UNIVERSIDADE LUTERANA DO BRASIL

PRÓ-REITORIA DE GRADUAÇÃO

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Gilmar José Zwirtes

SISTEMA DE CONTROLE DA POSIÇÃO DE UM FLUTUADOR IMPLEMENTADO EM PROCESSADOR DE 32 BITS

Canoas, Dezembro de 2011

Gilmar José Zwirtes

SISTEMA DE CONTROLE DA POSIÇÃO DE UM FLUTUADOR IMPLEMENTADO EM PROCESSADOR DE 32 BITS

Trabalho de Conclusão de Curso apresentado ao Departamento de Engenharia Elétrica da ULBRA como um dos requisitos obrigatórios para a obtenção do grau de Engenheiro Eletricista

Departamento:

Engenharia Elétrica

Área de Concentração

Sistemas de Controle.

Professor Orientador:

MSc. Eng. Eletr. Miriam N. Cáceres Villamayor – CREA-RS: 067.231-D

Professor Co-Orientador:

MSc. Eng. Eletr. Augusto Alexandre Durgante de Mattos – CREA-RS: 088.003-D

Canoas, Dezembro de 2011

FOLHA DE APROVAÇÃO

Nome do Autor: Gilmar José Zwirtes

Matrícula: 041017074-7

Título: Sistema de Controle da Posição de um Flutuador Implementado em Processador de 32 Bits

Trabalho de Conclusão de Curso apresentado ao Departamento de Engenharia Elétrica da ULBRA como um dos requisitos obrigatórios para a obtenção do grau de Engenheiro Eletricista

Professor Orientador:

MSc. Eng. Eletr. Miriam N. Cáceres Villamayor

CREA-RS: 067.231-D

Professor Co-Orientador:

MSc. Eng. Eletr. Augusto Alexandre Durgante de Mattos

CREA-RS: 088.003-D

Banca Avaliadora:

MSc. Eng. Eletr. Paulo César Cardoso Godoy

CREA-RS: 011.6822-D

Conceito Atribuído (A-B-C-D):

MSc. Eng. Eletr. André Luis Bianchi

CREA-RS: 089.197-D

Conceito Atribuído (A-B-C-D):

Assinaturas:

Autor Gilmar José Zwirtes	Orientadora Miriam N. Cáceres Villamayor
Avaliador Paulo César Cardoso Godoy	Avaliador André Luis Bianchi

Relatório Aprovado em:

DEDICATÓRIA

Dedico a minha esposa e aos meus pais.

AGRADECIMENTOS

Agradeço primeiramente a Deus por me proporcionar a vida e a saúde necessárias para poder concluir mais uma etapa da minha vida.

Agradeço imensamente a minha esposa, Simone Fink Zwirtes, pela paciência, carinho, compreensão e pelo apoio incondicional, incentivando para continuar sempre, mesmo com todas as adversidades e momentos difíceis durante estes anos de muita dedicação aos estudos.

A minha formação como profissional não poderia ter sido concretizada sem o apoio dos meus queridos pais Guido Arthur e Jacinta C. Zwirtes, que, no decorrer desta trajetória, proporcionaram-me, além de muito carinho e amor, os verdadeiros valores da vida.

Aos meus irmãos e familiares que sempre me apoiaram.

A todos que colaboraram direta ou indiretamente na elaboração deste trabalho, o meu reconhecimento.

Aos professores Augusto A. D. de Mattos, Andre L. Bianchi e Paulo C. C. Godoy pelos ensinamentos, conselhos e pela dedicação na orientação deste trabalho e aos professores da ULBRA pelo estímulo, em especial, às professoras Marília Amaral da Silveira e Miriam Noemi Cáceres Villamayor pela dedicação e esforço pessoal proporcionado principalmente durante o período de estágio e Trabalho de Conclusão de curso em Engenharia Elétrica.

Aos colegas Ely Cristófoli, Carlos Alberto Alves, Ricardo Masiel, Everton Bandeira pelas valiosas sugestões durante o curso de graduação.

Ao Professor Luis Fernando Espinosa Cocian pelas orientações e disposição para atendimento a todos os alunos.

A todos vocês, meu muito obrigado.

EPÍGRAFE

"A humildade é a única sabedoria verdadeira pela qual nós preparamos nossas mentes para todas as possíveis mudanças da vida."

George Arliss

RESUMO

Zwirtes, Gilmar José, Estudo e Desenvolvimento de um Sistema de Controle da Posição de um Flutuador Implementado em Processador de 32 Bits. Trabalho de Conclusão de Curso em Engenharia Elétrica - Departamento de Engenharia Elétrica. Universidade Luterana do Brasil. Canoas, RS. 2011.

O objetivo do projeto é o desenvolvimento de um sistema de controle da posição de um flutuador através de um controlador PID, implementado em um kit didático contendo um processador LPC 2138 de 32 bits núcleo ARM7 em linguagem C. O *hardware* do projeto é composto por um sistema mecânico acoplado a um tubo acrílico transparente, um flutuador, um ventilador de corrente contínua, um sensor de posição ultrasônico e circuitos de acionamento. Para sintonizar o sistema de controle foram usadas as regras de Ziegler-Nichols. Foram realizados diversos testes no controlador com ajustes P, PI e PID, possibilitando o controle da posição do flutuador suspenso no tubo pela variação de fluxo de ar ascendente, que empurra o flutuador para cima contrabalanceando a força descendente da gravidade. A implementação e os testes realizados possibilitaram o controle da posição do flutuador obtendo resultados satisfatórios com erro de posição em regime permanente inferior a 6% e máximo *overshoot* de 5%.

Palavras chave: PID. ARM7. Controle Digital. LPC. Ziegler-Nichols.

ABSTRACT

Zwirtes, Gilmar José, Study and Development of a Position Control System Implemented on a float 32-bit processor. Completion of course work in Electrical Engineering - Electrical Engineering Department. Lutheran University of Brazil. Canoas, RS. 2011.

The project's goal is to develop a system to control the position of a floater using a PID controller implemented in a educational kit containing a LPC 2138 processor 32-bit ARM7 core in C language. The *hardware* design consists of a mechanical system coupled to a transparent acrylic tube, a floater, a DC fan, an ultrasonic position sensor and drive circuits. To tune the control system were used Ziegler Nichols rules. Several tests were performed with the controller settings P, PI and PID controllers, controlling the position of the floater suspended inside the tube. A flow of ascending air pushes the floater up compensating the force of gravity. Implementation and tests have done and the control of the position of the float achieved satisfactory results with the position error in steady state under 6% and maximum *overshoot* under 5%.

Keywords: PID. ARM7. Digital Control. LPC. Ziegler-Nichols.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

A/D: Analógico / Digital

CLP: Controladores Lógicos Programáveis

I/O: Entrada / Saída

g: grama

Cm: Centimetros

LCD: Display Líquido

M_o: máximo valor alcançado (*Overshoot*)

K_p: Ganho Proporcional

T_i: Constante de Tempo Integrativa

T_d: Constante de Tempo Derivativa

P: Proporcional

PI: Proporcional Integral

PID: Proporcional Integral Derivativo

R: Resistência

t_r: tempo de subida (*Rise Time*)

t_s: tempo de estabilização (*Setting Time*)

SUMÁRIO

1. INTRODUÇÃO

O crescente avanço tecnológico nas mais diversas áreas da indústria tem se mostrado, nos últimos anos, surpreendente. A utilização da automação nas indústrias tem sido cada vez maior, proporcionando um aumento na qualidade e quantidade da produção e, ao mesmo tempo, proporcionando uma oferta de preços mais atrativos.

A utilização da automação aumenta a eficiência, tornando as empresas competitivas num mercado cada vez mais exigente. Para fazer frente à concorrência procura-se aumentar a produtividade, reduzir custos de produção e aumentar a qualidade dos produtos oferecidos. Ao mesmo tempo, para atender às exigências de diversidade do mercado consumidor e a gradativa redução da vida útil dos produtos, procura-se ampliar a flexibilidade na utilização dos sistemas produtivos [1].

O avanço da Automação está ligado, em grande parte, ao avanço da microeletrônica que se deu nos últimos anos e pouco a pouco, invadiu os setores produtivos das indústrias, propiciando a automação. O processo de automação não atinge apenas a produção em si, substituindo o trabalho braçal por robôs e máquinas computadorizadas, mas permite enormes ganhos de produtividade ao integrar tarefas distintas com a elaboração de projetos, o gerenciamento administrativo e a produção.

A automação industrial pode ser definida como um conjunto de técnicas destinadas a tornar automáticos vários processos na indústria, substituindo o trabalho muscular e mental do homem por equipamentos diversos.

O conceito de automação inclui a idéia de usar a potência elétrica ou mecânica para acionar algum tipo de máquina, sendo necessário acrescentar algum tipo de inteligência para executar tarefas de modo mais eficiente e com vantagens econômicas e de segurança.

Existem, basicamente, dois segmentos da automação industrial segundo a manipulação das variáveis a serem controladas, podendo ser do tipo analógicas ou de tempo contínuo. Presente na maioria dos processos industriais, tem-se um controle de processo do tipo contínuo (Controle de Processos, Controle Regulatório); caso as variáveis sejam do tipo discretas, ou digital, tem-se um Controle do tipo discreto (Controle Discreto) [1].

O controle do tipo discreto, voltado aos processos digitais, teve seu início marcado pela utilização de dispositivos eletromecânicos a relés e com o advento dos dispositivos microprocessados, vieram os Controladores Lógicos Programáveis (CLPs), cuja forma básica de programação é oriunda da lógica de programação dos diagramas elétricos a relés.

Já o controle do tipo analógico desenvolveu-se, com o surgimento dos amplificadores operacionais, por meio do qual as ações de controle eram implementadas, assim os controladores de processos contínuos evoluíram juntamente com a microeletrônica e passaram a utilizar circuitos mais complexos, microprocessados, de forma a poderem utilizar poderosos recursos e efetuarem técnicas de controle dos mais diversos tipos, tais como: Proporcional + Integral + Derivativo (PID), PID adaptativo (não linear), Lógica Fuzzy (lógica nebulosa), Preditivo, entre outros, surgindo também várias técnicas de sintonia de controladores, [2].

Atualmente a maioria dos controladores industriais utilizam técnicas de controle, como o controle On-Off ou PID que normalmente utilizam sintonia automática.

Visando esta tendência nas indústrias, o objetivo principal deste trabalho é desenvolver um sistema de controle da posição de um flutuador através de um controlador PID com sintonia automática. O processador escolhido para implementar o *software* deste projeto é o LPC 2138 de núcleo ARM7 – 32 bits.

O *hardware* do projeto é composto por um sistema mecânico acoplado a um tubo acrílico transparente, um flutuador, um ventilador DC Fan, um sensor de posição ultrasônico e circuitos de acionamento. A implementação do sistema, o *software*, resultados práticos obtidos, bem como as considerações e propostas de melhoria, serão apresentados nos capítulos seguintes.

2. REFERENCIAL *TEÓRICO*

Este capítulo descreve a fundamentação teórica dos sistemas de controle PID, tanto analógica como digital, bem como suas funcionalidades, apresentando a função de um controlador contínuo e cada uma das variáveis (proporcional, diferencial e integral), além da realização do controle com a união desses valores dentro do algoritmo. Também está descrito o método de Ziegler-Nichols e um modelo de linearização que permite a aplicação do controle linear para o flutuador, princípios estes indispensáveis para a compreensão e o desenvolvimento do *hardware* e *software* do sistema de controle. São apresentadas ainda as principais características de funcionalidade, bem como o tipo de sensor utilizado na planta.

2.1. Modelo de Linearização do Sistema

Os modelos matemáticos para uma bola em um tubo segundo Theodore P. Pavlic (http://www2.ece.ohio-state.edu/~passino/ee758.html) são obtidos de forma experimental utilizando uma bola flutuando em um tubo, a qual é impulsionada para cima e para baixo pelo empuxo gerado por um ventilador, sendo possível representar a performance da bola por um modelo simples através da linearização *feedback,* permitindo a aplicação do controle linear (por exemplo, controle PID).

Sobre uma hipótese de coeficiente de elevação, pode-se assumir que o empuxo é proporcional ao quadrado da tensão aplicada ao ventilador conforme Equação 2.1.

$$T = C.v_{in}^2$$

(Equação 2.1)

Onde T é o empuxo gerado por um ventilador controlado pela tensão *Vin,* assim desde que a impedância de saída da tensão *Vin* gerada pelo amplificador seja

suficientemente baixa, pode-se assumir que os efeitos da resistência elétrica são desprezíveis.

Um modelo de ponto material excessivamente simples para a bola é representado pela Equação 2.2 e mostrado na Figura 2.1, onde o empuxo T controla a bola de massa m (peso W) com aceleração a para cima.

$$\frac{\mathrm{F} = \mathrm{T} - \mathrm{W} = C.v_{in}^2 - \mathrm{m}\,.\,\mathrm{g}}{\mathrm{F} = \mathrm{m}\,.\,\mathrm{a}}$$

(Equação 2.2)

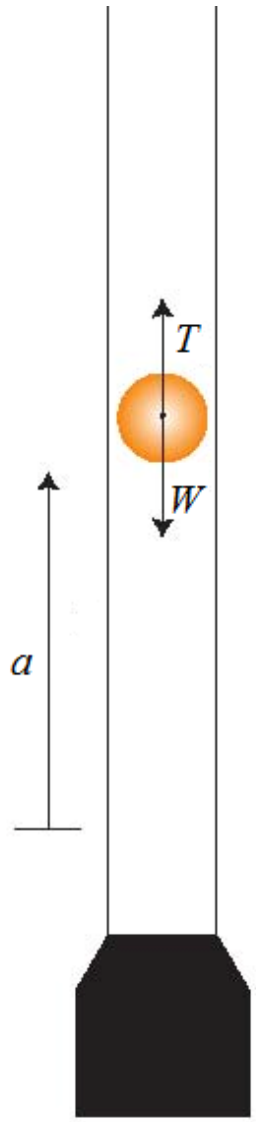


Figura 2-1 – Ilustração de ponto material de uma bola.

Neste modelo, a bola de massa *m* é forçada para cima pelo empuxo *T* e puxada para baixo pela gravidade e o peso representado pela Equação 2.3, assim a força resultante sobre a bola para cima é *T – W*, a qual é igual a *m.a* pela segunda lei de Newton, onde *a* é a magnitude da aceleração ascendente da bola, sendo que seu movimento é modelado conforme Equação 2.4.

$$W = m \,.\, g$$

(Equação 2.3)

$$\overbrace{m.a}^{F} = \overbrace{C.v_{in}^2}^{T} - \overbrace{m.g}^{W}$$

(Equação 2.4)

Mas como:

$$a = \dot{v} = \ddot{x}$$

(Equação 2.5)

Na modelagem matemática representada na Equação 2.5, ***x*** é a posição relativa da bola. Usando a posição ***x*** como uma saída, a Equação 2.4 pode ser representada conforme Equação 2.6.

$$\begin{cases} \dot{x} = v \\ \dot{v} = \dfrac{C}{m} v_{in}^2 - g \end{cases}$$

(Equação 2.6)

Para simplificar, pode-se forçar *Vin* ≥ 0, obtendo desta forma a Equação 2.7, onde *u* ≥ *0*.

$$V_{in} = \sqrt{u}$$

(Equação 2.7)

Assim:

$$\begin{cases} \dot{x} = v \\ \dot{v} = -g + \dfrac{C}{m} u \end{cases} \qquad \left(\text{i.e.}, \alpha(x,v) \triangleq -g \quad \text{e} \quad \beta(x,v) \triangleq \frac{C}{m}\right)$$

(Equação 2.8)

Este sistema já está na forma normal. Assim, sem nenhuma transformação de coordenadas, fica claro que este sistema de segunda ordem tem grau relativo 2 quando a posição *x é* usada como uma saída.

$$u = \frac{m}{C}(w + g) \qquad \left(\text{i.e.}, u = \frac{w - \alpha(x,v)}{\beta(x,v)}\right)$$

(Equação 2.9)

O controle com $w \geq -g$, é possível converter a Equação 2.8 através do sistema duplo integrador LTI, conforme mostrado na Equação 2.10.

$$\begin{cases} \dot{x} = v \\ \dot{v} = w \end{cases}$$

(Equação 2.10)

O parâmetro *g* é conhecido 9,80665 m/s², o parâmetro *m* pode ser medido (por ex. com uma escala), e o parâmetro *C* pode ser estimado do sistema (por ex. analisando a aceleração da bola quando a entrada *u* é constante). Assim obtém-se a Equação 2.11.

$$V_{in} = \sqrt{\frac{m}{C}(w + g)} \qquad \text{com} \quad w \geq -g$$

(Equação 2.11)

Então o controle lineariza o sistema $w - x$ e não necessita de *feedback* neste caso. Naturalmente, as aproximações do ponto material e coeficiente de elevação podem ser muito pequenos e desta forma desprezados.

2.2. Sistemas de Controle na Indústria

2.2.1. Conceitos Gerais

Os controladores PID (Proporcional - Integral - Derivativa) são bastante utilizados nas indústrias para controle de sistemas em geral, como dito anteriormente. De fato, na indústria, 95% das malhas de controle utilizam controladores assim, em sua maioria apenas a parte proporcional e integral é utilizada [2].

O controlador PID é um diferencial no controle de processos em função de ser facilmente programável, necessitar de poucos parâmetros de ajuste e pouco suporte de recursos para operação. Este controlador possui um conjunto de

equações que são aplicadas a fim de se produzir ações de controle através de um atuador sobre um processo. A Figura 2.2 apresenta o diagrama em blocos de um sistema de controle [2].

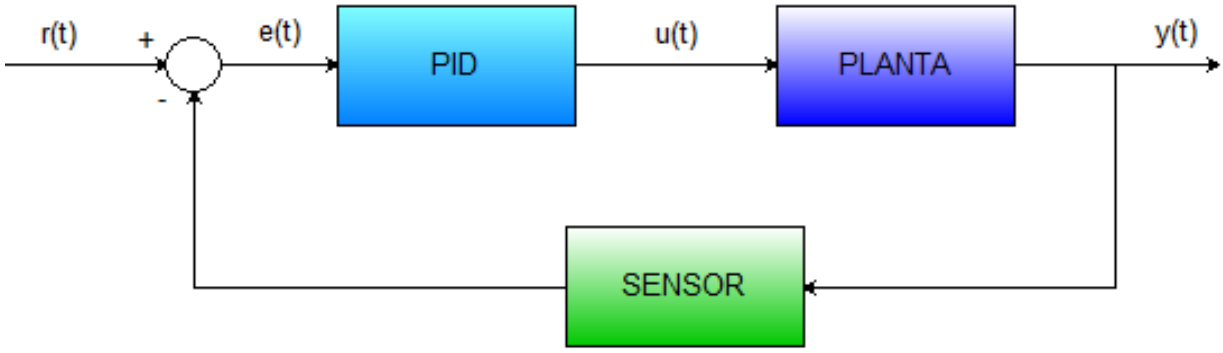


Figura 2-2 – Diagrama em blocos de um sistema de controle.

Uma versão clássica do controlador PID pode ser apresentada como na equação 2.12 [2].

$$U(t) = K_P.e(t) + \frac{K_P}{T_I}.\int_0^t e(t)dt + K_P.T_d\frac{de(t)}{dt}$$

(Equação 2.12)

Onde *e* representa o erro entre a variável medida e a variável controlada e *u* é a variável de controle que é a soma dos termos: proporcional, integral do erro e derivada do erro. Os parâmetros de controle considerados para este tipo de controlador são o ganho do controlador Kp, a constante de tempo integrativa Ti e a constante de tempo derivativa Td.

Esta equação correlaciona os termos os quais executam as ditas ações de controle, proporcional, integral e derivativa. A união destes três modos básicos de controle contínuo, vistos a seguir, produz um dos mais eficientes algoritmos de controle já desenvolvidos, o controlador PID (Controle Proporcional + Integral + Derivativo)[2]. Pois ele concilia simplicidade e atendimento às necessidades de controle para a grande maioria dos casos industriais. Contudo, a maneira como ocorre esta combinação pode variar significativamente, repercutindo em alterações no algoritmo dos controladores PID de diferentes fabricantes.

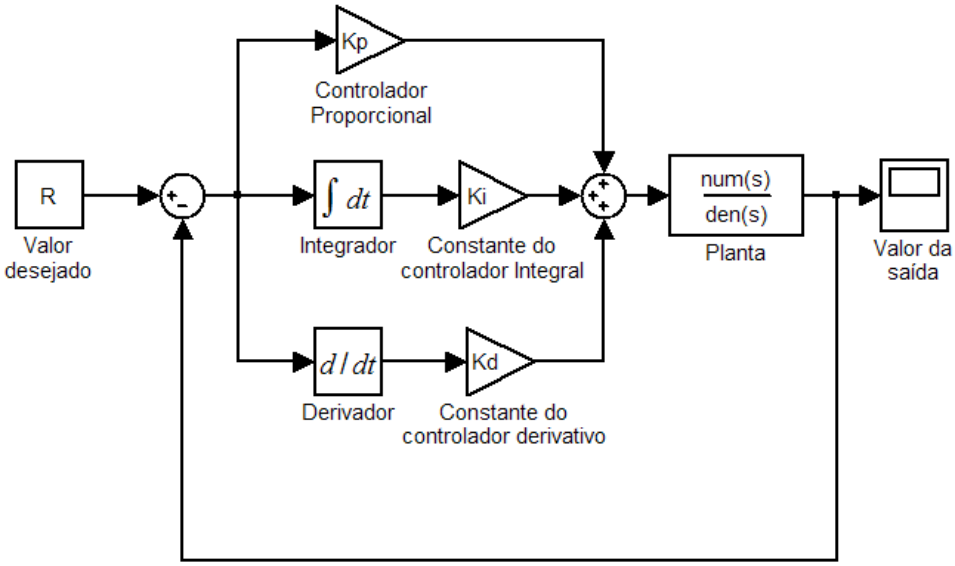


Figura 2-3 – Sistema em malha fechada com controlador PID.

Algumas alterações constituem em melhorias no algoritmo, outras, entretanto, remontam aos antigos mecanismos pneumáticos [4]. Apesar de existirem muitas formas, duas delas são mais difundidas e, assim, merecem especial atenção: a forma série ou interativa e a paralela ou não interativa, assim muitas parametrizações constituem em simples variação destas formas.

Um bom referencial teórico para métodos de ajuste de controladores PID pode ser encontrada em livros como ASTRÖM e HÄGGLUND (1995), NORMAN S. NISE e OGATA (1998).

2.2.2. Modo Proporcional

O modo de controle proporcional pode ser considerado como uma evolução do modo de controle de duas posições. A ação de controle gerada pelo modo proporcional é diretamente proporcional a sua entrada, ou seja, o sinal de erro em função do tempo, como mostrado na Equação 2.13.

$$U(t) = K_P . e(t) + u_0$$

(Equação 2.13)

A saída de um controlador proporcional pode assumir qualquer valor desde que compreendido entre os limites de saída máxima e mínima, em função do *off-set* verificado. Na Figura 2.4 ilustra-se um sistema característico em malha fechada utilizando uma dessas partes, ou seja, utilizando um controlador proporcional [11].

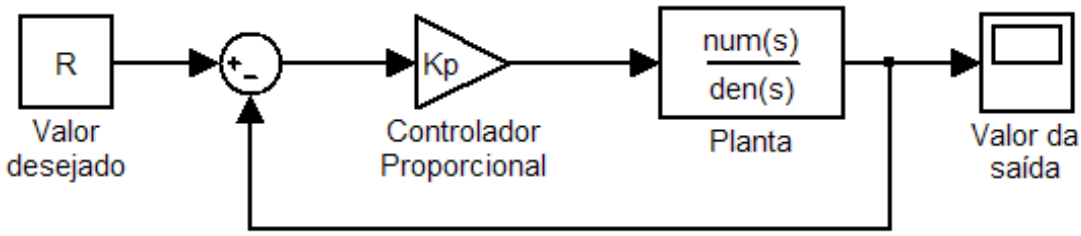


Figura 2-4 – Sistema de controle com controlador proporcional.

A Figura 2.5 mostra a relação entre o sinal de erro e a ação de controle gerada pelo modo de controle proporcional. Excluída a faixa de saturação da variável manipulada (sinal de erro fora da banda proporcional), cada valor de erro tem um único valor correspondente de ação de controle e vice-versa.

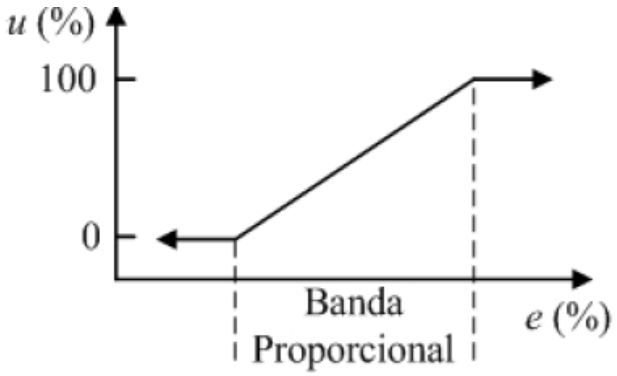


Figura 2-5 – Ação de controle em função do sinal de erro gerada pelo controlador proporcional [13].

