 2.1* apresenta a teoria base de *Sistemas de Controle*, dividindo-os na *Seção 2.1.1* sobre *Sistemas de Primeira Ordem*, e na *Seção 2.1.2* sobre *Sistemas de Segunda Ordem*. Na *Seção 2.2* é realizada a modelagem do *Sistema de Tanques Acoplados*, começando pelo *Modelo Não-Linear* na *Seção 2.2.1*, e depois o *Modelo de Tanques Acoplados* na *Seção 2.2.2*. Por fim, a *Seção 2.3* apresenta a fundamentação sobre o *Controlador PID* explicando cada ação de controle individualmente e depois todas juntas, onde a *Ação Proporcional (K_p)* está na *Seção 2.3.1*, a *Ação Integral (T_i)* está na *Seção 2.3.2*, e a *Ação Derivativa (T_d)* está na *Seção 2.3.3*. Todas as ações juntas, *Proporcional-Integral-Derivativa*, estão na *Seção 2.3.4*.

2.1 Sistemas de Controle

Na indústria chama-se o *sistema* a ser controlado de *processo* ou *planta*. Segundo Bazanella (2005), o sinal que é aplicado na entrada do *processo G(s)* é chamado de *sinal de controle u(s)* ou *variável manipulada MV*, pois seu valor pode ser manipulado. O sinal de *saída y(s)* do *processo* é chamado de *variável controlada* ou *variável de processo PV*, por que corresponde à variável que se deseja controlar o comportamento. Para alcançar a máxima eficiência desejada da *planta* é necessário o uso de um *controlador C(s)* na entrada do *processo*, a fim de garantir estabilidade do sinal de *saída*. O sinal aplicado na entrada do *controlador* é o valor de referência chamado de *referência de entrada r(s)* ou *set point SP*.

Um sistema sem realimentação na *saída y(s)* configura uma *malha-aberta*, ou seja, a *saída* não tem efeito na ação do controle. A *saída* não é medida ou realimentada para comparação com o valor de *referência de entrada r(s)* como demonstrado na Figura 2.1, consequentemente, cada entrada de *referência* corresponde a uma condição de operação fixa. Um sistema em *malha-aberta* não é capaz de rejeitar perturbações, e para ser utilizado e obter bons resultados deve-se conhecer muito bem o processo e não pode haver perturbações externas ou internas (OGATA, 1982).

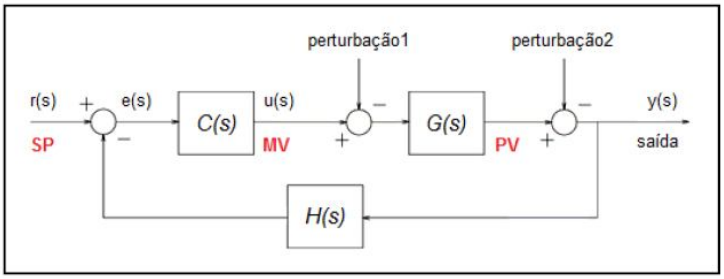
Figura 2.1 – Sistema de controle em *malha-aberta*



Fonte: Elaborado pelo autor.

Quando o sistema tem realimentação da saída ou forma um laço como apresentado no diagrama de blocos da Figura 2.2, ele configura uma *malha-fechada*, isto ocorre através do *sensor H(s)* que é normalmente um sensor que converte o sinal de *saída y(s)* para mesma unidade do sinal de *referência r(s)* para comparação. O *controlador C(s)* recebe o resultado da subtração entre *r(s)* e o valor real de *y(s)* do *processo*, gera o sinal de *erro e(s)*, e produz um *sinal de controle* que tenta reduzir o *erro* a um valor muito pequeno ou nulo (OGATA, 1982), independente se tiver sinais de *perturbação* ou não.

Figura 2.2 – Sistema de controle em *malha-fechada*



Fonte: Elaborado pelo autor.

Quando o *sistema* opera com controle em *malha-fechada* é possível obter ganhos em seu comportamento, como por exemplo, aumento da precisão do *sistema de controle*, rejeição dos efeitos de perturbação sobre a variável de processo, e diminuição da sensibilidade do comportamento do sistema a variações dos parâmetros do processo (aumenta robustez do sistema).

2.1.1 Sistemas de Primeira Ordem

Grande parte da teoria de Sistemas de Controle está fortemente embasada em conceitos matemáticos consolidados. A teoria de Equações Diferenciais descreve uma *Equação Diferencial de Primeira Ordem (EDO)* como a derivada de primeira ordem de uma função. Segundo DEQ/UFSCar (2016a), Sistemas de Controle visando o controle de uma *planta* descrita por uma *Equação Diferencial de Primeira Ordem* são chamados de *Sistemas*

de Primeira Ordem (SPO), estes possuem a *dinâmica de saída y(t)* descrita por uma *EDO*. Considerando a condição inicial *y(0)=0* e $a_0 \neq 0$, descreve-se pela Equação 2.1 a formulação de um *SPO*.

$$a_1 \frac{dy}{dt} + a_0 y = b\,u, \; y(0) = 0$$
$$\frac{a_1}{a_0}\frac{dy}{dt} + y = \frac{b}{a_0}\,u, \qquad y(0) = 0 \tag{2.1}$$

Substituindo o valor das constantes por $\tau_p = \frac{a_1}{a_0}$ e $K_p = \frac{b}{a_0}$, tem-se a forma padrão de representar um *SPO*. O valor τ_p é a *constante de tempo do sistema* que indica a rapidez com que a resposta do sistema reage a uma variação da entrada, já o valor K_p é o *ganho DC* do *processo* que é a razão entre os valores finais da resposta e de uma determinada entrada considerada. A partir destas variáveis a Equação 2.1 pode ser escrita como a Equação 2.2.

$$\tau_p \frac{dy}{dt} + y = K_p\,u, \qquad y(0) = 0 \tag{2.2}$$

A *Tranformada de Laplace* serve para transformar *Equações Diferenciais* em *Equações Algébricas*, mudando o *domínio tempo (t)* para o *domínio freqüência (s)*. Aplicando a *Transformada de Laplace* em ambos os lados da *EDO*, obtem-se o formato da equação esperada para um *SPO*. Observa-se que a *função de transferência* $G(s)$ na Equação 2.3 representa a relação entre o valor da saída e a entrada do sistema.

$$\tau_p\, s\, Y(s) + Y(s) = K_p\, U(s)$$
$$(\tau_p s + 1) Y(s) = K_p\, U(s)$$
$$G(s) = \frac{Y(s)}{U(s)} = \frac{K_p}{(\tau_p)\, s + 1} \tag{2.3}$$

2.1.2 Sistemas de Segunda Ordem

Para encontrar o formato da equação de um *Sistema de Segunda Ordem (SSO)* são tomados os mesmos passos descritos para o *SPO*. Segundo DEQ/UFSCar (2016b), Um *Sistema de Segunda Ordem* é aquele cuja *resposta y(t)* é descrita por uma *Equação Diferencial de Segunda Ordem*. Considerando as condições iniciais *y'(0)=0*, *y(0)=0*, e $a_0 \neq 0$, um *SSO* pode ser descrito pela Equação 2.4.

$$a_2 \frac{d^2y}{dt^2} + a_1 \frac{dy}{dt} + a_0 y = b\,u, \qquad y'(0) = y(0) = 0$$

$$\frac{a_2}{a_0}\frac{d^2y}{dt^2} + \frac{a_1}{a_0}\frac{dy}{dt} + y = \frac{b}{a_0}\,u, \qquad y'(0) = y(0) = 0 \tag{2.4}$$

Substituindo as constantes por $\tau^2_p = \frac{a_2}{a_0}$, $2\zeta\tau_p = \frac{a_1}{a_0}$ e $K_p = \frac{b}{a_0}$, tem-se a forma padrão para representação de um *SSO*, conforme a Equação 2.5. O valor τ_p agora é o *tempo característico* ou *período natural de oscilação* e indica a rapidez com que a resposta do sistema reage a uma perturbação na entrada, o valor ζ é o *fator de amortecimento* que tem valor adimensional e mede o grau de amortecimento da resposta do sistema. Em outras palavras ele mede o grau de oscilação na resposta após uma variação em alguma variável de entrada; por fim o valor K_p novamente chamado de *ganho DC* do *processo*, que é a razão entre os valores finais da resposta e de uma determinada entrada considerada.

$$\tau^2_p\frac{d^2y}{dt^2} + 2\zeta\tau_p\frac{dy}{dt} + y = K_p\,u, \qquad y'(0) = y(0) = 0 \tag{2.5}$$

Aplicando a *Transformada de Laplace* em ambos os lados da *Equação Diferencial de Segunda Ordem* é possível encontrar a relação frequencial que descreve um *SSO*, demonstrado na Equação 2.6.

$$\begin{gathered}\tau^2_p\,s^2\,Y(s) + 2\zeta\tau_p\,s\,Y(s) + Y(s) = K_p\,U(s)\\ (\tau^2_p\,s^2 + 2\zeta\tau_p\,s + 1)Y(s) = K_p\,U(s)\\ G_p(s) = \frac{Y(s)}{U(s)} = \frac{K_p}{(\tau^2_p)\,s^2 + (2\zeta\tau_p)\,s + 1}\end{gathered} \tag{2.6}$$

2.2 Sistemas de Tanques Acoplados

Para realizar o controle sobre a planta de tanques acoplados é preciso detalhar e considerar algumas características importantes do sistema, como por exemplo, a não-linearidade do modelo e também a necessidade de linearização do mesmo. O controle somente será possível depois de linearizar o modelo e obter a função de transferência, como apresentado a seguir.

2.2.1 Modelo Não-Linear

Entender matematicamente o comportamento de tanques acoplados é de fundamental importância, pois faz parte da modelagem do sistema. Para obter o controle sobre o nível do tanque é necessário criar uma relação matemática entre o nível e o fluxo de recebimento deste

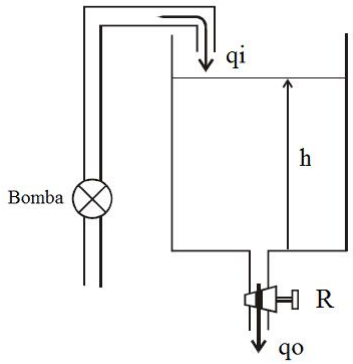
líquido, a qual permitirá ao projetista determinar um *controlador C(s)* mais adequado para a obtenção de uma resposta desejada.

Baseado na modelagem apresentada em Ogata (1982), sistemas envolvendo fluxos de fluídos são distinguidos em regime de fluxo *laminar* e *turbulentos*, de acordo com o valor do *número de Reynolds*. Se este número for menor do que 2.000 o fluxo é *laminar* e pode ser representado por Equações Diferenciais lineares, se o número for maior que 3.000, o fluxo é *turbulento* e tem de ser representado por Equações Diferenciais não-lineares. Usualmente em aplicações de tanques acoplados o fluxo é *turbulento*.

A classificação de um sistema como *LTI (linear e invariante no tempo)* significa que o mesmo é *linear* quando satisfaz as propriedades de linearidade e superposição, e *invariante no tempo* quando se comporta da mesma forma independente do instante de tempo onde o sinal de referência é aplicado (HAYKIN, 1999). O sistema de tanques acoplados é um sistema *invariante no tempo* e *não-linear*, logo, precisará passar pelo método de linearização por *expansão em série de Taylor* em torno de um ponto de operação (a relação linear será válida apenas para pequenas variações em torno deste ponto de operação).

Em geral as equações obtidas são *invariantes no tempo*, porém, nem todas são *lineares*. Segundo Bazanella (2005), a importância da linearização de equações se justifica por causa da necessidade de torná-las como um sistema *LTI* para que seja possível aplicar métodos importantes de cálculo, como a *Transformada de Laplace* para obter a *função de transferência* do sistema. Observando a Figura 2.3, e considerando C como a *multiplicação da área do reservatório pela densidade do líquido*, q_i como a *vazão de entrada*, q_o a *vazão de saída*, e h o *nível do tanque*, é possível obter a Equação 2.7 que modela a *variação instantânea do nível* em um tanque.

Figura 2.3 – Modelagem do nível considerando apenas um tanque



Fonte: Adaptado de Laubwald (2015, p. 2).

$$\frac{dh}{dt} = \frac{1}{C}\,(q_i - q_o) \tag{2.7}$$

Considerando a Equação (2.7), e sabendo que a *vazão de saída* q_o depende do *nível do tanque*, onde p_t é a *pressão hidrostática no fundo do tanque* e p_o é a *pressão hidrostática ao final da tubulação*, calcula-se as expressões a seguir. A *pressão hidrostática no fundo do tanque* é dada em função da *aceleração da gravidade* g e da *pressão atmosférica* p_{atm}, lembrando que a *pressão da saída da tubulação* é igual à *pressão atmosférica* (anulam-se no cálculo). Substituindo a Equação 2.8 na 2.9, e logo após substituindo o resultado na Equação (2.7), é possível encontrar a Equação 2.10.

$$p_t = pgh + p_{atm} \tag{2.8}$$

$$q_o = \frac{1}{R}\,(p_t - p_o)^{\alpha} \tag{2.9}$$

$$\frac{dh}{dt} = \frac{1}{C}\,(q_i - \frac{1}{R}\,(pgh - p_o)^{\alpha}\,) \tag{2.10}$$

Considerando o sistema agora com um regime de fluxo *turbulento*, isto é, com *número de Reynolds* superior a 10^5, então $\alpha = \frac{1}{2}$. Substituindo em (2.10) tem-se a Equação 2.11 a seguir, que é *não-linear* por causa do termo raiz quadrada.

$$\frac{dh}{dt} = \frac{1}{C}\,(q_i - \frac{1}{R}\,\sqrt{pgh - p_o}\,) \tag{2.11}$$

Suponha-se que h_o é o nível desejado para o processo em *regime permanente*, então, o nível h_o é atingido por uma vazão q_{io}. As derivadas em *R.P.* se anulam e o resultado é a Equação 2.12 demonstrada a seguir, onde é possível determinar q_{io} em função de h_o.

$$\frac{1}{C}\left(q_{i0} - \frac{1}{R}\,\sqrt{pgh_0 - p_o}\right) = 0 \tag{2.12}$$

Expandindo a função *não-linear* em *série de Taylor* em torno de h_o é possível obter:

$$f(h) = \sqrt{pgh_0 - p_o} \approx \sqrt{pgh_0 - p_o} + \frac{1}{2\sqrt{pgh_0 - p_o}}(h - h_0) \tag{2.13}$$

Definindo as variações das variáveis em torno de seus valores em *R.P.*, a equação linearizada será escrita em função dessas novas variáveis, considerando h_0 constante.

$$\Delta h \triangleq h - h_0 \tag{2.14}$$

$$\Delta q_i \triangleq q_i - q_{io} \tag{2.15}$$

$$\frac{d\Delta h}{dt} \triangleq \frac{dh}{dt} \tag{2.16}$$

Substituindo (2.14), (2.15) e (2.16) em (2.13), obtém-se:

$$\frac{d\Delta h}{dt} = \frac{1}{C}\,[(\Delta q_i + q_{io}) - \frac{1}{R}(\sqrt{pgh_0 - p_o} + \frac{1}{2\sqrt{pgh_0 - p_o}}\Delta h)\,]$$

$$\frac{d\Delta h}{dt} = \frac{1}{C}\left[\Delta q_i - \frac{1}{R}\frac{1}{2\sqrt{pgh_0 - p_o}}\Delta h\right] + \frac{1}{C}\left[q_{io} - \frac{1}{R}\sqrt{pgh_0 - p_o}\right] \quad (2.17)$$

O último termo de (2.17) é zero segundo a relação em *R.P.* (2.12), assim obtém-se a Equação 2.18 que é uma relação *linear*, por que os coeficientes de Δh e Δq_i são constantes:

$$\frac{d\Delta h}{dt} = -\frac{1}{2CR\sqrt{pgh_0 - p_o}}\Delta h + \frac{1}{C}\Delta q_i \quad (2.18)$$

Por fim aplicando a *Transformada de Laplace* em (2.18) encontra-se a *função de transferência* entre as variações de pequeno sinal da variável de processo e da manipulada.

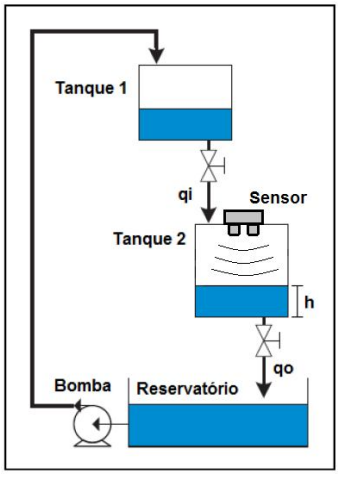
$$\frac{\Delta h(s)}{\Delta q_i(s)} = \frac{2R\sqrt{pgh_0 - p_o}}{\left(2CR\sqrt{pgh_0 - p_o}\right)s + 1} \quad (2.19)$$

A função obtida (2.19) é de primeira ordem (*SPO*), pois se assemelha com (2.3), e percebe-se que o valor da *constante de tempo* $2CR\sqrt{pgh_0 - p_o}$ e do valor do *ganho estático do processo* $2R\sqrt{pgh_0 - p_o}$ dependem do ponto de operação.

2.2.2 Modelo de Tanques Acoplados

A disposição dos tanques neste trabalho está em cascata como mostrado na Figura 2.4, porém, a modelagem encontrada através da pesquisa em referências bibliográficas para descrever o modelo matemático foi a de dois tanques acoplados fixados no mesmo nível, que se diferencia do modelo utilizado apenas por não ter a diferença de altura entre os tanques. Apesar disto, todo o conceito de modelagem é o mesmo e pode ser considerado como teoria sem prejudicar o funcionamento correto do sistema.

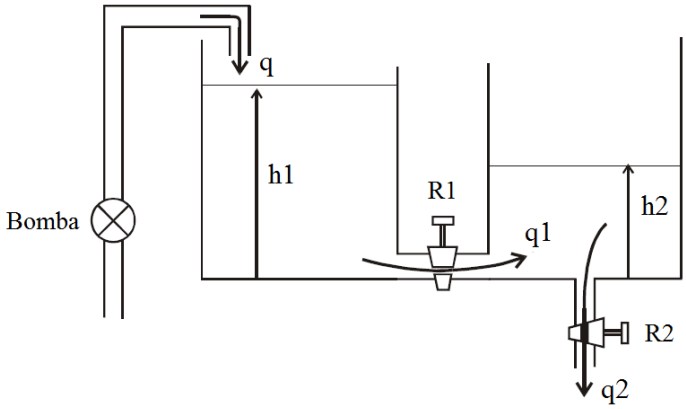
Figura 2.4 – Disposição de dois tanques acoplados em cascata



Fonte: Elaborado pelo autor.

Observa-se que segundo Laubwald (2015), em tanques acoplados o estado do sistema depende do nível no primeiro tanque, e do nível do segundo tanque. Neste sistema os dois tanques interagem e a *função de transferência* resultante não é o produto de duas funções de primeira ordem (OGATA, 1982). Considerando apenas pequenas variações dos sinais em relação aos valores de *R.P.*, observa-se a Figura 2.5 a seguir e assumindo o modelo linearizado (2.19), é possível obter as seguintes equações do sistema:

Figura 2.5 – Modelagem do nível considerando dois tanques acoplados



Fonte: Adaptado de Laubwald (2015, p. 4).

$$\frac{\Delta h_1 - \Delta h_2}{R_1} = \Delta q_1 \tag{2.23}$$

$$C_1 \frac{d\Delta h_1}{dt} = \Delta q - \Delta q_1 \tag{2.24}$$

$$\frac{\Delta h_2}{R_2} = \Delta q_2 \tag{2.25}$$

$$C_2 \frac{d\Delta h_2}{dt} = \Delta q_1 - \Delta q_2 \tag{2.26}$$

Considerando Δq como sinal de entrada e Δh_2 como sinal de saída, e realizando operações de álgebra entre (2.23), (2.24), (2.25) e (2.26), a *função de transferência* é a seguinte:

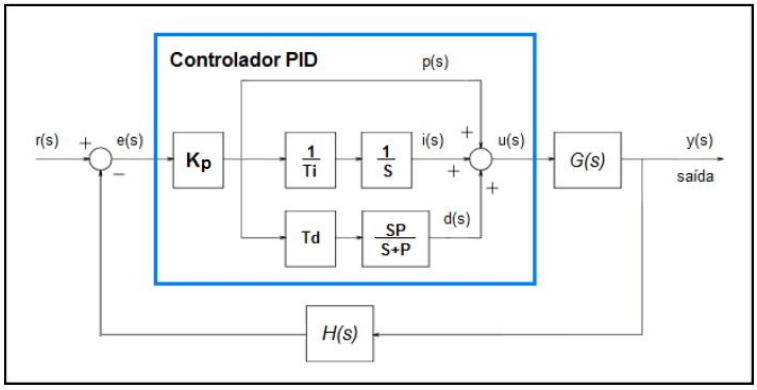
$$\frac{\Delta h_2(s)}{\Delta q(s)} = \frac{R_2}{(R_1 C_1 R_2 C_2)\, s^2 + (R_1 C_1 + R_2 C_2 + R_2 C_1)\, s + 1} \tag{2.27}$$

O resultado encontrado para a *função de transferência* de tanques acoplados neste caso tem formato de um *SSO* e satisfaz a condição de linearidade, pois se assemelha à (2.6) mostrado anteriormente.

2.3 Controlador PID

O *controlador PID* (Proporcional-Integral-Derivativo) une as ações *proporcional*, *integral* e *derivativa* para gerar um só *sinal de controle*, onde cada ação possui uma forte característica que auxilia no controle da saída do sistema. A *ação integral* torna o controlador capaz de eliminar erros em *R.P.* para entrada do tipo salto ou degrau, a *ação derivativa* antecipa o comportamento do processo, e a *ação proporcional* faz com que o sistema reaja ao *erro* presente conferindo uma reação imediata diante de perturbações ou variações da referência (BAZANELLA, 2005). O sinal de controle gerado pelo *controlador PID* pode ser visto como a soma de três sinais obtidos a partir do sinal de *erro* (*p(s)*, *i(s)*, *d(s)*) conforme mostra a Figura 2.6. O *ganho proporcional* K_p corresponde à *ação proporcional*, o *tempo integral* T_i à *ação integral*, e o *tempo derivativo* T_d à *ação derivativa*.

Figura 2.6 – Diagrama de blocos de um *controlador PID*



Fonte: Elaborado pelo autor.

2.3.1 Ação Proporcional (K_p)

Não é possível aplicar sinais de amplitudes ilimitadas para a variável de controle por motivos de restrição de ordem física ou de segurança. É estipulado um *limite máximo* e *mínimo* para a *variável de controle MV*, e a largura da faixa de operação entre estes limites é chamada de *banda proporcional*. Se o sinal de controle opera fora dos limites determinados diz-se que está em comportamento de *saturação de controle*, neste caso opera com comportamento *não-linear*. Quanto maior for a *banda proporcional*, mais dificilmente ocorrerá *saturação de controle*, pois maior é a região de comportamento linear para o *controlador PID*.

Quanto maior o valor do *ganho proporcional* K_p, menor será a *banda proporcional* e maior o esforço de controle (energia de controle). Isto faz com que o sistema responda com mais agilidade, mas um valor de K_p muito alto pode fazer com que o valor de *saída* ultrapasse o valor de *referência*, o que é chamado de *sobrepasso*. Esta situação pode ser indesejável, pois pode gerar elevados valores do sinal de saída na resposta transitória, ou seja, antes de entrar em *R.P.*

- ***Comportamento em R.P.*:** em sistemas sem pólos (s=0) quanto maior o valor de K_p, menor será o valor de *erro*, porém o *erro* nunca será completamente anulado; o comportamento da *ação proporcional* somente é nulo junto com a *ação integral*, assim o *erro* tende à zero.