Como o ganho do controlador é dado pela inclinação da reta sobre a banda proporcional percentual, a relação entre ambos é dada pela Equação 2.14. Esta representação é genérica para o caso onde a saída do controlador varia entre 0 a 100%, sendo que para casos específicos onde isto não ocorre, esta relação não é válida.

$$BP = \frac{100\%}{K_p}$$

(Equação 2.14)

Neste caso, o controlador é apenas um amplificador com ganho constante, quanto maior o erro, maior a ação de controle gerada. Assim, ele prevê um rápido ajuste da variável manipulada, tornando mais rápida a dinâmica do processo.

A principal desvantagem deste modo é que ele apresenta erro em regime permanente. O erro em regime permanente diminui com o aumento do ganho proporcional *Kp,* no entanto isto diminui a faixa correspondente à banda proporcional, tornando o controlador mais oscilatório.

Cabe ressaltar que o controlador liga-desliga pode ser definido como sendo um controlador proporcional no limite onde a banda proporcional tende a zero.

2.2.3. Modo Integral

A ação de controle gerada pelo modo integral é proporcional à integral do sinal de erro no tempo, como mostrado na Equação 2.15. O grande benefício da sua utilização é a eliminação do erro em regime permanente, contudo, ela reduz a estabilidade da malha de controle [11].

$$U(t) = K_I . \int_0^t e(t)dt + u_0$$

(Equação 2.15)

Esta equação mostra que a ação de controle depende do histórico do erro, desde que o processo de integração foi iniciado ($t = 0$) até o instante atual. Na Figura 2.6 apresenta-se um exemplo ilustrativo de um sistema com um controlador PI.

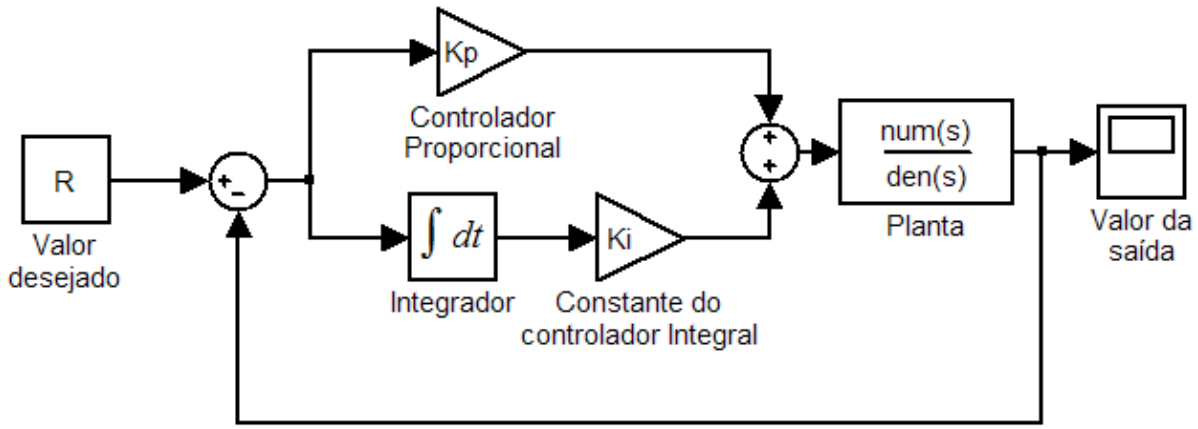


Figura 2-6 – Sistema de controle com controlador PI.

A ação integral também pode ser vista como um mecanismo que atualiza automaticamente o valor base do controlador com ação proporcional. Devido a isto, ela foi originalmente denominada de *reset action* (Controle com *Reset*) [2].

Os controladores com ação integral (Controle com *Reset*) são considerados de ação dinâmica, pois a saída dos mesmos é uma função do tempo da variável de entrada.

2.2.4. Modo Derivativo

A ação de controle gerada pelo modo derivativo é proporcional à taxa de variação do sinal de erro, ou seja, a sua derivada no tempo, conforme Equação 2.16, ela estima a tendência de aumento ou diminuição do erro futuro.

Assim, este modo é capaz de aumentar a velocidade de correção do processo, pois atua de forma antecipatória quando são detectadas variações no sinal de erro, no entanto, este modo dificilmente é utilizado sozinho, mas associado com outros modos de controle [11].

$$U(t) = K_D . \frac{de(t)}{dt} + u_0$$

(Equação 2.16)

Na Figura 2.7 apresenta-se um exemplo ilustrativo de um sistema com um controlador PD.

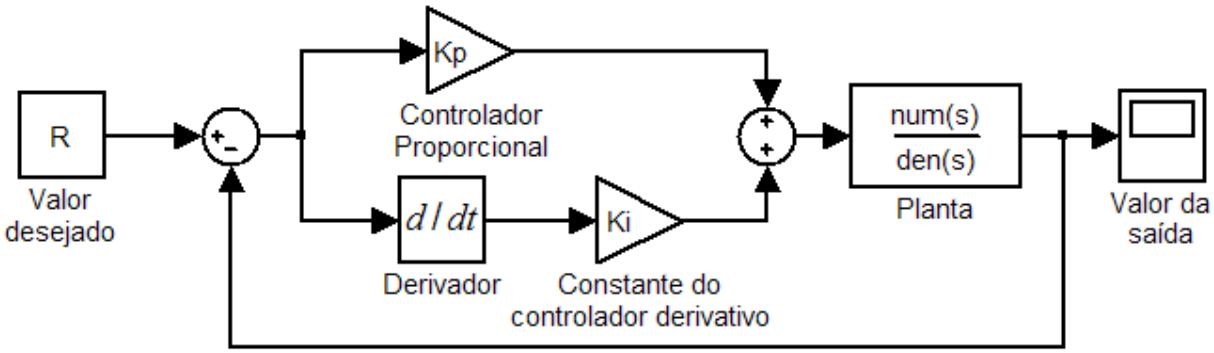


Figura 2-7 – Sistema em malha fechada com controlador PD.

2.3. *Método de Ziegler-Nicholds*

No projeto de um controlador, é desejável que existam alguns parâmetros que possam influenciar no desempenho do sistema. Estes parâmetros são normalmente definidos a partir de certas especificações.

Um bom método de ajuste de parâmetros leva em conta várias destas especificações de forma equilibrada, apesar da existência de vários métodos de ajuste, em muitos casos, o ajuste manual ainda é utilizado, onde parâmetros são

ajustados independentemente por tentativa-e-erro, ou com um procedimento básico para cada caso especifico. Com este procedimento, a ação derivativa normalmente não é utilizada, e o desempenho só é adequado para sistemas simples ou que não requeiram controle rigoroso.

No entanto, para sistemas complexos ou onde deseja-se um melhor desempenho, faz-se necessária a utilização de algum método sistemático para ajuste dos parâmetros de um controlador PID. Vários métodos de ajuste de controladores foram propostos nos últimos sessenta anos. Eles podem ser classificados em empíricos, analíticos ou obtidos através de algum tipo de otimização.

Existem métodos baseados em modelos do processo operando em malha aberta ou modelos em malha fechada, e existem ainda métodos no domínio da frequência. Dentre vários tipos de métodos, os mais difundidos são os métodos que apresentam regras de ajuste baseadas em um modelo aproximado do processo, representado normalmente por uma função de transferência de primeira ordem com tempo morto, conforme mostrado na Equação 2.17.

$$G = \frac{K}{\tau . s + 1} . e^{-\theta . s}$$

(Equação 2.17)

Os métodos de Ziegler-Nichols foram introduzidos já em 1942 e hoje são considerados clássicos [5]. Estes métodos continuam a ser largamente aplicados até hoje, mesmo em sua forma original, mas normalmente são utilizados em alguma forma modificada. Os dois métodos básicos de ajuste de Ziegler-Nichols visam obter uma mesma resposta pré-especificada para o sistema em malha fechada, e diferem no que diz respeito à natureza da informação sobre a dinâmica do processo que é exigida por cada um deles, as mesmas foram desenvolvidas para o ajuste de controladores P, PI e PID, colocados em cascata com o sistema ou processo controlado, conforme mostrado na Figura 2.8.

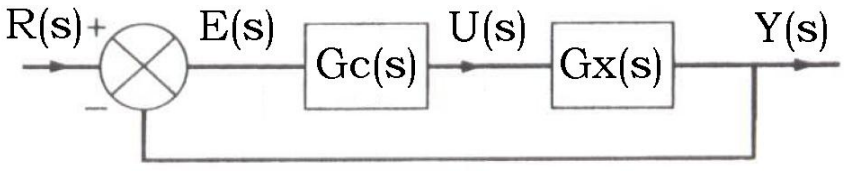


Figura 2-8 – Processo Ziegler-Nichols [13].

Onde:

R(s) - referência

E(s) - erro

Gc(s) – função de controle

U(s) - entrada do processo controlado

Gx(s) – processo controlado

Y(s) - resposta do processo

2.3.1 Método da Resposta ao Salto

O método da resposta ao salto consiste de equações derivadas a partir dos parâmetros da curva de reação [5]. Ela é obtida do sistema, quando o mesmo está em malha aberta. Pode-se ver na Figura 2.9a, a excitação do tipo salto e na figura 2.9b a resposta do sistema e como são obtidos os parâmetros de retardo e tempo.

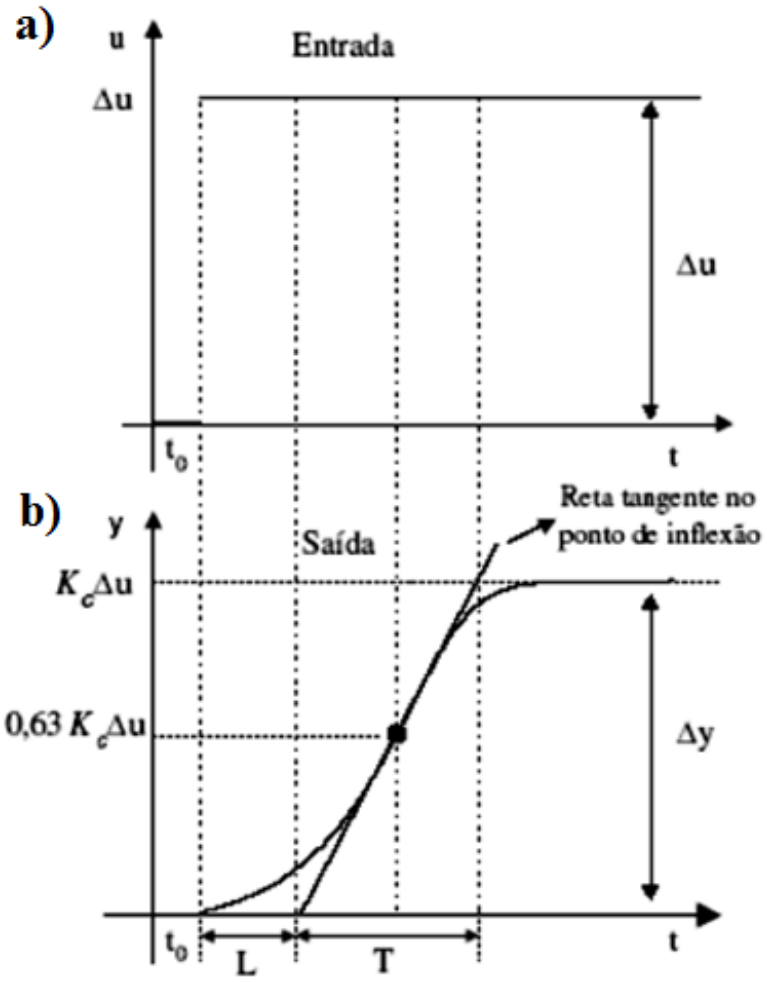


Figura 2-9 – Características da resposta ao salto do processo relevante para o ajuste de Ziegler-Nichols [13].

No entanto, sua aplicação é mais indicada para curva de reação na forma de um "S" e que atenda o seguinte critério: $0{,}1 < T/L < 1$. Para esta curva, os parâmetros, tempo de retardo L, e constante de tempo T, são determinados passando-se uma reta tangente no ponto de inflexão da curva, como ilustrado na Figura 2.9.

Tendo-se obtido experimentalmente os valores de L e T, pode-se recorrer à Tabela 2.1 para determinar os valores dos parâmetros do controlador PID.

Tabela 2-1 – Tabela de Ziegler e Nichols pelo método da resposta ao salto [3].

Tipo de Controlador	KP	Ti	Td
P	$\frac{1}{a}$	∞	0
PI	$\frac{0{,}9}{a}$	3 . L	0
PID	$\frac{1{,}2}{a}$	2 . L	$\frac{L}{2}$

Os valores da tabela 2.1 foram determinados de forma empírica com objetivo de obter uma resposta com amortecimento de 1/4 na resposta à referência para processos industriais típicos (um amortecimento de 1/4 leva a um valor máximo de *overshoot* de 25%). Enquanto a rejeição a perturbações muitas vezes apresenta um comportamento satisfatório, este amortecimento usualmente não é satisfatório na resposta à referência, causando em muitos casos um *overshoot* excessivo e baixa tolerância a variações na dinâmica do processo.

Este método limita-se a plantas de primeira ordem. Sistemas de segunda ordem, ou superior, por exemplo, não se enquadram nesta categoria. Por outro lado, este método baseia-se em identificação de formas de onda, o que pode ser problemático na prática, particularmente em aplicações com baixa relação sinal-ruído. Ainda assim, o método é utilizado em alguns processos industriais [2].

2.3.2 Método do Período Crítico

Se um processo é colocado em laço fechado com controle proporcional e o valor do ganho proporcional é aumentado progressivamente, em um determinado instante de tempo o processo oscilará. O ganho necessário para causar esta oscilação é chamado ganho crítico (K_{cr}) do processo e o período da oscilação observada é dito o seu período crítico (P_{cr}) [3].

Para aplicação deste método primeiramente devem ser ajustados os valores de T_i igual a infinito e T_d igual a zero. Utilizando somente a ação proporcional, aumenta-se o valor de K_p de zero a um valor crítico para o qual o sinal de saída apresente oscilações mantidas (se o sinal de saída não apresenta oscilações, quaisquer que sejam os valores de K_p, então este método não pode ser aplicado) [3].

Em consequência são determinados experimentalmente os valores de ganho crítico e o período crítico correspondente Figura 2.10. De posse do ganho crítico e do período crítico basta aplicar as fórmulas da Tabela 2.2, que apresenta as condições originalmente apresentadas por Ziegler e Nichols quando da proposta do método.

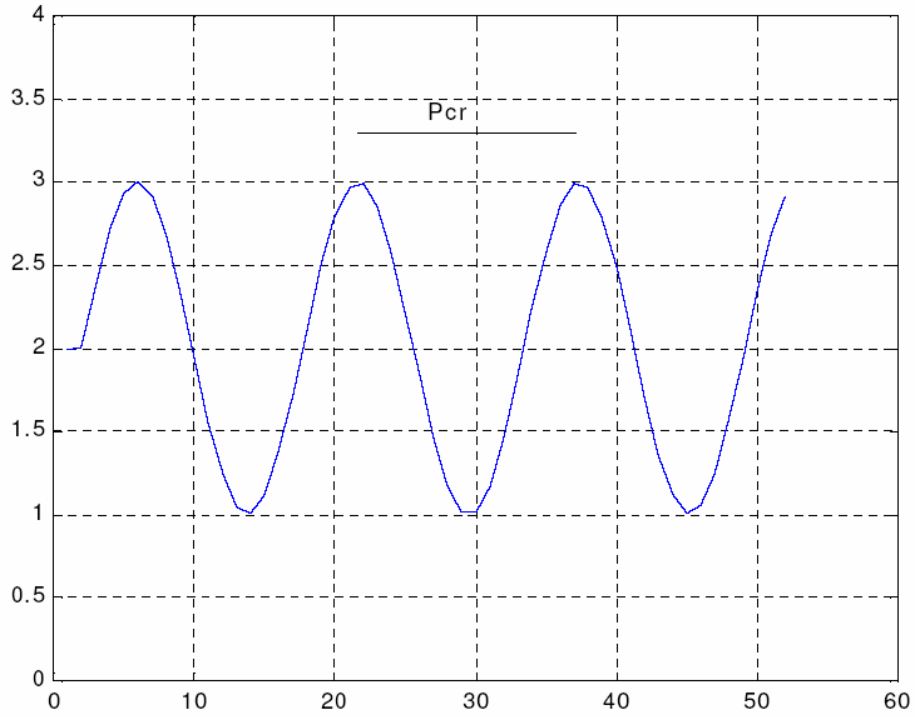


Figura 2-10 – Oscilações estáveis com períodos Pcr.

Tabela 2-2 – Fórmulas de Ziegler e Nichols para ajuste pelo método do período crítico [3].

Tipo de Controlador	KP	Ti	Td
P	0,5 . Kcr	∞	0
PI	0,45 . Kcr	$\frac{1}{1,2}$. Pcr	0
PID	0,6 . Kcr	0,5 . Pcr	0,125 . Pcr

O procedimento para obtenção do ganho crítico e do período critico visto até o momento é pouco eficiente por diversos motivos. Primeiramente, uma vez que o ganho deve ser aumentado de forma gradativa, o procedimento torna-se bastante

demorado. Em segundo lugar, é preciso ter de antemão alguma informação sobre a dinâmica do processo a fim de determinar o valor inicial do ganho e sua taxa de variação. Finalmente, a natureza linear da oscilação faz com que ela nunca seja sustentada, mas sempre amortecida ou instável.

Uma maneira muito mais eficiente de determinar estes parâmetros é o ensaio de realimentação com relé, que não sofre de nenhum dos problemas citados acima. Considerando que o processo esteja em laço fechado com um relé na realimentação, como na Figura 2.11. A figura 2.12 mostra a curva típica deste sistema [5].

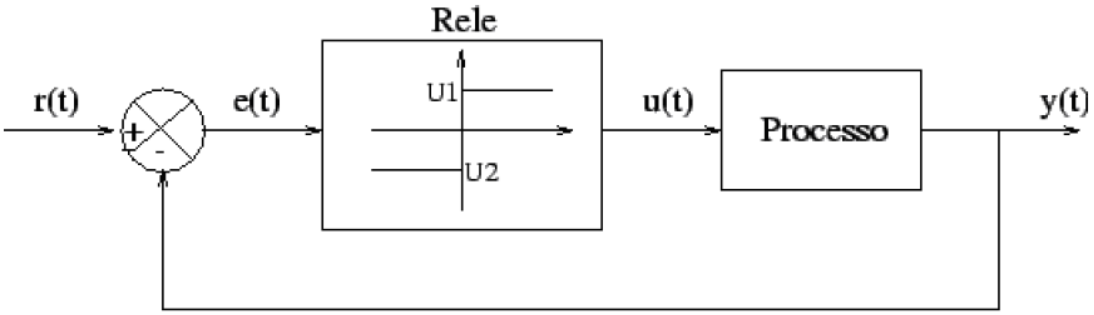


Figura 2-11 – Diagrama em bloco do controle com relé [5].

.

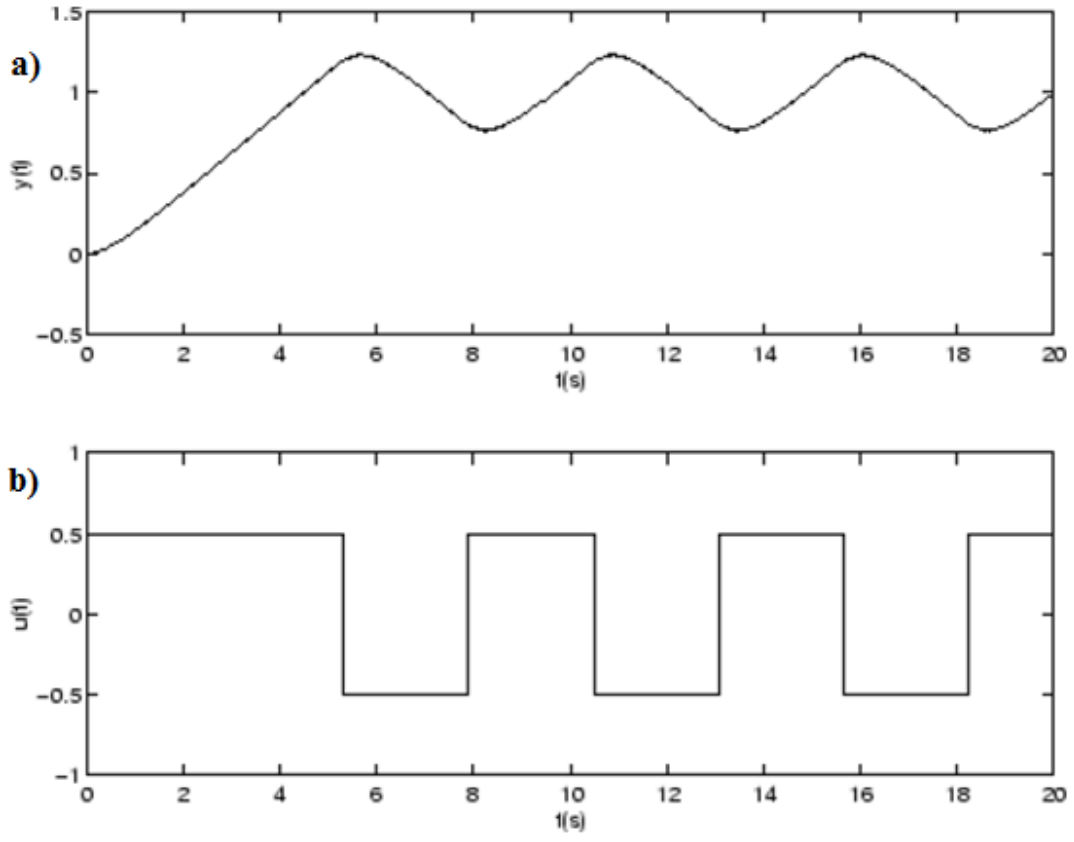


Figura 2-12 – Figuras a e b, representam a resposta típica do controle de realimentação com relé [5].

Em um processo de controle com relé, a variável controlada oscila em torno do valor de referência e a variável manipulada chaveia periodicamente entre seus dois valores possíveis. As características desta oscilação dependem das características do processo, mas também dos valores de u_1 e u_2 escolhidos para o controle de realimentação com relé [6].

Uma primeira característica da oscilação a ser analisada é sua simetria. Por simetria entende-se entre os tempos em que a variável manipulada fica em seu valor máximo e mínimo: a oscilação da variável controlada será simétrica com relação ao valor de referência. Se essa oscilação é simétrica, então seu período é igual ao período crítico do processo e o ganho crítico pode ser calculado pela equação 2.18.

$$K_{cr} = \frac{4.d}{\pi.A}$$

(Equação 2.18)

Onde:

k_{cr} = ganho crítico;

$d = u_1 - u_2$ (Figura 2.11);

A = amplitude pico-a-pico da oscilação observada na variável controlada.

2.4. Controle Digital

O controle digital caracteriza-se pelo uso de um computador, microcontrolador, ou microprocessador, que gera a lei de controle e exerce a função de controlador, conforme pode ser visto na Figura 2.13. Controladores digitais são flexíveis e as funções de controle podem ser facilmente modificadas. Leis de controle mais complexas também podem ser implementadas sem grandes dificuldades.

Figura 2-13 – Diagrama em Blocos do controlador Digital. Fonte: Nise, (2002).

Em um controle digital o sinal de saída e amostrado e convertido em uma sequência de pulsos expressos em um código numérico (código binário, por exemplo).

A função de transferência do controlador é convertida em uma equação de diferença, implementada como um programa no computador. A saída do computador por sua vez que é expressa também no mesmo código binário, e convertida para um sinal continuo. Esta saída e o sinal de atuação.

A implementação do controlador PID pode ser realizada com aproximações numéricas das derivadas e da integral que aparecem na lei de controle. Desta forma, é possível descrever cada uma das ações por uma equação de recorrência.

As equações de recorrência descrevem as operações matemáticas a serem programadas no microcontrolador ou no microcomputador onde será implementado o PID digital.

Sistemas de controle digital são utilizados quando um elevado grau de precisão é requerido. As vantagens com relação ao controle analógico são:

- Redução de custos.
- Flexibilidade para realizar mudanças no projeto.
- Imunidade a ruídos.

Por outro lado algumas desvantagens também se apresentam:

- Erros são introduzidos pelos processos de amostragem e quantização, e podem degradar o desempenho do sistema.
- O projeto pode se tornar mais complexo para compensar esta degradação.

2.5. Sensores

Os sensores são partes cruciais dos sistemas de controle, pois eles fornecem as informações necessárias sobre as quais as ações de controle são baseadas. Eles são os “olhos” dos controladores e, assim, qualquer problema ou defeitos nos instrumentos de medida terão impacto significativo na performance do controlador, alguns modelos de sensores podem ser vistos na figura 2.14.

Um dos problemas mais comuns é o ruído de medição, que pode mascarar a real tendência da curva do processo. Como o ruído é tipicamente dominado por altas frequências, na prática, ele acarreta um limite superior na largura de banda da malha [12]. Seus efeitos podem ser minimizados através da utilização de filtros específicos.

Figura 2-14 – Imagem ilustrativa de modelos de sensores.

Outra limitação bastante comum introduzida pelos instrumentos de medida surge do fato que eles possuem uma dinâmica própria. Isto pode introduzir uma defasagem adicional à resposta do sistema e, em alguns casos, ser até dominante.

Quando esta limitação precisa ser levada em conta no projeto do controlador, ela pode ser modelada como uma dinâmica de primeira ordem, com uma constante de tempo representativa do instrumento de medida (T_{im}), conforme mostrado na Equação 2.19.