2.3.2 Ação Integral (T_i)

A principal característica desta ação é fazer com que o processo siga com *erro* nulo em *R.P.* um sinal de *referência* constante (do tipo salto), porém quando esta é aplicada isoladamente, a tendência é piorar a estabilidade relativa do sistema. Geralmente para contornar este problema a *ação integral* é utilizada em conjunto com a *ação proporcional*. Quanto maior o *tempo integral* T_i, menor será o *sobrepasso* e mais devagar o sistema alcança o valor de *referência* em *regime permanente*.

- ***Comportamento em R.P.*:** a *ação integral* será constante quando o sistema em *malha-fechada* for estável; quando o *erro* em *R.P.* é nulo, o integrador pára de integrar e sua saída assume o valor armazenado até então; o valor de *saída* depende do *ganho estático* do processo e do valor de *referência* constante.

2.3.3 Ação Derivativa (T_d)

Normalmente o processo apresenta uma inércia com relação às modificações na *variável de entrada*, ou seja, o tempo que uma alteração na *variável de controle MV* provoca uma mudança na *saída* do processo *PV*; este fato causa transitórios com grande amplitude e período de oscilação. A *ação derivativa* antecipa a *ação de controle* para que o processo reaja mais rápido do que de costume e que transitórios com característica excessivamente oscilatória sejam evitados. Esta ação preditiva aumenta a estabilidade relativa do sistema e

torna a resposta transitória mais rápida e menos oscilatória conforme maior for o *tempo derivativo* T_d.

- ***Comportamento em R.P.*:** Para sistemas estáveis em malha fechada operando com uma *referência* constante, a *ação derivativa* será nula em *R.P.*, pois o sinal de *erro* será constante em *R.P.*

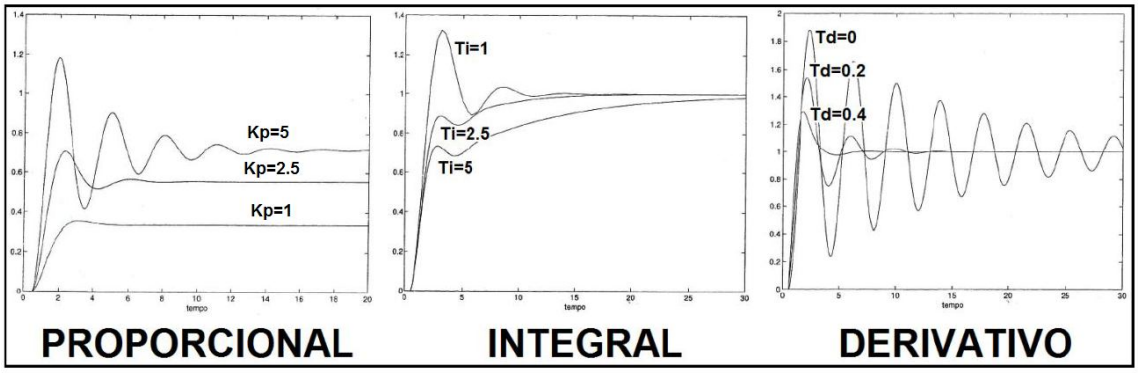
2.3.4 Ação Proporcional-Integral-Derivativa

O *controlador PID* combina as três ações descritas, onde cada uma compensa as características não desejáveis da outra. Considerando Bazanella (2005), a *ação integral* está relacionada à precisão do sistema, responsável pelo erro nulo em *R.P.* O efeito desestabilizador do controlador integral é suprido pela *ação derivativa*, pois o efeito antecipatório torna a resposta mais rápida e aumenta a estabilidade relativa do sistema. Cada sinal individualmente possui um parâmetro de sintonia correspondente a sua *ação de controle*. Assim, o *sinal de controle* gerado pelo *controlador PID* pode ser genericamente expresso como a Equação 2.27:

$$u(t) = K_p\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}\right) \tag{2.27}$$

Para entender melhor as características de cada ação do *controlador PID*, é interessante analisar a Figura 2.7, que apresenta o comportamento do sinal de *saída* mediante a variação dos parâmetros já descritos de cada ação de controle.

Figura 2.7 – Comportamento de cada ação *PID* conforme parametrização



Fonte: Adaptado de Bazanella (2005, p. 54,57,61).

3 DESCRIÇÃO DO PROTÓTIPO

Este capítulo é o maior do relatório e tem objetivo de detalhar todas as principais partes do protótipo bem como apresentar as definições que foram tomadas durante o desenvolvimento, ele é dividido em três partes: *protótipo*, *software* e *hardware*. A *Seção 3.1* trata de *Definições Preliminares* e divide seu assunto em duas seções, sendo elas a *Seção 3.1.1* sobre a *Descrição da Bancada Experimental*, e a *Seção 3.1.2* sobre as *Características da Bomba e do Sensor Ultrassônico*. Na *Seção 3.2* são apresentadas as principais características do *Projeto em Software*, esta seção também é dividida em duas seções. Na *Seção 3.2.1* são apresentados maiores detalhes sobre a *Navegação na IHM* (Interface Homem-Máquina) *e Fluxograma do Sistema*, já na *Seção 3.2.2*, é demonstrado o detalhamento das principais funções e trechos de *Código do Sistema*. Finalmente, a *Seção 3.3* trata do *Projeto em Hardware* que é dividido na *Seção 3.3.1* sobre o *Projeto e Esquema Eletrônico*, e na *Seção 3.3.2*, sobre a *Interface IHM*.

3.1 Definições preliminares

As definições durante o desenvolvimento do projeto esclarecem qual o objetivo do trabalho. Entre as principais definições está o uso de uma *planta de tanques acoplados*, o *controle de nível* apenas do segundo tanque da planta, a utilização de uma *bomba d'água* de parabrisa como *atuador*, o uso de um *sensor ultrassônico* como *sensor* para realimentação do sinal de *saída*, entre outras que serão descritas a seguir.

3.1.1 Descrição da Bancada Experimental

O protótipo dos tanques acoplados consiste basicamente em quatro partes, sendo elas três recipientes feitos de acrílico para acúmulo de água mais um suporte feito em chapa de madeira mdf para sustentar o peso de todos os recipientes, sejam cheios ou vazios. Cada parte recebe um nome para facilitar a identificação do protótipo neste relatório, conforme demonstrado na Figura 3.1. É chamado de *estrutura* o suporte de madeira mdf, chamado de *reservatório* o recipiente de acrílico que alimenta a *bomba* e armazena água quando o sistema está inativo, chama-se de *tanque 1* o primeiro acrílico que recebe água da *bomba*, e finalmente, de *tanque 2* o segundo acrílico que recebe a água por gravidade e mede o nível através do *sensor*.

Figura 3.1 – Protótipo de Tanques Acoplados

Fonte: Elaborado pelo autor.

A *estrutura* tem a finalidade de dar sustentação, e definir uma altura e posição fixa para colocar o *tanque 1*, *tanque 2* e o *reservatório*. A altura dos degraus para acomodar os tanques foi estabelecida de forma que a melhor dinâmica entre os dois tanques fosse obtida. A melhor dinâmica foi obtida quando a altura do *tanque 1* era maior, isto porque quando a água chega inicialmente a este tanque é formada uma coluna de acúmulo até que exista pressão suficiente para empurrar a água para o segundo tanque. Se os tanques permanecessem no mesmo nível a altura da coluna no primeiro tanque para obter pressão seria muito maior. Quando o primeiro tanque está em um degrau de nível mais alto a ação da gravidade colabora com o envio d'água e não é preciso um acúmulo muito grande de líquido para conseguir a pressão necessária, o que resulta em um nível de acomodação de água mais baixo e consequentemente maior liberdade para excursão do nível de água no primeiro tanque. Quanto maior é a excursão do nível d'água no primeiro tanque, maior será a excursão do nível no *tanque 2*. Após muitos testes, foi conveniente projetar a estrutura de tal forma que o *tanque 1* ficasse acima do *tanque 2*, mais especificamente falando cerca de 35cm em relação a base

do segundo tanque, e o *tanque 2* em um nível de 15cm acima da base do *reservatório*. Assim configura-se um sistema cascata em função do caminho que o líquido percorre sempre descendo de um nível superior a um inferior.

O *reservatório* tem a função de acumular o líquido utilizado no controle quando o sistema esta em repouso ou desligado. A *bomba* d'água está fixa junto ao reservatório para ser alimentada pela água do recipiente e conseguir abastecer os tanques. A capacidade de acumulo de líquido do *reservatório* é superior quando comparada a dos tanques, isto por que foi definido que este recipiente deve ter a capacidade dos dois tanques mais 15% para justamente eliminar a possibilidade de a *bomba* trabalhar a seco e estragar, a capacidade do *reservatório* é de 4,5 litros. O projeto do tamanho de cada recipiente mais a estrutura pode ser conferido com detalhes no apêndice ao final deste relatório.

Quando a *bomba* envia água para executar o controle de nível em *malha-fechada*, o líquido passa primeiro pelo *tanque 1*, que tem a função de aumentar a complexidade do controle do sistema. A água chega ao *reservatório* e, no primeiro momento, fica acumulada por que não há pressão suficiente para abastecer o *tanque 2* por gravidade. Dependendo da intensidade que a *bomba* abastece o primeiro tanque, depois de algum tempo, a coluna de água formada tem pressão suficiente para empurrar o líquido que desce por gravidade para o segundo tanque. Para controlar a vazão de entrada e saída dos recipientes é utilizado torneiras, e para estipular um caminho de passagem do líquido foi utilizado tubo de silicone de aproximadamente 8mm de diâmetro. Foi projetada uma torneira para evitar transbordamento do *tanque 1* (popularmente chamado de *ladrão*), mas durante os testes percebeu-se que o caminho para fuga de água impossibilita a formação de uma grande coluna de água no *tanque 1*. Isto implica na limitação de variação de nível no *tanque 2*, por que quanto maior for a coluna do *tanque 1* maior será a pressão para envio de água, e conseqüentemente, maior será o nível do *tanque 2* em função desta pressão. Portanto, foi decidido deixar o *ladrão* totalmente fechado. O *tanque 1* funciona praticamente como um buffer de água com capacidade para 2,4 litros, introduzindo assim uma dinâmica adicional no controle de nível no *tanque 2*, isto de fato aumenta a complexidade do sistema.

O *tanque 2* é o principal elemento deste sistema, pois nele acontece o controle de nível de água; ele tem capacidade para 1,6 litros. Depois de algum tempo após o sistema acionar a *bomba*, o primeiro tanque começa a escoar líquido para o *tanque 2*, neste período transitório os dois tanques começam a encher e buscar um nível de equilíbrio. Em determinado momento o fluxo estabiliza e o nível dos tanques acomoda-se em um ponto mais alto ou mais baixo conforme a intensidade da *bomba*. Não é desejado a presença de bolhas de ar no *tanque 2* no

momento que o fluxo e o nível estabilizam, por que isto causa turbulência no espelho d'água do nível e dificulta a execução do controle. A torneira de saída do *tanque 2* é a única que não está totalmente aberta, justamente para restringir o fluxo em busca de um valor de nível fixo que determina o ponto zero (inicial) do controle. O nível neste tanque aumenta através do controle cerca de 3cm a partir do ponto zero, e de fato esta é a faixa que deseja-se controlar por intermédio de valores do *set point* inseridos pelo usuário conforme será detalhado posteriormente. No *tanque 2* ainda existe um *sensor ultrassônico* para obter medidas do nível de água, o qual é fixado na tampa do recipiente; também é importante relatar que ambas as tampas do *tanque 1* e *tanque 2* possuem um furo para igualar a pressão atmosférica de dentro dos recipientes com a de fora, isto para não interferir no controle. Por fim, o segundo tanque escoa água para o reservatório e com isso fecha o ciclo do caminho do líquido.

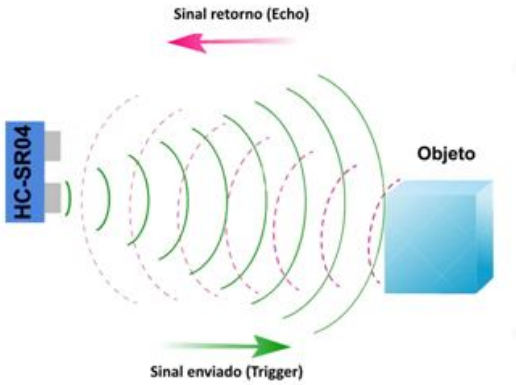
3.1.2 Características da Bomba e do Sensor Ultrassônico

A *bomba* d'água de parabrisa é um componente utilizado em automóveis para enviar água com detergente para o parabrisa do carro com a finalidade de limpar e melhorar a visibilidade do motorista. Este componente é muito simples, e foi escolhido para este projeto por causa da possibilidade de variar a intensidade de acionamento, bombeando mais ou menos água, conforme a tensão contínua em que é alimentado. Funciona com até 12Vdc e consome cerca de 3A de corrente, ou seja, um valor considerável de corrente. Para alimentar o circuito eletrônico e a *bomba*, foi comprado uma fonte chaveada comercial de 12Vdc com capacidade de 5A de saída.

A finalidade da *bomba* neste projeto é poder controlar a intensidade do envio d'água do *reservatório* até o *tanque 1*, para obter o nível desejado no *tanque 2*. Ela é o elemento *atuador* que converte o sinal elétrico em intensidade de fluxo. Auxilia na busca do *set point* configurado pelo usuário, variando para mais ou menos a intensidade de seu funcionamento conforme for o nível em relação onde se deseja estabilidade. Segundo ABC Engenharia (2010), o funcionamento da *bomba* ocorre com a água entrando no centro do rotor de plástico, que possui canais em alta rotação impulsionados pelo motor elétrico e movimentam o líquido criando força centrífuga, que se transforma em energia de pressão. A vazão máxima atingida pode ser de 0,08561 litros por segundo, ou 85,61 mililitros por segundo, e para as exigências deste projeto a vazão é suficiente, analisando o comportamento do sistema e resultados dos testes práticos realizados.

O *sensor de distância ultrassônico HC-SR04* serve como sensor de nível neste projeto e tem objetivo de medir constantemente o comportamento da variação do nível de água no *tanque 2*. Seu funcionamento se baseia no envio de sinais *ultrassônicos* pelo *sensor* e espera do retorno, conforme demonstrado na Figura 3.2; calcula-se a distância entre o *sensor* e o que se deseja medir, que neste projeto é o nível d'água, com base no tempo entre o envio e retorno do sinal (FILIPEFLOP, 2011).

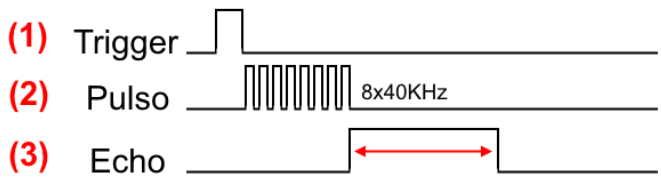
Figura 3.2 – Exemplo de funcionamento do Sensor Ultrassônico



Fonte: FilipeFlop (2011).

O *sensor ultrassônico* é um equipamento eletrônico comercial utilizado em diversos projetos eletrônicos, ele tem capacidade de medir distâncias de 2cm a 4m com precisão de 3mm. O módulo do *sensor* possui um circuito pronto com emissor e receptor acoplados e 4 pinos para medição, sendo 2 pinos para alimentação +5Vdc e *GROUND*, e mais 2 pinos para envio do sinal *ultrassônico TRIGGER* e recebimento do sinal *ECHO*. Segundo FilipeFlop (2011), para obter a distância o *sensor* em primeiro momento coloca o pino *TRIGGER* em nível alto para enviar um pulso de 10µs indicando o início da transmissão de dados, depois são enviados 8 pulsos de 40KHz, e finalmente o pino *ECHO* é colocado em nível alto aguardando o retorno do sinal enviado; estes passos estão demonstrados na Figura 3.3.

Figura 3.3 – Esquema de envio dos sinais do *sensor*



Fonte: FilipeFlop (2011).

Para determinar a distância entre o *sensor* e o nível da água, considerando a velocidade do som 340 metros por segundo, utiliza-se a Equação 3.1 a seguir.

$$distância = \frac{(tempo\ pino\ ECHO\ nível\ alto \times velocidade\ do\ som)}{2} \quad (3.1)$$

Quando o sinal é enviado percorre o caminho de ida até o nível do espelho de água, e volta até o pino *ECHO* receber retorno, isto justifica na equação a necessidade da divisão por dois para obter como resultado apenas a distância até o nível de água.

3.2 Projeto em Software

Para facilitar o entendimento foi detalhada cada tela de navegação da *IHM*, assim como indicado quais botões estão habilitados para cada interação. O Fluxograma ilustra e complementa a informação sobre a possível navegação. Cada função principal do código fonte que roda na *IHM* também será detalhada para compreensão da lógica do código.

3.2.1 Navegação na IHM e Fluxograma do Sistema

A navegação no *software* que foi desenvolvido é muito simples, fácil, e intuitiva. Foi criado uma lógica de navegação na *IHM* que contempla apenas o necessário para atingir o objetivo deste projeto. Por definição, a parte de *software* é bastante sucinta tanto nas opções de navegação quanto no próprio código do software, pois como este projeto tem cunho didático, não haverá maiores preocupações em desenvolver um produto final. O foco principal é o funcionamento e apresentação dos resultados.

A caixa da *IHM* tem na parte traseira um botão de *liga e desliga* que é responsável por alimentar todo o protótipo. Houve uma preocupação maior em centralizar toda alimentação em um só lugar no protótipo a fim de proporcionar maior confiabilidade e segurança na execução. Existem quatro botões responsáveis pela navegação na interface do usuário, todas as decisões são tomadas através deles, cada um possui uma finalidade específica, conforme demonstrado na Figura 3.4. O primeiro botão é chamado de *VOLTA* e tem a finalidade de *retroceder* para um estado anterior, *invalidar* a configuração de algum parâmetro, ou *retornar* da tela de *Controle PID* para tela de *Menu Principal*; aqui chamaremos de parâmetro qualquer valor referente as configurações que alteram o funcionamento do controle em si, tais como o *Set Point*, *Proporcional* (K_p), *Integral* (T_i) e *Derivativo* (T_d). Os próximos dois botões são

chamados de *MENOS* e *MAIS*, isto por que a principal finalidade deles é em primeiro momento possibilitar a navegação entre as opções do *Menu Principal*, ou quando em uma opção de configuração eles auxiliam na *mudança de valores* do *Set Point* ou valores *PID*. Por último existe o botão *OK* que é talvez o principal botão de navegação pelo fato de *selecionar e confirmar escolhas* do usuário. Na tela do *Menu Principal* este botão *seleciona* as opções disponíveis; quando dentro de uma opção de configuração o botão *confirma* o valor selecionado, e finalmente na tela de informações, este botão realiza o *avanço* de tela.

Figura 3.4 – Interface *IHM* e botões de navegação

Fonte: Elaborado pelo autor.

Para apresentar as informações ao usuário é utilizado um *display LCD* com luz de fundo *azul* de 2 linhas por 16 colunas, suficiente para possibilitar a navegação e cumprir os propósitos deste projeto. Por vezes, quando não houve espaço para colocar uma palavra no *display,* foi utilizada alguma abreviatura que remetesse ao significado da mesma. Na tela do *Menu Principal* procurou-se ocupar todo espaço disponível colocando as quatro opções nos cantos do *display*. Quando nas telas de configuração de *Set Point* e *PID*, optou-se por colocar o nome do parâmetro na primeira linha do *display*, e o valor que pode ser alterado, na segunda linha. Quando utilizados os botões de *MENOS* e *MAIS* o valor do parâmetro altera-se na tela, e através do botão *OK* confirma-se a escolha feita e ocorre o retorno para a tela do *Menu Principal*. Quando na tela de configuração de algum parâmetro é pressionado o botão *VOLTA*, o valor alterado ou não pelo usuário não será salvo. O programa volta para a tela de *Menu Principal* se o usuário estiver configurando o *Set Point*, ou volta para o parâmetro anterior se estiver configurando K_p, T_i e T_d; se estiver configurando o *Proporcional* (K_p), como a tela anterior é o *Menu Principal*, a ação será retornar para o mesmo. Na tela de *Controle PID*, que pode ser acessada pela opção *(X)Start*, existem quatro informações que servem para o usuário verificar a execução em tempo real, sendo elas: *Set Point (SP)* configurado pelo usuário, *Nível*

(N) da água instantâneo no *tanque 2*, *Erro (e)* entre o *Set Point* e o *Nível*, e o valor instantâneo do *pwm* aplicado na bomba. O valor do *Set Point* é apresentado na escala de centímetros, assim como o do *Nível* e do *Erro*. O valor de *pwm* (pulse width modulation) realiza o controle da variação da tensão de saída do pino do microncontrolador através da variação da largura de pulso de uma onda quadrada. A largura de pulso é modulada conforme o número de bits definidos para excursão do sinal (2^{10} = 1024). Por sua vez, o *pwm* varia entre os limites definidos no projeto de *software*, ou seja, entre 0 e 1023 respectivamente, que equivale entre 0 e aproximadamente +5Vdc na saída do pino (5Vdc / 1024 = 0,0048Vdc/unidade).

Quando é ligada a *IHM* pelo botão *on/off*, a apresentação começa pela *animação de abertura* mostrando o nome do projeto em letras deslizantes, algo muito simples que introduz a idéia do que se trata o projeto e o seu objetivo. Esta animação termina e fica na tela o nome do projeto até que o usuário pressione o botão *OK* para prosseguir para a tela do *Menu Principal*. A Figura 3.5 detalha esta etapa assim como quais botões estão habilitados (indicados por letras na cor azul) neste momento.

Figura 3.5 – Tela de *animação de abertura*

Fonte: Elaborado pelo autor.

A *animação de abertura* é o único momento em que não é executado, seja em *background* ou explícito, o *controle PID* com parâmetros configurados por *default* ou pelo usuário. O *controle PID* é permanentemente executado durante o tempo em que a *IHM* está ligada depois de passado a tela de *animação de abertura*. Após o usuário pressionar o botão *OK*, o sistema de controle começa a funcionar em *background* configurado por *default* com busca por *Set Point* igual a zero. O objetivo desta configuração é garantir o nível de água no segundo tanque igual ao zero definido enquanto o usuário configura parâmetros no *Menu Principal*. A tela de *Menu Principal* apresenta quatro opções resumidas, mas auto-descritivas, sendo elas: *( )SP, ( )Start, ( )PID, ( )Info*. Cada opção quando selecionada é demarcada com *'X'* para indicar para o usuário qual opção ele está selecionando, conforme a Figura 3.6.

Figura 3.6 – Tela de *Menu Principal*

BOTÕES HABILITADOS:
[ ] VOLTA [ ] MENOS [ ] MAIS [ ] OK

Fonte: Elaborado pelo autor.

Se o cursor *'X'* está sobre a opção de *(X)SP* e o usuário pressionar o botão *OK*, ele terá acesso a tela de configuração do *Set Point* do programa. A opção de *SP* disponibiliza a opção de configuração de *Set Point* ao usuário variando o valor de configuração entre 0 e 3cm, que é aproximadamente a variação obtida pelo método de *resposta ao salto* em *malha-aberta* que será apresentado na *análise de resultados*. A variação do valor no *SP* acontece na ordem de ±0.1, para possibilitar maior liberdade de excursão entre os valores disponíveis de 0 à 3cm, sempre para mais ou menos conforme o usuário pressionar os botões. Existe a possibilidade de entrar na tela e ao pressionar o botão *VOLTA*, retornar ao *Menu Principal* sem salvar as alterações de *Set Point*, do contrário se o usuário pressionar o botão *OK* na tela de alteração do valor de *Set Point*, o programa retorna para o *Menu Principal* agora com a informação salva.