Por simplificação, na maioria dos casos, esta dinâmica é inserida no modelo do processo na forma de um tempo morto adicional ao do próprio processo.

$$Y_m(s) = \frac{1}{T_{im}.s + 1}.Y(s)$$

(Equação 2.19)

Para medir distância é comum a utilização de sensores de ultra-som, cujo termo refere-se às ondas com frequências mais altas que a audibilidade humana, com valor máximo em torno de 18kHz. Já o sensor ultrasônico transmite e emite seus sons não audíveis na frequência usual de ~20 a 300Khz.

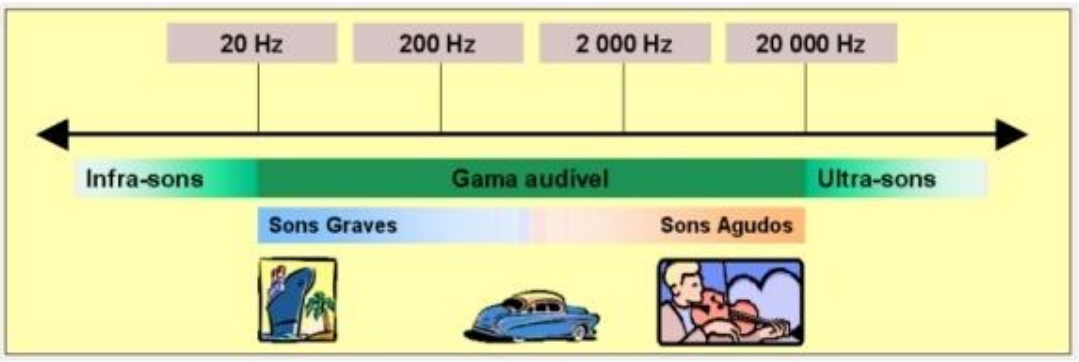


Figura 2-15 – Imagem ilustrativa das faixas dos Sons.
Fonte: http://www.mecatrons.xpg.com.br/Artigos/ArtigoVIII/artigoVIII.html.

As ondas ultra-sônicas obedecem às leis básicas da acústica, porém possuem comprimento de onda curto, o que reduz a difração e a dispersão pelos obstáculos e facilita o direcionamento da emissão.

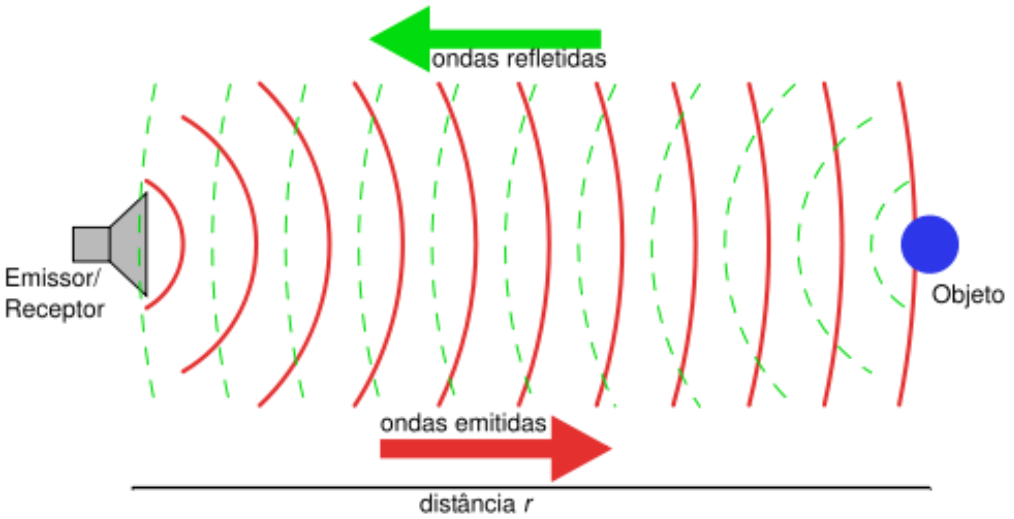


Figura 2-16 – Características do sinal de um sonar.
Fonte: http://www.mecatrons.xpg.com.br/Artigos/ArtigoVIII/artigoVIII.html.

A velocidade do som depende do meio que ele atravessa. Em geral, a velocidade do som é proporcional (a raiz quadrada da diferença) da "dureza" do meio e sua densidade. Esta é uma propriedade fundamental do meio. As propriedades físicas e a velocidade do som mudam de acordo com as condições do ambiente. A velocidade do som no ar depende da temperatura. No ar, a velocidade é de aproximadamente 345 m/s, na água de 1500 m/s e em uma barra de aço, de 5000 m/s. Um uso comum do ultrasom é para medição de distâncias, isto também é chamado de sonar. Sonar trabalha de um modo similar ao radar. Um pulso ultrasônico é gerado em uma direção. Se existir um objeto no caminho deste pulso, o pulso é refletido de volta para o emissor como um eco, e é detectado. Medindo e

diferença de tempo entre a emissão do pulso e a recepção do eco conforme figura 2.16, é possível determinar a distância do objeto. Os morcegos utilizam uma variação deste método para detectar suas presas.

O dispositivo mais usado para transmissão de ultra-som é o cristal piezo-elétrico, que converte energia mecânica em elétrica e vice-versa. Dessa forma, a aplicação de uma tensão senoidal no transmissor produz deformação senoidal correspondente. A vibração do cristal piezo-elétrico passa para o meio de transmissão e neste se propaga. Para aplicação em medição de distâncias, são usadas ondas de pequena potência, que não deformem permanentemente o meio. No receptor, ocorre o processo inverso, de modo que a vibração provocada por uma onda ultra-sônica é transformada em tensão.

2.6. O Microcontrolador LPC 2138 ARM7

Para a implementação do projeto foi utilizado o kit de desenvolvimento da empresa Lab Tools, cujo *hardware* permite o uso do microcontrolador LPC 2138. O esquema elétrico deste *hardware* está apresentado na Figura 2.17.

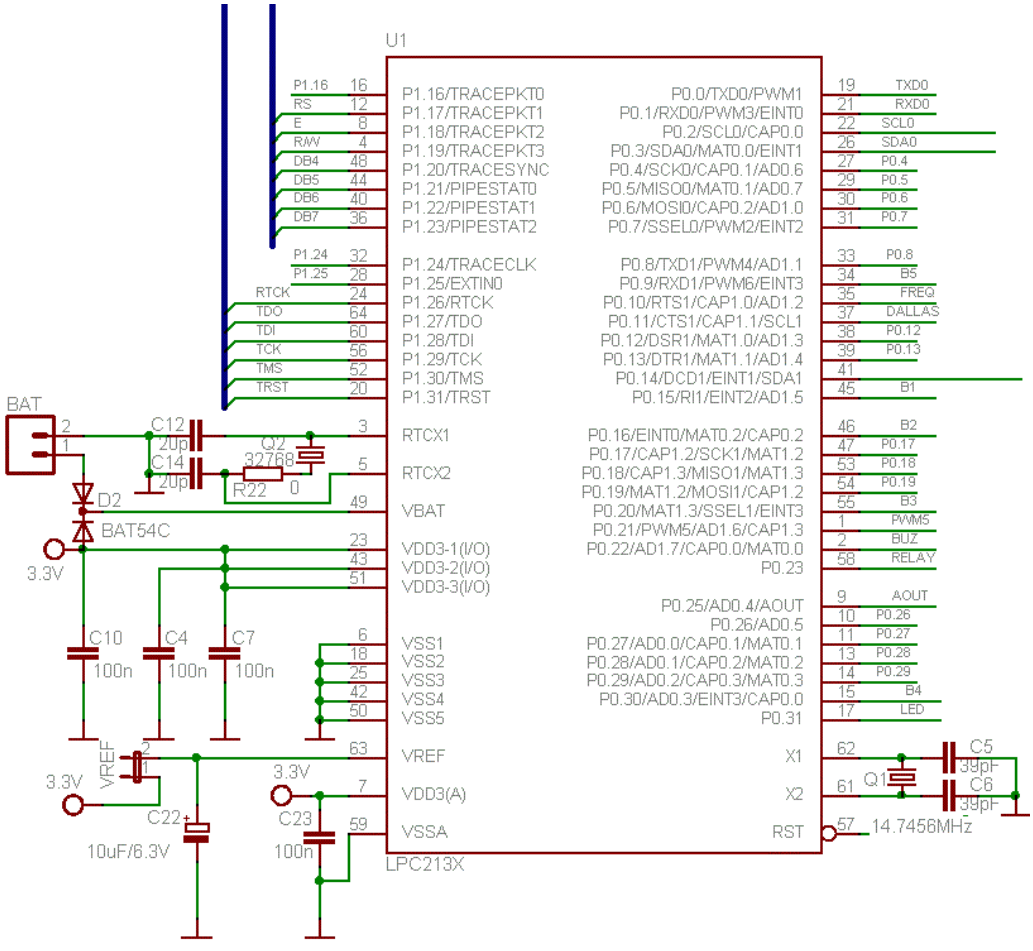


Figura 2-17– Circuito LPC 2138.

A placa utilizada para desenvolver este projeto está provida de um LCD alfanumérico padrão com 16 colunas e 2 linhas sem backlight. A comunicação é paralela com 4 vias de dados. Além das 4 vias de dados, mais duas vias são utilizadas para controlar o LCD, uma denominada de EN (enable) e a outra de RS.

A comunicação com o LCD é somente de escrita, desta forma, o pino de R/W do LCD está diretamente ligado ao terra (GND), não permitindo a leitura do mesmo.

A placa de desenvolvimento McBoard é dotada de um microcontrolador ARM7 de 32 bits LPC2138 da Philips, disposto sob a forma de uma bandeira (uma espécie de cartucho, similar a uma memória de microcontrolador) conforme Figura 2.18, cujos recursos serão apresentados a seguir.

Figura 2-18 – Bandeira LPC 2138.

Figura 2-19 – Kit de Desenvolvimento Utilizado no Projeto.

2.6.1 Apresentação do Núcleo ARM 7

Uma nova tendência no mercado são produtos com baixo consumo de energia e alto poder de processamento, uma qualidade desejável em todos os equipamentos eletrônicos. Para atingir este objetivo surgiram os microprocessadores ARM, em que ARM significa *Advanced RISC Machine* (máquina RISC avançada) e RISC significa *Reduced Instruction Set Computer* (computador com set de instruções reduzido) [9].

Os microprocessadores ARM de 32 bits são usados em celulares, PDA's, videogames portáteis, etc. O sucesso desses produtos se deve ao fato de o ARM ser uma nova tendência de mercado e pode ser mais explorado. O núcleo (core) ARM7 está disponível em vários microprocessadores de diversos fabricantes como a Philips, Analog Devices, OKI, dentre outros.

As características principais do processador ARM7TDMI-S são:

- **Modo Thumb:** o modo Thumb do ARM7TDMI-S nada mais é do que um segundo *set* de instruções com 16 bits e ajuda a economizar memória de programa.
- **Multiplicação longa:** a utilização de um *hardware* dedicado para multiplicação longa possibilita ao ARM7TDMI-S realizar operações mais complexas normalmente feitas por um DSP. Com esse *hardware* é possível realizar multiplicações de 32 bits por 32 bits, apresentando o resultado em 64 bits. Ainda esse módulo é capaz de realizar multiplicação-acumulação (MAC) de 32 bits por 32 bits com resultado de 64 bits.
- **Depuração:** uma região do ARM7TDMI-S possui uma extensão de *hardware* capaz de realizar uma depuração dentro da aplicação através de porta JTAG.
- **Embedded ICE:** é uma extensão das funções de depuração. Esse módulo estende as funções de breakpoint (pontos de parada), visualização de registros e outros pontos do programa, o que torna muito fácil o trabalho de depuração. Esse módulo é acessado pela porta JTAG.
- **Alta capacidade de processamento:** o núcleo ARM7TDMI-S tem alta capacidade de processamento.

A família LPC 213X da Philips é baseada no núcleo ARM7TDMI-S com suporte à emulação (usando a porta J-TAG) aliada a uma memória de programa Flash de alta velocidade e uma interface de alta velocidade permitindo a execução em 60 MHz. Para aplicações em que o tamanho do programa é importante, podemos contar também com o modo Thumb, com o qual é possível reduzir 30% do espaço da memória de programa [9].

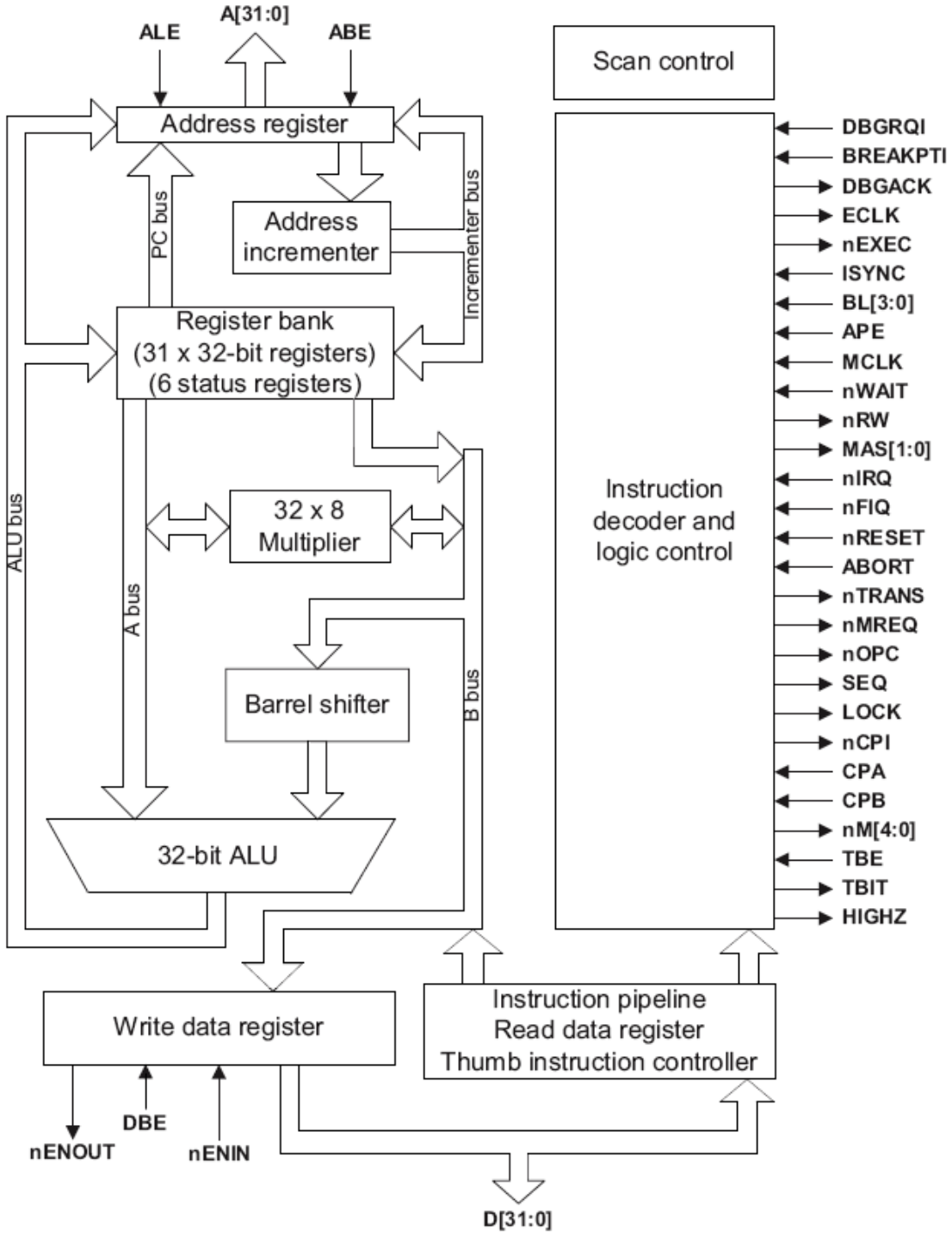


Figura 2-20 – Diagrama de Blocos da Arquitetura ARM 7 [9].

2.6.2 Características do LPC 2138

A família de processadores LPC213x é baseada no núcleo ARM7TDMI-S. O seu tamanho, consumo e desempenho são ideais para as mais diversas aplicações como: equipamentos médicos, controladores industriais de uso geral, controle de acesso, conectividade e aplicações de uso geral [9].

A aplicação proposta por este trabalho utilizará o processador LPC2138 da Philips. Segundo o fabricante as principais características desse processador são:

- Núcleo ARM7TDMI-S;
- Alimentação de 3,0V a 3,6V;
- 512 Kbytes de memória de programa;
- 32 Kbytes de memória de dados volátil (RAM);
- 22 interrupções (quatro externas);
- 47 I/Os;
- Dois timers ou contadores de eventos externos de 32 bits;
- Unidade de PWM com seis saídas;
- Watchdog Timer;
- Duas UART`S;
- Dois barramentos I²C`S;
- Um barramento SPI;
- Um módulo SSP;
- RTC interno;
- Dois conversores A/D de 10 bits com oito canais cada;
- Um canal de conversão D/A de 10 bits;
- Opera com cristal de 1 MHz até 30 MHz ou oscilador externo de 1 MHz até 50 MHz;
- 60 MHz de operação máxima via PPL interno.

2.6.3 Pinagem do Processador LPC 2138

O processador ARM é do tipo SMD (Superficial Monting Device), o que significa que ele fica sobre a placa de circuito impresso, ao contrário dos processadores antigos que eram fixados no lado oposto da placa. Esse detalhe permite que os processadores SMD possam ser menores, porém a troca do mesmo pode ser muito mais trabalhosa.

A Figura 2.21 apresenta a distribuição dos pinos do processador ARM7 LPC2138.

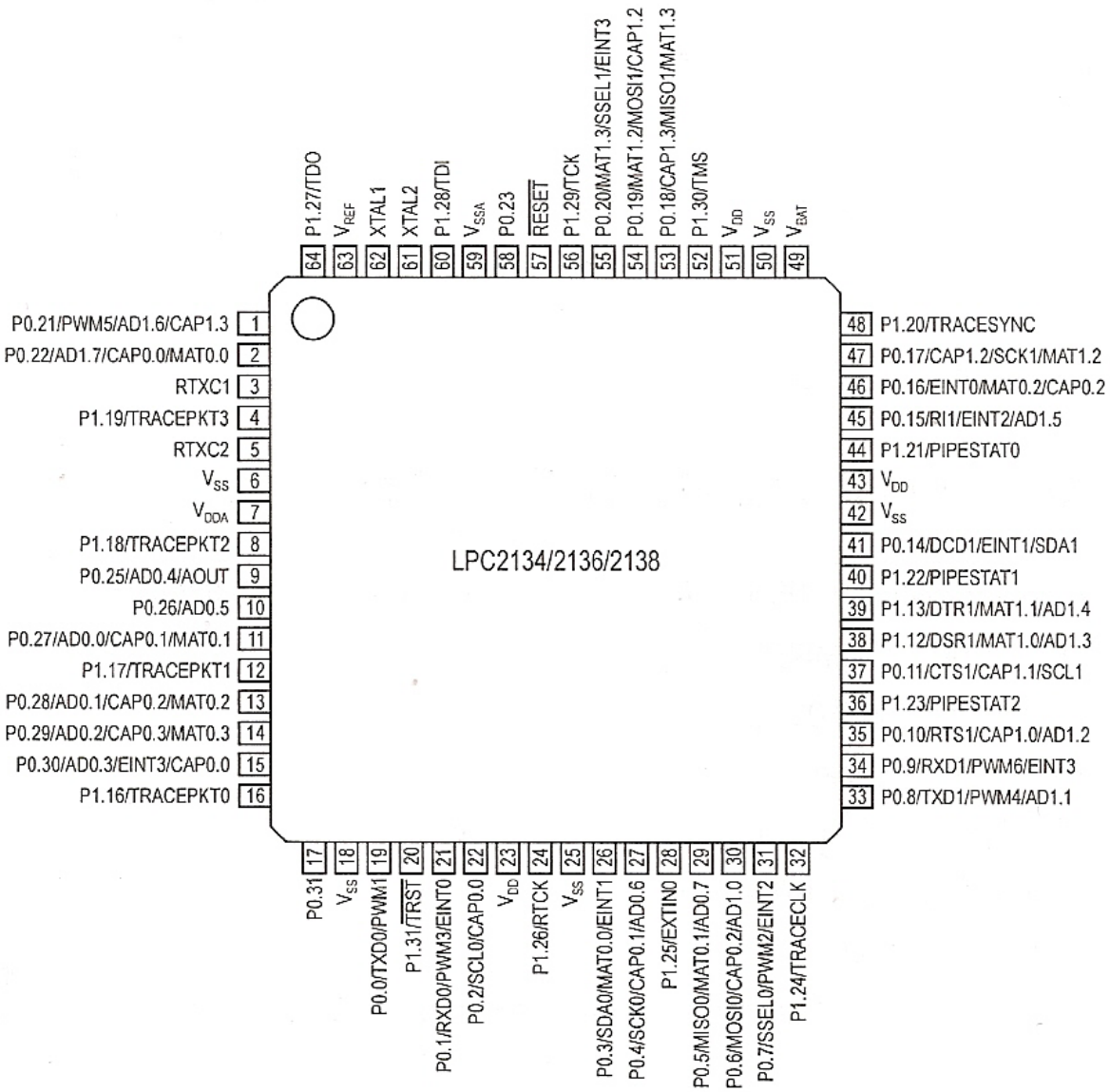


Figura 2-21 – Distribuição de pinos do Processador LPC 2138.
lpc2131.lpc2132.lpc2134.lpc2136.lpc2138-Datasheet.

2.6.4 Mapa de Memória do Processador ARM7

Na Figura 2.22 é apresentado o mapa de memória do processador ARM7. Para entender esse mapeamento é importante ressaltar que a memória pode ser: Flash (NON-VOLATILE MEMORY) ou SRAM (Static RAM). Analisando a figura de baixo para cima, é possível observar que, conforme o modelo do processador, a memória é maior. Por exemplo a capacidade do modelo LPC2131, é de 32 Kbytes e a do modelo LPC2138 é de 512 Kbytes. Acima da memória flash estão definidos os endereços da memória SRAM, também organizada pelos modelos de processador, e por último estão os endereços destinados aos periféricos e as interrupções [9].

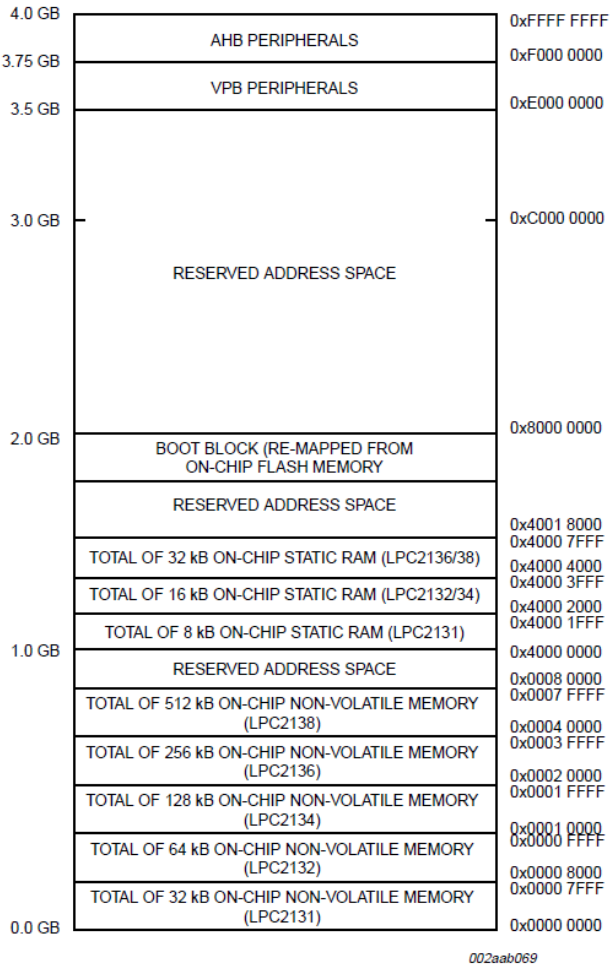


Figura 2-22 – Mapa de Memória do Processador ARM7 [9].

2.6.5 Comparativos Técnicos Entre Alguns Processadores

Considerando que o processador não é o item mais importante deste projeto e sim o meio para a resolução do objetivo principal, abaixo segue uma tabela com as principais características dos principais processadores ARM fabricados pela Philips.

A Tabela 2.3 tem o objetivo de demonstrar o grande número de microcontroladores existentes no mercado. Foram listados apenas alguns processadores ARM produzidos pela Philips.

Tabela 2-3 – Comparativo entre alguns processadores ARM fabricados pela Philips

Modelo	CPU Clock	Mem. Flash	RAM	Temperatura
ARM7 / LPC2104BBD48	60 MHz	128 KB	16 KB	0 °C to +70 °C
ARM7 / LPC2105FBD48/00	60 MHz	128 KB	32 KB	0 °C to +70 °C
ARM7 / LPC2106FHN48	60 MHz	128 KB	64 KB	40 °C to +85 °C
ARM7 / LPC2106FHN48/00	60 MHz	128 KB	64 KB	40 °C to +85 °C
ARM7 / LPC2131FBD64/01	60 MHz	32 KB	8 KB	40 °C to +85 °C

ARM7 / LPC2132FBD64/01	60 MHz	64 KB	16 KB	40 °C to +85 °C
ARM7 / LPC2132FHN64	60 MHz	64 KB	16 KB	40 °C to +85 °C
ARM7 / LPC2134FBD64	60 MHz	128 KB	16 KB	40 °C to +85 °C
ARM7 / LPC2136FBD64	60 MHz	256 KB	32 KB	40 °C to +85 °C
ARM7 / LPC2138FBD64	60 MHz	512 KB	32 KB	40 °C to +85 °C
ARM9 / LH7A400N0F076B5	250 MHz		80 KB	40 °C to +85 °C

Fonte: www.nxp.com, página oficial da Philips.

O estudo do processador utilizado neste projeto é fundamental, pois, para desenvolver o projeto foi necessário um conhecimento aprofundado das funcionalidades do mesmo.

A escolha deste processador LPC2138 núcleo ARM7 se deu em função da vasta gama de periféricos disponíveis combinada com um excelente custo-benefício, seu poder de processamento em relação ao consumo de energia e a disponibilidade do mesmo, pois muitos fabricantes de semicondutores oferecem microcontroladores baseados na tecnologia ARM.

3. IMPLEMENTAÇÃO DO *HARDWARE*

Este capítulo apresenta a metodologia utilizada no desenvolvimento do projeto, com as implementações realizadas no Laboratório da Universidade Luterana do Brasil, abordando também as descrições das características construtivas do sistema de controle em estudo, bem como a sua implementação através de um controlador PID com sintonia automática, utilizando um processador LPC 2138 de 32 bits núcleo ARM7, apresentando todas as características da sua aplicação neste projeto de pesquisa.

No decorrer deste capítulo estão apresentadas as medições do fluxo de ar, leituras do sensor em relação à tensão aplicada, bem como as características do sistema em diversas posições do elemento móvel, com valores de tensão compreendidos na saída do microcontrolador entre 0 e 3,3V.

3.1. Descrição do Sistema Desenvolvido

3.1.1 Diagrama em Blocos

O diagrama em blocos construtivo mostrado na Figura 3.1, permite visualizar a forma como o sistema foi implementado. O desenvolvimento e a implementação de cada bloco estão descritos no decorrer deste capitulo.

A descrição em forma de bloco do *hardware* utilizado para implementação do microcontrolador é mostrado na Figura 3.2, sendo que os blocos são compostos da seguinte forma;

- A: Computador para realizar a programação e gravar o software.
- B: Microcontrolador LPC 2138.
- C: Circuito de alimentação do ventilador, sensor e fonte externa para o somador e drive para reforço de corrente.
- D: Ventilador DC Fan de 12 Volts.

- E: Estrutura mecânica com tubo de acrílico transparente.
- F: Sensor ultra-som.
- G: Display para mostrar os resultados.

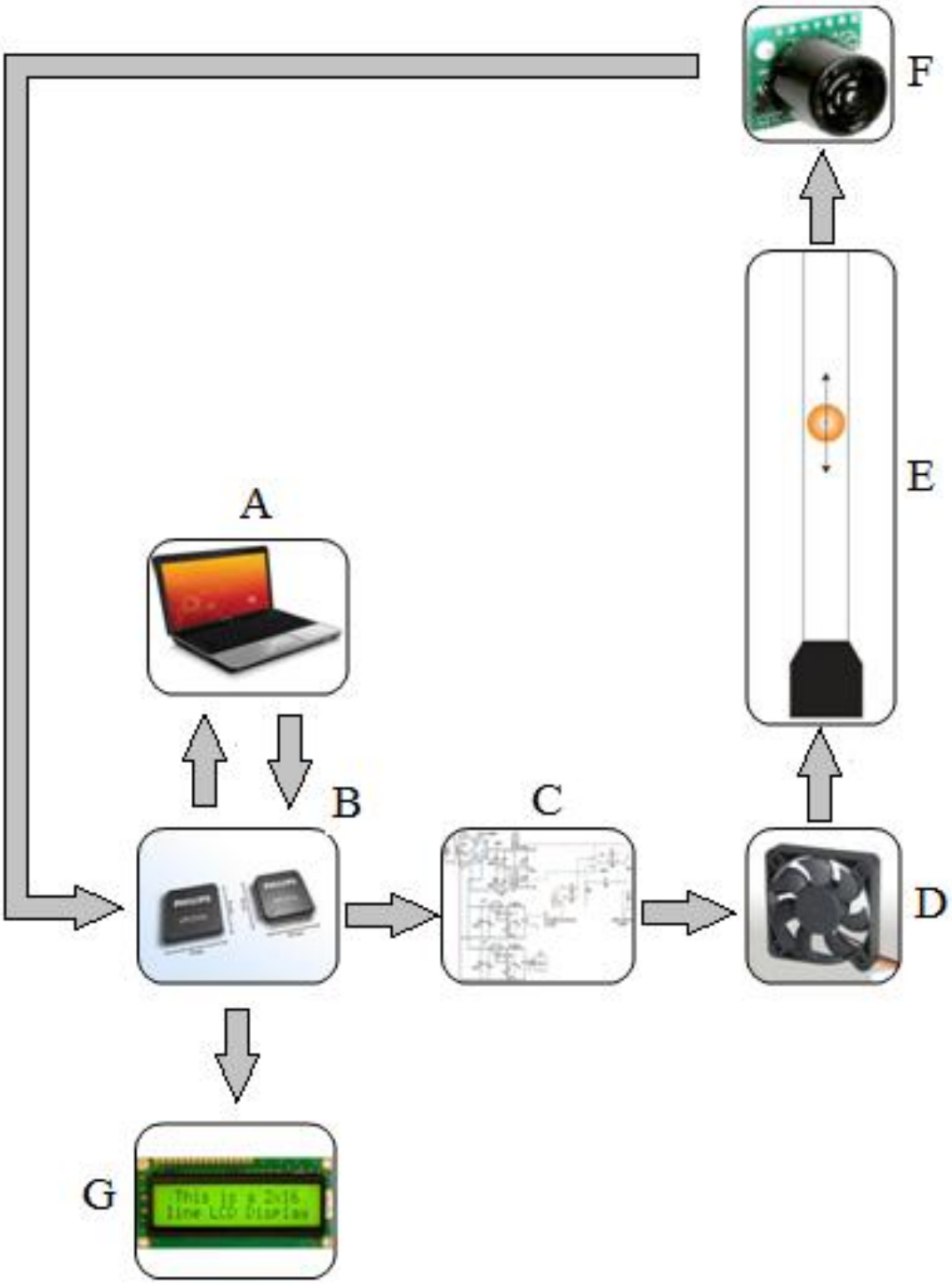


Figura 3-1 – Imagem Ilustrativa do Diagrama em Blocos do Projeto.

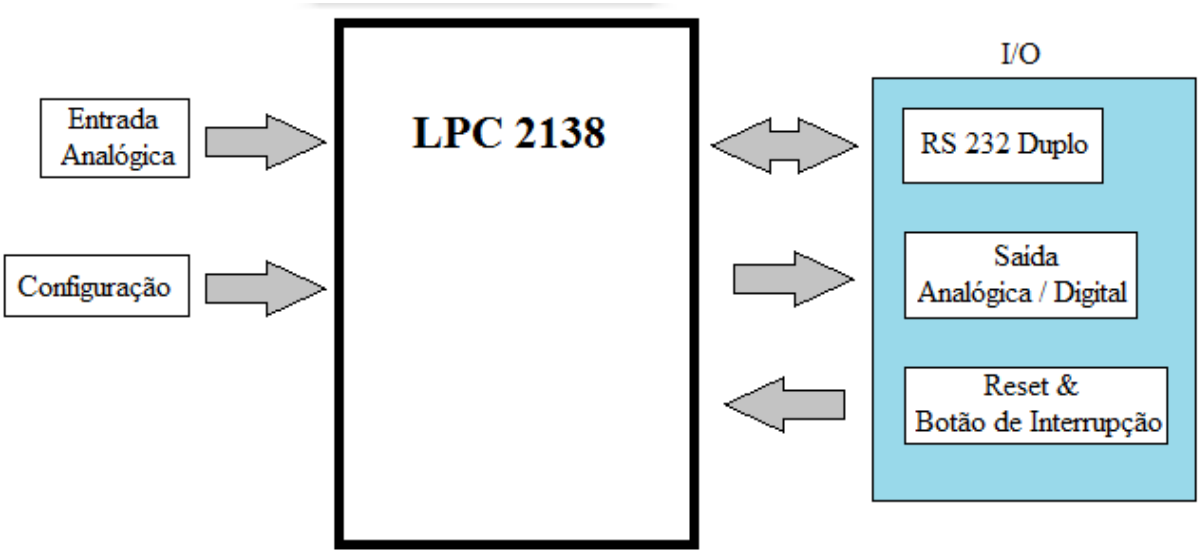


Figura 3-2 – Diagrama em Blocos do Kit de Desenvolvimento.

O kit de desenvolvimento do projeto foi implementado junto com a planta através dos pinos 9 e 11 conforme está apresentado no diagrama em blocos da Figura 3.3.

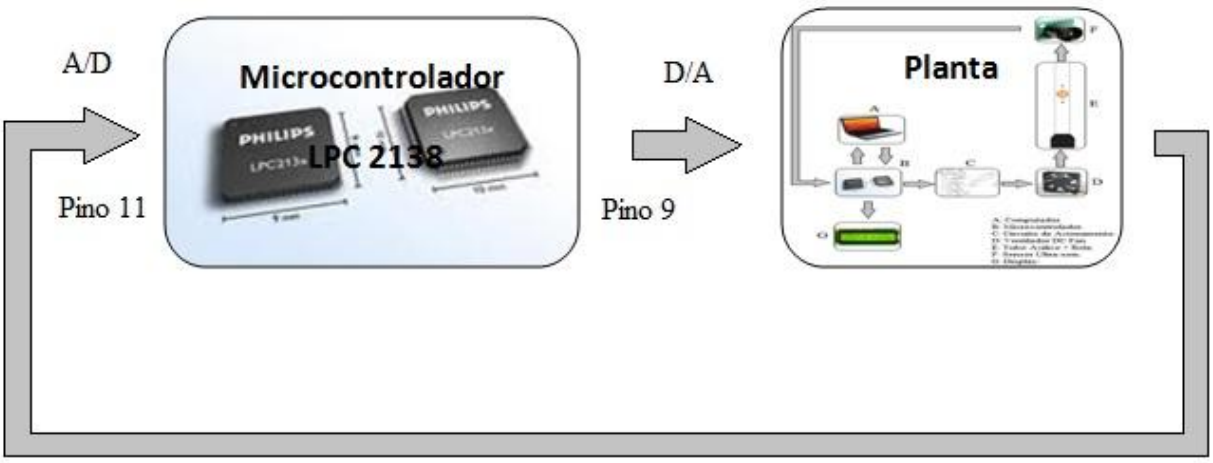


Figura 3-3 – Diagrama em Blocos do Microcontrolador Ligado a Planta.

3.1.2 Estrutura Mecânica

Para tornar o projeto realidade, foi necessário planejar e confeccionar uma estrutura metálica contendo perfil em aço quadrado e uma chapa circular também de aço, ambas soldadas para possibilitar a fixação e sustentação dos demais componentes conforme ilustração na Figura 3.4.

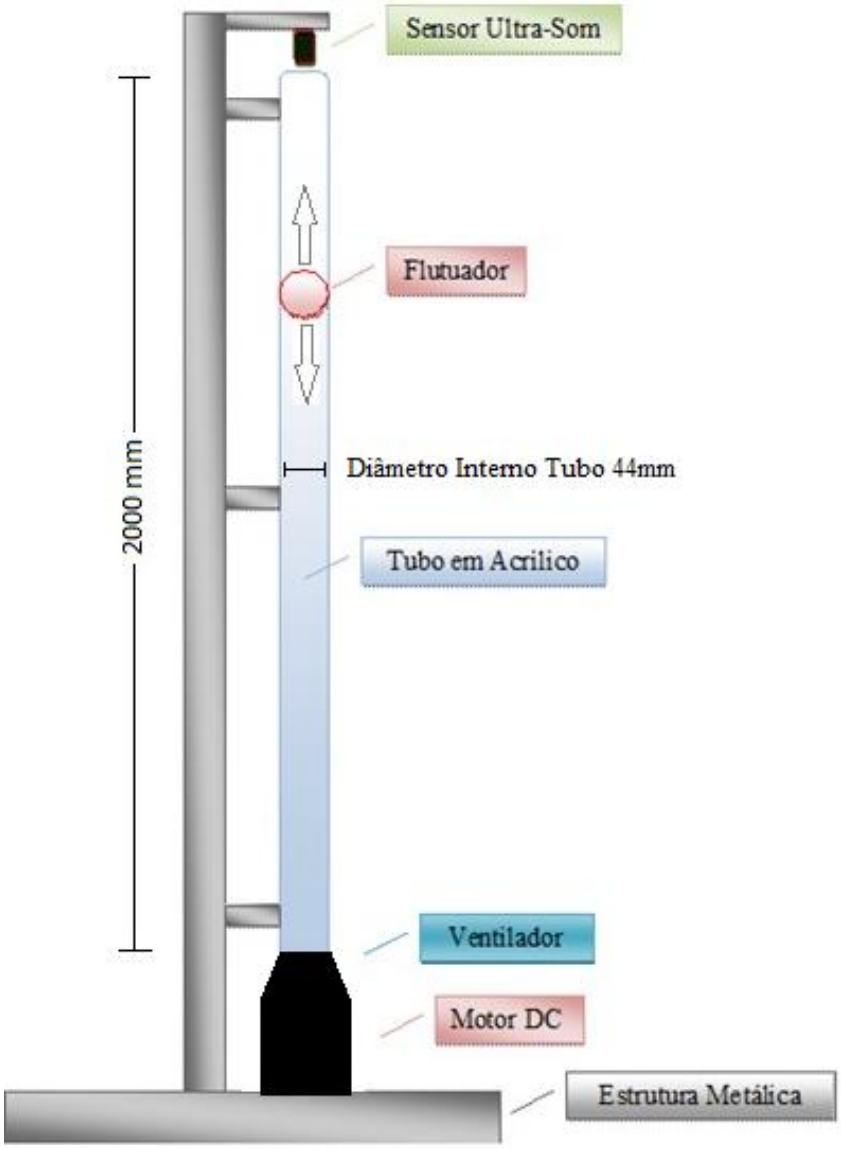


Figura 3-4 – Ilustração da Estrutura do Projeto.

Como se pode observar, a estrutura metálica foi desenvolvida de tal forma a possibilitar a fixação do tubo de acrílico transparente contendo 2000 (dois mil) milímetros de comprimento, 50 (cinquenta) milímetros de diâmetro externo e 44 (quarenta e quatro) milímetros de diâmetro interno.

Foi fundamental a escolha deste tubo transparente, pois desta forma é possível visualizar a posição do flutuador e assim estabelecer os parâmetros de controle da posição do mesmo. O flutuador utilizado neste projeto é uma simples esfera de isopor com massa de 0,96 gramas e raio 19 milímetros.

Na parte inferior da estrutura foi necessário construir uma redução conforme Figura 3.5, para acoplar o ventilador ao tubo, pois o mesmo possui diâmetro circular, já o ventilador tem o formato quadrado de 120mm x 120mm x 38mm.

Figura 3-5 – Ilustração da redução do ventilador para o tubo.

Na Figura 3.5, também é possível visualizar a base feita em metal para dar estabilidade à estrutura, bem como o suporte de sustentação do ventilador aparafusado no fundo da redução do tubo, sendo que o mesmo fornece um fluxo de ar ascendente que empurra a bola para cima de maneira a contrabalançar a força descendente da gravidade. A velocidade do ventilador (Figura 3.6) pode ser controlada para mudar a velocidade da corrente de ar causando uma mudança na altura da bola.

Figura 3-6 – Foto do ventilador DC Fan 12V.

Com a utilização de uma redução do ventilador para o tubo, as características da direção do fluxo de ar foram alteradas, gerando uma elevada turbulência no mesmo, comprometendo desta forma o desempenho do sistema.

Para tentar solucionar esta alteração, foi necessária a confecção de uma colméia composta por 16 tubos que possuem um comprimento de 100 milímetros e diâmetro de 16 milímetros, colocados na parte inferior do tubo Figura 3.7 com o objetivo de direcionar o sentido do fluxo de ar e minimizar a turbulência do mesmo.

Figura 3-7 – Colméia colocada no tubo.

3.1.3 Sensor Ultra-Som

O sensor ultra-som implementado no projeto (figura3.8), utiliza para seu funcionamento um circuito formado por quatro LM324, uma matriz de diodos, um PIC16F676, junto com uma variedade de componentes passivos conforme Figura 3.9.

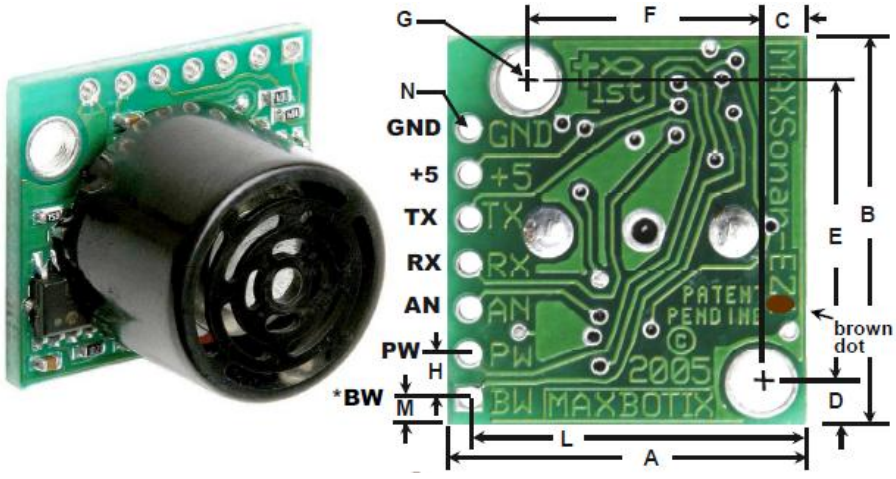


Figura 3-8 – Imagem sensor ultra-som utilizado.

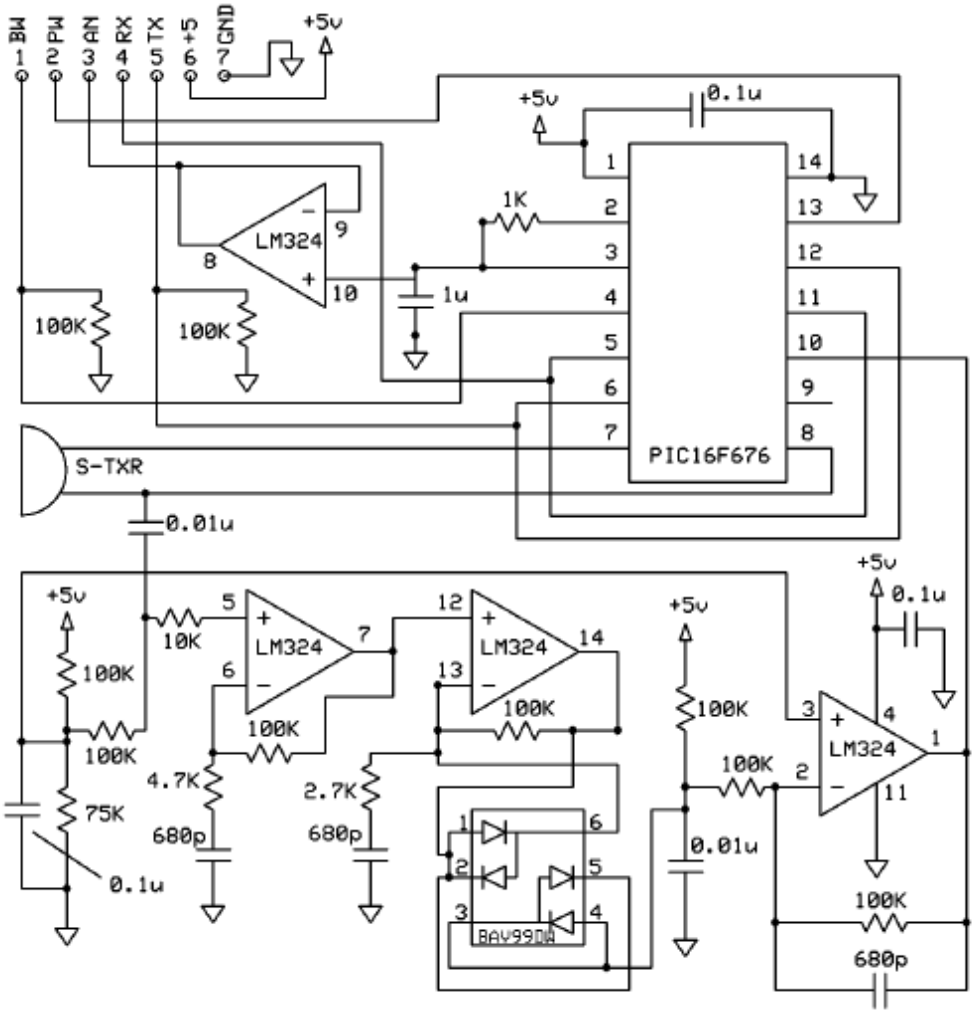


Figura 3-9 – Circuito do Sensor Ultra-Som.
www.tato.ind.br

Este circuito apresenta as seguintes características:

- Ganho variável ajustado continuamente para controle preciso do feixe de ultra-som.
- A detecção de objetos inclui objetos encostados no sensor.
- Alimentação de 2.5V a 5.5V com baixo consumo de 2mA.
- Medição a cada 50ms, (20-Hz rate).
- Operação automática, mede e retorna os valores continuamente.
- Operação manual permite receber os valores apenas quando desejado.
- Todas as saídas são ativas simultaneamente.
- Baixo custo e consumo.
- Dados de saída precisos e estáveis.
- Sem “zona morta” de medição.
- Feixe de qualidade.

- Furos de montagem na placa.
- Ideal para operar a baterias.
- Pode ser disparado externamente ou internamente.
- O sensor retorna o valor da medição de acordo com sua alimentação, sendo que para 3,3V retorna com aproximadamente 6,9mV por polegada, e alimentação de 5V retorna com aproximadamente 9,8mV por polegada.

Na Figura 3.10 é ilustrada a instalação do sensor ultra-som na estrutura do projeto, sendo que o mesmo é um dispositivo muito sensível, tornando-se necessário que o mesmo fosse colocado dentro de uma caixa de proteção para evitar danos durante o transporte e a realização dos testes.

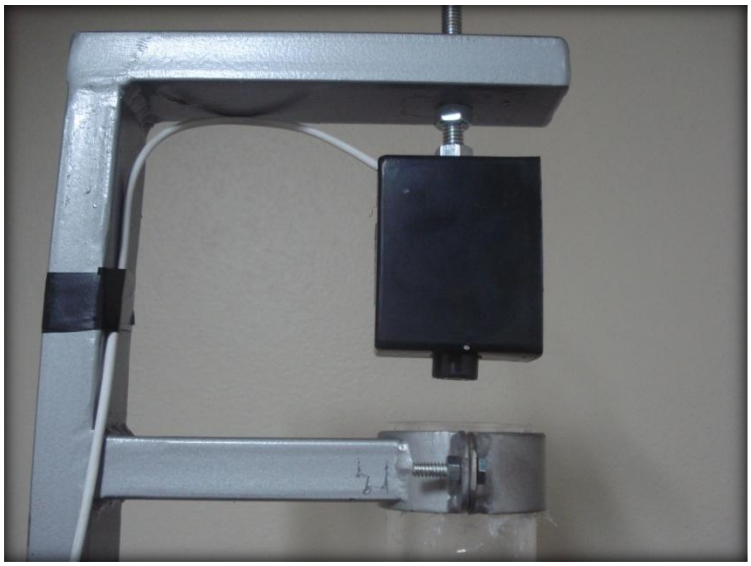


Figura 3-10 – Imagem da instalação do sensor.

Este sensor permite a detecção e medição de obstáculos em uma placa de pequeno tamanho, detectando objetos entre 0 e 254 polegadas (6.45 metros) e permite a medição de distância de 6 polegadas até 254 polegadas com uma resolução de 1 polegada. Objetos entre 0 e 6 polegadas são medidos como 6 polegadas, sendo que os sinais de saída são: Largura de pulso, tensão analógica e saída serial digital.

Para tornar possível a variação da tensão de alimentação do sensor foi implementado um circuito conforme a Figura 3.11, com o objetivo de proporcionar nos testes o melhor rendimento, de forma a verificar qual o valor de tensão na alimentação resulta nas melhores respostas com o seu uso no sistema.