Figura 3.7 – Tela de configuração do *Set Point*

BOTÕES HABILITADOS:
[ ] VOLTA [ ] MENOS [ ] MAIS [ ] OK

Fonte: Elaborado pelo autor.

Para cada vez em que é salva uma informação em alguma tela de alteração de parâmetros, seja ela de *Set Point* ou no final da configuração dos valores de *PID*, se o parâmetro que foi selecionado é diferente do valor já salvo anteriormente, o programa salva em memória volátil o novo valor e informa isto ao usuário através da tela de *animação de upload*. Observa-se na Figura 3.8 a demonstração deste momento.

Figura 3.8 – Tela de *animação de upload*

Fonte: Elaborado pelo autor.

No *Menu Principal*, quando o cursor estiver sobre a opção de configuração dos valores *(X)PID* e o usuário pressionar o botão *OK*, ele terá acesso às telas de configuração dos parâmetros *Proporcional* (K_p), *Integral* (T_i) e *Derivativo* (T_d). Cada parâmetro vem configurado com valor *default*, que segundo cálculos demonstrados no capítulo 4, é o melhor valor de configuração segundo o funcionamento que se espera nesta planta didática de tanques acoplados. Se o usuário quiser alterar os valores de *PID* é possível, mesmo que não haja necessidade por que os parâmetros estão cuidadosamente estipulados para uma resposta de controle relativamente rápida (mas ainda um pouco lenta) e com mínimo *sobressalto*. As telas de configuração dos valores *PID* estão apresentadas em seqüência conforme mostra a Figura 3.9, onde primeiro configura-se o parâmetro K_p, depois T_i e por fim o T_d, durante a configuração é possível navegar nas telas através dos botões *VOLTA* e *OK*. Quando configura o valor *Proporcional* (que é o primeiro parâmetro que aparece partindo do *Menu Principal*), se o usuário apertar o botão *VOLTA* não será salvo o valor inserido e o programa volta para o *Menu Principal*, ou seja, retorna ao estado anterior das telas. Quando pressiona a tecla *OK*, ocorre o avanço para configuração do próximo parâmetro que é o valor de *Integral*, o mesmo conceito de *VOLTA* e *OK* valem para esta tela, agora quando pressiona *VOLTA* retorna para *Proporcional*, e quando *OK*, avança para o último parâmetro que é o *Derivativo*. Na tela do valor de *Derivativo* quando o usuário pressiona *VOLTA*, ele retorna para a configuração do parâmetro *Integral*, e quando pressiona *OK* finalmente o programa salva todos os valores configurados se forem diferentes dos que já estavam salvos ou configurados anteriormente. Lembrando que para mudança dos valores de cada parâmetro é possível utilizar os botões *MENOS* e *MAIS*; estes botões têm configuração sensível ao pressionar, como por exemplo, se o valor desejado for muito maior que o inicial, aumenta-se a velocidade com que ocorre a soma ou subtração para chegar logo no valor desejado a fim de não fatigar o usuário.

Figura 3.9 – Tela de configuração de K_p, T_i e T_d

Fonte: Elaborado pelo autor.

O valor do parâmetro *Proporcional* varia na ordem de ±0.001 cm/V dentro dos limites definidos entre 0.000 e 1.000, para o valor do parâmetro *Integral* a variação ocorre na ordem de ±1 e dentro dos limites definidos entre 0 e 200, e por último para o *Derivativo* o valor varia na ordem de ±1 também entre os limites definidos de 0 e 200. O *controlador* utilizado para obter o nível desejado neste projeto é um *PI* como será detalhado na *análise de resultados*, portanto, é necessário que o valor do *Derivativo* permaneça sempre em zero para que o controle seja de fato realizado conforme se espera.

Mais uma alternativa disponível no *Menu Principal* que já foi citada é a opção *(X)Info* que tem a finalidade de trazer maiores informações sobre o *aluno desenvolvedor do projeto*, *semestre em que o trabalho acontece* e do *nome do projeto* (novamente, igual ao da *animação de abertura*), conforme ilustrado na Figura 3.10. Esta possui conteúdo informativo para o usuário que interage com a *IHM*, e não altera qualquer valor de parâmetro utilizado. É importante informar que o *controle PID* também executa em *background* nesta tela de informações, o *PID* funciona constantemente e não pára. Quando a navegação está nas telas de *Info*, a única opção disponível para o usuários pressionar é o botão de *OK* que avança as telas até retornar ao *Menu Principal* novamente.

Figura 3.10 – Tela de *Info* (informações)

Fonte: Elaborado pelo autor.

O usuário, quando seleciona a opção *(X)Start* no *Menu Principal*, está concordando com todas configurações inseridas nas opções de parametrização, e mesmo se o usuário não inserir nenhuma informação sobre o *Set Point* que deseja, ou alterar os valores *PID*, o

programa inicia o controle com valores *default*. Como *default* tem-se zero para o valor *Set Point*, 0.13 para o *Proporcional*, 100 para o *Integral* e zero para o *Derivativo*, pois o controlador estabelecido inicialmente é um PI e o valor para D deve ser nulo. Se os parâmetros foram alterados, quando o usuário seleciona *Start* e pressiona o botão *OK* no *Menu*, automaticamente o programa coloca todos valores alterados nas variáveis que o *controle PID* usará, e passa para a tela de *Controle PID*. Na tela de *Controle* são apresentados os principais valores de forma instantânea que o *software* utiliza para cálculo conforme mostrado na Figura 3.11, mais especificamente falando o valor de *Set Point* que está configurado, o valor atual do *Nível* de água no *Tanque 2*, e o valor do *Erro* é que a subtração do *Set Point* menos o *Nível*; estes últimos três valores estão com as unidades de medidas omitidas no *display*, mas todos estão em centímetros (cm). Por último, ainda é apresentado nesta tela o valor instantâneo correspondente ao *sinal de controle pwm* aplicado na *bomba* para acionamento, ou seja, o efetivo valor de controle que está sendo aplicado na busca pelo *Set Point*. Pelo valor do *erro* e do *pwm* é possível entender o comportamento do *controle PID* projetado na integra, se é lento ou rápido, se tem *sobrepasso* ou não, observar quanto tempo leva para o nível atingir o *Set Point* configurado, etc.

Figura 3.11 – Tela de visualização do controlador em *malha-fechada*

Fonte: Elaborado pelo autor.

Algo interessante que deve ser informado é a possibilidade de alterar os parâmetros do *controle PID* depois de iniciar o controle na opção *Start* do *Menu Principal*. Por exemplo, o usuário liga a *IHM* e configura os valores de *Set Point* e *PID* que deseja (não se esqueça que enquanto isso o *controle PID* já está atuando na busca pelo nível zero definido), depois pressiona *Start* e executa o *controle PID* com seus valores configurados. Por algum motivo o usuário deseja alterar os parâmetros *PID*, pois achou muito lenta a resposta por exemplo. Então ele deve retornar através do botão *VOLTA* para o *Menu Principal* e modificar os parâmetros. Após realizar a alteração no *Menu Principal* seleciona *Start* novamente, mas o *controle PID* já está executando na busca por um *Set Point* configurado anteriormente, então quais parâmetros de controle serão válidos? Para resolver este problema o programa cada vez

que executa a opção *Start* no *Menu* realiza uma verificação se o usuário reconfigurou os valores *PID*, se esta condição for verdadeira o programa apresenta uma tela de espera ao usuário e busca o zero definido como *Set Point* com os valores de *PID* configurados no estado anterior. A tela de espera apresenta a seguinte mensagem: *'PID updated! wait ZERO level'* conforme apresentado na Figura 3.12, isto significa que o valor de *PID* foi alterado e o usuário deve aguardar o programa colocar o nível em zero.

Figura 3.12 – Tela de *PID updated* e espera pelo nível zero

Fonte: Elaborado pelo autor.

Quando o programa verifica que o nível se encontra no zero, avisa por meio de uma mensagem e libera o controle para o usuário, que ao pressionar o botão *OK* coloca em funcionamento a busca pelo *Set Point* configurado agora com os novos valores de *PID*. A mensagem que aparece para o usuário quando é liberado controle é a seguinte: *'Ready! press OK'* demonstrada na Figura 3.13, que significa o estado pronto do programa e a espera pelo usuário pressionar *OK* para continuar o controle com os novos valores de *PID*.

Figura 3.13 – Tela de liberação do controle para usuário

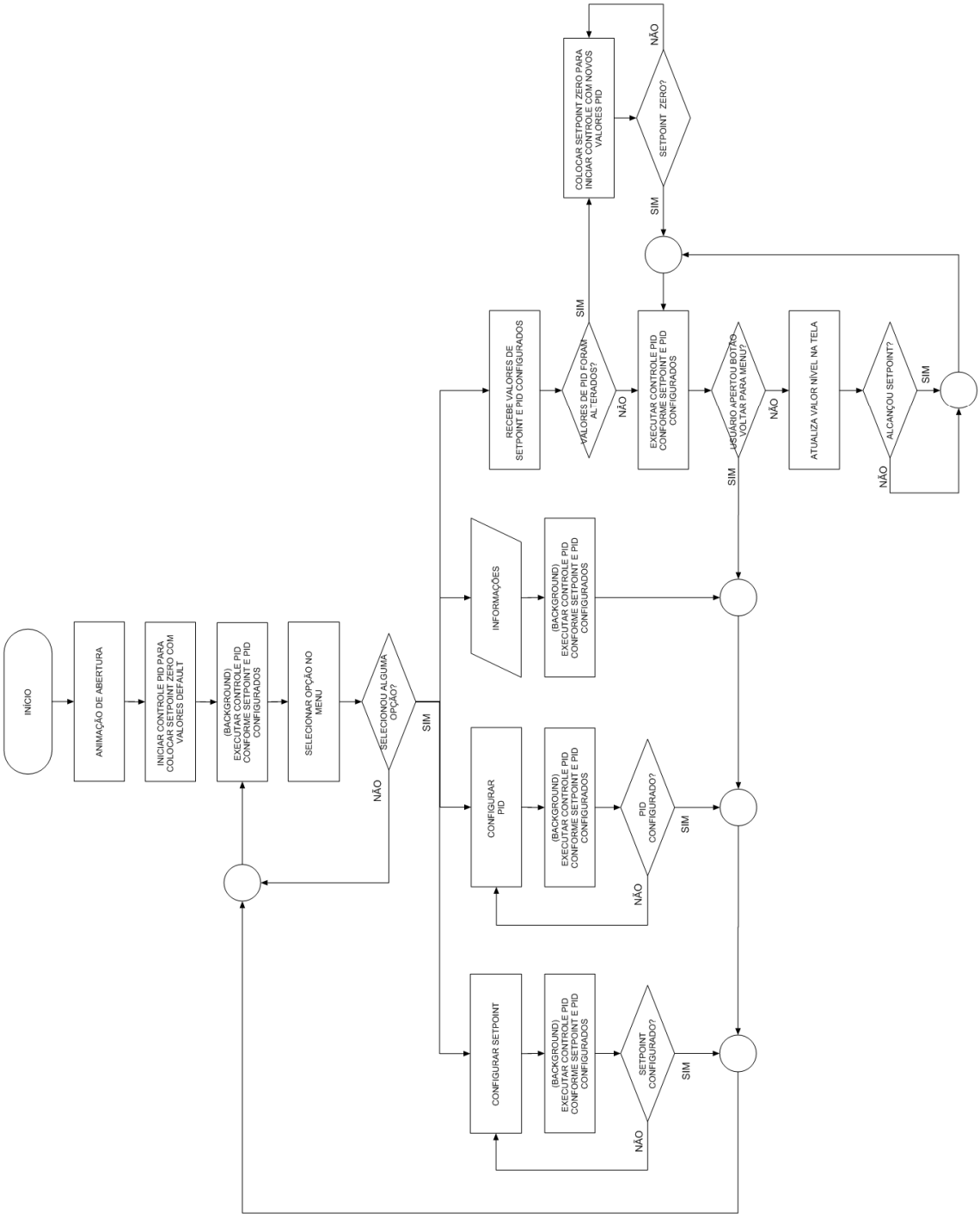
Fonte: Elaborado pelo autor.

Para o *Set Point* esta regra de verificação não se aplica, pois o usuário pode voltar da tela de *Controle PID* para o *Menu*, alterar o valor do *SP*, e quando selecionar a opção *Start* no *Menu*, automaticamente é alterado o valor e o controle começa a nova busca, ou seja, não é necessário colocar o nível em zero para modificar a busca de algum *Set Point* configurado posteriormente.

Existe uma lógica de funcionamento por trás de toda a extensa explicação sobre a navegação na *IHM*, sobre o fluxo e os caminhos para onde vai cada escolha feita. A descrição das telas proporciona um entendimento amplo do funcionamento, mas para entender com

maior clareza como funciona o fluxo de navegação é interessante analisar o *fluxograma* da Figura 3.14. Este *fluxograma* tem objetivo de validar o entendimento sobre como de fato funciona todo programa, todas as partes já descritas estão apresentadas na imagem. É importante ressaltar novamente que a execução do *controle PID* é constante e sempre está operando seja explicita na tela de *Controle PID* ou em *background* em outros momentos. A tela de *Menu Principal* é o caminho para todas decisões que o usuário desejar tomar, seja ela em relação as alterações de parâmetros quanto de iniciar o controle, é representado no fluxograma todos momentos que retornam para a tela de *Menu*. O laço de controle pela busca do *Set Point* é um condicional que se alcançado o nível ou não, sempre permanece no mesmo laço. Perceba também que o programa só tem início e não tem fim, isto por que foi desenvolvido para executar constantemente, a idéia é ter um controle contínuo que deve ser executado em todo momento. Não há preocupação em criar uma condição no programa para terminar o *controle* por que este projeto é apenas didático, como já foi citado, e seu propósito maior é a demonstração do *controle PID* atuando sobre a *planta*. Para terminar o *controle* é possível desligar a *IHM* através do botão *on/off*. Por fim, o fluxograma da Figura 3.14 representa a ideia geral sobre a navegação entre telas.

Figura 3.14 – Fluxograma sobre a navegação na *IHM*

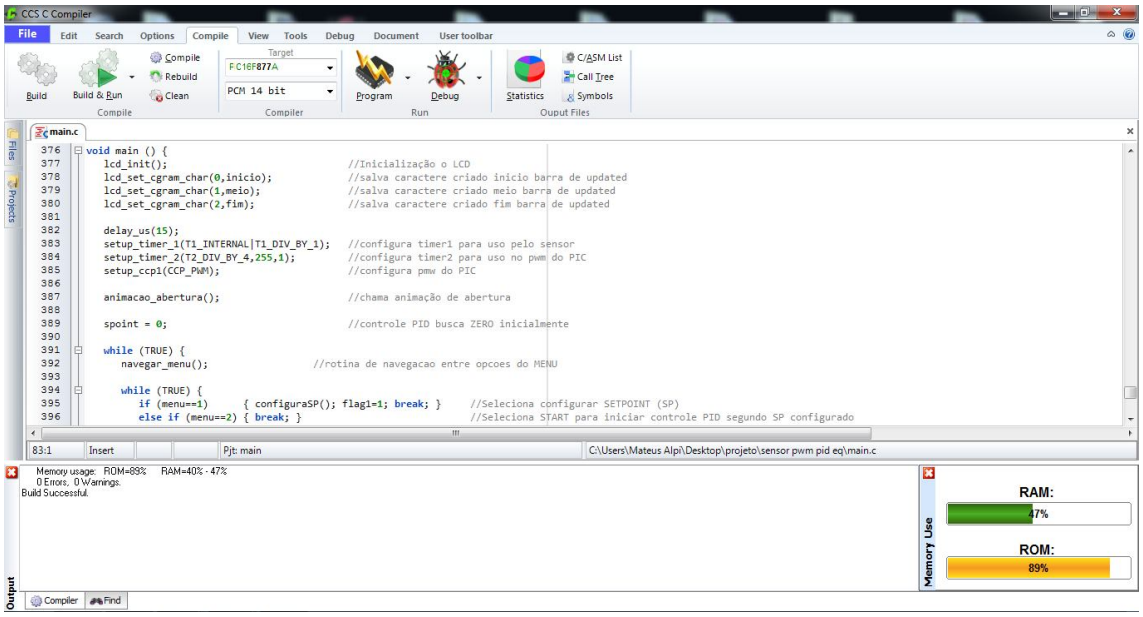


Fonte: Elaborado pelo autor.

3.2.2 Código do Sistema

O código fonte do projeto foi estruturado em uma rotina *main* principal que chama outras pequenas rotinas para executar, chamadas de *funções*. Dividir a execução do programa em *funções* foi positivo no sentido de garantir maior flexibilidade e independência das rotinas, onde cada uma é bem definida e para alterar algum parâmetro muda-se dentro de cada respectiva *função*. Cada rotina trata de alguma etapa especifica do funcionamento do programa. Fica intuitivo entender o código depois de analisar o *fluxograma* já visto anteriormente. O programa e suas funções foram desenvolvidos no ambiente de programação *CCS C Compiler 5.010* demonstrado na Figura 3.15, que oferece a possibilidade do desenvolvimento de um programa em *linguagem C*, uma linguagem de mais alto nível que o Assembler, por exemplo.

Figura 3.15 – Ambiente de programação

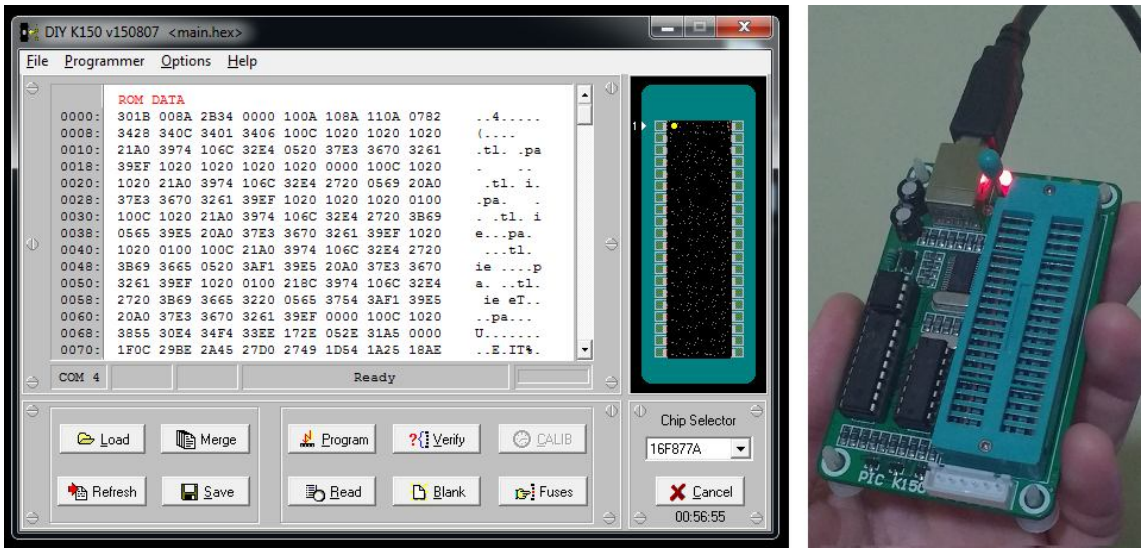


Fonte: Elaborado pelo autor.

Durante o desenvolvimento do programa, o processo de compilar o código e testar na prática era contínuo, principalmente para validar as etapas de funcionamento do projeto, como por exemplo: levantamento de amostras do *sensor*, excursão do *pwm* atuando sobre a *bomba*, teste da navegação nas telas, etc. Ao compilar o código sem erros no ambiente do *CCS*, é gerado um arquivo em formato ASCII (*.HEX*) que serve para programar o microcontrolador. Este arquivo contém as configurações e informações sobre o código fonte. Foi utilizado um gravador de *PIC* comercial que veio com software próprio chamado *DIY K150 v150807*, este recebe o arquivo *hexadecimal* e programa eletricamente o microcontrolador. O gravador de

PIC chama-se *PIC K150* e foi comprado por um site de compras coletivas brasileiro, a aquisição é de fácil acesso e baixo valor comparado a sua funcionalidade. Na Figura 3.16 é demonstrado o software e gravador.

Figura 3.16 – Software e hardware do gravador de microcontrolador *PIC*



Fonte: Elaborado pelo autor.

Será descrito a seguir o nome das principais funções que são chamadas pela rotina principal *main*, assim como o seu funcionamento e finalidade. O código fonte completo do projeto em *software* está no apêndice ao final deste relatório.

- ***animacao_abertura( ):*** Esta é a primeira função que é chamada quando o programa inicia, ela é executada apenas uma vez, e só pode ser executada novamente quando a *IHM* é reiniciada. A *função* imprime na tela o nome do projeto com uma animação de letras deslizantes, ao mesmo tempo verifica se o usuário pressionou o botão *OK*, pois esta ação indica que o mesmo dispensa a animação e deseja prosseguir rapidamente para o *Menu Principal*. Se não houver interação do usuário, a animação permanecerá na tela até que seja pressionado o botão *OK*, para então prosseguir ao *Menu Principal*.
- ***navegar_menu( ):*** Após a tela de abertura animada, esta é a primeira função que o programa chama quando entra no *Menu Principal*. Cada vez que o usuário pressiona os botões *MENOS* e *MAIS* é alterado o valor da variável auxiliar *menu* com limites entre 1 e 4 (1-*SP*, 2-*Start*, 3-*PID*, 4-*Info*). Conforme for o valor de *menu* é impresso uma nova tela mudando de lugar o cursor *'X'*, para indicar a opção que está sendo selecionada na navegação. Quando é pressionado *OK*, esta função termina e volta para a rotina principal *main*; o último valor de *menu* corresponde a opção que o usuário digitou *OK* para entrar, e será acessada da rotina principal.

- ***atualiza_valor_nivel( ):*** Serve para obter periodicamente o valor de nível do *sensor ultrassônico*. No início envia um pulso colocando o pino *TRIGGER* do *sensor* em nível alto, espera 10µs, coloca o pino *TRIGGER* em nível baixo, e espera o retorno do pulso no pino *ECHO*; no mesmo tempo que começa a esperar é setado o *timer1* para contar o tempo. O programa fica em loop até receber retorno pelo pino *ECHO*, quando recebe, coloca o valor de tempo do *timer1* na variável *time*, que é utilizada posteriormente em cálculos para descobrir a distância desejada, este valor é calculado e salvo na variável *nivel_agua*. Em primeiro momento a variável *media_nivel* recebe diretamente o valor de *nivel_agua*, depois é feito um filtro da média de três amostras para levantar as amostras, a fim de eliminar acionamentos discrepantes na bomba. Os valores de nível atualizado sempre estarão na variável *media_nivel*.
- ***controlePID( ):*** Rotina muito específica e que executa todo *controle PID* do sistema, é uma das rotinas mais importantes do projeto. Primeiro calcula o valor do *Erro* que é igual ao *Set Point* escolhido, mais o zero definido, menos o valor do *Nível* de água (*media_nivel*). O valor do *Erro* é o quão distante o *Nível* d'água está do *Set Point* pode-se dizer assim, e ele influencia diretamente o calculo dos valores de *p, i* e *d*, que ocorrem em seguida nesta função. A soma dos valores de *p, i* e *d*, geram o sinal de controle que deve ser aplicado na *bomba*, esta soma ainda é multiplicada pela divisão da variação do valor pwm (260 – 230) pela variação da tensão (1,22 – 1,079), que é igual a 30 / 0.141 = 212.8. Estas variações são referentes ao ensaio da *resposta ao salto* que será apresentado com maiores detalhes na *análise de resultados*. Depois ocorre a verificação se o valor resultante deste cálculo do sinal de controle ultrapassou os limites definidos, ou seja, se for menor que zero ou maior que 1023. Se isto acontecer o sinal de controle será o limite. Para finalizar esta *função* são salvos os valores de *erro atual* como agora *erro anterior*, *i atual* como *i anterior*, e *d atual* como *d anterior*, isto é feito para efeito de cálculo das próximas vezes em que esta função será chamada.
- ***executa_ctrl( ):*** Esta é a *função* mais chamada em todo programa, isto por que ela agrega funções muito importantes para o controle. Dentro dela é chamado primeiro a função *atualiza_valor_nivel( )* que já foi explicada acima, depois chama a função *controlePID( )*, por fim seta a *bomba* pela saída *pwm* do *PIC* com o valor *sinal_ctrl* resultante do cálculo baseado nas duas funções chamadas nesta rotina.

As funções a seguir estão relacionadas com as opções de navegação apresentadas no *Menu Principal* da *IHM*, quando selecionada alguma opção, são estas funções que serão chamadas.

- ***configuraSP( ):*** É chamada depois do termino da função *navegar_menu( )* quando a variável *menu=1*, esta função entra em um loop que primeiramente chama a função *executa_ctrl( )* e depois espera receber alteração do valor de *Set Point* pelo usuário. Se pressionado botão *VOLTA*, o programa sai do loop e volta para tela de *Menu*; os botões *MENOS* e *MAIS* alteram o valor do *Set Point*, e o *OK* salva o valor configurado e retorna para o *Menu Principal*.
- ***configuraP, I, D( ):*** Cada rotina configura o valor do respectivo parâmetro, as três funções são chamadas em seqüência depois do termino da função *nevagar_menu( )* quando a variável *menu=3*, estas funções funcionam da mesma forma e tem o mesmo objetivo que a função *configuraSP( )* já explicada. A única diferença é que a navegação na tela destes três parâmetros é em seqüência, onde na tela do K_p, *VOLTA* retorna para o *Menu* e *OK* avança para tela do T_i, e na tela T_d, *VOLTA* retorna para tela do T_i, e *OK* salva todas configurações e retorna para o *Menu Principal.*
- ***info( ):*** Esta função é chamada no retorno de *navegar_menu( )* quando *menu=4*, enquanto imprime as informações do *aluno desenvolvedor do projeto* na tela *executa em paralelo* em todas telas, constantemente, a função *executa_ctrl( )* e também espera o usuário digitar *OK* para avançar para próxima tela de informações. Ao total são três telas impressas nesta função e cada uma depende de *OK* para prosseguir e finalmente retornar ao *Menu*.
- ***start( ):*** Ao retornar da função *navegar_menu( )* com *menu=2*, então é dado início ao *controle PID* do sistema, agora com valores de *Set Point* e *PID* configurados pelo usuário, mas antes é verificado se houve configuração do *PID* quando o controle já estava executando com algum valor setado anteriormente. Se isto for verdade é preciso colocar o nível em zero primeiro para depois liberar o controle para o usuário ver a busca do nível com as novas configurações em vigência. Ao final desta função, independente da verificação ou não ser verdadeira, o principal objetivo é colocar os valores salvos de *SP* e *PID* nas efetivas variáveis que serão utilizadas durante o controle do sistema.

- ***salvando( ):*** Esta é mais uma função de animação que serve apenas para informar ao usuário que foram salvas as suas configurações. Esta rotina só é chamada se ao final da parametrização de *SP* ou *PID* os valores selecionados são diferentes do que estava salvo inicialmente, se forem iguais, não irá chamá-la e retorna para ao *Menu*.

3.3 Projeto em Hardware

O projeto em hardware é dividido em duas partes principais que correspondem ao projeto eletrônico e o projeto mecânico da *IHM*. A parte eletrônica trata-se da alimentação, controle e acionamento da bomba. A parte mecânica apresenta uma descrição do trabalho para desenvolver uma *IHM* que fosse de utilização prática pelo usuário.

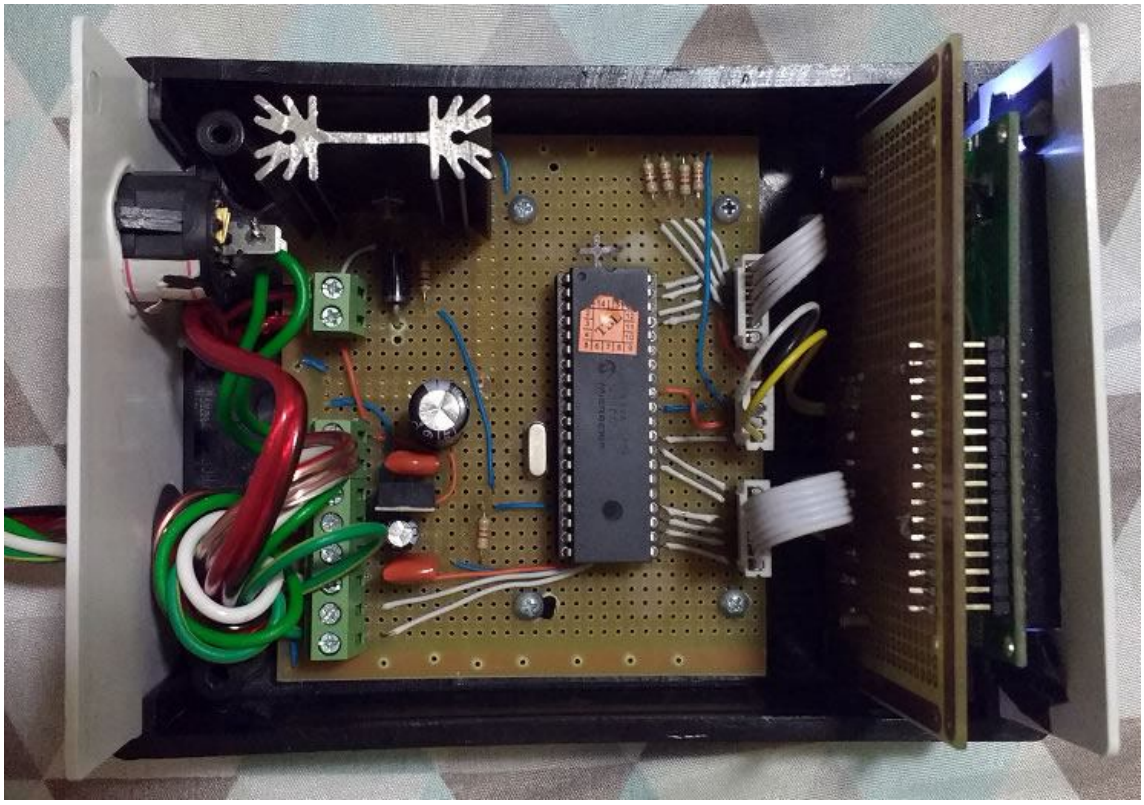
3.3.1 Projeto e Esquema Eletrônico

O programa desenvolvido é executado no microcontrolador *PIC 16F877A* desenvolvido pela *Microchip*, que é um dos mais utilizados em projetos microcontrolados devido a sua simplicidade e facilidade de programação. Citando as características do chip que interessam para este projeto, pode-se mencionar que o mesmo tem freqüência de operação de 4MHz (mas pode ir até 20MHz). No projeto faz uso de 47% da memória RAM (368 bytes disponíveis) para execução de rotinas, e 89% da ROM (256 bytes disponíveis) para armazenamento das informações inicias de programa, como variáveis, valores, definições e rotinas. Trabalha com alimentação de 5Vdc (pode trabalhar entre 2 e 5,5 Vdc), possui pinagem de 40 pinos no encapsulamento PDIP. Os periféricos podem ser divididos em 5 conjuntos de portas de entrada e saída que somam um total de 33 portas, porém estão sendo usadas apenas 14 portas, ou seja, aproximadamente 40% da capacidade. Utiliza-se 1 módulo CCP do total de 2 disponíveis (Comparação, Captura e PWM), é utilizado 2 dos 3 Timers disponíveis, um para *pwm* e outro para obter a distância pelo sensor, por fim, usa-se 1 conversor analógico-digital de 10 bits de resolução utilizado para o *pwm*.

Optou-se por utilizar uma placa de circuito impresso já furada e pronta para soldar os componentes conforme demonstrado na Figura 3.17, esta placa também é facilmente encontrada em qualquer loja de componentes eletrônicos. Por causa desta escolha, não foi feito um projeto a nível de *PCB* (Printed circuit board), apenas foi analisado o dimensionamento da placa e qual melhor lugar para se colocar cada componente de forma que ficasse mais perto das suas respectivas ligações eletrônicas. Outra escolha de projeto foi de

colocar um botão *on/off* (liga e desliga) na parte traseira da *IHM*, o objetivo é de centralizar neste botão toda alimentação do circuito como uma medida de segurança, para que seja possível desligar caso ocorra alguma anomalia de funcionamento. Ao pressioná-lo, fecha-se o caminho da alimentação positiva da fonte chaveada de +12Vdc/5A e toda *IHM* é ligada.

Figura 3.17 – Circuito Eletrônico da parte de hardware da *IHM*



Fonte: Elaborado pelo autor.

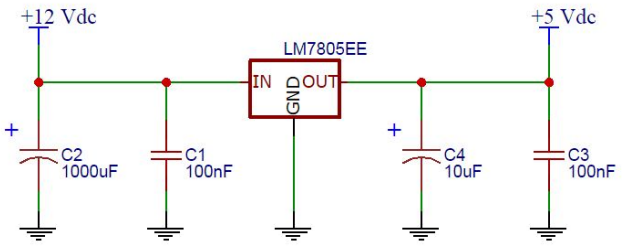
Cabos elétricos com cerca de 4 metros estão sendo utilizados para ligar a *IHM* ao protótipo, estes cabos possuem conector de engate rápido bem no meio do comprimento, para facilitar o manuseio das ligações. A estrutura do protótipo tem fixado uma placa eletrônica que recebe o plug de alimentação da fonte chaveada de +12Vdc/5A, cabos são ligados à esta placa e vão alimentar a *IHM*. Os cabos entre a *IHM* e o protótipo são em um total de sete, sendo eles: *GROUND* e +12Vdc são entradas; +5Vdc, *TRIGGER*, *ECHO*, *+Bomba*, *-Bomba*, são as saídas do circuito da *IHM*.

O circuito eletrônico pode ser explicado por etapas para facilitar o entendimento sobre o funcionamento da parte prática de *hardware* do projeto, sendo assim divide-se entre três partes a seguir: *Alimentação*, *Controle* e *Acionamento da bomba*.

Na etapa de *Alimentação*, o objetivo é proporcionar ao circuito da *IHM* recursos para um funcionamento confiável e sem ruído. A própria fonte chaveada de +12Vdc que alimenta

todo o sistema já possui uma saída que é fixa e livre de ruídos. Em primeiro momento quando o circuito recebe alimentação contínua, ele possui partes que devem ser ligadas em 12Vdc e outras em 5Vdc; toda parte de Controle é alimentada em +5Vdc, como por exemplo, o *microcontrolador*, *display LCD*, etc. A parte de *acionamento da bomba* já trabalha com +12Vdc, para esta a alimentação vem direto da fonte chaveada. Para criar uma tensão de +5Vdc confiável, foi projetado um circuito de capacitores em paralelo com as entradas e saídas do regulador de tensão *LM7805*, responsável por receber a tensão de +12Vdc e gerar uma saída de +5Vdc/1A que alimenta toda parte de controle do circuito. Os capacitores em paralelo eliminam ruídos e fixam o valor da tensão em meio a variações. É interessante comentar que o capacitor eletrolítico de 1000uF também tem a função de eliminar os ruídos gerados pelo acionamento da *bomba*. Os ruídos podem atrapalhar o funcionamento e controle do *PIC*, e inclusive gerar comportamento anômalo durante a navegação na *IHM* através da propagação de ruídos. Os demais capacitores servem de filtro e estabilidade da saída de 5Vdc. Observa-se a etapa de *Alimentação* do circuito na Figura 3.18.

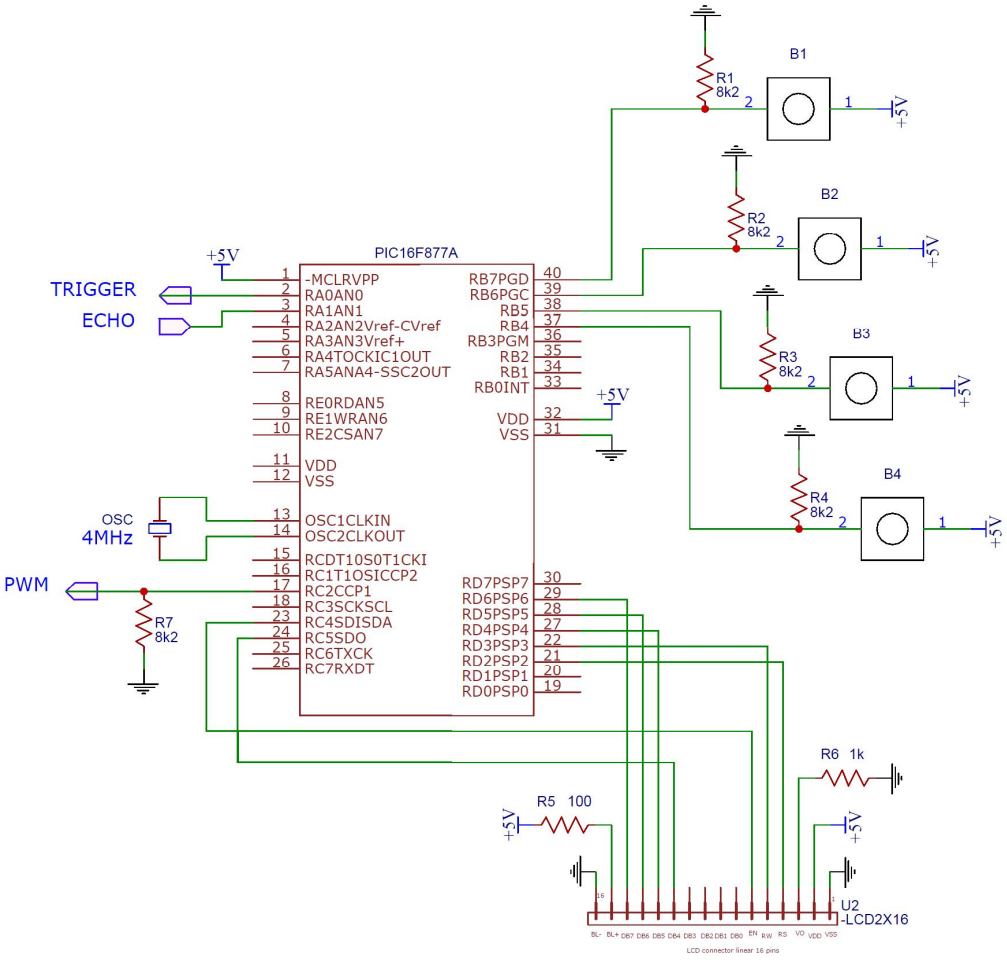
Figura 3.18 – Etapa de *Alimentação* da *IHM*



Fonte: Elaborado pelo autor.

A parte de *Controle* é a mais extensa por que envolve vários elementos importantes no funcionamento do projeto, como: *Microcontrolador*, *Sensor Ultrassônico*, *display LCD* e botões de navegação. Todos estes elementos são alimentados com +5Vdc, sem exceção. Foram utilizadas duas placas de circuito impresso para desenvolver a IHM, uma para o *PIC* e outra para o *display LCD* e os botões de navegação. Utiliza-se barra de pinos e conectores para ligar as duas placas. A parte de *Controle* caracteriza-se pelo microcontrolador e suas entradas e saídas. São utilizadas *4 entradas digitais* para os botões mais *7 saídas digitais* para comunicação com o *display LCD*. Para envio e recebimento do sinal do Sensor são utilizadas *1 saída analógica* para o *TRIGGER* e *1 entrada analógica* para *ECHO*. Finalmente para comunicação com a parte de *Acionamento da bomba* é utilizado *1 saída PWM*, conforme demonstrado na Figura 3.19.

Figura 3.19 – Etapa de *Controle* da *IHM*



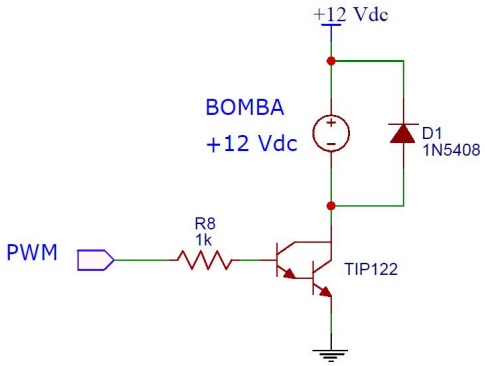
Fonte: Elaborado pelo autor.

Cada pino do *PIC* que está configurado como entrada tem um resistor de *pull-down* de 8k2Ω para evitar um valor de entrada flutuante enquanto o pino não está recebendo sinal em sua entrada, este conceito se aplica também ao pino de saída *PWM* para criar um divisor de tensão no acionamento da *bomba* através do transistor. Os pinos de alimentação do *microcontrolador* devem ser alimentados com *GROUND* e +5Vdc, assim como o pino *masterclear* em +5Vdc. Para obter a freqüência de operação foi utilizado um oscilador cristal de *4MHz* nos respectivos pinos do *PIC*. No *display LCD* as alimentações estão nas extremidades, este também recebe *GROUND* e +5Vdc; na alimentação da luz de fundo do *display* (*backlight*) é utilizado um resistor em série de 100Ω para melhor ajuste; na definição do contraste do *display* após alguns testes foi usado um resistor de 1kΩ que define um contraste aceitável de fundo para as informações no display. Os dados são enviados pelos pinos de saída digital do *PIC* para os pinos do *display* de DB4 até DB7, assim como para os

pinos de controle para escrita no *display*, mais especificamente falando: EN (*enable*), RW (*read/write*), RS (*register select*). Os botões são do tipo *push-botton* e quando pressionados formam caminho da alimentação de +5Vdc até uma entrada digital do PIC. A configuração mais detalhada das definições do projeto é possível acompanhar pelo código fonte do programa no apêndice deste documento.

O *Acionamento da bomba* acontece através da variação do sinal *pwm* enviado pelo pino do *PIC*. O circuito é relativamente simples onde um transistor TIP 122 recebe a variação do *pwm* pela base com um resistor de 1kΩ em série. O TIP é um transistor NPN do tipo Darlington que equivale a dois transistores ligados em cascata no mesmo invólucro, para ter um componente de potência e alto ganho. Este transistor transforma um sinal muito fraco na entrada (base) em um sinal potente na saída (*coletor* ou *emissor*). Ele foi desenvolvido para aplicações de chaveamento e possui um diodo interno de proteção na junção *coleto-emissor*. Quando aumenta-se a tensão de saída do pino do microcontrolador (através da variação do valor *pwm)* recebida pela entrada do TIP, é possível obter maior circulação de corrente entre a junção *coletor-emissor* e isto faz com que o acionamento da *bomba* aumente a intensidade e vice-versa. O circuito de acionamento da *bomba* é apresentado na Figura 3.20.

Figura 3.20 – Etapa de *Acionamento da bomba* da *IHM*



Fonte: Elaborado pelo autor.

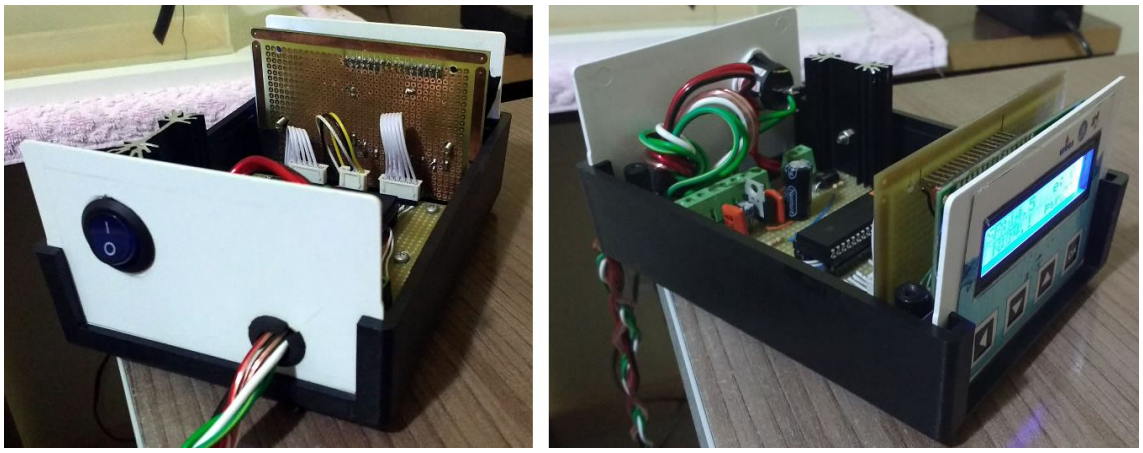
A *bomba* com seus dois terminais +*Bomba* e –*Bomba* está ligada respectivamente entre a alimentação +12Vdc e o terminal *coletor* do TIP. O terminal *emissor* do transistor é ligado direto no *GROUND* do circuito. Como proteção contra corrente reversa no acionamento da *bomba* foi utilizado, além do diodo de proteção que já existe no TIP, mais o diodo semicondutor 1N5408 que suporta correntes de até 3A; este é ligado em paralelo com a *bomba*. Em função da corrente de acionamento da *bomba* ser relativamente alta é necessário utilizar no transistor TIP um dissipador de calor para não danificar o componente por superaquecimento.

3.3.2 Interface IHM

Para fixar as placas de circuito impresso do microcontrolador, *display* LCD e botões, foi utilizada como suporte uma caixa plástica comercial. Através de um trabalho artesanal foi recortado encaixes nas tampas laterais para o *display LCD* e botões de navegação, assim como para o botão de *on/off*. A caixa é da cor preta com tampas laterais brancas, e possui fendas para possibilitar a circulação de ar. Cabos serão ligados entre a *IHM* e o protótipo.

As dimensões da caixa utilizada para desenvolver a *IHM* são de 11cm de largura, 8cm de altura e 15cm de profundidade, conforme mostra Figura 3.21. Na tampa frontal foi colada uma arte auto-explicativa sobre a função de cada botão para induzir o significado ao usuário. A interface é de fácil manuseio, o comprimento dos cabos possibilita maior liberdade de movimentação ao usuário.

Figura 3.21 – Caixa plástica em que foi desenvolvida a *IHM*



Fonte: Elaborado pelo autor.

4 ANÁLISE DE RESULTADOS

O capítulo sobre análise de resultados demonstra a parte de protótipo, software e hardware interagindo juntas para alcançar o objetivo do projeto, ou seja, o controle de nível do segundo tanque. Todos os resultados e definições estão detalhados na *Seção 4.1* sobre os *Resultados Obtidos*, e os resultados estão divididos nas subseções a seguir: *Seção 4.1.1* sobre a *Identificação do Modelo da Planta*, *Seção 4.1.2* sobre o *Projeto do Controlador PI*, *Seção 4.1.3* sobre a *Implementação Digital*, e por fim a *Seção 4.1.4* sobre a *Comparação entre Malha Aberta e Malha Fechada*.

4.1 Resultados Obtidos

É necessário detalhar os aspectos mais importantes deste projeto, a fim de esclarecer algumas definições e decisões assumidas. Os casos de teste representam todas as interações com a *planta* e o *controle* no que se refere aos testes, mais especificamente: levantar o *modelo da planta*, verificar a eficiência do *controlar* projetado e também para comprovar o *rejeite de perturbações* na entrada da *planta*. Nas próximas seções serão descritos todos os procedimentos de teste realizados, qual a sua importância e o objetivo de aplicação.