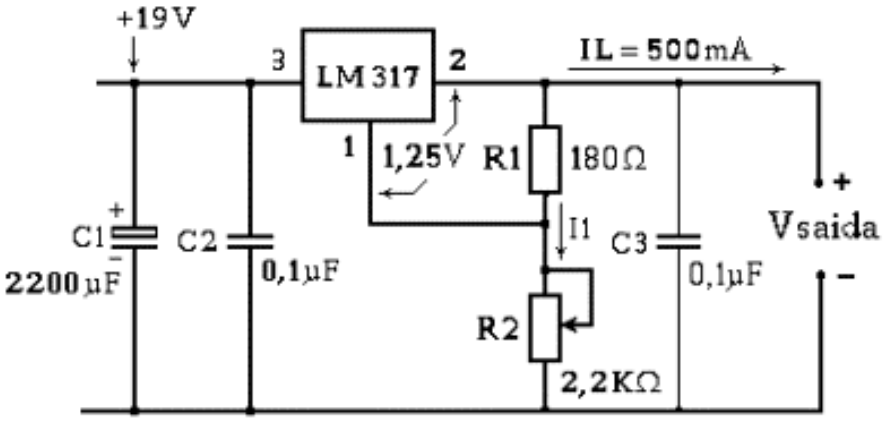


Figura 3-11 – Circuito de alimentação do sensor.
http://ivairijs.vilabol.uol.com.br/regulador1.html#Ivair

3.1.4 Circuito de Alimentação do Ventilador

Como a saída D/A do microcontrolador possibilita somente uma variação na tensão de 0 a 3,3V, e por sua vez o ventilador possui variação de 0 a 12V, foi necessário a implementação de um circuito de alimentação auxiliar para efetuar o controle do sistema. Um circuito com as características desejadas foi implementado com a utilização de um amplificador operacional somador não inversor, sendo que o mesmo possibilita realizar a soma da tensão de saída do kit com outra gerada por uma fonte externa.

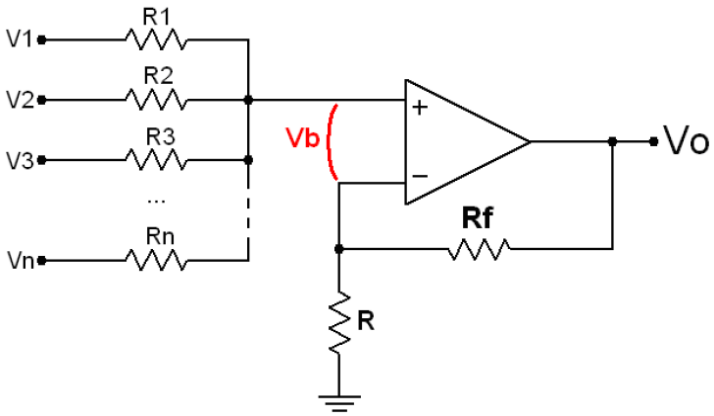


Figura 3-12 – Circuito Somador Não-Inversor.

As características da saída do amplificador operacional somador não-inversor, podem ser obtidas através da Equação 3.1, sendo que seu circuito foi montado utilizando resistores de 10k Ohms.

$$V_0 = \left(1 + \frac{R_f}{R}\right).(V_1 + V_2 + \cdots + V_n)$$

(Equação 3.1)

A alimentação do amplificador é realizada com um circuito utilizando composto por uma fonte simétrica com tensão de saída regulada conforme Figura 3.13.

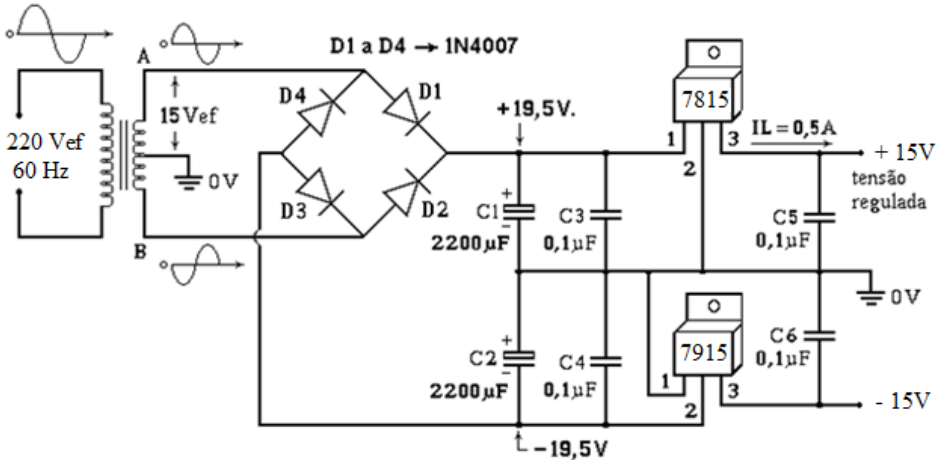


Figura 3-13 – Circuito de Alimentação do Somador Não-Inversor.
http://ivairijs.vilabol.uol.com.br/regulador1.html#Ivair

Para estabelecer os parâmetros iniciais do circuito somador é necessária a variação da tensão de tal forma que seja possível com a soma das tensões do microcontralador e da fonte externa realizar a variação do fluxo de ar proveniente do ventilador. Esta soma se faz necessária, pois o ventilador possui tensão de trabalho que pode ser variada de 0 a 12 volts, no entanto a saída do microcontrolador permite apenas a variação de 0 a 3 volts.

Como a movimentação da esfera de isopor teve seu início com a aplicação de 7,5 volts, o circuito somador possibilitou o ajuste desta tensão inicial, permanecendo o *range* do microcontrolador para realizar a variação de tensão necessária para estabelecer o controle do sistema.

A alimentação externa para realizar a variação da tensão estabelecendo os parâmetros iniciais do sistema foi realizada conforme Figura 3.14.

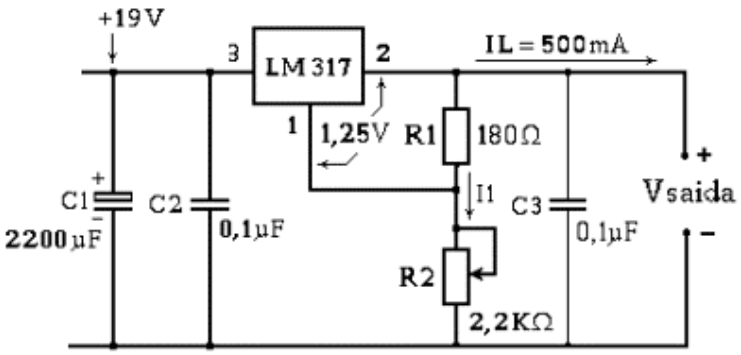


Figura 3-14 – Circuito fonte Externa.
http://ivairijs.vilabol.uol.com.br/regulador1.html#Ivair

Após a implementação dos circuitos mencionados anteriormente, verificou-se que os amplificadores operacionais fornecem correntes relativamente baixas na saída. Para aumentar a corrente foi necessário acrescentar um drive de reforço na saída do amplificador conforme Figura 3.15.

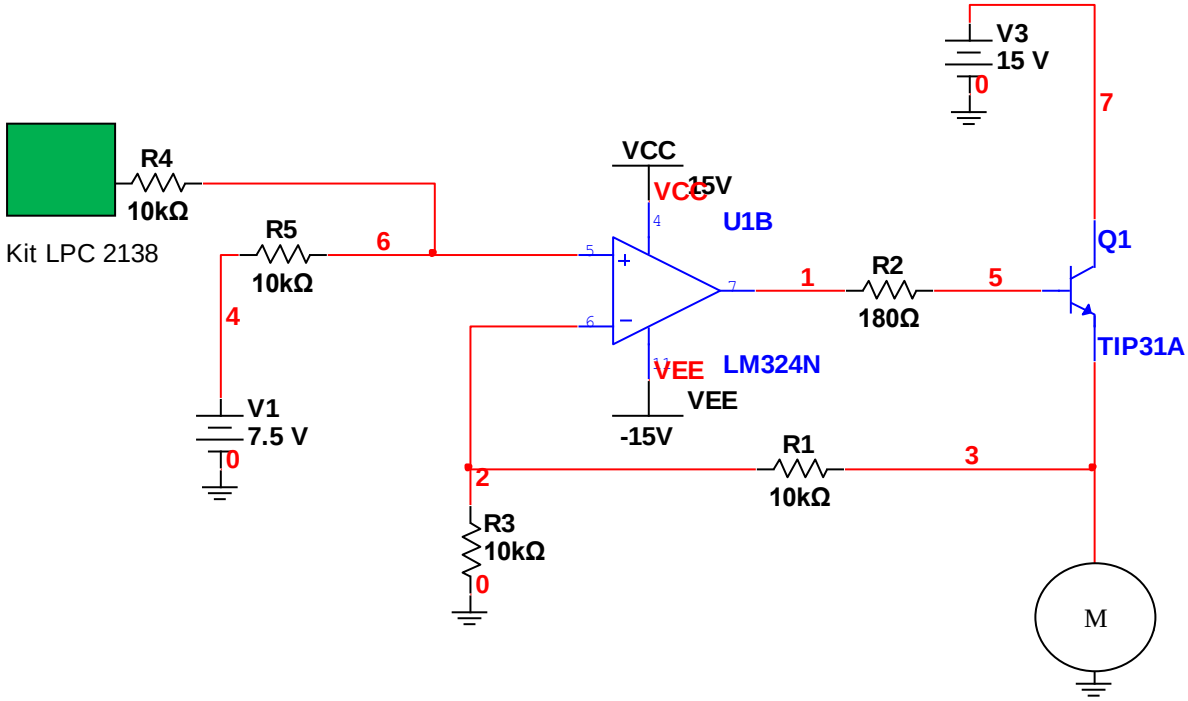


Figura 3-15 – Circuito Reforço de Corrente.

A figura 3.16 mostra a montagem completa do circuito de alimentação do amplificador operacional, alimentação do sensor e da fonte externa para o circuito somador não inversor.

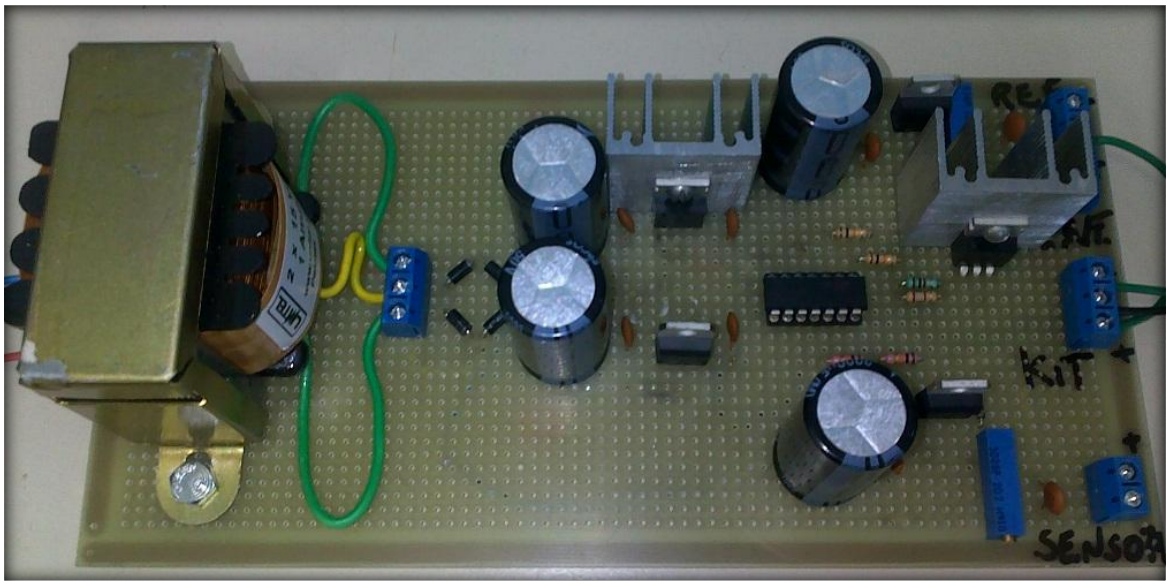


Figura 3-16 – Ilustração Circuito Completo e Montado.

4. IMPLEMENTAÇÃO DO SOFTWARE E DISCUSSÃO DOS RESULTADOS

No decorrer deste capítulo serão apresentadas as características, bem como o funcionamento do *software* e os resultados práticos obtidos.

Para iniciar o projeto e facilitar a programação do LPC 2138, utilizou-se a ferramenta IAR MakeApp criada pela empresa IAR cujo ambiente de programação pode ser visto na figura 4.1, cuja finalidade é gerar códigos em linguagem C conforme a configuração visual feita pelo usuário.

Estas características facilitam a criação de projetos, já que é possível não só fazer uma configuração inicial do programa, mas modificá-la a qualquer momento durante o desenvolvimento.

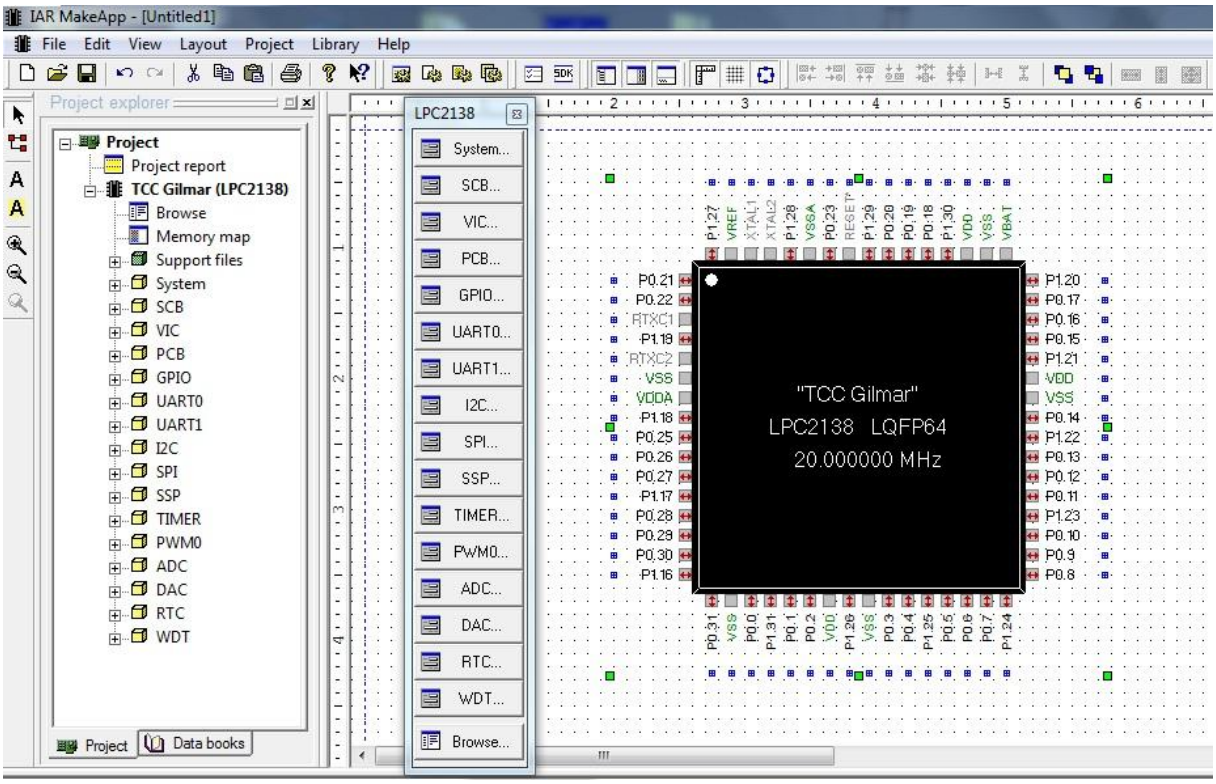


Figura 4-1 – Início do projeto no IAR MakeApp.

Para dar sequência à programação e compilação do *software* do microcontrolador LPC2138 família ARM7, foi utilizado o compilador com escrita em linguagem C IAR Embedded Workbench também da empresa IAR. A figura 4.2 apresenta um trecho do código do programa neste ambiente.

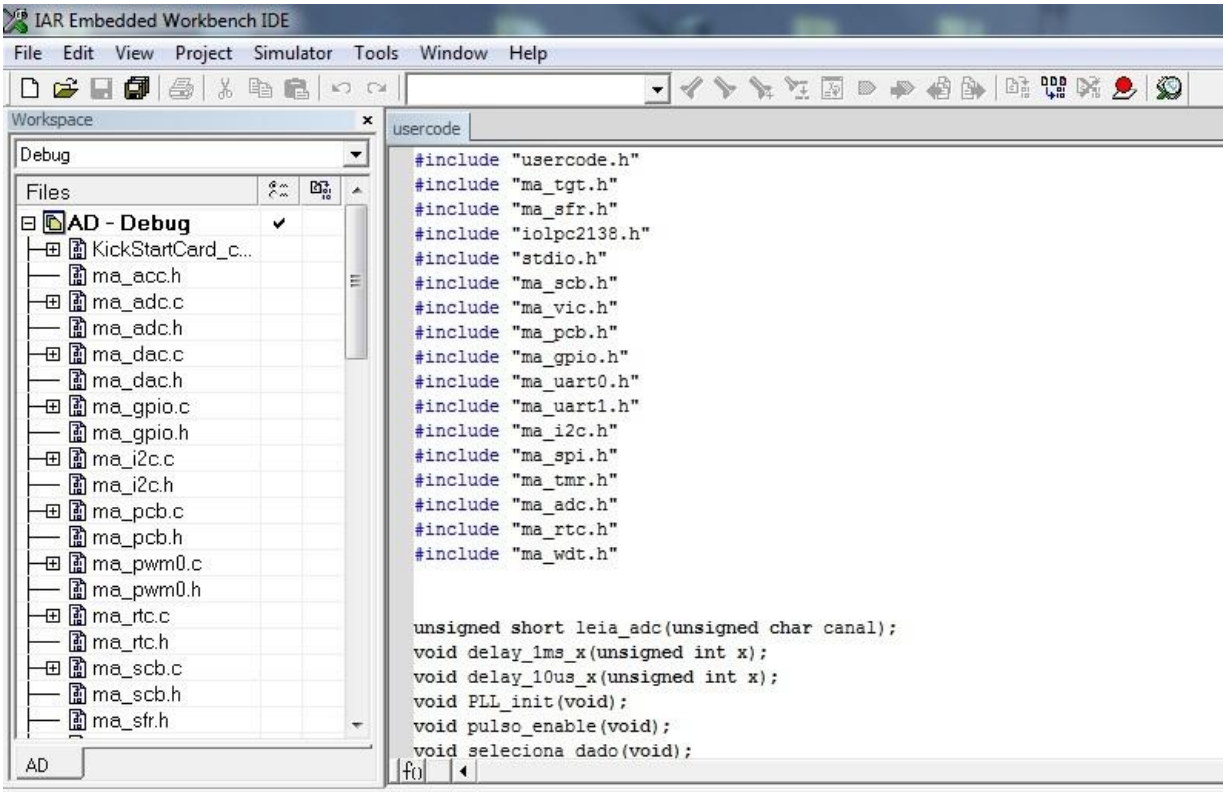


Figura 4-2 – Ambiente de desenvolvimento IAR Embedded Workbench.

Para gravar o código compilado no microcontrolador foi utilizado o programa LPC2000 Flash Utility da Philips (Figura 4.3).

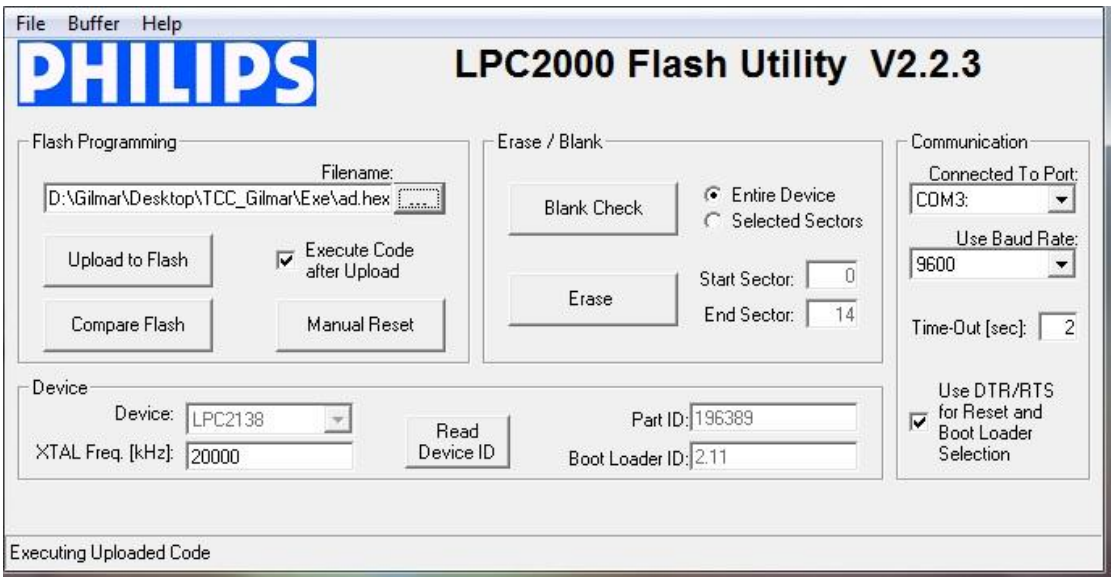


Figura 4-3 – Ambiente para Gravação LPC2000 Flash Utility.

A configuração correta deste aplicativo é muito importante para obter o resultado desejado, sendo que devem ser observados os seguintes parâmetros:

- A velocidade recomendada da porta serial é de 9600bps.
- Seleção correta da porta serial do computador.
- Habilitação dos pinos DTR/RTS para a geração dos *resets* durante a gravação.
- O tempo de espera recomendado para o sistema é de 2s.
- Identificar corretamente o microcontrolador utilizado.
- A frequência do oscilador em 20MHz.

Se a habilitação dos pinos DTR/RTS não for realizada, o programa informa ao usuário que em determinados momentos será preciso pressionar o botão de reset para efetuar a gravação.

Para gravar o programa, primeiramente deve-se apagar o conteúdo da memória flash clicando no botão Erase e para verificar se a memória flash do LCP2138 está limpa deve-se clicar no botão Blank Check. Na caixa Flash Programming em Filename, pode-se localizar e selecionar o arquivo a ser gravado com extensão .HEX e depois clicando em Upload to Flash inicia-se a gravação do mesmo.

4.1. Modelagem do Sistema de Controle

4.1.1. Função Transferência

Atualmente nas indústrias muitas plantas são controladas diretamente por um único computador digital, onde a maioria das malhas de controle podem ser manipuladas por um controle PID. A ação de controle PID nos controladores analógicos é dada pela equação 4.1 [14].

$$m(t) = K\left[e(t) + \frac{1}{T_i}\int_0^t e(t)dt + T_d\frac{de(t)}{dt}\right]$$

(Equação 4.1)

Onde *e(t)* representa a entrada do controlador, *m(t)* é a saída do controlador, *K* é o ganho proporcional, T_i é o tempo integral (ou tempo de *reset*) e T_d é a parcela do tempo derivativo. A função transferência pode ser obtida aproximando o termo integral pelo somatório trapezoidal e o termo derivativo pela diferença entre dois pontos.

A transformada z é descrita conforme equação 4.2.

$$M(Z) = K\left[1 + \frac{T}{2T_i}\frac{1+Z^{-1}}{1-Z^{-1}} + \frac{T_d}{T}(1-Z^{-1})\right]E(Z)$$

(Equação 4.2)

Sendo que a mesma pode ser reescrita conforme equação 4.3.

$$M(Z) = K\left[1 - \frac{T}{2T_i}\frac{1}{1-Z^{-1}} + \frac{T_d}{T}(1-Z^{-1})\right]E(Z) = \left[K_p + \frac{K_i}{1-Z^{-1}} + K_d(1-Z^{-1})\right]E(Z)$$

(Equação 4.3)

Assim:

$$K_P = K - \frac{KT}{2T_i} = K - \frac{K_i}{2} = \text{ganho proporcional}$$

(Equação 4.4)

$$K_I = \frac{KT}{T_i} = \text{ganho integral}$$

(Equação 4.5)

$$K_D = \frac{KT_D}{T} = \text{ganho derivativo}$$

(Equação 4.6)

Um bom referencial teórico para modelagem dos controladores PID digitais pode ser encontrada em livros como OGATA (1987).

4.1.2. Modelagem do Controlador

A realização de um controle digital envolve *software* e *hardware* ou ambos, sendo que por realização entende-se a organização das operações de soma, armazenamento e multiplicação. Um controlador digital é um filtro digital implementado na forma de um algoritmo.

Este converte uma sequência de números da entrada em outra sequência na saída. Pode-se verificar que Z^{-1} representa o atraso de uma constante de tempo ($Z^{-1} = e^{-Ts}$), considerando uma função transferência genérica.

$$G(Z) = \frac{Y(Z)}{X(Z)} = \frac{b_0 + b_1Z^{-1} + b_2Z^{-2} + \cdots + b_mZ^{-m}}{1 + a_1Z^{-1} + a_2Z^{-2} + \cdots + a_nZ^{-n}}$$

(Equação 4.7)

Onde n é menor ou igual a m, assim, para um PID tem-se:

$$Gc(Z) = K_P + \frac{K_I}{(1 - Z^{-1})} + K_D(1 - Z^{-1})$$

(Equação 4.8)

$$Gc(Z) = \frac{(K_P + K_I + K_D) - (K_P + 2K_D)Z^{-1} + K_D Z^{-2}}{1 - Z^{-1}}$$

(Equação 4.9)

Logo comparando com a função genérica:

$$a_1 = -1 \quad a_2 = 0 \quad b_0 = K_P + K_I + K_D \quad b_1 = -(K_P - 2K_D) \quad b_2 = K_D$$

Os coeficientes a_i, b_i são quantidades reais que aparecem como multiplicadores. Este controlador implementado via programação direta seria:

$$Gc(Z) = \frac{U(Z)}{E(Z)} = \frac{b_0 + b_1 Z^{-1} + b_2 Z^{-2}}{1 - Z^{-1}}$$

(Equação 4.10)

Onde:

U(z) = Atuação

E(z) = Erro

Assim:

$$U(Z) = b_0 E(Z) + b_1 Z^{-1} E(Z) + b_2 Z^{-2} E(Z) + Z^{-1} U(Z)$$

(Equação 4.11)

Onde:

U(z) = Atuação

$b_0 E(Z)$ = Erro Atual

$b_1 Z^{-1} E(Z)$ = Erro Anterior

$b_2 Z^{-2} E(Z)$ = Erro Anterior ao erro anterior

$Z^{-1} U(Z)$ = Atuação Anterior

4.2. Principais recursos de software utilizados

O software implementado é composto de várias funções, cada uma com um objetivo específico, no decorrer deste capítulo cada uma delas será apresentada. O código fonte completo do software se encontra no Apêndice A.