4.1.1 Identificação do Modelo da Planta

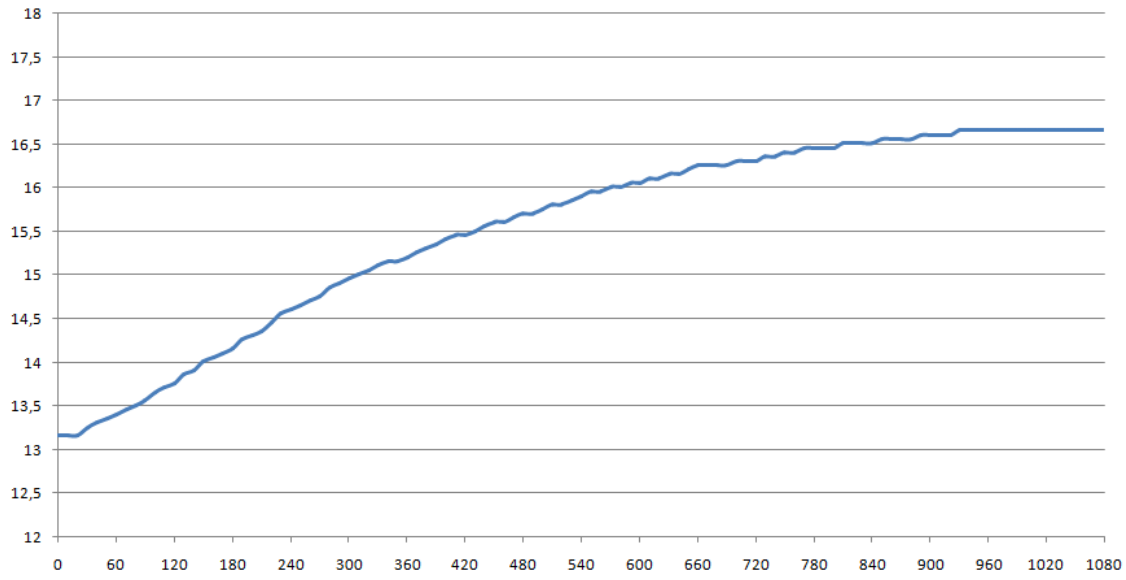
Para que seja possível obter um ajuste ótimo do *controlador* para o processo, é necessário conhecer previamente o comportamento do mesmo, seja no período *transitório* como em *regime permanente*. Conhecendo estes comportamentos é possível determinar a ação de controle para obter o melhor desempenho do sistema de controle. Segundo Bazanella (2005), o desempenho de um sistema de controle, falando em relação ao ajuste com exatidão, é tão melhor quanto mais preciso for o modelo disponível para o processo.

O método da *resposta ao salto* é realizado em *malha-aberta*, e caracteriza-se pela aplicação de um salto na *variável manipulada MV* da *planta*. O objetivo é levantar a *função de transferência* do comportamento da *planta*, para isto deve-se avaliar o comportamento da variação do nível após aplicação de um salto até a estabilização novamente. Foram realizados vários testes para determinar a intensidade de operação mínima da *bomba* que deixa o nível do segundo tanque estabilizado em *R.P.* Ao descobrir esta intensidade, foi aplicado um salto

na entrada da *planta*, ou seja, foi somado um valor ao sinal de *referência r(t)* que equivale ao valor *pwm* no *PIC* para acionamento da *bomba*.

Inicialmente o valor do *pwm* era de 230 (escala de 0 a 1023), conforme definido no programa uma excursão de 10 bits para o sinal *pwm* ($2^{10} = 1024$). O sinal de $pwm_i = 230$ equivale a uma *tensão* de saída no pino do microcontrolador de $V_i = 1{,}079$Vdc, sendo que a *tensão* máxima aplicada na saída é 5Vdc. No estado inicial o nível do segundo tanque estava estabilizado em h_i=13,16cm, sendo que a altura total do segundo tanque é de 25cm. O salto aplicado é igual ao valor de pwm_f =260, que corresponde a V_f =1,22Vdc. O valor do nível d'água começa a variar, e estabiliza novamente após o salto aplicado em h_f =16,66cm, portanto, tem-se uma variação do valor *pwm* de 30, variação da *tensão* aplicada na *bomba* de 0,141Vdc, e variação de 3,5cm no nível. O tempo total contabilizado para acomodação do nível foi de 1080 segundos, equivalente a 18 minutos. Foi utilizado o programa *Microsoft Excel* para gerar o gráfico demonstrado na Figura 4.1. A curva foi obtida anotando manualmente os valores de nível levantado pelo sensor ultrassônico na tela da IHM, a cada 10 segundos, até o nível estabilizar novamente.

Figura 4.1 – Curva da resposta ao salto da *planta* (*cm x seg*)

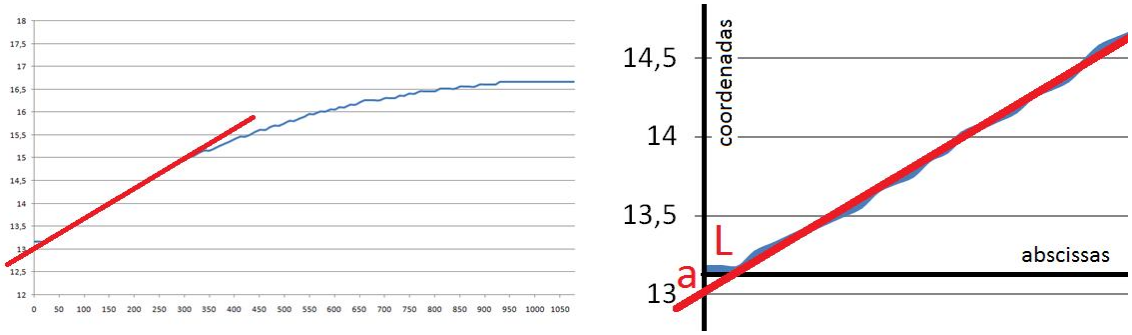


Fonte: Elaborado pelo autor.

A curva da resposta ao salto pode ser caracterizada por três parâmetros: o *atraso aparente L*, o *ganho integral equivalente a* e a *constante de tempo dominante T*. Os parâmetros representam o modelo do processo que será utilizado no presente método de ajuste. São traçados *eixos coordenados* sobre o gráfico e tomado como referência no *eixo das abscissas* o instante de tempo que o salto foi aplicado, e no *eixo das coordenadas* o valor

variável de processo antes do salto. Ao traçar uma reta que coincide com o maior ponto de inclinação no gráfico da curva, o parâmetro a corresponde a intersecção dessa reta com o *eixo das coordenadas*, e o L, corresponde a intersecção com o *eixo das abscissas.* A *constante de tempo dominante T* é definida segundo a teoria de sistemas lineares como o tempo em que a variável de processo demora em atingir 63% da sua variação total (BAZANELLA, 2005). A Figura 4.2 a seguir demonstra o procedimento para encontrar os valores.

Figura 4.2 – Demonstração dos parâmetros da resposta ao salto



Fonte: Elaborado pelo autor.

Realizando os passos descritos, os valores obtidos para os três parâmetros $a, L\ e\ T$ são respectivamente as Equações 4.1, 4.2 e 4.3.

$$a = 1{,}2023 \tag{4.1}$$

$$L = 18{,}018\ seg \tag{4.2}$$

$$h_{63\%} = h_i + 0{,}63 * (h_f - h_i) = 13{,}16 + 0{,}63 * (3{,}5) = 15{,}365\ cm$$

Ao olhar no gráfico, o valor de tempo que corresponde ao valor de nível equivalente a 15,36 cm é igual a $t_{63\%} = 390\ seg$, assim é possível calcular o valor do *tempo dominante T.*

$$T = t_{63\%} - L = 390 - 18{,}018 = 371{,}982 \cong 372\ seg \tag{4.3}$$

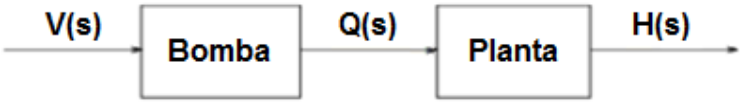
Analisando a curva do comportamento da *resposta ao salto* e conhecendo o comportamento da *resposta ao salto* de *Sistemas de Primeira Ordem*, é possível afirmar que o formato de um *SPO* é uma boa aproximação para realizar o controle. Assim, a *função de transferência* para o segundo tanque terá o modelo já demonstrado na *revisão bibliográfica*, igual ao demonstrado a seguir na Equação 4.4.

$$G_i(s) = e^{-Ls}\frac{\bar{k}}{Ts + 1} \tag{4.4}$$

A *função de transferência* é apenas para o tanque, porém o sistema também precisa ser modelado considerando a *bomba*. A Bomba tem uma dinâmica mais rápida que o sistema, por isso pode ser considerada como uma constante. Na Figura 4.3 a seguir a *bomba* transforma o sinal de *tensão* recebido *V(s)* em vazão *Q(s)*, e por isso a *função de transferência* do sistema

relaciona centímetros por volts. A representação matemática está demonstrada a seguir na Equação 4.5.

Figura 4.3 – Diagrama de blocos da modelagem considerando a *bomba*



Fonte: Elaborado pelo autor.

$$G(s) = K_{bomba} * G_i(s) = e^{-Ls}\frac{\bar{k}K_{bomba}}{Ts+1} = e^{-Ls}\frac{k_{dc}}{Ts+1} \tag{4.5}$$

O ganho DC do sistema é dado pela Equação 4.6:

$$k_{dc} = \frac{h_{max} - h_{min}}{V_{max} - V_{min}} = \frac{3{,}5}{0{,}141} = 24{,}8227\ \frac{cm}{V} \tag{4.6}$$

Em particular o *ganho* da *bomba* é o cálculo da divisão $\frac{\Delta pwm}{\Delta V}$ e está representado apenas no código fonte do *software* como ajuste do valor gerado pelo *sinal de controle*. O valor é igual ao da Equação 4.7.

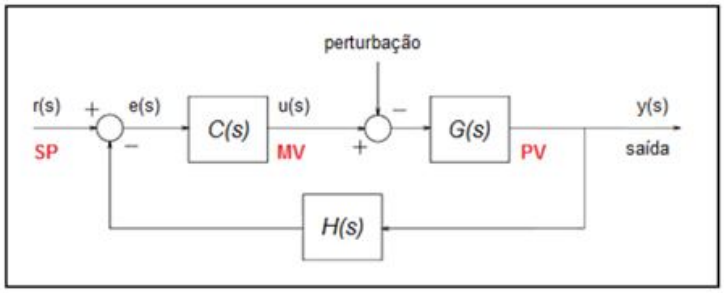
$$\frac{\Delta pwm}{\Delta V} = \frac{pwm_{max} - pwm_{min}}{V_{max} - V_{min}} = \frac{30}{0{,}141} = 212{,}766 \cong 212{,}8\ \frac{pwm}{V} \tag{4.7}$$

Finalmente é possível obter a *função de transferência* que se aproxima da curva de *resposta ao salto* da *planta* conforme Equação 4.8. O valor do *tempo dominante T* foi ajustado para 370 seg por motivo de aproximação do gráfico entre a função de transferência e a resposta ao salto.

$$G(s) = e^{-18s}\frac{24{,}823}{370s+1} = e^{-18s}\frac{0{,}06709}{s+0{,}002703}\ \frac{cm}{V} \tag{4.8}$$

A planta é uma *função* de um *Sistema de Primeira Ordem* e tem *pólo* em *-0,002703* e *ganho* de *0,06709*. O sistema completo pode ser analisado no diagrama de blocos a seguir da Figura 4.4.

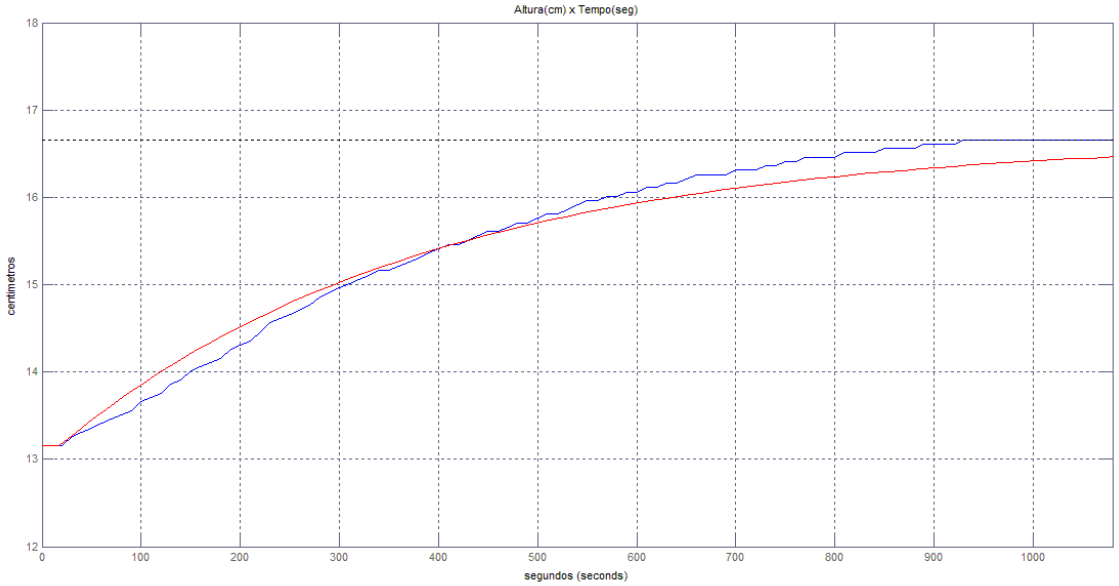
Figura 4.4 – Diagrama de blocos do sistema completo



Fonte: Elaborado pelo autor.

O software *Matlab* foi utilizado para realizar a comparação entre a *resposta ao salto* da *planta* e a *função de transferência*, que está demonstrado na Figura 4.5. O valor de *T* foi ajustado para otimizar a aproximação como já mencionado, mas nas extremidades da curva o *controle* se comporta na teoria quase como o esperado.

Figura 4.5 – *Resposta ao salto* (azul) x *função de transferência* (vermelho)



Fonte: Elaborado pelo autor.

4.1.2 Projeto do Controlador PI

A função do *controle* é garantir o sinal de saída esperado, mas, além disso, *rejeitar assintoticamente* qualquer *perturbação* que possa surgir no *processo*. Segundo Bazanella (2005), para um sistema rejeitar uma *perturbação* ele deve ser capaz de gerar internamente o *sinal de perturbação*, com sinal oposto para gerar um *cancelamento*. Para o sistema rejeitar uma *perturbação do tipo salto*, a *função de transferência* do *controlador* deve ser no *mínimo* de um *SPO*, ou seja, possuir um *pólo* na origem. Isto considerando uma *perturbação* na entrada do processo.

Considerando as especificações definidas para o projeto:

- **Overshoot ≤ 20%:** Como não se tem uma dinâmica controlável de saída de água, o overshoot deve ser o mínimo possível.

- **Tempo de acomodação ≤ 90% da resposta em malha aberta:** A planta é muito lenta, então considera-se que ela pode ser mais rápida em malha-fechada.

- **Erro de regime permanente ≤ 2%:** Escolhido por ser um valor padrão utilizado.

Percebe-se que existem duas entradas no sistema, a de *referência* e *perturbação*. Com base nesta informação e sabendo que as entradas serão do *tipo salto*, define-se o *controlador PI* para o sistema, justificando a escolha segundo teoria de cada ação de controle, *proporcional* e *integral*, já apresentada anteriormente. Desta forma a *saída* seguirá a *referência* e as *perturbações* serão amortecidas. O modelo do controlador PI está representado na Equação 4.9.

$$C_{PI}(s) = \frac{U(s)}{E(s)} = \frac{K_p(s + \frac{1}{T_i})}{s} \tag{4.9}$$

Para facilitar o entendimento é possível considerar o valor $\frac{1}{T_i}$ como *x*, pois a divisão identifica na Equação 4.10 o *zero* do controlador que atua sobre o *pólo* da *planta*.

$$C_{PI}(s) = K_p\left(\frac{s + x}{s}\right) \tag{4.10}$$

Com este *controlador* a *função de transferência* do sistema em relação a *referência* tem formato igual a Equação 4.11:

$$T_r(s) = \frac{C_{PI}(s)*G(s)}{1+C_{PI}(s)*G(s)} \tag{4.11}$$

Para realizar a análise sobre os possíveis valores de K_p e T_i é utilizada a técnica do *Lugar Geométrico das Raízes* para isso com base na análise de *F(s)*. O método é uma ferramenta gráfica que permite visualizar tendências de variação dos *pólos* da *T(s)* na medida em que o parâmetro variável na função de laço aumenta ou diminui. O cálculo da *F(s)* é demonstrado através da Equação 4.12.

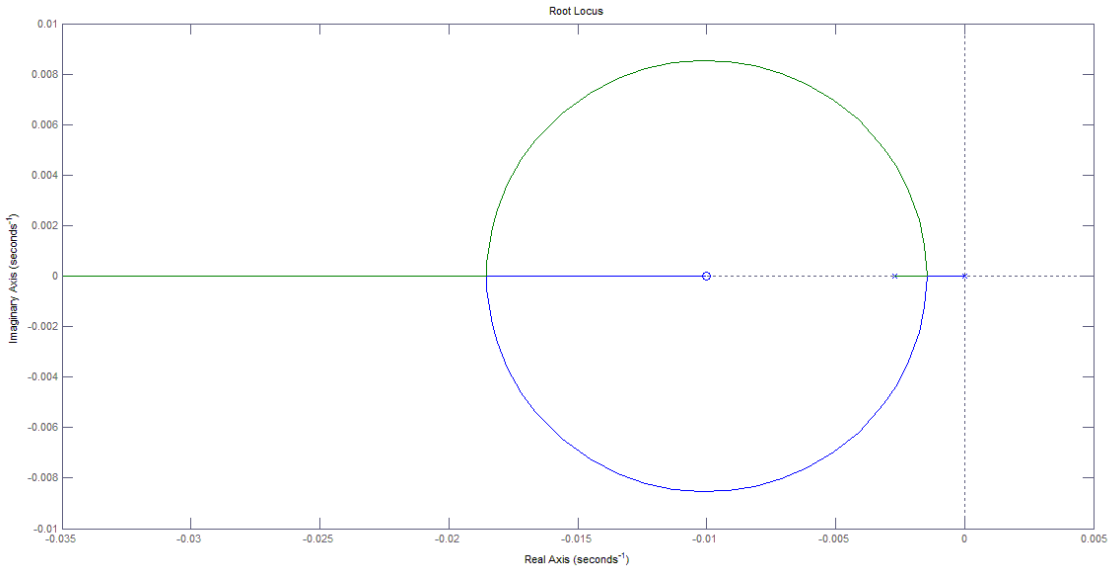
$$F(s) = C_{PI}(s) * G(s) = K_p\left(\frac{s + x}{s}\right)\left(\frac{0{,}06709}{s + 0{,}002703}\right) \tag{4.12}$$

Sabendo que a planta tem *pólo* igual a *-0,002703*, é possível escolher o mesmo valor para *x* com objetivo de anular o *pólo* da *planta*. Porém, não é aconselhável eliminar este *pólo* por que isto pode interferir drasticamente na *dinâmica de perturbação* do sistema causando uma *resposta muito mais lenta* do que a de *referência*. Se porventura for escolhido um valor diferente ao do *pólo* da *planta*, obter-se-á os mesmos *pólos* tanto para a *função de transferência* de *referência* quanto para a de *perturbação*. Como trabalha-se com controle de nível e foi escolhido como definição de projeto ter *overshoot* ≤ 20% na *dinâmica de controle*, optou-se por fixar o valor de T_i em 100, e projetar o *lugar das raízes* para escolher o melhor valor para K_p. O alto valor de T_i colabora para haver o mínimo *overshoot* na *dinâmica de controle*, mas também deixa a *resposta* do sistema lenta na busca do *set point* em *R.P.* O valor de K_p quanto mais alto, menor é o *tempo de resposta* e maior o *overshoot*, portanto, busca-se

um valor entre estes dois extremos priorizando o menor *tempo de resposta*. Lembrando que com o valor de T_i escolhido, o valor de x é igual a 0,01, sendo assim, calcula-se o valor de *F(s)* conforme a Equação 4.13 e depois é demonstrado o *Lugar das Raízes* na Figura 4.6.

$$F(s) = K_p\left(\frac{s + 0{,}01}{s}\right)\left(\frac{0{,}06709}{s + 0{,}002703}\right) = K_p\left(\frac{0{,}06709\,s + 0{,}0006709}{s^2 + 0{,}002703\,s}\right) \quad (4.13)$$

Figura 4.6 – *Lugar Geométrico das Raízes* de *F(s)*



Fonte: Elaborado pelo autor.

O *ganho máximo* é demonstrado apenas para fins de cálculo na Equação 4.14, por que o valor para K_p não pode ser tão alto em função da diminuição do *overshoot*.

$$U_{pi}(0) = U_{p(0)} + U_{i(0)} = K_p r$$

$$K_{pmax} = \frac{U_{max}}{r_{max}} = \frac{5\,V}{3{,}5\,cm} = 1{,}428\,\frac{V}{cm} \quad (4.14)$$

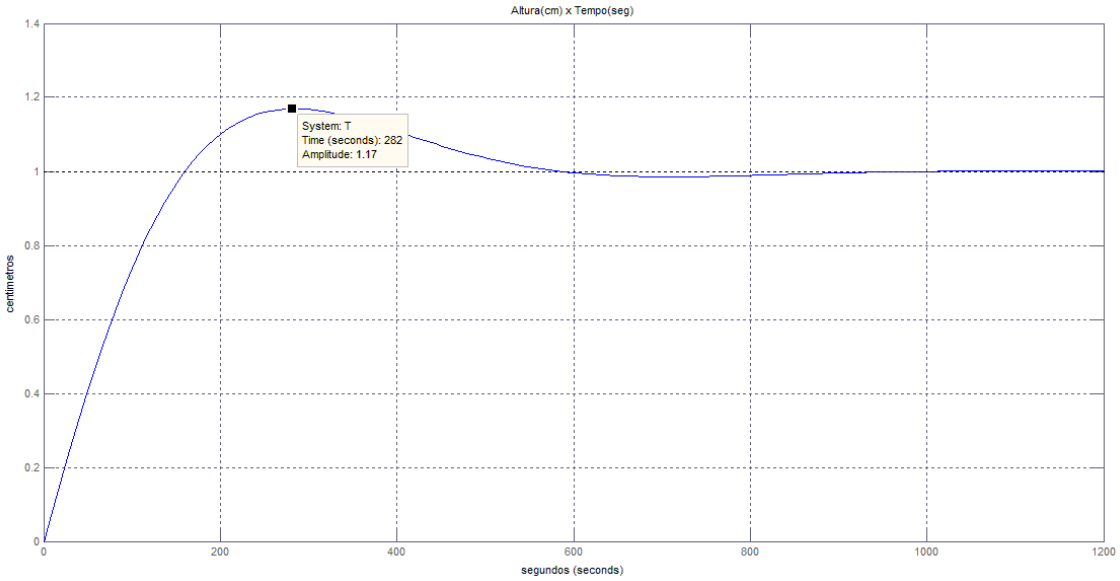
Considerando o que implica o valor de K_p, foi escolhido através da análise do *Lugar das Raízes* o valor de 0,01, que é o *pólo* mais rápido e fica perto do centro do círculo da figura que limita o tempo de acomodação. Após alguns testes práticos verificou-se que com este valor para K_p o sinal de controle calculado não era suficiente para vencer a zona-morta e outras não linearidades da bomba. O valor calculado se comportou de maneira um pouco diferente do esperado na prática, portanto, ao realizar testes práticos com outros valores maiores em busca de agilidade na resposta, obteve-se sucesso com o valor de 0,13, ou seja, o *controle* é relativamente rápido e possui *overshoot* dentro de uma faixa aceitável. Agora analisando a resposta *T(s)* do sistema com K_p igual a 0,13, foi obtida a curva de resposta da Figura 4.7 e o *Lugar Geométrico das Raízes* na Figura 4.8. O *controlador* $C_{PI}(s)$ utilizado no

projeto com este valor de K_p é o descrito a seguir na Equação 4.15, assim como a *função de transferência* $T(s)$ na Equação 4.16.

$$C_{PI}(s) = \frac{K_p(s+\frac{1}{T_i})}{s} = \frac{0{,}13(s+\frac{1}{100})}{s} = \frac{0{,}13(s+0{,}01)}{s} \quad (4.15)$$

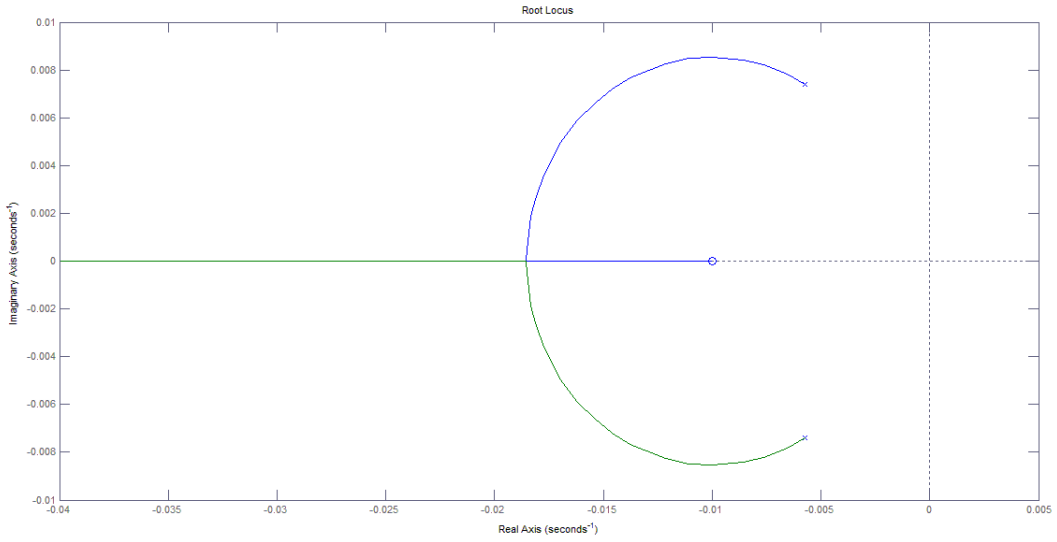
$$T(s) = \frac{C(s)*G(s)}{1+C(s)*G(s)} = \frac{0{,}0087217\ s + 0{,}000087217}{s^2 + 0{,}0114247\ s + 0{,}000087217} \quad (4.16)$$

Figura 4.7 – Resposta teórica da *T(s)* para K_p igual a 0,13



Fonte: Elaborado pelo autor.

Figura 4.8 – *Lugar Geométrico das Raízes* de *T(s)*



Fonte: Elaborado pelo autor.

Conforme análise da resposta de *T(s)* o *overshoot* obtido foi de 17% e o tempo de acomodação de aproximadamente 1000 segundos, que é equivalente a 16min. Sendo assim, o valor de *overshoot* encontra-se aproximadamente dentro do esperado, assim como o tempo de acomodação em malha-fechada, que representa aproximadamente 90% do tempo de acomodação em malha-aberta demonstrado na Figura 4.1. O tempo em malha–aberta é cerca de 1080 segundos (18min). A dinâmica do sistema é lenta para buscar o *set point* em função do alto valor de T_i, o que torna aceitável um tempo longo de *acomodação*.

4.1.3 Implementação Digital

Segundo Bazanella (2005), a implementação digital do *controlador PID* é feita através de aproximações numéricas da *derivada* e da *integral* que aparecem na rotina de *controle*. Cada *ação de controle* é descrita por uma *equação de recorrência*, que será programada no *microcontrolador* para implementar o *PID* digital.

O *sinal de controle* é atualizado apenas no instante de tempo que ocorre a *amostragem*, que no caso deste projeto foi estabelecido um *período de amostragem* de *100ms*. O *período de amostragem* foi escolhido tão pequeno quanto possível para fornecer informações frequentes ao controlador sobre a saída do processo.

Para implementação digital cada *ação de controle* deve ser discretizada, ou seja, deve ser aproximada numericamente por algum método de aproximação. Considerando o instante de tempo em que ocorre a *k-ésima amostragem*, denomina-se o sinal genericamente como $t_k = kT$, onde T é o período de amostragem, e define-se p como o *pólo derivativo*, $y(k)$ para *saída* do processo, $r(k)$ para o sinal de *referência*, $u(k)$ para o *sinal de controle* e o *erro* no instante t_k como $e(k)$. Utilizando o método de *Tustin* para isto, apresenta-se nas Equações 4.17, 4.18 e 4.19, o formato da equação para cada *ação de controle* (BAZANELLA, 2005).

$$P(k) = K_p(e(k)) \tag{4.17}$$

$$I(k) = I(k-1) + \frac{K_p T}{T_i}\frac{e(k) + e(k-1)}{2} \tag{4.18}$$

$$D(k) = \frac{(2-pT)}{(2+pT)}D(k-1) + \frac{2pK_pT_d}{(2+pT)}(e(k) - e(k-1)) \tag{4.19}$$

Considerando o *controlador* $C_{PI}(s)$ e os valores encontrados em seu projeto, os valores *default* definidos no programa da *IHM* para as *ações de controle* são de K_p igual a 0,13, T_i igual a 100 e T_d igual a zero, justamente pelo controlador ser um PI. Todos os valores citados podem ser alterados com a finalidade de gerar um novo *controlador* para o sistema,

isto através do acesso a opção de *configuração* na *IHM*. Por fim, como a rotina de implementação do *controle PID* é uma das principais do programa, é de interesse apresentá-la a seguir na Figura 4.9; a rotina é implementada em *linguagem C* e utiliza as *ações de controle* discretizadas anteriormente.

Figura 4.9 – Rotina de implementação do *controle PID*

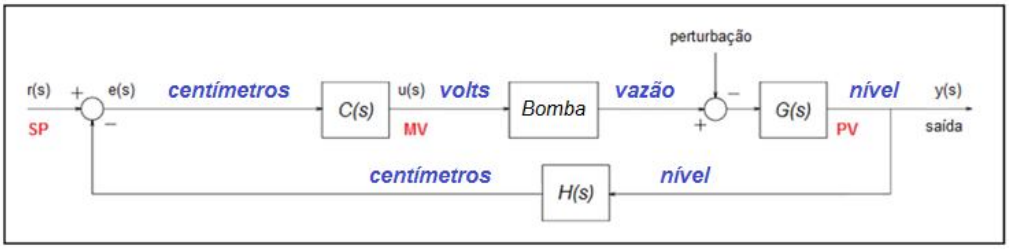
```
void controlePID() {          //rotina que executa o controle PID
    erro = (ZERO + spoint) - (media_nivel);   //calcula o Erro entre o SETPOINT e o Nível de água

    p = kp*erro;                                              //calcula valor PROPORCIONAL
    i = i_anterior + (((kp*TS)/ti)*((erro + erro_anterior)/2));    //calcula valor INTEGRAL
    d = (((2 - (POLO*TS))*d_anterior)/(2 + (POLO*TS))) + (((2*POLO*kp*td)/(2 + (POLO*TS)))*(erro - erro_anterior));   //calcula valor DERIVATIVO

    sinal_ctrl = (212.8)*(p + i + d);    //calcula sinal de controle PI (D=0) para ser aplicado no PWM

    if (sinal_ctrl > CTRL_MAX) sinal_ctrl = CTRL_MAX;     //teste para não exceder os limites do sinal de controle
    if (sinal_ctrl < CTRL_MIN) sinal_ctrl = CTRL_MIN;

    erro_anterior = erro;      //salva valor de erro atual para calculo posterior
    i_anterior = i;            //salva valor de integral para calculo posterior
    d_anterior = d;            //salva valor de derivada para calculo posterior

}
```

Fonte: Elaborado pelo autor.

4.1.4 Comparação Malha Aberta x Malha Fechada

Para executar um controle em *malha-aberta* o operador da planta deve conhecer muito bem o processo para configurar um valor de *referência* e ter a certeza que a *saída* do processo será da forma que ele espera. O funcionamento em *malha-aberta* neste projeto pode ser exemplificado pelo levantamento da planta através do método da resposta ao salto. Foram muitas tentativas alterando valores de saída *pwm* para encontrar o valor ideal que aciona a *bomba* e deixa o nível de água no segundo tanque fixo. Após encontrar o valor, era seguro que o nível atingiria o comportamento esperado, ou seja, houve muitos testes para entender a *planta* previamente e depois conseguir o comportamento da *saída* esperado. A *malha-fechada* por sua vez tem caracterização no projeto através do sinal do *sensor* que realimenta uma comparação na entrada do sistema, conforme demonstrado na Figura 4.10. A importância da *malha-fechada* é fundamental, pois sem é impossível implementar o *controle PID*, ou qualquer outro controle realimentado.

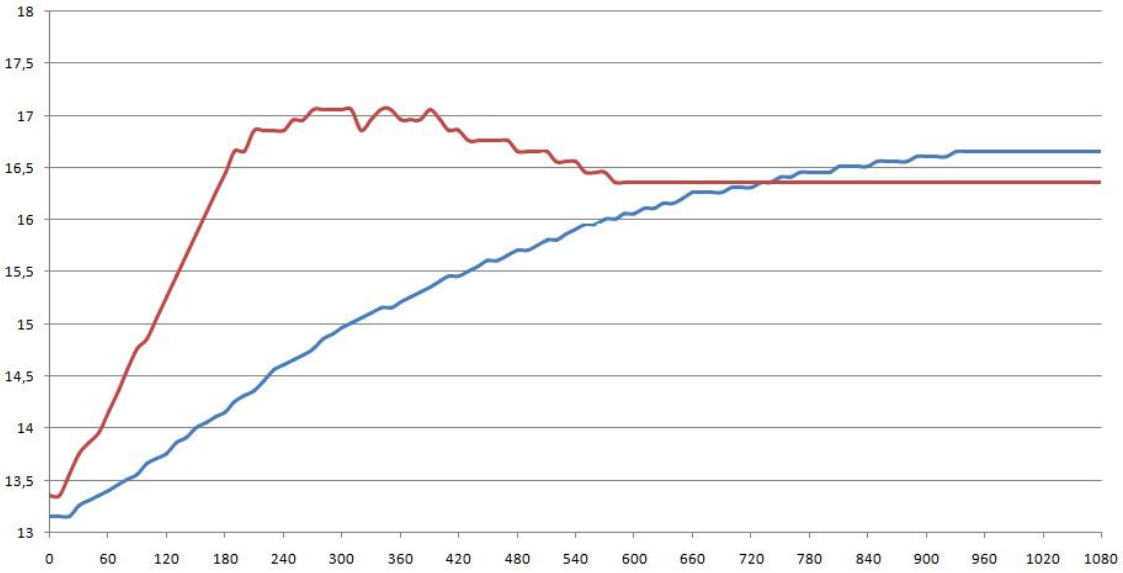
Figura 4.10 – Diagrama de blocos do sistema e conversão de unidades



Fonte: Elaborado pelo autor.

Variando cerca de 3cm, que é aproximadamente a distância que foi obtida no método da resposta ao salto, só que agora em malha-fechada, é possível comparar o tempo de acomodação do nível em malha-aberta e malha-fechada. A Figura 4.11 apresenta a acomodação do nível em malha-aberta em azul, e em malha-fechada, em vermelho. Em malha-fechada o sistema tem overshoot de aproximadamente 23% (considerando mais o erro de 3mm do sensor este percentual pode diminuir) e menor tempo de acomodação de 580 segundos, quase cerca de metade do tempo de acomodação em malha-aberta, que é de 930 segundos. A curva em malha-fechada a seguir foi obtida anotando manualmente os valores de nível levantado pelo sensor ultrassônico na tela da IHM, a cada 10 segundos, até o nível estabilizar novamente. A curva em malha-aberta é a mesma demonstrada anteriormente.

Figura 4.11 – *malha-aberta* (azul) x *malha-fechada* (vermelho)

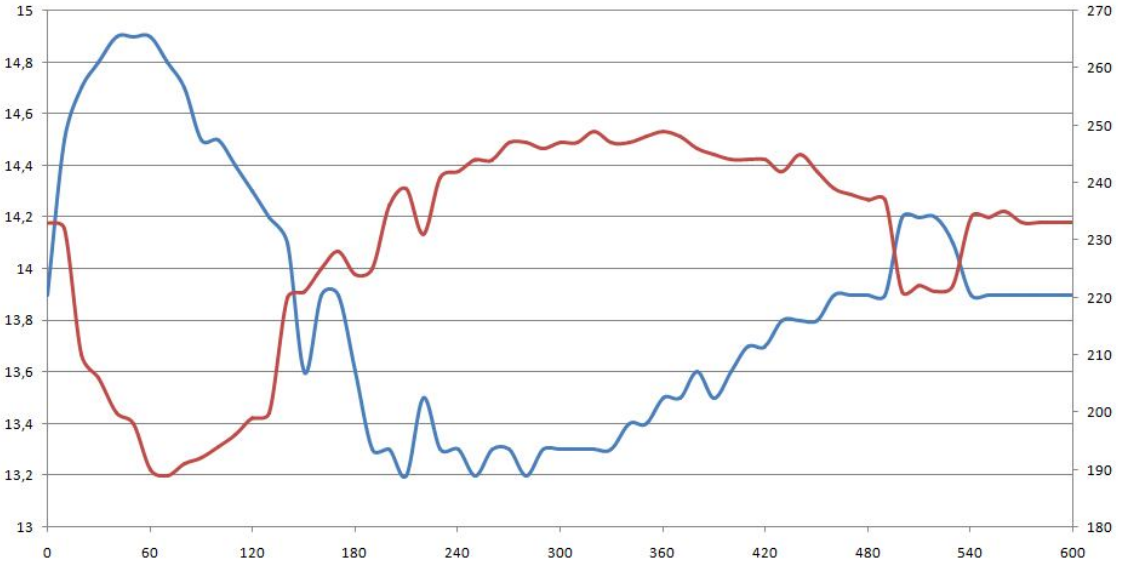


Fonte: Elaborado pelo autor.

A *malha-fechada* também possibilita ao *controle PID* rejeitar *perturbações* indesejadas na entrada do processo. Para demonstrar isto, foi feito um teste colocando o sistema em um valor de *set point* e injetando uma *perturbação* no primeiro tanque do sistema. A *perturbação* é o acréscimo de mais água no *tanque 1*, quando tira-se a tampa do recipiente e coloca-se manualmente mais água (cerca de meio litro), que faz com que o nível do segundo tanque suba inesperadamente em relação ao *controle*. A *ação* esperada é que a intensidade do acionamento da *bomba* diminua para poder acomodar o nível do segundo tanque novamente no valor inicial. Acompanhe na Figura 4.12 o comportamento do nível do *Tanque 2* comparado ao sinal de controle *pwm* para acionamento da *bomba* quando é injetada esta *perturbação*. As curvas a seguir foram obtidas anotando manualmente os valores de *pwm* e

nível levantado pelo sensor ultrassônico na tela da IHM, a cada 10 segundos, até o nível estabilizar novamente.

Figura 4.12 –Nível no *Tanque 2* (azul) x sinal de controle *pwm* (vermelho)



Fonte: Elaborado pelo autor.

Novamente, agora na prática, são atingidas as especificações definidas para o projeto, os valores encontrados vão de encontro ao que foi definido. O valor de overshoot do sistema é igual a 23%, mas com o erro do sensor de 3mm o valor levantado do nível pode estar dentro de uma margem menor, e isto pode implicar em um percentual menor ainda de overshoot, portanto, o valor encontrado está dentro do esperado que é overshoot ≤ 20%. O tempo de acomodação em malha fechada é igual a 62% do tempo de acomodação em malha aberta, e também está dentro do esperado, que por definição é o tempo de acomodação em malha-aberta ≤ 90% do tempo de acomodação em malha-fechada.

5 CONCLUSÃO

O projeto de um modo geral possui uma complexidade considerável, mas se divido em partes fica intuitivo o seu desenvolvimento. Dividindo em três grandes partes foi possível estruturar uma lógica de desenvolvimento, sendo elas: a parte de *software*, a parte de *hardware* e a parte de levantamento do *modelo da planta*, bem como foi estruturado neste documento.

A parte de *software* envolveu maior complexidade na parte de teste e funcionamento dos elementos principais que o programa controla, como por exemplo: executar o levantamento do nível de água através do *sensor* e acionar a *bomba* através do sinal *pwm*. O acionamento da *bomba* tem um controle satisfatório por que o valor de excursão do sinal *pwm* de 10 bits possibilita variar a tensão de saída do pino do microcontrolador (máxima de 5Vdc) com base em 1024 níveis de tensão, variando aproximadamente 0,005Vdc por unidade, o que resulta na maior precisão do controle. A parte de *software* praticamente não gerou problemas neste projeto, e cada problema que surgiu foi resolvido com ajustes no código fonte.

O *hardware* do projeto foi criado para satisfazer as necessidades de funcionamento, ele foi dividido em três partes, sendo elas: *alimentação*, *controle* e *acionamento da bomba*. A parte crítica que deu mais trabalho foi a parte de acionamento da bomba que por trabalhar com uma corrente alta por vezes gerava ruído e atrapalhava a parte de controle de *hardware*. O problema foi resolvido utilizando um capacitor eletrolítico na entrada da alimentação e pelo isolamento da parte de acionamento da *bomba* do resto do circuito através de uma barreira formada pelo *GROUND* na placa de circuito impresso. A parte mecânica da preparação da caixa plástica chamada de *IHM* foi simples, e foi preciso apenas alguns ajustes para fixar as placas de circuito impresso, foi uma parte relativamente boa de desenvolver.

Por fim a parte de levantamento do *modelo da planta* foi a que deu mais trabalho, uma vez que todo o processo de reconhecimento e interação com a mesma foi feito através de testes de tentativa-e-erro. Segundo Ogata (1982), a abordagem básica para qualquer projeto de sistema de controle prático envolve necessariamente processos de *tentativa-e-erro*. A síntese de sistemas de controle lineares é teoricamente possível, mas na prática, entretanto, o sistema pode estar sujeito a muitos vínculos ou pode ser não linear, como já visto anteriormente. A parte de encontrar um nível acomodado com o acionamento da *bomba* para executar a resposta ao salto foi uma das que mais demorou por causa da dificuldade em definir um acionamento fixo, a tubulação para o líquido, e uma posição de abertura para cada torneira na

planta, para isso foram feitos incansáveis testes que duraram semanas. Após isto, foi intuitivo levantar o *modelo da planta* assim como realizar os testes do *controle PID*.

O controle de nível de tanques acoplados utilizando o *controle PID* representa a consolidação dos conhecimentos teóricos aplicados na prática. Desenvolver este projeto envolve um grande aprendizado, pois engloba conceitos das cadeiras de *Algoritmos e Programação, Arquitetura e Organização de Computadores*, *Análise de Circuitos I, Análise de Circuitos II*, *Microcontroladores*, *Eletrônica I*, *Eletrônica II*, *Sistemas e Sinais*, e *Sistemas de Controle I*, todas estas compõem a grade curricular do Curso de *Engenharia de Computação* da *UFRGS*. O projeto abrange muitos conceitos e serve também como aprendizado para entender como estes conceitos se comportam na prática, e quais dificuldades podem surgir. Considerando a *planta* e o *controle* em si, foi possível obter um controle satisfatório e dentro das especificações desejadas, obedecendo definições de projeto com algumas considerações; em um geral o projeto conclui-se com êxito.

REFERÊNCIAS

ABC Engenharia. “Bomba de esguicho”. Disponível: http://abc-engenharia.blogspot.com.br/2010/05/bomba-de-esguicho.html. Acesso: novembro/2016.

Aström, K. J.; Hägglund, T. H. “PID Controllers”: theory, design and tuning. 2.ed. Research Triangle Park, Instrument Society of America, 1995.

Barroso, H. C.; Lima, R. B. C.; Barros, P. R. “Processo Didático Flexível de Quatro Tanques Acoplados para Ensino e Pesquisa em Controle”, 2014. Disponível: http://www.abenge.org.br/cobenge-2014/Artigos/129110.pdf. Acesso: outubro/2015.

Bazanella, A. S.; Silva Júnior, J. M. G. Da. “Sistemas de Controle”: princípios e métodos de projeto. 1.ed. Editora UFRGS, 2005.

Cocota Júnior, J. A. N.; Monteiro, P. M. De B.; Sanchez, M. De S.; Cruz, E. B. Da; D’Angelo, T.; Brito, R. P. De; Ireno, T. “Análise de Diferentes Controladores para o Processo de Dois Tanques Acoplados”, 2014. Disponível: http://www.abenge.org.br/cobenge-2014/Artigos/130358.pdf. Acesso: outubro/2015.

DEQ/UFSCar. “Comportamento Dinâmico de Sistemas de Primeira Ordem”. Disponível: http://www.professores.deq.ufscar.br/ronaldo/cp1/pdf/aula_primeira.pdf. Acesso: novembro/2016.

DEQ/UFSCar. “Comportamento Dinâmico de Sistemas de Segunda Ordem”. Disponível: http://www.professores.deq.ufscar.br/ronaldo/cp1/pdf/aula_segunda.pdf. Acesso: novembro/2016.

FILIPEFLOP, Blog. “Como Conector o Sensor Ultrassônico HC-SR04 ao Arduino”. Disponível: http://blog.filipeflop.com/sensores/sensor-ultrassonico-hc-sr04-ao-arduino.html. Acesso: novembro/2016.

Galdino, J. C. Da S. “Controle Robusto de um Sistema de Tanques Acoplados”, 2011. Disponível: http://www3.ifrn.edu.br/~jeangaldino/dokuwiki/lib/exe/fetch.php?media=poster_congic2011_controle_robusto_jean_parnamirim.pdf. Acesso: outubro/2015.