4.2.1. Função MA_ADC

O conversor analógico-digital do LPC 2138 funciona por aproximação sucessiva de 10 bits, com um tempo de conversão de 2,44µs, sendo que o microcontrolador possui dois circuitos de conversão e cada um deles possui oito entradas multiplexadas.

As características desse conversor são:

- Configurável para conversões de 3 a 10 bits por *hardware*.
- Trabalha com tensão de 0 a 3 Volts.
- Tempo de conversão de 2,44µs ou 410kcps (410.000 conversões por segundo).
- Possibilidade de trabalhar juntamente com o timer para conversões automáticas ou através de um evento externo.

Esta função conforme figura 4.1 faz a conversão A/D do microcontrolador, iniciando com a configuração e em seguida a conversão A/D através do registrador ADCR, e depois, grava o valor do registrador ADDR. Após gravar o valor, transforma este em um número que corresponde ao valor de tensão e o retorna para a função principal.

usercode *

```
unsigned short leia_adc(unsigned char canal);

          //leitura do AD (float)leia_adc(0);
           resultado = (float)leia_adc(0);  //Chama a funcao de leitura do A/D.
           resultado = (resultado)/10; //Converte valor do A/D em valores de 0 a 74 polegadas.
```

Figura 4-4 – Habilitação da função A/D.

4.2.2. Função MA_DAC

Este é o módulo mais simples de todos, pois tem apenas um único registrador, um conversor de 10 bits, podendo ser selecionadas duas velocidades:

- Bits 6 a 15: Bits VALUE, um valor de 0 a 1023 representa uma tensão cujo valor é dado por (VALUE/1024)*Vref.
- Bit 16: Bit BIAS, este bit em 1 configura o tempo de conversão de 1µs e em 0 configura o tempo de conversão de 2,5µs.

Esta função transforma o valor da atuação do PID, em um valor correspondente em bits, enviando este valor para o pino 11 do microcontrolador, sendo que sua habilitação está descrita na figura 4.5.

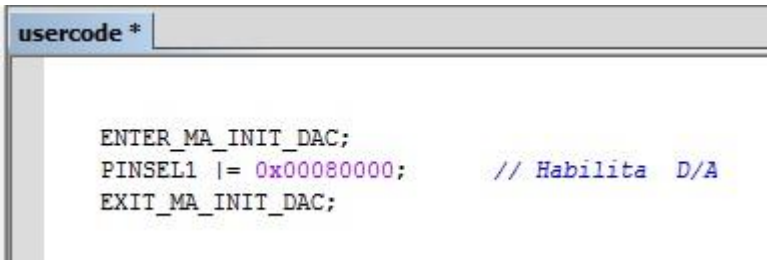


Figura 4-5 – Habilitação do conversor D/A.

4.2.3. Função Aquisição do Tempo de Atuação

Como é fundamental saber o tempo de atuação para realizar os cálculos do sistema de controle, foi necessário habilitar um pino digital ligado a um led conforme figura 4.6 para realizar a aquisição e validação do mesmo.

Figura 4-6 – Rotinas para aquisição do tempo de atuação.

Com esta implementação foi possível verificar que o tempo de atuação é muito pequeno, na faixa de 6ms, portanto, já que o sensor realiza uma aquisição a cada 50ms, foi inserido um *delay* de 25ms para que sejam feitos conforme figura 4.7 dois cálculos de 28ms para cada aquisição.

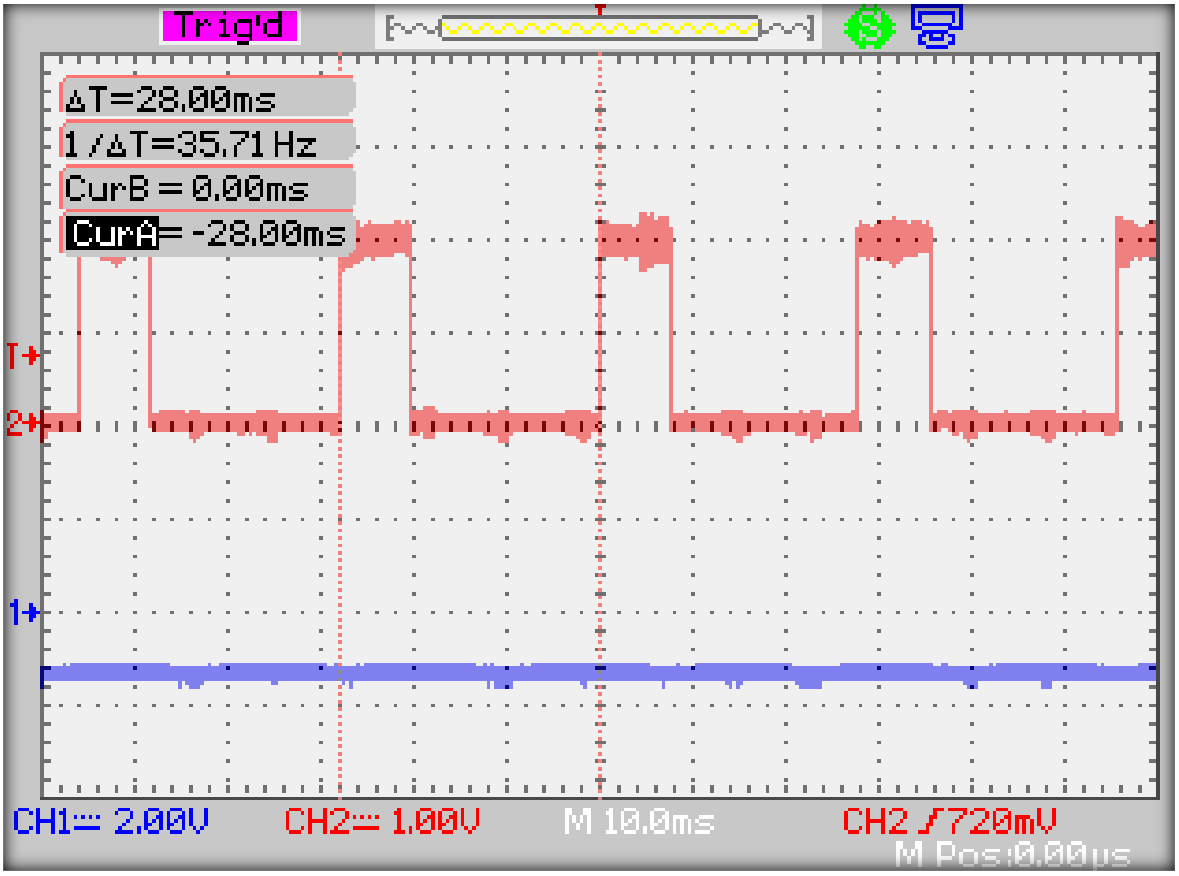


Figura 4-7 – Validação do Tempo de Atuação.

4.2.4. Atuação do PID Digital

No sistema de controle foi utilizado um controle PID de posição em malha fechada conforme modelagem vista no Item 4.1, sendo que esta etapa tem como finalidade fechar o laço e determinar a atuação que será aplicada na entrada da planta. Nesta parte do programa está à equação PID no plano Z e a atuação do ensaio com realimentação com relé.

Os testes foram iniciados utilizando o método 2 de Ziegler-Nichols para identificação dos parâmetros do PID, pois se trata de um sistema que opera em malha fechada, não sendo possível o uso do método 1.

O controle PID é constituído por três componentes principais:

P – Controle Proporcional. A saída varia em função do erro entre o valor do processo e o valor desejado.

I – Controle Integral. A saída varia em função do tempo de correção do erro até o valor desejado.

D – Controle Derivativo. A saída varia em função da variação do erro.

Quanto maior a variação, maior a resposta, sendo ideal para amortecer picos e saltos.

O mais simples dos três controles é o proporcional. Ele tem sua saída baseada na variação do erro entre a posição atual e a posição alvo.

A fórmula para a atuação Proporcional é:

$$Atua\_prop = Kp * (Erro\_atual - Erro\_anterior)$$

(Equação 4.12)

A fórmula para Erro_atual é:

$$Erro\_atual = Resultado - SetPoint$$

(Equação 4.13)

Atua_prop é o resultado, K_P é a constante proporcional e Erro_ Atual é o valor do erro atual entre o valor desejado e o valor do processo. O controle P é simplesmente uma multiplicação do erro pelo ganho. Quanto maior o ganho, uma maior reação obtém-se por unidade de erro.

Isto gera um problema chamado "estado estável de Erro", o que torna impróprio para a maioria das correções de forma independente. Normalmente, é onde o controle integral é utilizado para a correção deste erro. Quanto mais tempo se leva para atingir o valor desejado, maior será o controle integral, isto em função da soma do erro anterior com o erro atual.

A fórmula para atuação integral é:

$$atua\_int = Ki * Erro\_atual;$$

(Equação 4.14)

O controle derivativo (D), às vezes chamado de controle delta, isto porque é aumentada em função da mudança ou delta do Erro_atual. Como tal, ele pode ser usado para reagir a mudanças repentinas no erro, e é bom para manter uma determinada posição ou velocidade em um sistema de ciclo fechado, assim para torná-lo mais preciso são utilizados dois erros anteriores a ele próprio.

A fórmula atuação derivativa é:

$$atua\_deriv = Kd*(Erro\_atual - 2 * Erro\_anterior + Erro\_ante\_ante)$$

(Equação 4.15)

O controle PID é simplesmente a combinação ou somatório dos resultados de todas as três fórmulas.

Atuacao = Atuacao_anterior + atua_prop + atua_int + atua_deriv

(Equação 4.16)

A implementação do algoritmo de controle PID está descrito na figura 4.9.

usercode *

```
 // Calculo do Erro.
Erro_atual = resultado - SetPoint; //ERRO ATUAL

// Calculo do atuacao Proporcional.
atua_prop = Kp * (Erro_atual - Erro_anterior);   // Multiplica erro pelo ganho Proporcional.

//Calculo do atuacao Integral.
atua_int = Ki* Erro_atual;

// Calculo da atuacao Derivativa.
atua_deriv = Kd*(Erro_atual-2*Erro_anterior+Erro_ante_ante);

 // Calculo do atuacao.
Atuacao = Atuacao_anterior + atua_prop + atua_int + atua_deriv;

//Atualizacao das variaveis
Atuacao_anterior = Atuacao;
Erro_ante_ante = Erro_anterior;
Erro_anterior = Erro_atual;
```

Figura 4-8 – Algoritmo do PID.

4.3. *Descrição da Função Principal do Software*

No primeiro bloco do programa é apresentada a habilitação do conversor A/D do microcontrolador, cuja função é ler a tensão gerada pelo canal analógico do sensor ultra-som e em seguida converter o resultado e mostrar a posição da esfera de isopor no display, sendo que a rotina do mesmo é apresentada na figura 4.9, sendo que sua implementação está representada no diagrama em blocos na Figura 4.10.

usercode *

```
PLL_init();
inicializa_LCD();         //INICIALIZAÇÃO DO LCD
comando_LCD(0x80);        //POSICIONA LCD NA LINHA 0 COLUNA 0
escreve_frase_LCD("TCC Gilmar 2011.");    //ESCREVE FRASE TCC Gilmar 2011

            sprintf(buf, "POS=%3.1f ER=% 3.1f", resultado, Erro_atual);//CONVERTE EM STRING
            comando_LCD(0xC0);        //POSICIONA CURSOR NA LINHA 1 COLUNA 9
            escreve_frase_LCD((const unsigned char *)buf);//ENVIA STRING PARA O LCD
            sprintf(buf1, "=%3.1f\r\n", resultado);
```

Figura 4-9 – Inicialização e escrita do LCD.

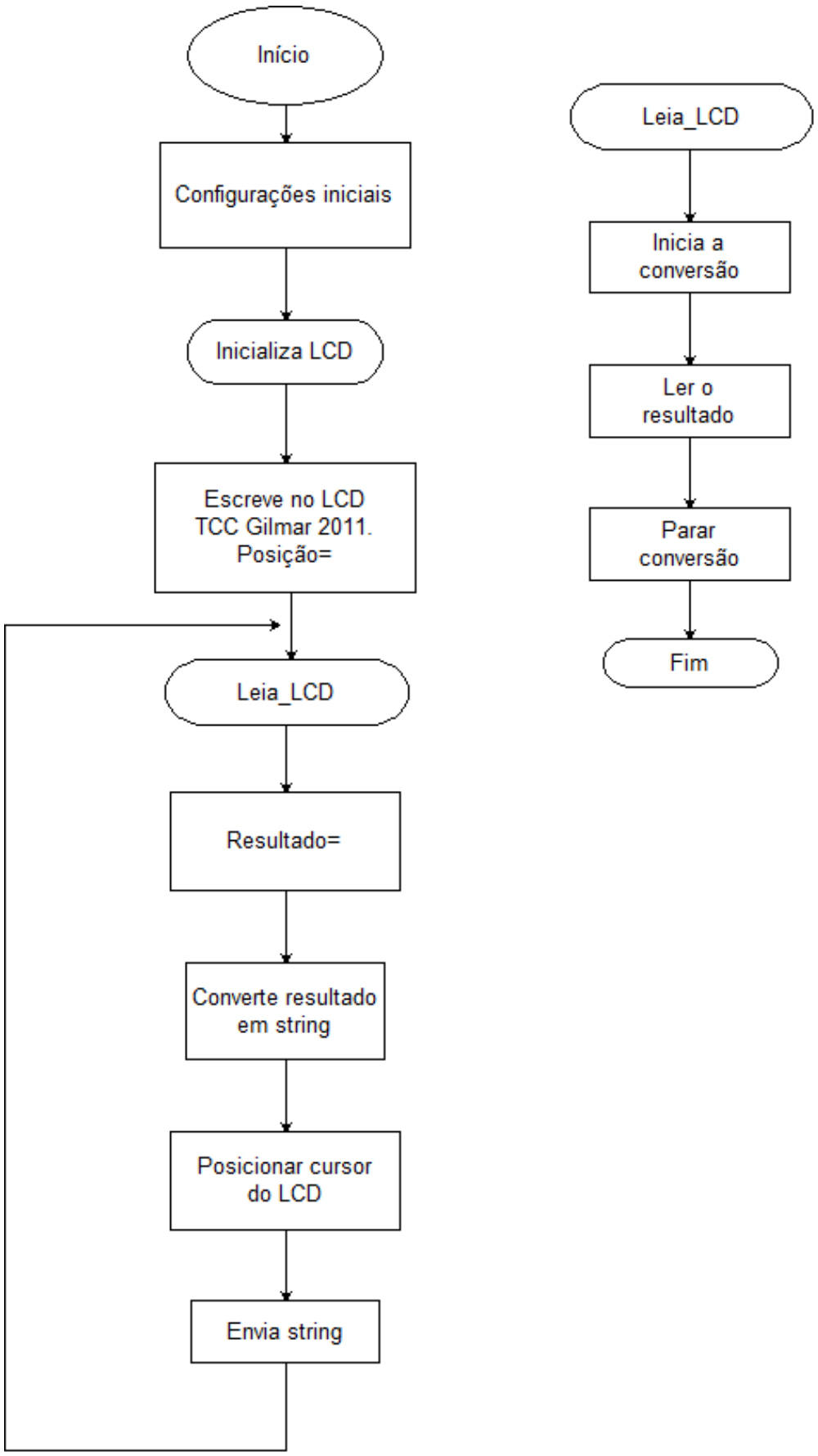


Figura 4-10 – Diagrama em blocos do software A/D e LCD.

Para verificar o funcionamento correto do conversor A/D, foi aplicada uma tensão no ventilador com a utilização de uma fonte simétrica, obtendo a resposta da posição esperada em um display conforme Figura 4.11.

Figura 4-11 – Posição e Erro do Flutuador.

4.4. Resultados dos Testes Realizados

Foram realizados diversos testes com o controlador em modo proporcional. Estes testes permitiram a validação das rotinas de aquisição do sensor através do conversor A/D, assim como a análise e atuação em malha fechada. Utilizando o segundo método de Ziegler e Nichols.

4.4.1. Testes com o Método 2 de Ziegler e Nicholds

Conforme visto no capítulo 2, para estabelecer as características de controle o processo é colocado em laço fechado com atuação proporcional e o valor do ganho é aumentado progressivamente, o que a certa altura provocará a oscilação do processo. O ganho necessário para causar esta oscilação é chamado ganho crítico (K_{cr}) do processo e o período da oscilação observada é dito o seu período crítico (P_{cr}).

Verificou-se que a saída não oscila para qualquer valor de ganho aplicado. O ganho crítico K_{cr}, e o correspondente período P_{cr} foram, então, obtidos experimentalmente através de um ensaio de realimentação a relé (controle bang-bang).

Em um processo de controle com relé a variável controlada oscila em torno do valor de referência e a variável manipulada chaveia periodicamente entre seus

dois valores possíveis conforme mostrado no gráfico com variação de 20 a 40 polegadas conforme Figura 4.4.

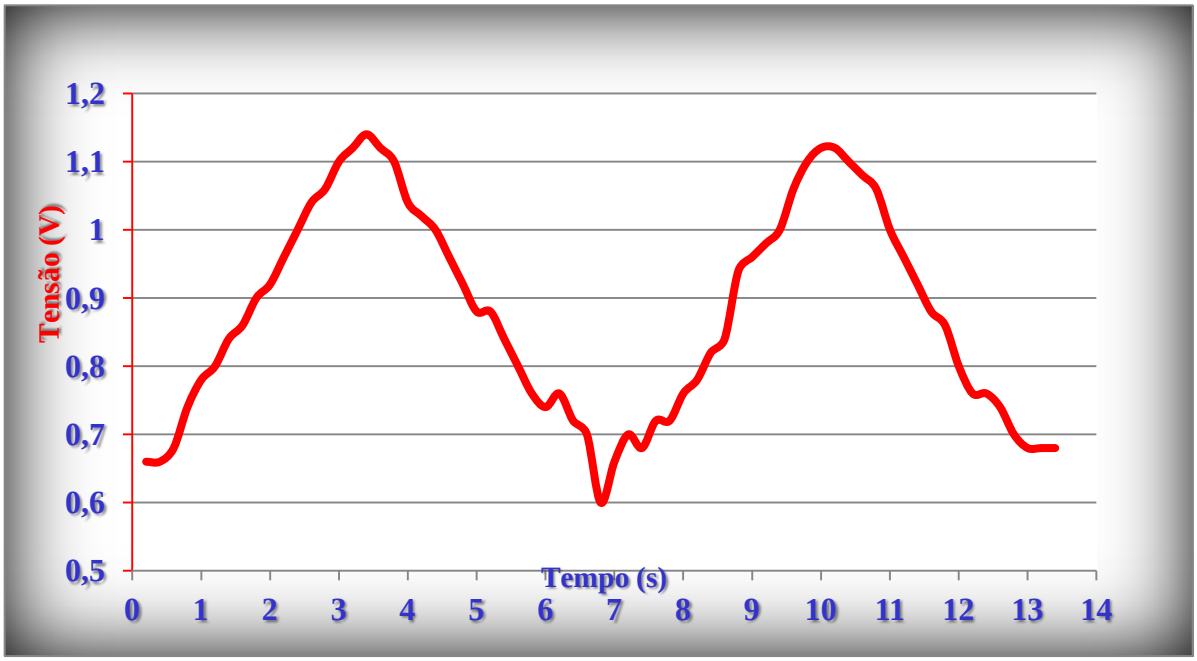


Figura 4-12 – Oscilação Forçada da Planta.

Uma primeira característica da oscilação a ser analisada é sua simetria. Por simetria entende-se entre os tempos em que a variável manipulada fica em seu valor máximo e mínimo: a oscilação da variável controlada será simétrica com relação ao valor de referência. Se essa oscilação é simétrica, então seu período é igual ao período crítico do processo e o ganho crítico pode ser calculado pela equação 4.6.

$$K_{cr} = \frac{4d}{\pi A}$$

(Equação 4.17)

Com a aplicação do método foi verificado que a planta oscilou de fato entre as posições de 19 e 38 polegadas, sendo que o valor de referência era entre 20 e 40 polegadas, tendo uma pequena diferença devido às características da mesma.

Utilizando as características do conversor D/A de 10 bits e a amplitude da posição obtida foi possível calcular o valor de K_{cr} conforme equação 4.18.

$$K_{cr} = \frac{4(1023 - 0)}{\pi(38 - 19)} = 68{,}5538$$

(Equação 4.18)

Após obter os parâmetros da realimentação a relé e validar o tempo de atuação do sistema, foram realizados os cálculos com aplicação dos valores nas fórmulas de Ziegler e Nichols para ajuste pelo método do período crítico conforme tabela 2.2 vista no capitulo 2.

Aplicando os valores obtidos nas equações anteriores foram determinados os parâmetro conforme tabela 4.1.

Tabela 4-1 – Valores obtidos após aplicação do método.

	T	Kcr	Pcr	K	Ti	Td	Kp	Ki	Kd
P	0,028	68,5538	6,51	34,276	-	-	34,276	-	-
PI	0,028	68,5538	6,51	27,42	5,208	-	27,346	0,1474	-
PID	0,028	68,5538	6,51	41,132	3,255	0,8138	40,955	0,3538	1195,3

Os valores calculados foram inseridos no sistema para possibilitar a atuação proporcional, integral e derivativa, sendo que foram obtidos os resultados de forma individual para cada controle.

Os resultados dos ensaios foram analisados através de um osciloscópio, que permitiu obter todos os valores referentes às respostas fornecidas pelo sistema de controle. O osciloscópio possui um recurso que permite salvar os dados coletados em uma planilha Excel, a fim de usar os valores e criar gráficos para visualizar as respostas obtidas.

Após coletar os valores e realizar os cálculos necessários para estabelecer os parâmetros, os mesmos foram inseridos no sistema para possibilitar a atuação proporcional, integral e derivativa, sendo que as características das respostas foram obtidas de forma individual para cada controle. Os resultados serão apresentados de acordo com as medições realizadas utilizando um osciloscópio e também em forma de gráficos conforme demonstrado a seguir.

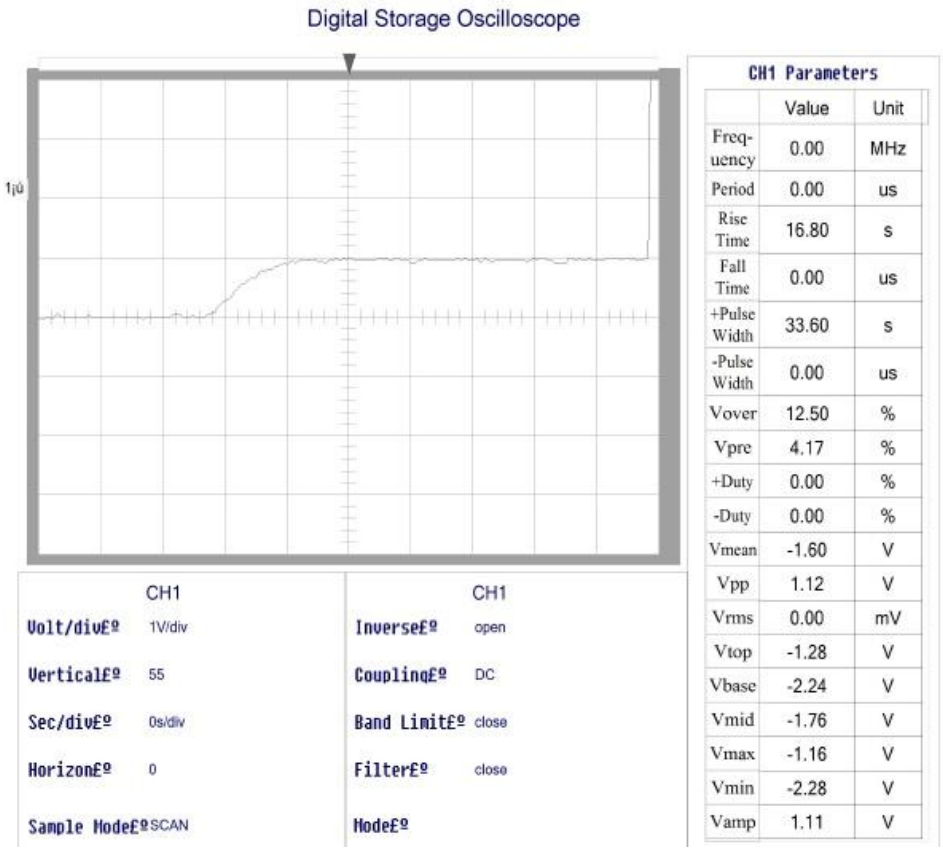


Figura 4-13 – Leitura do Osciloscópio com atuação proporcional (P).

As figuras 4.13 e 4.14 mostram as características com a aplicação da atuação proporcional, sendo possível verificar um erro em regime permanente significativo.

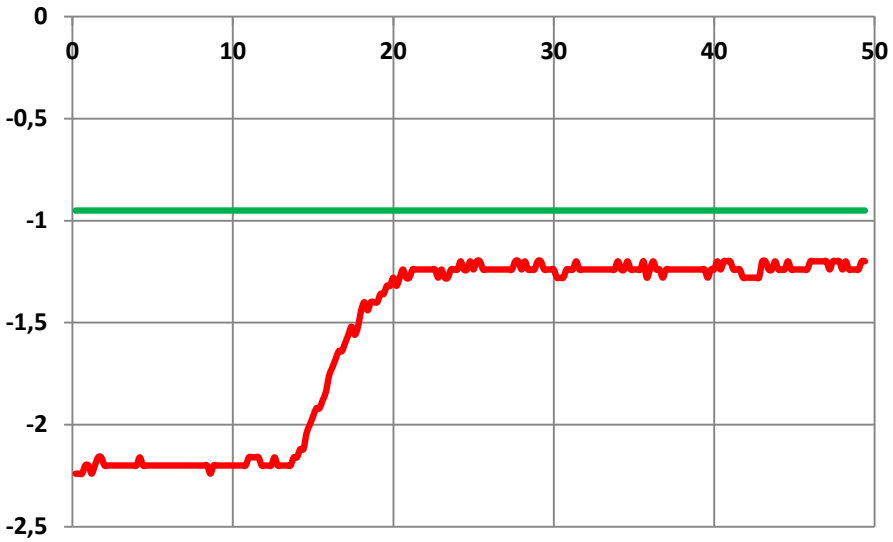


Figura 4-14 – Gráfico do controle proporcional (P).

As Figuras 4.15 e 4.16 mostram as características obtidas com a aplicação da atuação proporcional + integral.

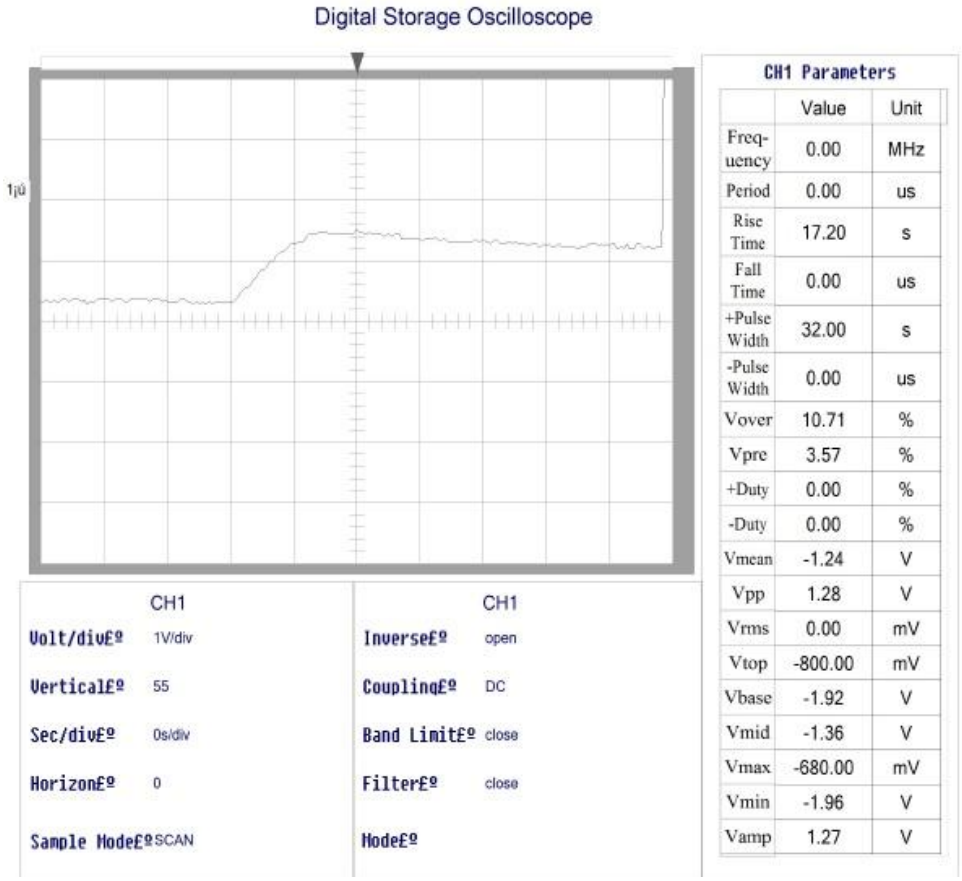


Figura 4-15 – Leitura do Osciloscópio com atuação proporcional + integral (PI).

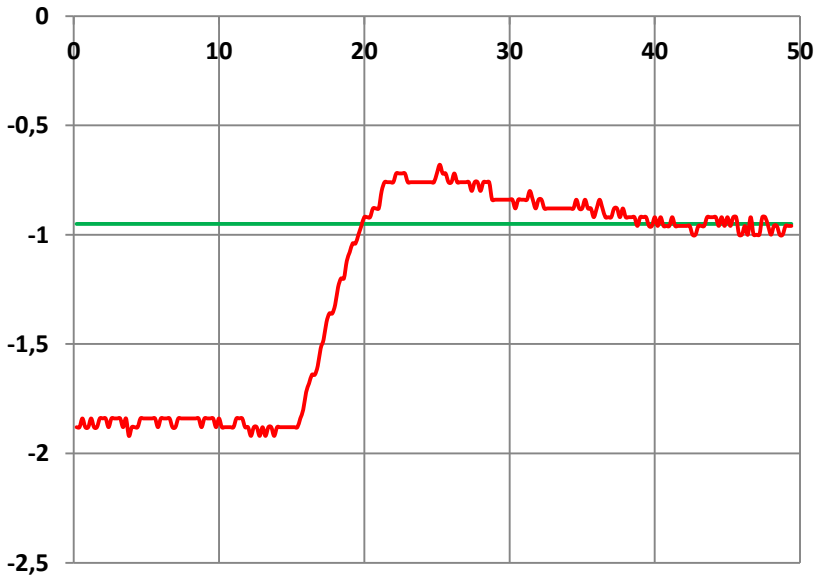


Figura 4-16 – Gráfico do controle PI.

Analisando a resposta do sistema utilizando a atuação de controle PI (Figura 4.16), é possível verificar um *Overshoot* na faixa de 11% e um tempo de estabilização de aproximadamente 25 segundos.

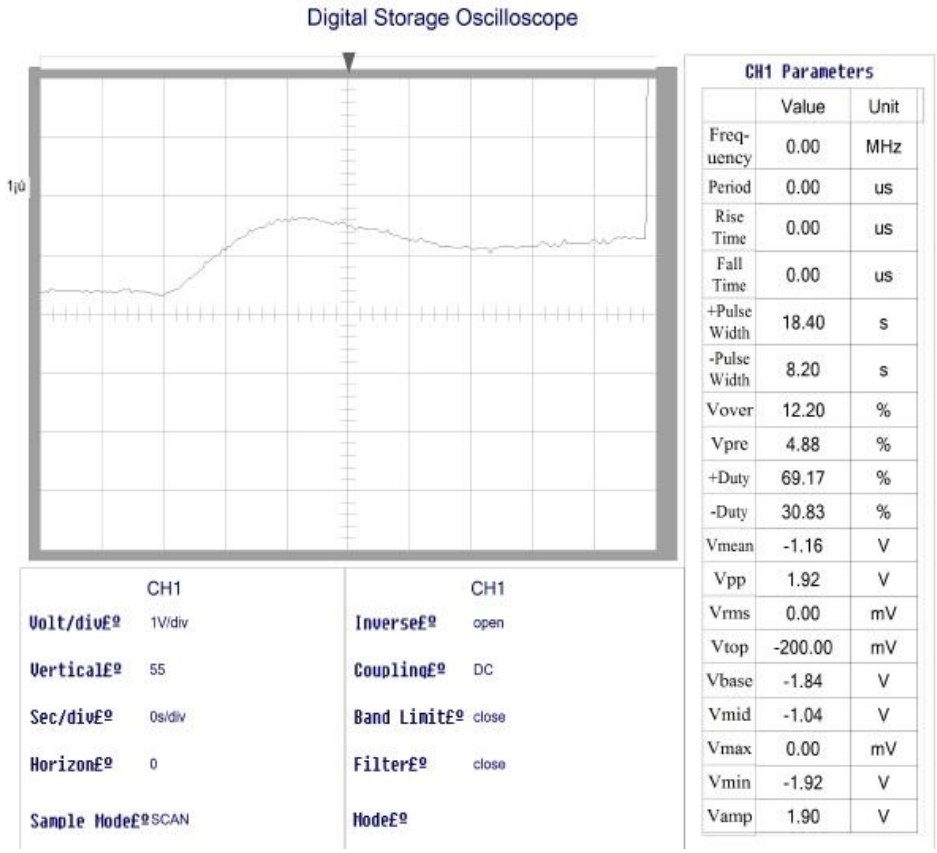


Figura 4-17 – Leitura do Osciloscópio atuação proporcional + integral + Derivativo (PID).

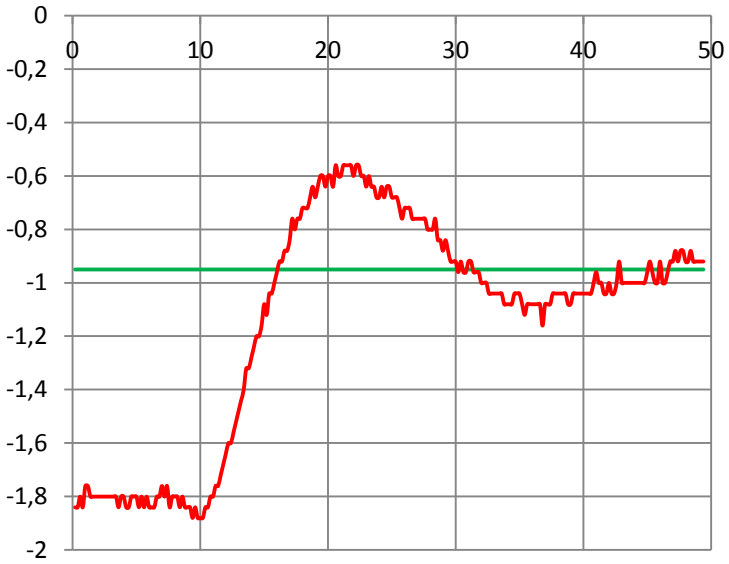


Figura 4-18 – Gráfico do controle PID.

Analisando a resposta do sistema utilizando a atuação de controle PID (Figuras 4.17 e 4.18), é possível verificar um *Overshoot* na faixa de 22% e um tempo de estabilização de aproximadamente 35 segundos.

Os resultados obtidos até aqui foram considerados satisfatórios em relação à proposta do projeto apresentada, cujo objetivo foi realizar o controle em um ponto pré determinado. No entanto, foram realizados testes complementares alterando os valores de K_P, K_I e K_D com o objetivo de verificar as mudanças nas características da resposta do sistema de controle.

4.4.2. Ajuste Manual

Os modelos utilizados para o projeto nem sempre são suficientemente completos e os métodos, por se tratar de aplicação genérica, muitas vezes fornecem ajustes que podem ser melhorados. Por esta razão, por vezes é conveniente, após ter obtido um ajuste para o PID por meio de um dos métodos apresentados, efetuar manualmente um *ajuste fino* dos parâmetros do controlador tendo em conta o desempenho observado do sistema.

Para tanto é preciso saber o efeito de cada uma das ações de controle sobre o desempenho do processo, sendo que a tabela abaixo apresenta um sumário que pode servir de guia ao operador efetuando o ajuste manual, sempre tendo em mente que este ajuste manual tem por objetivo unicamente refinar o ajuste já feito do controlador, portanto, as variações efetuadas nos parâmetros devem ser pequenas [6].

Tabela 4-2 – Guia para Ajuste Manual [6].

Problema	Medida de Ajuste
Resposta muito Lenta	Aumentar ganho proporcional
Resposta excessivamente Oscilatória	Aumentar tempo derivativo
Sobrepassagem	Reduzir taxa integral
Resposta inicialmente rápida e em seguida muito lenta	Aumentar taxa integral

Utilizando a tabela 4.2, foram realizados alguns testes atribuindo valores de forma a possibilitar melhorias que podem ser obtidas realizando um ajuste fino, sendo que os testes realizados estão descritos na tabela 4.3.

Tabela 4-3 – Valores Aplicados nos Testes.

PID	**Kp**	**Ki**	**Kd**
Teste 1	40,955	0,10	2195,3
Teste 2	120	0,1538	1195,3
Teste 3	41	0,056	3000
Teste 4	41	0,02	500

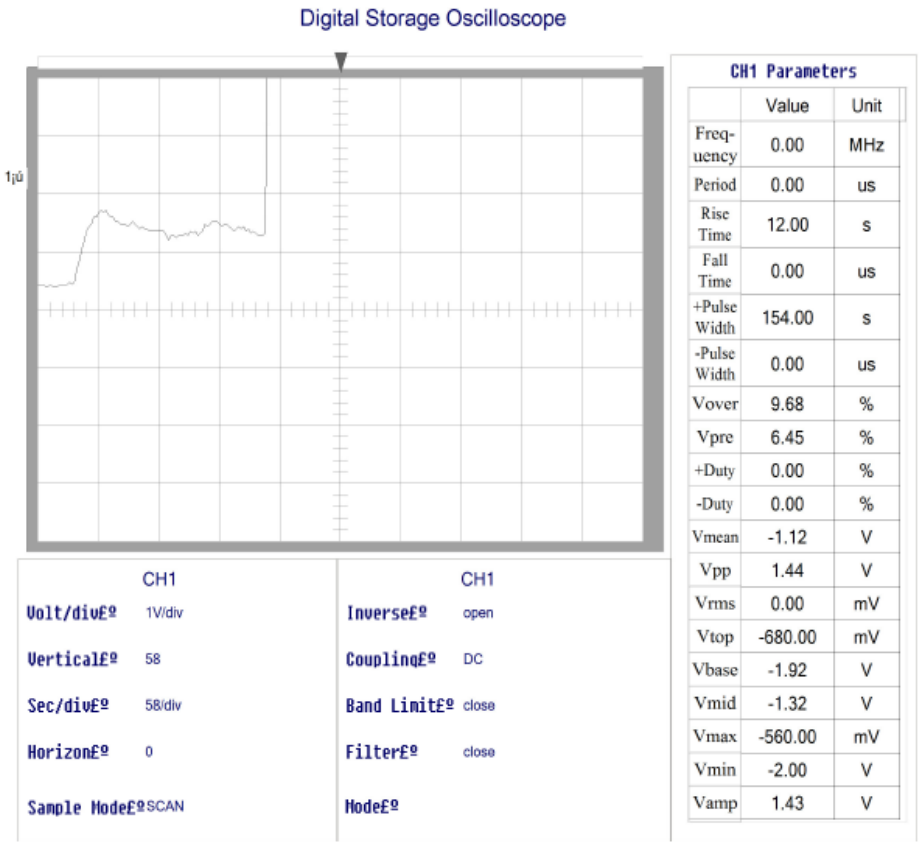


Figura 4-19 – Leitura do Osciloscópio no Teste 1.

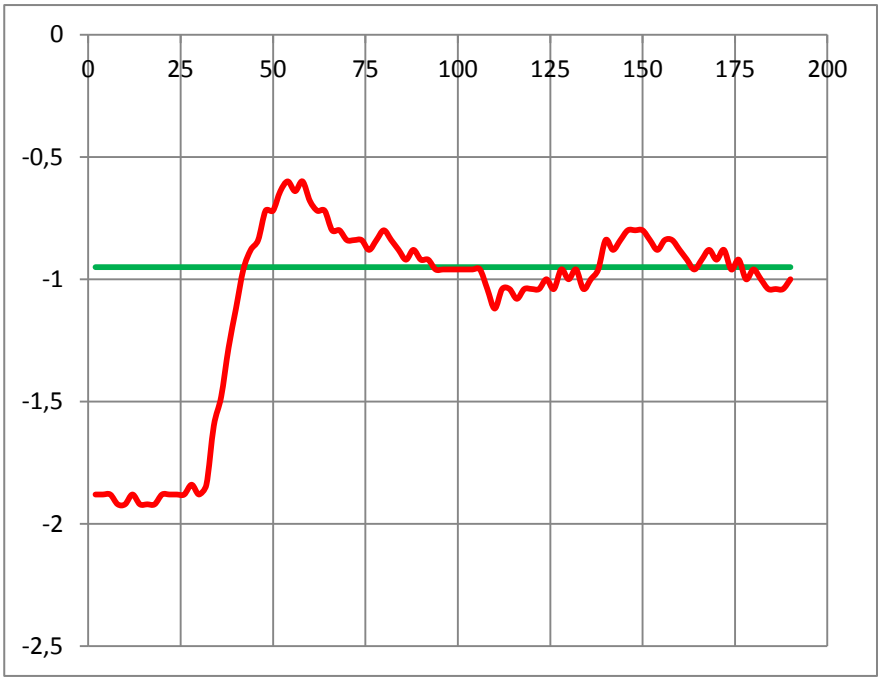


Figura 4-20 – Resposta Teste 1.

Analisando a resposta do teste 1 (Figuras 4.19 e 4.20) com atuação de controle PID, foi possível verificar um *Overshoot* na faixa de 22% e um tempo de estabilização de aproximadamente 38 segundos.

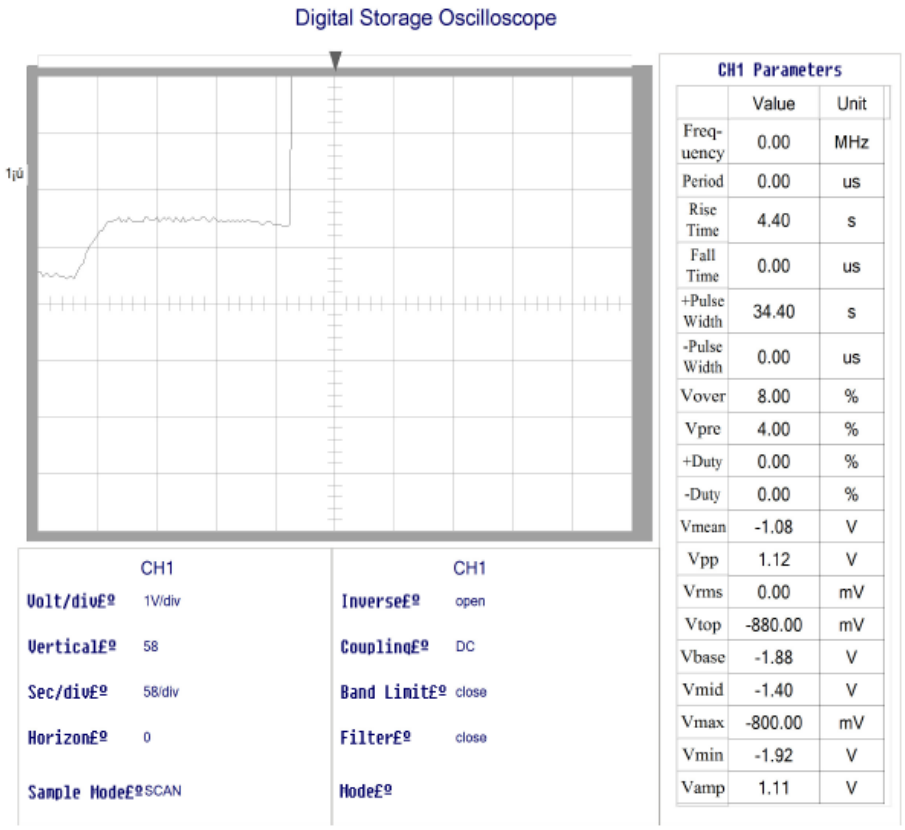


Figura 4-21 – Leitura do Osciloscópio no Teste 2.

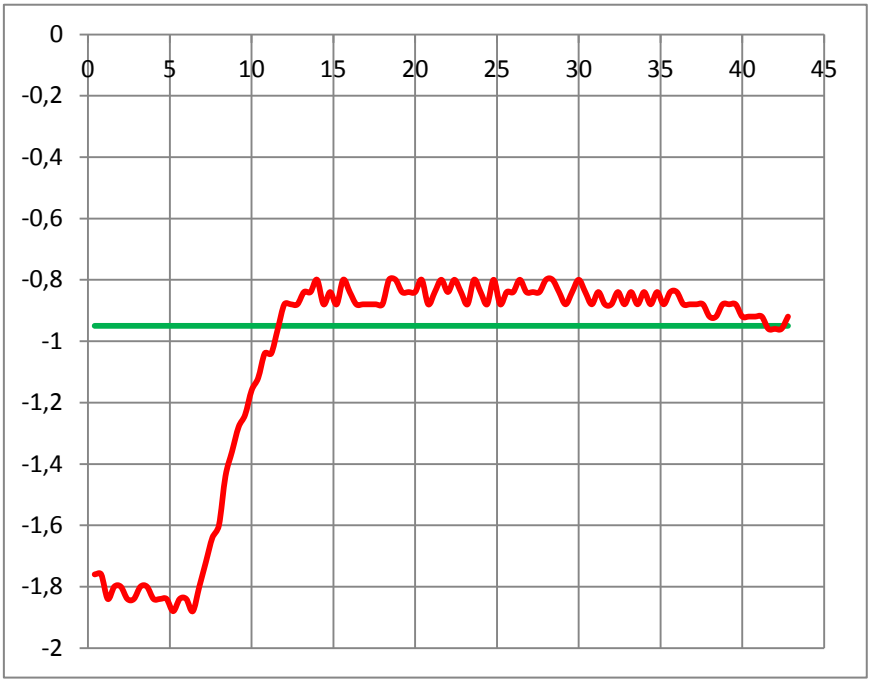


Figura 4-22 – Resposta Teste 2.

Na resposta do teste 2 (Figuras 4.21 e 4.22) com atuação de controle PID, onde verifica-se um *Overshoot* na faixa de 8,5% e um tempo de estabilização de aproximadamente 35 segundos.

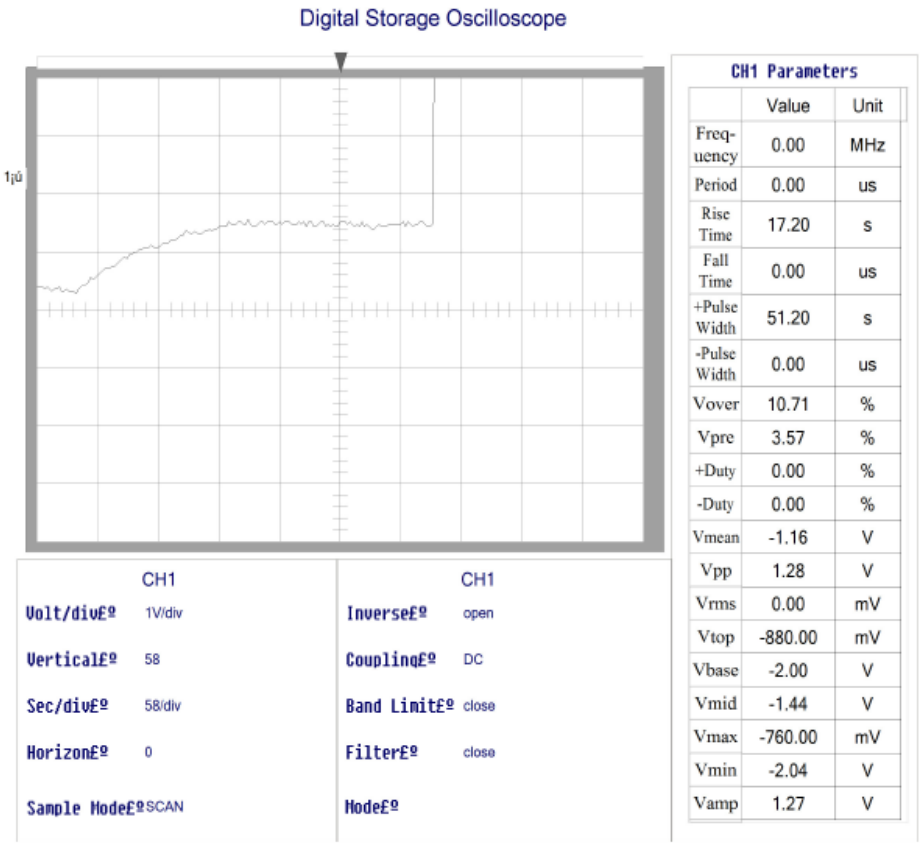


Figura 4-23 – Leitura do Osciloscópio no Teste 3.