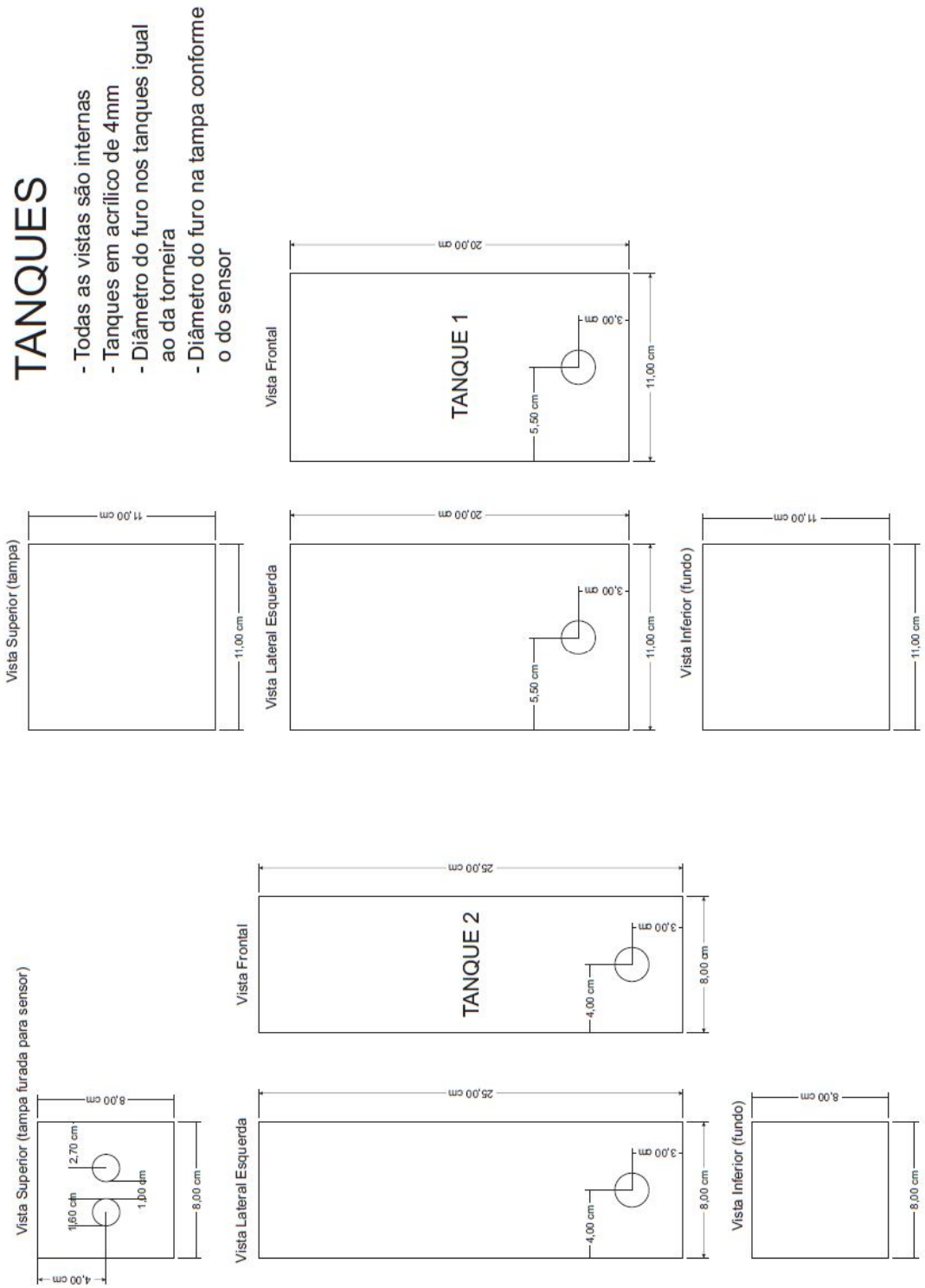
Haykin, S.; Van Veen, B. “Signals and Systems”. John Wiley & Sons, Inc, 1999.

Laubwald, E. “Coupled Tanks Systems 1”. Disponível: http://control-systemsprinciples.co.uk/whitepapers/coupled-tanks-systems.pdf. Acesso: outubro/2015.

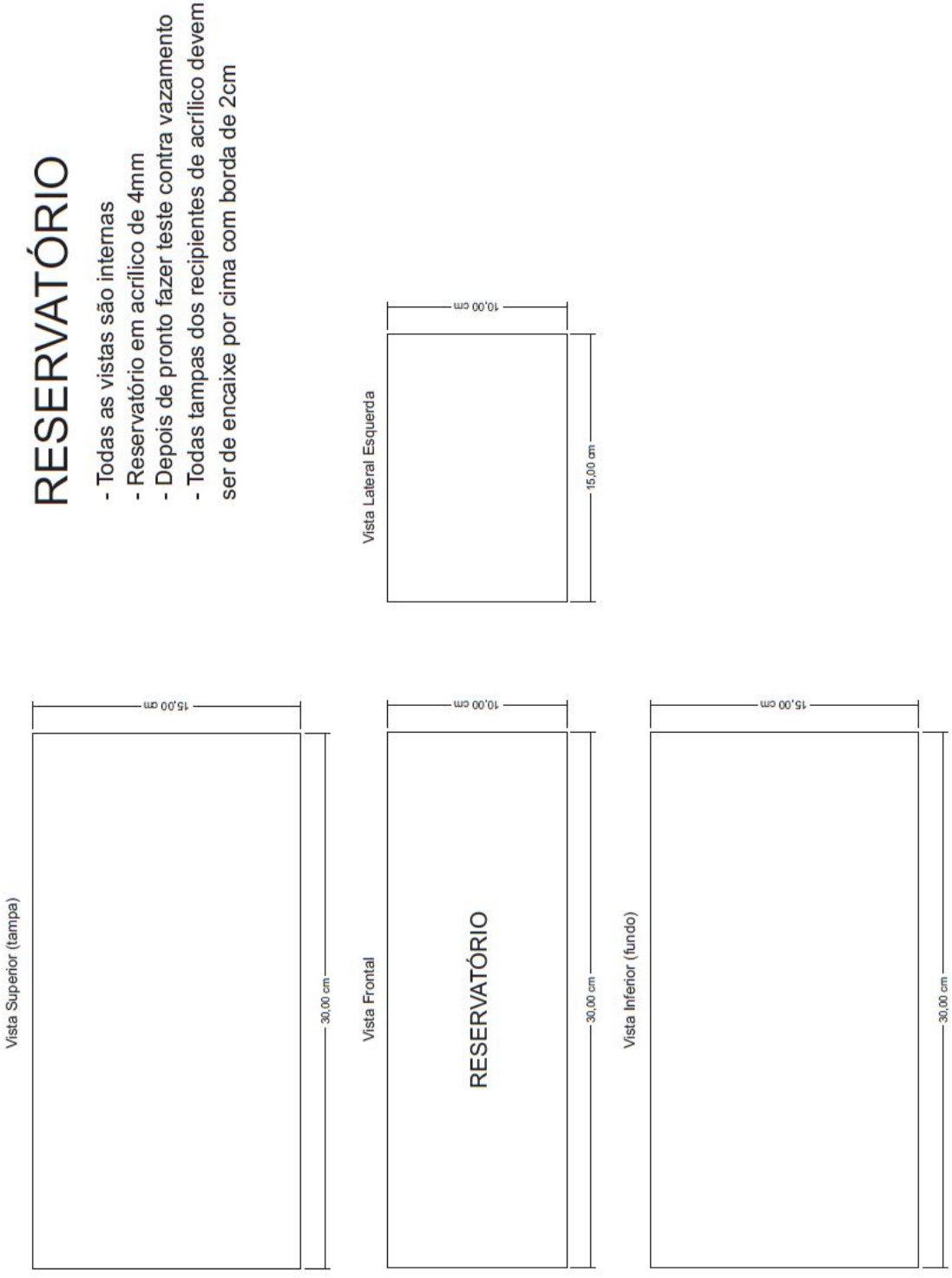
Ogata, K. “Engenharia de Controle Moderno”. Editado por André Fábio Kohn, José Carlos Teixeira de Barros Moraes. Editora Prentice/Hall do Brasil Ltda, 1982.

Souza, D. J. De; Lavinia, N. C. “Conectando o PIC”: recursos avançados. Editora Erica, 2003.

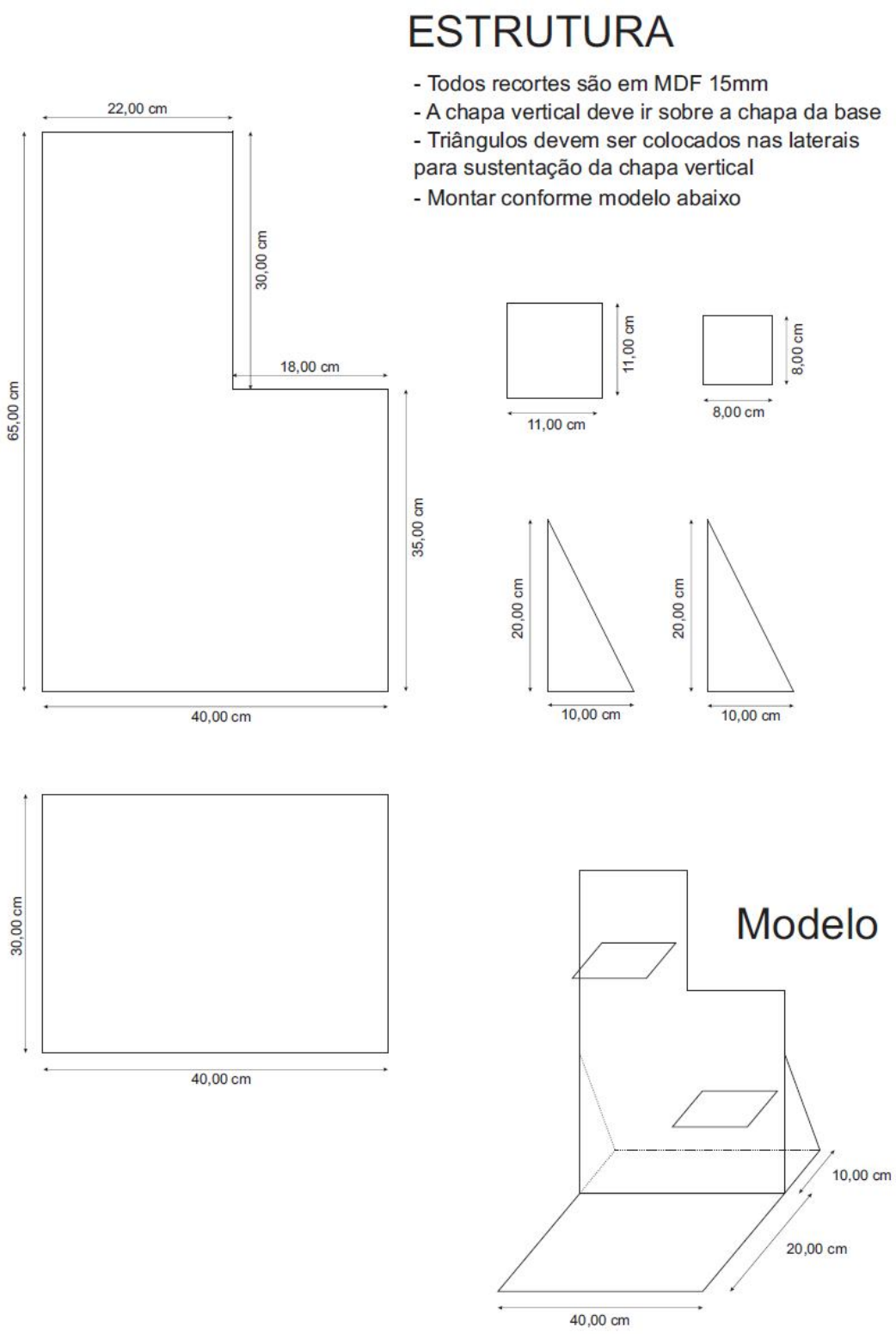
APÊNDICE SOBRE O PROTÓTIPO



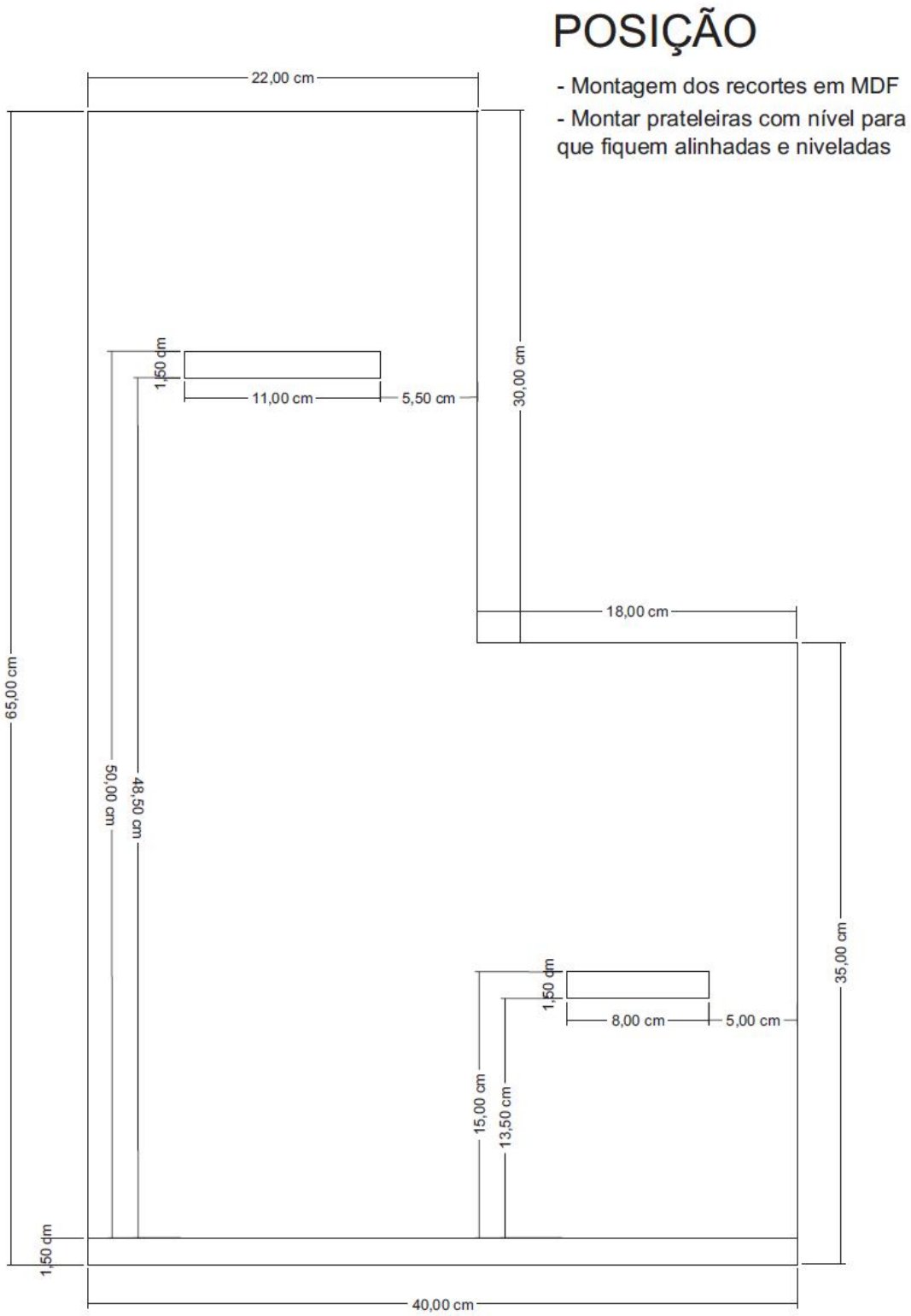
APÊNDICE SOBRE O PROTÓTIPO



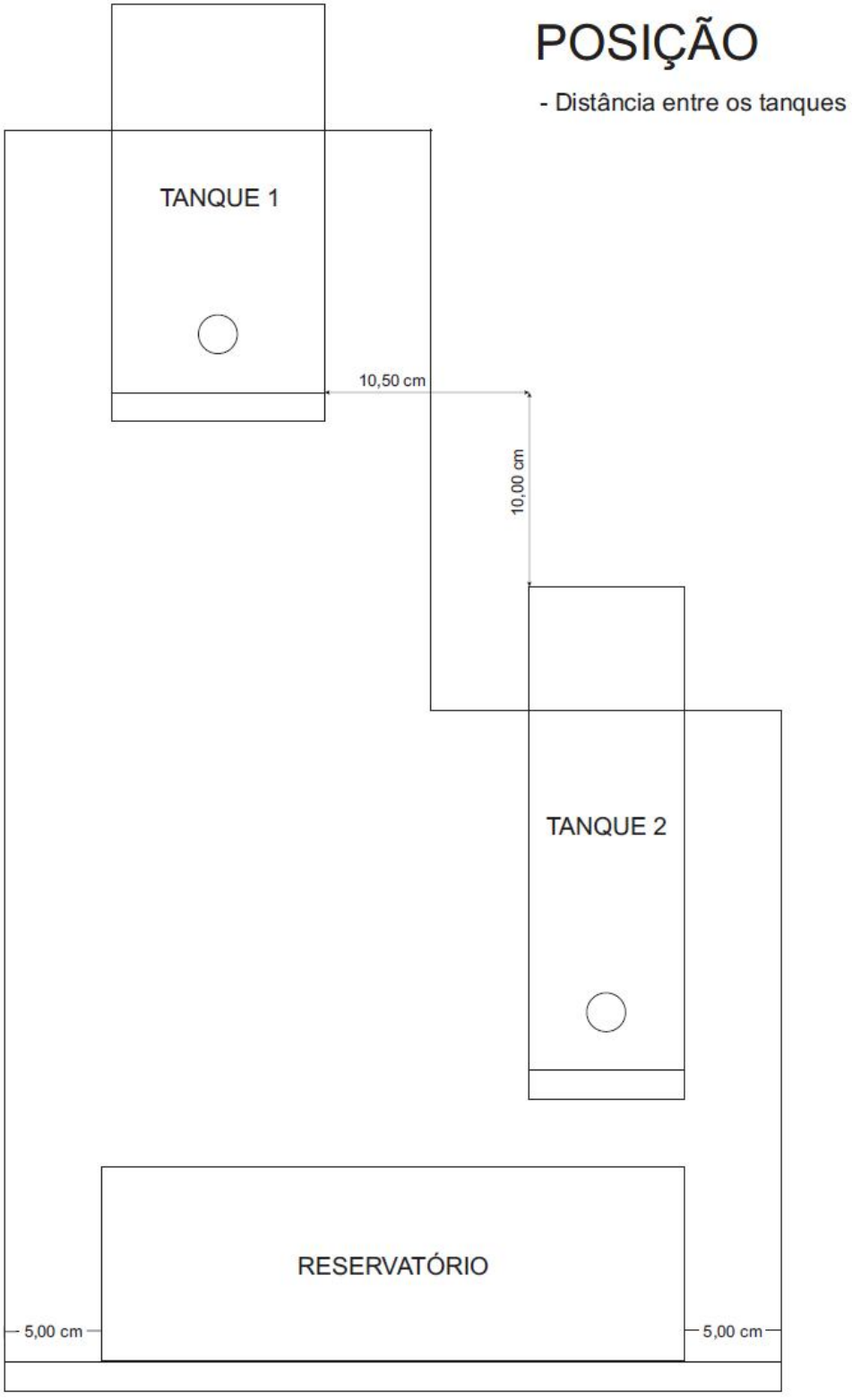
APÊNDICE SOBRE O PROTÓTIPO



APÊNDICE SOBRE O PROTÓTIPO



APÊNDICE SOBRE O PROTÓTIPO



APÊNDICE SOBRE CÓDIGO FONTE DO PROGRAMA

```
#INCLUDE <16F877A.H>                //INCLUI BIBLIOTECA DO MODELO DE PIC UTILIZADO
#DEVICE ADC=10                      //DEFINE 10 BITS PARA AMOSTRAGEM DO PWM (0-1023)
#USE DELAY(CRYSTAL=4000000)         //CRISTAL UTILIZADO

#FUSES NOWDT                        //DESABILITA WATCH DOG TIMER
#FUSES NOBROWNOUT                   //DESABILITA BROWNOUT RESET
#FUSES NOLVP                        //DESABILITA LOW VOLTAGE PRGMING, B3(PIC16) OR B5(PIC18) USED FOR I/O



////////////////////////////////////////////////////////////////////////////////////////////////////////////
// DEFINES
//
#DEFINE LCD_RS_PIN PIN_D2           //DEFINES PARA USO DO DISPLAY LCD
#DEFINE LCD_RW_PIN PIN_D3
#DEFINE LCD_ENABLE_PIN PIN_C4
#DEFINE LCD_DATA4 PIN_C5
#DEFINE LCD_DATA5 PIN_D4
#DEFINE LCD_DATA6 PIN_D5
#DEFINE LCD_DATA7 PIN_D6

#DEFINE TRIG    PIN_A0              //TRIGGER DO SENSOR QUE EMITE PULSO
#DEFINE ECHO    PIN_A1              //ECHO DO SENSOR QUE RECEBE PULSO

#DEFINE VOLTA   PIN_B4              //DEFINE DOS BOTOES DE NAVEGACAO
#DEFINE MENOS   PIN_B5
#DEFINE MAIS    PIN_B6
#DEFINE OK      PIN_B7

#DEFINE SPOINT_MAX      5.0         //LIMITES PARA CONFIGURACAO DO SETPOINT
#DEFINE SPOINT_MIN      0.0

#DEFINE KP_MAX          1.0         //LIMITES PARA CONFIGURACAO PROPORCIONAL
#DEFINE KP_MIN          0.001

#DEFINE TI_MAX          200.0       //LIMITES PARA CONFIGURACAO INTEGRAL
#DEFINE TI_MIN          0.0

#DEFINE TD_MAX          200.0       //LIMITES PARA CONFIGURACAO DERIVATIVO
#DEFINE TD_MIN          0.0

#DEFINE CTRL_MAX        1023        //LIMITES DO SINAL GERADO DE CONTROLE PWM
#DEFINE CTRL_MIN        0

#DEFINE ZERO            14.5        //ZERO DEFINIDO PARA CONTROLE DE NIVEL NO TANQUE2

#DEFINE POLO            100         //POLO PARA CALCULO DO DERIVATIVO

#DEFINE TS              0.1         //PERIODO DE AMOSTRAGEM

#INCLUDE <LCD.C>        //INCLUI BIBLIOTECA DO LCD



////////////////////////////////////////////////////////////////////////////////////////////////////////////
// VARIAVEIS
//
        //VARIAVEIS DE USO DA ATUALIZAÇÃO DE NIVEL
INT16 TIME;         //PEGA VALOR DE TEMPO DO TIMER 1 PARA CALCULAR DISTANCIA ENTRE SENSOR E AGUA
FLOAT NIVEL1, NIVEL2, NIVEL3;       //VARIAVEIS UTILIZADAS PARA CALCULO DE MEDIA COMO FILTRO
FLOAT NIVEL_AGUA, MEDIA_NIVEL;              //VARIAVEIS DE NIVEL D'AGUA

        //VARIAVEIS DE USO DO CONTROLE PID
FLOAT SPOINT_INICIAL=0, SPOINT_TEMP, SPOINT=0;              //TRABALHA COM VALORES DE SPOINT
FLOAT P=0, I=0, D=0, I_ANTERIOR=0, D_ANTERIOR=0, ERRO=0, ERRO_ANTERIOR=0;       //USO NO CONTROLE PID
FLOAT SINAL_CTRL=0;     //USO NO SINAL DE CONTROLE PWM NA BOMBA
```

```
            //IMPORTANTE: ESTAS VARIAVEIS DETERMINAM O FUNCIONAMENTO DO CONTROLE PID
FLOAT KP_INICIAL=0.13, KP_TEMP, KP=0.13;                //PROPORCIONAL
FLOAT TI_INICIAL=100, TI_TEMP, TI=100;                  //INTEGRAL
FLOAT TD_INICIAL=0, TD_TEMP, TD=0;                      //DERIVATIVO

INT MENU=1, FLAG1=1, FLAG2=0, FLAG3=0, FLAG4=0, CONT=1;       //VARIAVEIS DE USO DO MENU DE NAVEGACAO
/*
MENU  SERVE PARA SELECIONAR A OPCAO DESEJADA NO MENU; 1-CONFIGSP, 2-STARTPID, 3-CONFIGPID, 4-INFO
FLAG1 INDICA QUANDO USUARIO MUDOU OPCAO DO MENU PRINCIPAL E REIMPRIME TELA
FLAG2 SERVE PARA CONTROLE DA NAVEGACAO DE CONFIGURACAO DO PID
FLAG3 AUXILIA NA OBTENCAO DA PRIMEIRA MEDIDA DO NIVEL DE AGUA
FLAG4 AUXILIA NA VERIFICACAO SE USUARIO ALTEROU PARAMETROS PID ENQUANTO CONTROLE EXECUTA
CONT  AUXILIA NA OBTENCAO DAS AMOSTRAS PARA CALCULO DA MEDIA DO NIVEL DE AGUA
*/

INT INICIO[8]={     //CARACTERE CRIADO DO INICIO DA BARRA DE UPDATED DE PARAMETROS
  0B11111
  0B10000
  0B10111
  0B10111
  0B10111
  0B10111
  0B10000
  0B11111
};

INT MEIO[8]={       //CARACTERE CRIADO DO MEIO DA BARRA DE UPDATED DE PARAMETROS
  0B11111
  0B00000
  0B11111
  0B11111
  0B11111
  0B11111
  0B00000
  0B11111
};

INT FIM[8]={        //CARACTERE CRIADO FIM DA BARRA DE UPDATED DE PARAMETROS
  0B11111
  0B00001
  0B11101
  0B11101
  0B11101
  0B11101
  0B00001
  0B11111
};




////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// FUNCOES
//



VOID ANIMACAO_ABERTURA(){       //ANIMACAO DE ABERTURA DA IHM
  WHILE (TRUE) {
    PRINTF (LCD_PUTC, "\F         CTRL DE \NCOPLADOS          ");     //ESCREVE ANIMACAO AO MESMO TEMPO QUE
TESTA SE USUARIO PRESSIONA OK
    IF (INPUT(OK)) { BREAK; }; DELAY_MS(150); IF (INPUT(OK)) { BREAK; }; DELAY_MS(150); IF (INPUT(OK)) { BREAK; };
DELAY_MS(150);
    PRINTF (LCD_PUTC, "\F     CTRL DE NI\N ACOPLADOS      ");
    IF (INPUT(OK)) { BREAK; }; DELAY_MS(150); IF (INPUT(OK)) { BREAK; }; DELAY_MS(150); IF (INPUT(OK)) { BREAK; };
DELAY_MS(150);
    PRINTF (LCD_PUTC, "\F    CTRL DE NIVE\NES ACOPLADOS    ");
    IF (INPUT(OK)) { BREAK; }; DELAY_MS(150); IF (INPUT(OK)) { BREAK; }; DELAY_MS(150); IF (INPUT(OK)) { BREAK; };
DELAY_MS(150);
    PRINTF (LCD_PUTC, "\F  CTRL DE NIVEL \NQUES ACOPLADOS  ");
    IF (INPUT(OK)) { BREAK; }; DELAY_MS(150); IF (INPUT(OK)) { BREAK; }; DELAY_MS(150); IF (INPUT(OK)) { BREAK; };
DELAY_MS(150);
    PRINTF (LCD_PUTC, "\FCTRL DE NIVEL DE\NTNQUES ACOPLADOS");
    WHILE (!INPUT(OK)) {  }; BREAK;
  }
```

```
  WHILE (INPUT(OK)) { };  //ESPERA USUARIO PRESSIONAR OK PARA PROSSEGUIR PARA TELA DE MENU PRINCIPAL
  DELAY_MS(500);
}


VOID SALVANDO() {       //ANIMACAO DO UPDATED DAS INFORMACOES ALTERADAS DE SP OU PID
  INT I;
  PRINTF (LCD_PUTC, "\F   UPDATING...\N%C",0); DELAY_MS(100);
  FOR (I=2; I<=15; I++) { LCD_GOTOXY(I,0); PRINTF (LCD_PUTC,"%C",1); DELAY_MS(50); }
  LCD_GOTOXY(16,0); PRINTF (LCD_PUTC,"%C",2); DELAY_MS(400);
}


VOID ATUALIZA_VALOR_NIVEL() { //ROTINA QUE ATUALIZA VALOR DO NÍVEL D'ÁGUA NO LCD
  DELAY_MS(20);                     //ENVIA PULSO PELO TRIGGER
  OUTPUT_HIGH(TRIG);
  DELAY_US(10);
  OUTPUT_LOW(TRIG);

  WHILE(!INPUT(ECHO)) { }             //RECEBE PULSO PELO ECHO
  SET_TIMER1(0);
  WHILE(INPUT(ECHO)) { }

  TIME = GET_TIMER1();                //RECEBE VALOR DE DURAÇÃO DO TEMPO POR INTERRUPÇÃO DE TIMER1
  NIVEL_AGUA = 24.6 - (TIME*0.017);     //TRATA VALOR RECEBIDO E TRANSFORMA EM CENTIMETROS


  IF (FLAG3==0) {          //NAO TEM MEDIA DO NIVEL AINDA, PEGA PRIMEIRO VALOR OBTIDO
     MEDIA_NIVEL = NIVEL_AGUA;
     FLAG3=1;
  } ELSE IF (FLAG3==1) {    //PEGA 5 VALORES DE NIVEL E GERA A MEDIA
     SWITCH (CONT) {
        CASE 1: NIVEL1 = NIVEL_AGUA; CONT=2; BREAK;
        CASE 2: NIVEL2 = NIVEL_AGUA; CONT=3; BREAK;
        CASE 3: NIVEL3 = NIVEL_AGUA; CONT=1; MEDIA_NIVEL = ((NIVEL1 + NIVEL2 + NIVEL3) / 3); BREAK;
     }
  }
}


VOID CONTROLEPID() {        //ROTINA QUE EXECUTA O CONTROLE PID
  ERRO = (ZERO + SPOINT) - (MEDIA_NIVEL);   //CALCULA O ERRO ENTRE O SETPOINT E O NÍVEL DE ÁGUA

  P = KP*ERRO;                                  //CALCULA VALOR PROPORCIONAL
  I = I_ANTERIOR + (((KP*TS)/TI)*((ERRO + ERRO_ANTERIOR)/2));    //CALCULA VALOR INTEGRAL
  D = (((2 - (POLO*TS))*D_ANTERIOR)/(2 + (POLO*TS))) + (((2*POLO*KP*TD)/(2 + (POLO*TS)))*(ERRO -
ERRO_ANTERIOR));   //CALCULA VALOR DERIVATIVO

  SINAL_CTRL = (212.8)*(P + I + D);    //CALCULA SINAL DE CONTROLE PI (D=0) PARA SER APLICADO NO PWM

  IF (SINAL_CTRL > CTRL_MAX) SINAL_CTRL = CTRL_MAX;     //TESTE PARA NÃO EXCEDER OS LIMITES DO SINAL DE
CONTROLE
  IF (SINAL_CTRL < CTRL_MIN) SINAL_CTRL = CTRL_MIN;

  ERRO_ANTERIOR = ERRO;      //SALVA VALOR DE ERRO ATUAL PARA CALCULO POSTERIOR
  I_ANTERIOR = I;          //SALVA VALOR DE INTEGRAL PARA CALCULO POSTERIOR
  D_ANTERIOR = D;          //SALVA VALOR DE DERIVADA PARA CALCULO POSTERIOR
}


VOID EXECUTA_CTRL() {        //ROTINA QUE SIMPLIFICA A EXECUCAO DO CONTROLE
  ATUALIZA_VALOR_NIVEL();             //CHAMA ROTINA QUE ATUALIZA NIVEL DE AGUA
  CONTROLEPID();                    //CHAMA ROTINA QUE CALCULA NOVO SINAL DE CONTROLE PID
  DELAY_MS(100);
  SET_PWM1_DUTY((INT16)SINAL_CTRL);      //SETA VALOR PWM NA BOMBA
}


VOID CONFIGURASP() {        //ROTINA QUE RECEBE DO USUÁRIO O VALOR DE SET POINT
  SPOINT_TEMP = SPOINT_INICIAL;
  PRINTF (LCD_PUTC,"\F>>SETPOINT(SP):\N%4.1F CM", SPOINT_TEMP);
```

```
  WHILE (TRUE) {
    EXECUTA_CTRL();        //CHAMA ROTINA QUE EXECUTA CONTROLE PID EM BACKGROUND

    IF (INPUT(VOLTA)) { WHILE(INPUT(VOLTA)) { }; DELAY_MS(200); BREAK; }
    IF (INPUT(MENOS)) { DELAY_MS(100); IF ((SPOINT_TEMP - 0.1) > SPOINT_MIN) { SPOINT_TEMP -= 0.1;
LCD_GOTOXY(0,0);   PRINTF (LCD_PUTC, "%4.1F",SPOINT_TEMP); } }
    IF (INPUT(MAIS))      { DELAY_MS(100); IF (SPOINT_TEMP < SPOINT_MAX) { SPOINT_TEMP += 0.1;
LCD_GOTOXY(0,0);   PRINTF (LCD_PUTC, "%4.1F",SPOINT_TEMP); } }
    IF (INPUT(OK))       { IF (SPOINT_TEMP!=SPOINT_INICIAL) { SPOINT_INICIAL = SPOINT_TEMP; SALVANDO(); };
WHILE(INPUT(OK)) { }; DELAY_MS(200); BREAK; }
  }
}



VOID CONFIGURAP() {          //ROTINA QUE RECEBE DO USUÁRIO O VALOR DE PID
  INT CONT2=0;
  KP_TEMP = KP_INICIAL;     //RECEBE VALOR GRAVADO DE KP PARA SER ALTERADO

  IF (FLAG2==0) {
    PRINTF (LCD_PUTC,"\F>>PROPORC(KP):\N%4.3F", KP_TEMP);

    WHILE (TRUE) {
      EXECUTA_CTRL();     //CHAMA ROTINA QUE EXECUTA CONTROLE PID EM BACKGROUND

      IF (INPUT(VOLTA)) { FLAG2=3; WHILE(INPUT(VOLTA)) { }; DELAY_MS(200); BREAK; }
      IF (INPUT(MENOS)) {
          DELAY_MS(100);
          CONT2=0;
          IF ((KP_TEMP - 0.001) > KP_MIN) { KP_TEMP -= 0.001;    LCD_GOTOXY(0,0);    PRINTF (LCD_PUTC, "%4.3F",
KP_TEMP); }
          WHILE (INPUT(MENOS)) {
            CONT2+=1; DELAY_US(30);
            IF ((KP_TEMP > KP_MIN) && (CONT < 100))    { KP_TEMP-=0.001; LCD_GOTOXY(0,0); PRINTF (LCD_PUTC,
"%4.3F", KP_TEMP); }
             ELSE IF (KP_TEMP > (KP_MIN + 0.1))           { KP_TEMP-=0.01; LCD_GOTOXY(0,0); PRINTF (LCD_PUTC,
"%4.3F", KP_TEMP); }
          }
        }
      IF (INPUT(MAIS))  {
          DELAY_MS(100);
          CONT2=0;
          IF (KP_TEMP < KP_MAX) { KP_TEMP += 0.001;   LCD_GOTOXY(0,0);   PRINTF (LCD_PUTC, "%4.3F", KP_TEMP);
}
          WHILE (INPUT(MAIS)) {
            CONT2+=1; DELAY_US(30);
            IF ((KP_TEMP < KP_MAX) && (CONT < 100))    { KP_TEMP+=0.001; LCD_GOTOXY(0,0); PRINTF (LCD_PUTC,
"%4.3F", KP_TEMP); }
             ELSE IF (KP_TEMP < (KP_MAX - 0.1))           { KP_TEMP+=0.01; LCD_GOTOXY(0,0); PRINTF (LCD_PUTC,
"%4.3F", KP_TEMP); }
          }
        }
      IF (INPUT(OK))    { FLAG2=1; WHILE(INPUT(OK)) { }; DELAY_MS(200); BREAK; }
    }
  }
}



VOID CONFIGURAI() {          //ROTINA QUE RECEBE DO USUÁRIO O VALOR DE PID
  INT CONT2=0;
  TI_TEMP = TI_INICIAL;     //RECEBE VALOR GRAVADO DE TI PARA SER ALTERADO

  IF (FLAG2==1) {
    PRINTF (LCD_PUTC,"\F>>INTEGRAL(TI):\N%3.0F", TI_TEMP);
    WHILE (TRUE) {
      EXECUTA_CTRL();     //CHAMA ROTINA QUE EXECUTA CONTROLE PID EM BACKGROUND

      IF     (INPUT(VOLTA))     {     FLAG2=0;     WHILE     (INPUT(VOLTA))     {     };     DELAY_MS(200);     PRINTF
(LCD_PUTC,"\F>>PROPORC(KP):\N%4.3F CM/S", KP_TEMP); BREAK; }
      IF (INPUT(MENOS)) {
        DELAY_MS(100);
        CONT2=0;
        IF ((TI_TEMP - 1) > TI_MIN) { TI_TEMP -= 1;   LCD_GOTOXY(0,0);   PRINTF (LCD_PUTC, "%3.0F", TI_TEMP); }
        WHILE (INPUT(MENOS)) {
```

```
          CONT2+=1; DELAY_US(30);
          IF ((TI_TEMP > TI_MIN) && (CONT < 100))    { TI_TEMP-=1; LCD_GOTOXY(0,0); PRINTF (LCD_PUTC, "%3.0F",
TI_TEMP); }
          ELSE IF (TI_TEMP > (TI_MIN + 10))          { TI_TEMP-=10; LCD_GOTOXY(0,0); PRINTF (LCD_PUTC, "%3.0F",
TI_TEMP); }
        }
      }
      IF (INPUT(MAIS))  {
        DELAY_MS(100);
        CONT2=0;
        IF (TI_TEMP < TI_MAX) { TI_TEMP += 1;   LCD_GOTOXY(0,0);   PRINTF (LCD_PUTC, "%3.0F", TI_TEMP); }
        WHILE (INPUT(MAIS)) {
          CONT2+=1; DELAY_US(30);
          IF ((TI_TEMP < TI_MAX) && (CONT < 100))    { TI_TEMP+=1; LCD_GOTOXY(0,0); PRINTF (LCD_PUTC, "%3.0F",
TI_TEMP); }
          ELSE IF (TI_TEMP < (TI_MAX - 10))          { TI_TEMP+=10; LCD_GOTOXY(0,0); PRINTF (LCD_PUTC, "%3.0F",
TI_TEMP); }
        }
      }
      IF (INPUT(OK))    { FLAG2=2; WHILE(INPUT(OK)) { }; DELAY_MS(200); BREAK; }
    }
  }
}



VOID CONFIGURAD() {          //ROTINA QUE RECEBE DO USUÁRIO O VALOR DE PID
  INT CONT2=0;
  TD_TEMP = TD_INICIAL;     //RECEBE VALOR GRAVADO DE TD PARA SER ALTERADO

  IF (FLAG2==2) {
    PRINTF (LCD_PUTC,"\F>>DERIVATIV(TD):\N%3.0F", TD_TEMP);
    WHILE (TRUE) {
      EXECUTA_CTRL();     //CHAMA ROTINA QUE EXECUTA CONTROLE PID EM BACKGROUND

      IF     (INPUT(VOLTA))     {     FLAG2=1;     WHILE     (INPUT(VOLTA))     {     };     DELAY_MS(200);     PRINTF
(LCD_PUTC,"\F>>INTEGRAL(TI):\N%3.0F S", TI_TEMP); BREAK; }
      IF (INPUT(MENOS)) {
        DELAY_MS(100);
        CONT2=0;
        IF ((TD_TEMP - 1) > TD_MIN) { TD_TEMP -= 1;   LCD_GOTOXY(0,0);   PRINTF (LCD_PUTC, "%3.0F", TD_TEMP); }
        WHILE (INPUT(MENOS)) {
          CONT2+=1; DELAY_US(30);
          IF ((TD_TEMP > TD_MIN) && (CONT < 100))   { TD_TEMP-=1; LCD_GOTOXY(0,0); PRINTF (LCD_PUTC, "%3.0F",
TD_TEMP); }
          ELSE IF (TD_TEMP > (TD_MIN + 10))         { TD_TEMP-=10; LCD_GOTOXY(0,0); PRINTF (LCD_PUTC, "%3.0F",
TD_TEMP); }
        }
      }
      IF (INPUT(MAIS))  {
        DELAY_MS(100);
        CONT2=0;
        IF (TD_TEMP < TD_MAX) { TD_TEMP += 1;   LCD_GOTOXY(0,0);   PRINTF (LCD_PUTC, "%3.0F", TD_TEMP); }
        WHILE (INPUT(MAIS)) {
          CONT2+=1; DELAY_US(30);
          IF ((TD_TEMP < TD_MAX) && (CONT < 100))   { TD_TEMP+=1; LCD_GOTOXY(0,0); PRINTF (LCD_PUTC, "%3.0F",
TD_TEMP); }
          ELSE IF (TD_TEMP < (TD_MAX - 10))         { TD_TEMP+=10; LCD_GOTOXY(0,0); PRINTF (LCD_PUTC, "%3.0F",
TD_TEMP); }
        }
      }
      IF  (INPUT(OK))       {  FLAG2=3;  IF  ((KP_INICIAL!=KP_TEMP)||(TI_INICIAL!=TI_TEMP)||(TD_INICIAL!=TD_TEMP))  {
KP_INICIAL=KP_TEMP;    TI_INICIAL=TI_TEMP;    TD_INICIAL=TD_TEMP;    SALVANDO();    }    WHILE(INPUT(OK))    {    };
DELAY_MS(200); BREAK; }
    }
  }
}



VOID INFO() {              //INFORMA DADOS DO ALUNO DA OPCAO INFO DO MENU PRINCIPAL
      PRINTF (LCD_PUTC,"\FLUCAS BROLLO\N   ALPI   [201248]"); WHILE (!INPUT(OK)) { EXECUTA_CTRL(); } WHILE
(INPUT(OK)) { }; DELAY_MS(200);
      PRINTF (LCD_PUTC,"\FUFRGS 2016/2\NECP TG2"); WHILE (!INPUT(OK)) { EXECUTA_CTRL(); } WHILE (INPUT(OK))
{ }; DELAY_MS(200);
```

```
      PRINTF (LCD_PUTC, "\FCTRL DE NIVEL DE\NTNQUES ACOPLADOS"); WHILE (!INPUT(OK)) { EXECUTA_CTRL(); }
WHILE (INPUT(OK)) { }; DELAY_MS(200);
      FLAG1=1;
}



VOID NAVEGAR_MENU() {       //ROTINA DE NAVEGACAO ENTRE OPCOES DO MENU

  WHILE (TRUE) {
    EXECUTA_CTRL();     //CHAMA ROTINA QUE EXECUTA CONTROLE PID EM BACKGROUND

    IF      (INPUT(MENOS) && MENU<4) { DELAY_MS(200); MENU+=1; FLAG1=1; }       //SELECIONA OPCAO DO MENU
PRINCIPAL
    ELSE IF (INPUT(MAIS) && MENU>1)  { DELAY_MS(200); MENU-=1; FLAG1=1; }
    ELSE IF (INPUT(OK)) { WHILE (INPUT(OK)) { }; DELAY_MS(200); BREAK; }

    IF (FLAG1==1) {     //REIMPRIME OPCAO SELECIONADA NA TELA
      FLAG1=0;
      SWITCH (MENU) {
        CASE 1: PRINTF (LCD_PUTC, "\F(X)SP   ( )START\N( )PID  ( )INFO"); BREAK;
        CASE 2: PRINTF (LCD_PUTC, "\F( )SP   (X)START\N( )PID  ( )INFO"); BREAK;
        CASE 3: PRINTF (LCD_PUTC, "\F( )SP   ( )START\N(X)PID  ( )INFO"); BREAK;
        CASE 4: PRINTF (LCD_PUTC, "\F( )SP   ( )START\N( )PID  (X)INFO"); BREAK;
      }
    }
  }
}



VOID START() {                      //ROTINA QUE REALIZA VERIFICACAO E INICIA CONTROLE PID
    IF (((KP!=KP_INICIAL)||(TI!=TI_INICIAL)||(TD!=TD_INICIAL))&&(FLAG4==1)) {     //VERIFICA SE VALORES DE PI FORAM
ALTERADOS ENQUANTO CONTROLE ESTAVA EXECUTANDO
      FLAG4=0;
      SPOINT_INICIAL=0, SPOINT_TEMP=0, SPOINT=0;       //SE PID ALTERADO, BUSCA VALOR DE ZERO DEFINIDO
COMO SET POINT

      PRINTF (LCD_PUTC, "\F  PID UPDATED!\NWAIT ZERO LEVEL");  //PEDE PARA USUARIO AGUARDAR ATE BUSCAR
ZERO
      WHILE (MEDIA_NIVEL >= ZERO) { EXECUTA_CTRL(); }

      PRINTF (LCD_PUTC, "\F  READY!\N     PRESS OK");           //QUANDO PRONTO NIVEL ZERO LIBERA CONTROLE
PARA USUARIO
      WHILE (!INPUT(OK)) { EXECUTA_CTRL(); }; WHILE (INPUT(OK)) { }; DELAY_MS(200);

    }

    SPOINT = SPOINT_INICIAL;     //SALVA NOVOS VALORES PARAMETRIZADOS NAS VARIAVEIS PARA CONTROLE
    KP = KP_INICIAL; TI = TI_INICIAL; TD = TD_INICIAL;

    DELAY_MS(200);
}




////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// PROGRAMA PRINCIPAL
//



VOID MAIN () {
  LCD_INIT();                       //INICIALIZAÇÃO O LCD
  LCD_SET_CGRAM_CHAR(0,INICIO);          //SALVA CARACTERE CRIADO INICIO BARRA DE UPDATED
  LCD_SET_CGRAM_CHAR(1,MEIO);            //SALVA CARACTERE CRIADO MEIO BARRA DE UPDATED
  LCD_SET_CGRAM_CHAR(2,FIM);             //SALVA CARACTERE CRIADO FIM BARRA DE UPDATED

  DELAY_US(15);
  SETUP_TIMER_1(T1_INTERNAL|T1_DIV_BY_1);   //CONFIGURA TIMER1 PARA USO PELO SENSOR
  SETUP_TIMER_2(T2_DIV_BY_4,255,1);        //CONFIGURA TIMER2 PARA USO NO PWM DO PIC
  SETUP_CCP1(CCP_PWM);                  //CONFIGURA PMW DO PIC

  ANIMACAO_ABERTURA();                  //CHAMA ANIMAÇÃO DE ABERTURA
```

```
  SPOINT = 0;                        //CONTROLE PID BUSCA ZERO INICIALMENTE

  WHILE (TRUE) {
    NAVEGAR_MENU();              //ROTINA DE NAVEGACAO ENTRE OPCOES DO MENU

    WHILE (TRUE) {
      IF (MENU==1)     { CONFIGURASP(); FLAG1=1; BREAK; }     //SELECIONA CONFIGURAR SETPOINT (SP)
      ELSE IF (MENU==2) { BREAK; }                        //SELECIONA START PARA INICIAR CONTROLE PID SEGUNDO SP
CONFIGURADO
      ELSE IF (MENU==3) { WHILE (TRUE) { CONFIGURAP(); CONFIGURAI(); CONFIGURAD(); IF (FLAG2==3) { BREAK; } };
FLAG1=1; FLAG2=0; BREAK; }    //SELECIONA CONFIGURAR PID
      ELSE IF (MENU==4) { INFO(); BREAK; }                 //SELECIONA INFORMACOES DO ALUNO
    }

    IF (MENU==2) {
        START();    //CHAMA ROTINA QUE VERIFICA SE PID ALTERADO ENQUANTO CONTROLE EXECUTAVA

        WHILE (TRUE) {
          EXECUTA_CTRL(); IF (INPUT(VOLTA)) { BREAK; };   //PEGA VALORES DE NIVEL EM SEQUENCIA PARA GERAR
DELAY QUE ATUALIZA DISPLAY
          EXECUTA_CTRL(); IF (INPUT(VOLTA)) { BREAK; };    //VERIFICA CONSTANTEMENTE SE BOTAO VOLTA FOI
PRESSIONADO
          EXECUTA_CTRL(); IF (INPUT(VOLTA)) { BREAK; };
          EXECUTA_CTRL(); IF (INPUT(VOLTA)) { BREAK; };
          PRINTF(LCD_PUTC, "\FSP:%4.1F   E:%4.1F\N N:%4.1F  PWM%LU", (SPOINT), (ZERO + SPOINT - MEDIA_NIVEL),
(MEDIA_NIVEL - ZERO), (INT16)SINAL_CTRL);
        }; WHILE (INPUT(VOLTA)) { }; DELAY_MS(200);

        FLAG1=1; FLAG4=1;    //INDICA QUE DEVE REIMPRIMIR TELA  DE MENU PRINCIPAL E VERIFICAR SE PID FOI
ALTERADO
    }
  }
}
```