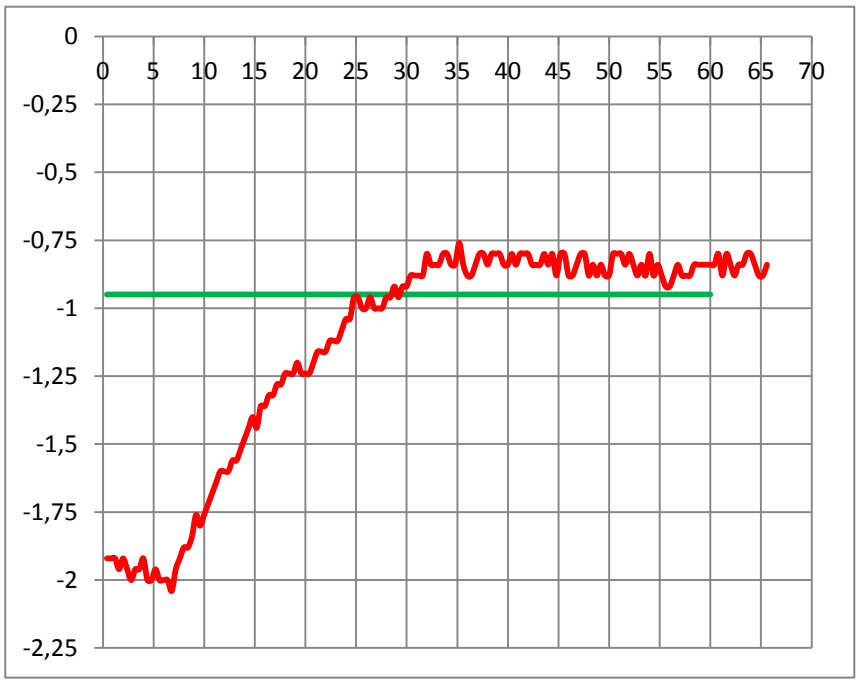


Figura 4-24 – Resposta Teste 3.

No teste 3 (Figuras 4.23 e 4.24) com aplicação de atuação de controle PID, foi possível verificar um *Overshoot* na faixa de 12% e um tempo de estabilização maior que 60 segundos.

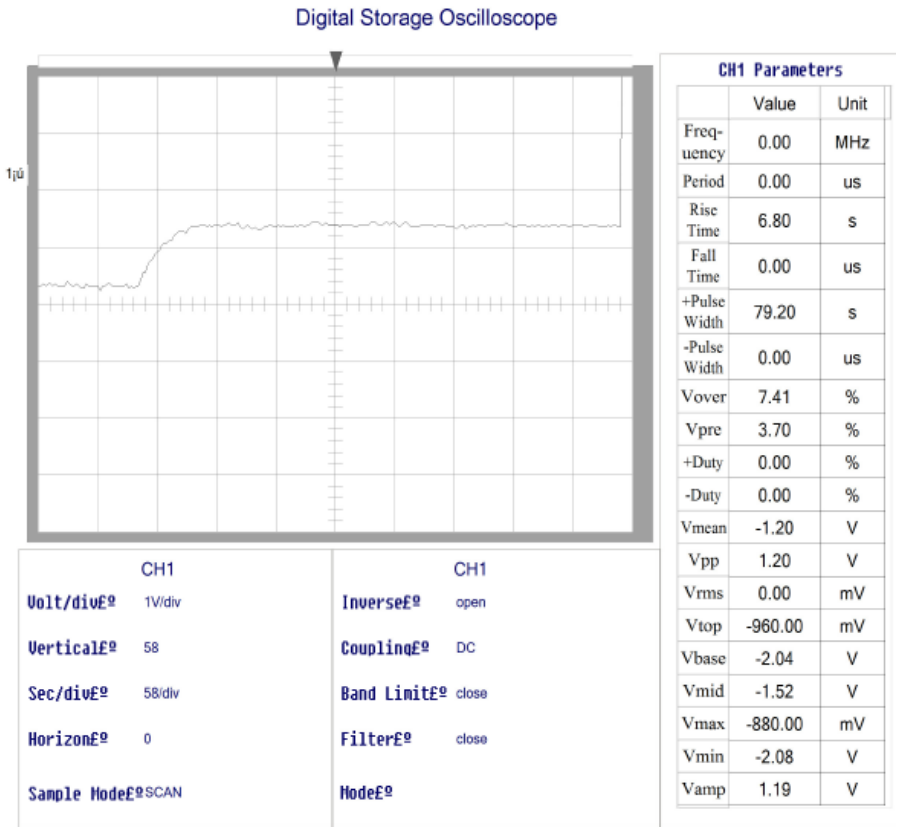


Figura 4-25 – Leitura do Osciloscópio no Teste 4.

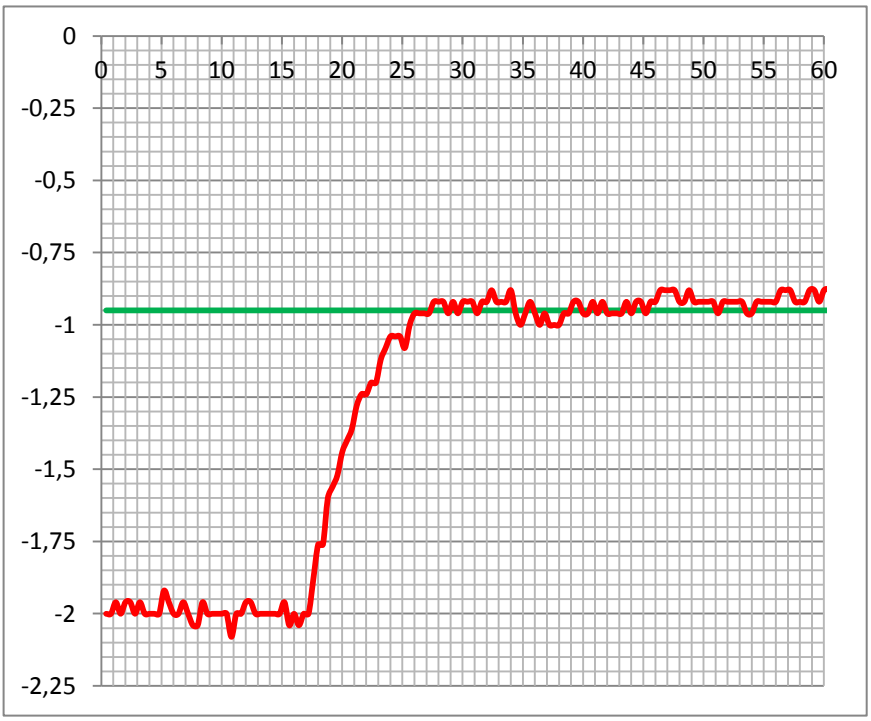


Figura 4-26 – Resposta Teste 4.

No teste 4 (Figuras 4.25 e 4.26) foi obtida uma resposta bastante satisfatória, onde verifica-se um *Overshoot* na faixa de 5% e um tempo de estabilização de aproximadamente 10 segundos.

5. CONSIDERAÇÕES FINAIS

5.1. Conclusões

A proposta deste projeto foi desenvolver um sistema de controle da posição de um flutuador através de um controlador PID, implementado em um kit didático contendo um processador LPC 2138 de 32 bits núcleo ARM7 em linguagem C.

O *hardware* do projeto é composto por um sistema mecânico acoplado a um tubo acrílico transparente com diâmetro interno de 44 milímetros, uma esfera de isopor com diâmetro de 38 milímetros e peso de 0,96 gramas, um ventilador de corrente contínua de 12 volts com dimensões de 12x12x3,8cm, um sensor de posição Maxsonar EZ1 da empresa Tato Equipamentos Eletrônicos e circuitos de acionamento.

Para sintonizar o sistema de controle foram usadas as regras de Ziegler-Nichols. Foram realizados diversos testes no controlador com ajustes P, PI e PID, possibilitando o controle da posição do flutuador suspenso no tubo pela variação de fluxo de ar ascendente, que empurra o flutuador para cima contrabalanceando a força descendente da gravidade.

Os objetivos do projeto deste trabalho de conclusão de curso foram alcançados com êxito, sendo que foram obtidos resultados satisfatórios em relação ao proposto, cujo objetivo era o controle do sistema em um ponto pré determinado, realizando ainda, testes buscando o aperfeiçoamento do sistema e seus parâmetros de respostas em relação aos métodos aplicados.

A partir dos testes realizados, foi comprovado que o método do período crítico em que se eleva o ganho Kp gradualmente até a planta começar a oscilar é pouco eficiente. Com este método não foi possível encontrar um ganho crítico preciso, sendo necessário o uso do método de realimentação por relé (bang-bang), obtendo resultados expressivos com erros de posição em regime permanente inferiores a 6%, porém a aplicação direta do método resulta em um *Overshoot*

bastante alto, mas a partir dos resultados obtidos, foi possível passar a uma etapa de refinamento das soluções com o uso de ajustes finos.

Foram realizados quatro testes com ajustes manuais conforme descrito no item 4.4.2, sendo que o sistema obteve a melhor relação entre o máximo *overshoot* e o tempo de estabilização utilizando um controlador PID. O sistema foi ajustado com ganho proporcional de 41, ganho integral de 0,02 e ganho derivativo de 500, obtendo uma resposta bastante satisfatória, verificando-se um *overshoot* na faixa de 5% e um tempo de estabilização de aproximadamente 10 segundos.

5.2. Trabalhos Futuros

Este sistema permite realizar diversos projetos futuros, iniciando por uma etapa de refinamento dos métodos de controle aplicados, e assim possibilitando o estudo de outras formas de ajuste bem como suas características em suas aplicações. A estrutura mecânica da planta pode ser alterada para possibilitar o uso de ventiladores mais potentes e consequentemente, possibilitar o uso de flutuadores com pesos maiores e características mais apuradas para o sistema.

Criar uma interface gráfica com o usuário que permita a interação total com o sistema em tempo real estabelecendo meios de escolher o tipo de controle a ser aplicado, determinar a posição, gráfico com aquisição das respostas, salvar aquisições, imprimir aquisições, etc. Criar um produto utilizando a interface gráfica, possibilitando sua aplicação em qualquer sistema de controle mudando apenas o *hardware* conforme as características do sistema, possibilitando o ajuste de controle de posição, nível, pressão, temperatura, velocidade, etc.

6. REFERÊNCIAS

[1] MAITELLI, A. L. – Apostila de Controladores Lógicos Programáveis – UFRN, Natal-RN, 2003.

[2] ASTRÖN, K. J.; HÄGGLUND T. – The Future of PID Control. – Control Engineering Pratice, 2001.

[3] OGATA, KATSUHIKO. – Engenharia de Controle Moderno – Editora Prentice Hall do Brasil - RJ, 1998.

[4] RHINEHART, R. R. – Control Modes – PID Variations – In: LIPTAK, B. G. (Edit.) Instrument Engineers' Handbook: Process Control. 3º ed. Boca Raton: CRC, 1999.

[5] BAZANELA, ALEXANDRE SANFELICE; SILVA, JOÃO MANOEL DA – Ajuste de Controladores PID. – WWW.ece.ufrgs.br/~jmgomes/pid/apostila/apostila/.

[6] BAZANELA, ALEXANDRE SANFELICE; SILVA, JOÃO MANOEL DA – Sistemas de Controle: Princípios e Métodos de Projeto. – Editora UFRGS, 1º Edição.

[7] NISE, NORMAN S. – Engenharia de Sistemas de Controle - 3a. Ed. – Rio de Janeiro: Editora LTC – Livros Tecnicos e Cientificos Editora S.A., 2002.

[9] SOUZA, DANIEL RODRIGUES DE. - Microcontroladores ARM7. - Editora Érica - São Paulo - 2006.

[10] RENALDI, FELIPE GENOS - Protótipo De Um Montador De Sistemas Operacionais para Sistemas Embarcados. Santa Catarina: Monografia FURB, 2006. Disponível em: http://www.bc.furb.br/docs/MO/2006/307628_1_1.pdf . Acesso em: maio 2008.

[11] MOORE, C. F. Control Modes – Closed-Loop Response. In: LIPTÁK, B. G. (Edit.) Instrument Engineers' Handbook: Process Control. 3ª ed. Boca Raton: CRC, 1999.

[12] GOODWIN, G. C.; GRAEBE, S. F.; SALGADO, M. E. Control System Design. Upper Saddle River: Prentice Hall, 2001.

[13] COELHO, C. A. D. – Revista Instrumentação & Controle – Valete, Março 2001.

[14] OGATA, K. Discrete Time Control Systems. Prentice Hall, 1987.

7. OBRAS CONSULTADAS

BALBINOT, A.; BRUSAMARELLO, V. J. – Instrumentação e Fundamentos de Medidas, Vol. 1 – Rio de Janeiro: Editora LTC, 2006.

COCIAN, LUIS F. E. – Manual da Linguagem C – 1ª Ed. – Canoas: Editora da Ulbra, 2004.

BOYLESTAD, R.; NASHELSKY, L. – Dispositivos Eletrônicos e Teoria de Circuitos - 3ª Ed. – Rio de Janeiro: PHB, 1982.

TOCCI, R. J.; WIDNER, N. S. – Sistemas Digitais – 7ª Ed. – Rio de Janeiro: Editora LTC, 2000.

APÊNDICE A – CÓDIGO FONTE DO SOFTWARE

```
*******************************************************************************
Trabalho de Conclusão de Curso, Gilmar José Zwirtes, Canoas 2011/2
Sistema de Controle da Posição de um Flutuador Implementado em
Microcontrolador de 32 Bits, (Air Ball).
*******************************************************************************
==========
*******************************************************************************
Trabalho de Conclusão de Curso apresentado ao Departamento de Engenharia
Elétrica da ULBRA como um dos requisitos obrigatórios para a obtenção
do grau de Engenheiro Eletricista.
*******************************************************************************
**=====================================================================
**  2.2     Application include files
**=====================================================================
#include "usercode.h"   /* Usercode macros (see <template.h>) */
#include "ma_tgt.h"     /* Target specific header file */
#include "ma_sfr.h"     /* Special function register bitfield macros */
#include "iolpc2138.h"  /* Defines Special function registers */
**---------------------------------------------------
**  IAR MakeApp peripheral modules header files
**  Include the header files used by the application
**---------------------------------------------------
#include "ma_scb.h"
#include "ma_vic.h"
#include "ma_pcb.h"
#include "ma_gpio.h"
#include "ma_uart0.h"
#include "ma_uart1.h"
#include "ma_i2c.h"
#include "ma_spi.h"
#include "ma_tmr.h"
#include "ma_pwm0.h"
#include "ma_adc.h"
#include "ma_rtc.h"
#include "ma_wdt.h"
**=====================================================================
**  3.2     Internal macros
**=====================================================================
#define   LIGA_LED1(void)     MA_WritePort_GPIO(0,0x00001000,0x00001000)
#define   DESLIGA_LED1(void)  MA_WritePort_GPIO(0,0x00000000,0x00001000)
**=====================================================================
**  3.5     Internal function prototypes (defined in Section 5)
**=====================================================================
void escreve_frase_UART0(const unsigned char *frase);
unsigned short leia_adc(unsigned char canal);
void delay_1ms_x(unsigned int x);
void delay_10us_x(unsigned int x);
void PLL_init(void);
void pulso_enable(void);
void seleciona_dado(void);
void seleciona_comando(void);
void envia_byte(unsigned char dado);
void limpar_LCD(void);
void dado_LCD(unsigned char dado);
void comando_LCD(unsigned char dado);
void escreve_frase_LCD(const unsigned char *frase);
void inicializa_LCD(void);
void executaPID(void);
void liga_leds(void);
void seta_leds(void);
**=====================================================================
**  3.6     Internal variables
**=====================================================================
char buffer_lcd[16];
unsigned char t, n, m;
unsigned char x;
```

```
unsigned char rx_byte;
unsigned char rx_status;
float resultado, Medida, Erro_atual= 0, Erro_anterior=0, Erro_ante_ante=0, SetPoint, Kp, Ki, Kd, ATant;
float  atua_prop=0, atua_int=0, atua_deriv=0, Atuacao = 0, Atuacao_anterior = 0;
unsigned int  Atuador=0;
char  buf[17];
char  buf1[17];
unsigned int i, i2;
union port
{
   struct
   {
      unsigned int :16;
      unsigned int BIT16:1;
      unsigned int BIT17:1;
      unsigned int BIT18:1;
      unsigned int BIT19:1;
      unsigned int :12;
   } ;
   struct
   {
      unsigned int BITS;
   } ;
};
union port PORT_COPY;
#define BOTAO_1   PORT_COPY.BIT16
#define BOTAO_2   PORT_COPY.BIT17
#define BOTAO_3   PORT_COPY.BIT18
#define BOTAO_4   PORT_COPY.BIT19
**=====================================================================
**  4.      GLOBAL FUNCTIONS (declared as 'extern' in some header file)
**=====================================================================
void main( void )
{
   MA_Init_SCB();
   MA_Init_PCB();
   MA_Init_GPIO();
   MA_Init_UART0();
   MA_Init_TIMER();

   PLL_init();
   inicializa_LCD();        //INICIALIZAÇÃO DO LCD
   comando_LCD(0x80);       //POSICIONA LCD NA LINHA 0 COLUNA 0
   escreve_frase_LCD("TCC Gilmar 2011.");    //ESCREVE FRASE TCC Gilmar 2011

   ENTER_MA_INIT_DAC;
   PINSEL1 |= 0x00080000;      // Habilita  D/A
   EXIT_MA_INIT_DAC;

      //valores das constantes
     SetPoint =30;
     Kp = 40.955;
     Kd = 1195.3;
     Ki = 0.3538;

  while( 1 )
      {
      PORT_COPY.BITS = MA_ReadPort_GPIO(0);
     if(!BOTAO_1)         //TESTA BOTÃO BT1
      {                   //QUANDO BOTÃO BT1 PRESSIONADO, VAI PARA POSIÇÂO 60 POLEGADAS
         SetPoint = 60;
      }
      if(!BOTAO_2)         //TESTA BOTÃO BT1
      {                   //QUANDO BOTÃO BT2 PRESSIONADO, VAI PARA POSIÇÂO 30 POLEGADAS
         SetPoint = 30;
      }
       LIGA_LED1();  //LIGA LED 1
         sprintf(buf1, "%3.1f\r\n", resultado);
         escreve_frase_UART0((const unsigned char *)buf1);

         sprintf(buf, "POS=%3.1f ER=% 3.1f", resultado, Erro_atual);//CONVERTE EM STRING
         comando_LCD(0xC0);       //POSICIONA CURSOR NA LINHA 1 COLUNA 9
         escreve_frase_LCD((const unsigned char *)buf);//ENVIA STRING PARA O LCD
         sprintf(buf1, "=%3.1f\r\n", resultado);

        //leitura do AD (float)leia_adc(0);
```

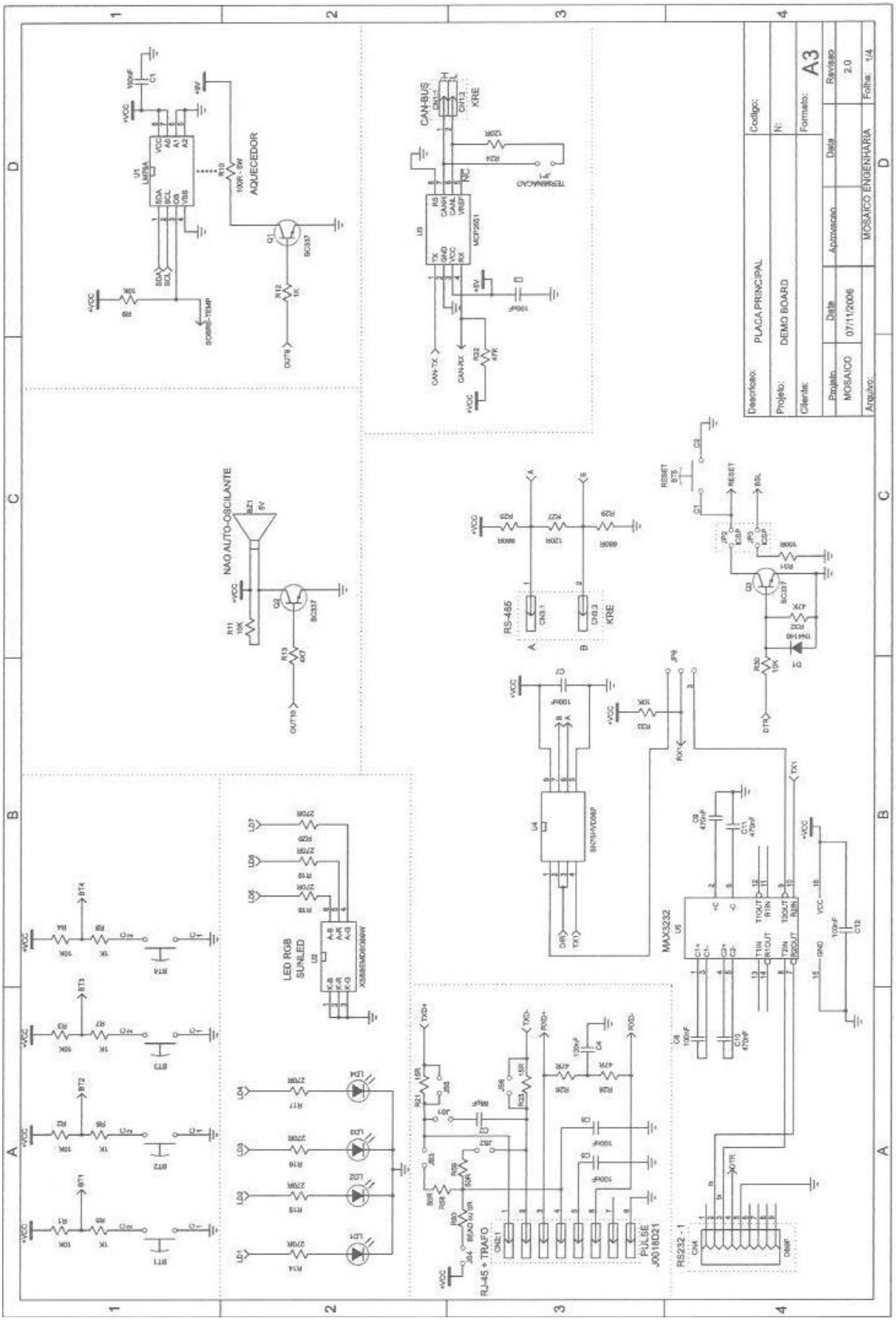
```
        resultado = (float)leia_adc(0);  //Chama a funcao de leitura do A/D.
        resultado = (resultado)/10; //Converte valor do A/D em valores de 0 a 74 polegadas.

         // Calculo do Erro.
        Erro_atual = resultado - SetPoint; //ERRO ATUAL

        // Calculo do atuacao Proporcional.
        atua_prop = Kp * (Erro_atual - Erro_anterior);   // Multiplica erro pelo ganho Proporcional.

        //Calculo do atuacao Integral.
        atua_int = Ki* Erro_atual;

        // Calculo da atuacao Derivativa.
        atua_deriv = Kd*(Erro_atual-2*Erro_anterior+Erro_ante_ante);

        // Calculo do atuacao.
       Atuacao = Atuacao_anterior + atua_prop + atua_int + atua_deriv;
       //Atuacao = atua_prop;

        if (Atuacao > 1023)
          Atuador = 1023;

        if (Atuacao < 0)
          Atuador = 0;

        if ((Atuacao>=0) && (Atuacao<=1023))
          Atuador = (unsigned int)Atuacao;

        ENTER_MA_SETVALUE_DAC;
        DACR_bit.VALUE = Atuador;
        EXIT_MA_SETVALUE_DAC;
           //Atualizacao das variaveis
       Atuacao_anterior = Atuacao;
       Erro_ante_ante = Erro_anterior;
       Erro_anterior = Erro_atual;

        delay_1ms_x(20);
        DESLIGA_LED1();
        }
   }
**=====================================================================
**        PLL_init()
**
**        Configura o LPC2138 para trabalhar com Cclk = 40MHz e Pclk = 20MHz
**        com a frequência do cristal de 20MHz
**=====================================================================
void PLL_init(void)
{
   PLLCFG  = 0X00000021;   //Configura M = 2 e P = 2 para uma frequência
                  //de 40MHz com cristal de 20MHz
   PLLCON  = 0X00000001;   //Habilita PLL
   PLLFEED = 0X000000AA;
   PLLFEED = 0X00000055;   //Atualiza os registradores
   while(!(PLLSTAT & 0X00000400)); //Testa PLOCK
   PLLCON  = 0X00000003;   //Conecta ao PLL
   PLLFEED = 0X000000AA;
   PLLFEED = 0X00000055;   //Atualiza os registradores
   VPBDIV  = 0X00000002;   //Configura Pclk = 20MHz (Cclk/2)
}
**=====================================================================
**        unsigned short leia_adc(unsigned char canal)
**
**        Efetua uma conversão AD no conversor 0, passando número do canal
**        para leitura.
**=====================================================================
unsigned short leia_adc(unsigned char canal)
{
   unsigned short  resultado;
   signed char     ret_code;    // INICIA CONVERSÃO
   MA_Start_ADC(MA_AD_CONVERTER_0, MA_AD_SOFTWARE_MODE,      AD0CR_START_START_NOW,
MA_AD_CHANNEL_0);  // LER O RESULTADO DA CONVERSÃO
   ret_code = MA_EMPTY;
   while(ret_code == MA_EMPTY) ret_code = MA_Read_ADC(MA_AD_CONVERTER_0, &canal, &resultado);
   if(ret_code != MA_OK) resultado = 0;   // PARAR CONVERSÃO
   MA_Stop_ADC(MA_AD_CONVERTER_0);
   return(resultado);  //RETORNA O VALOR CONVERTIDO
```

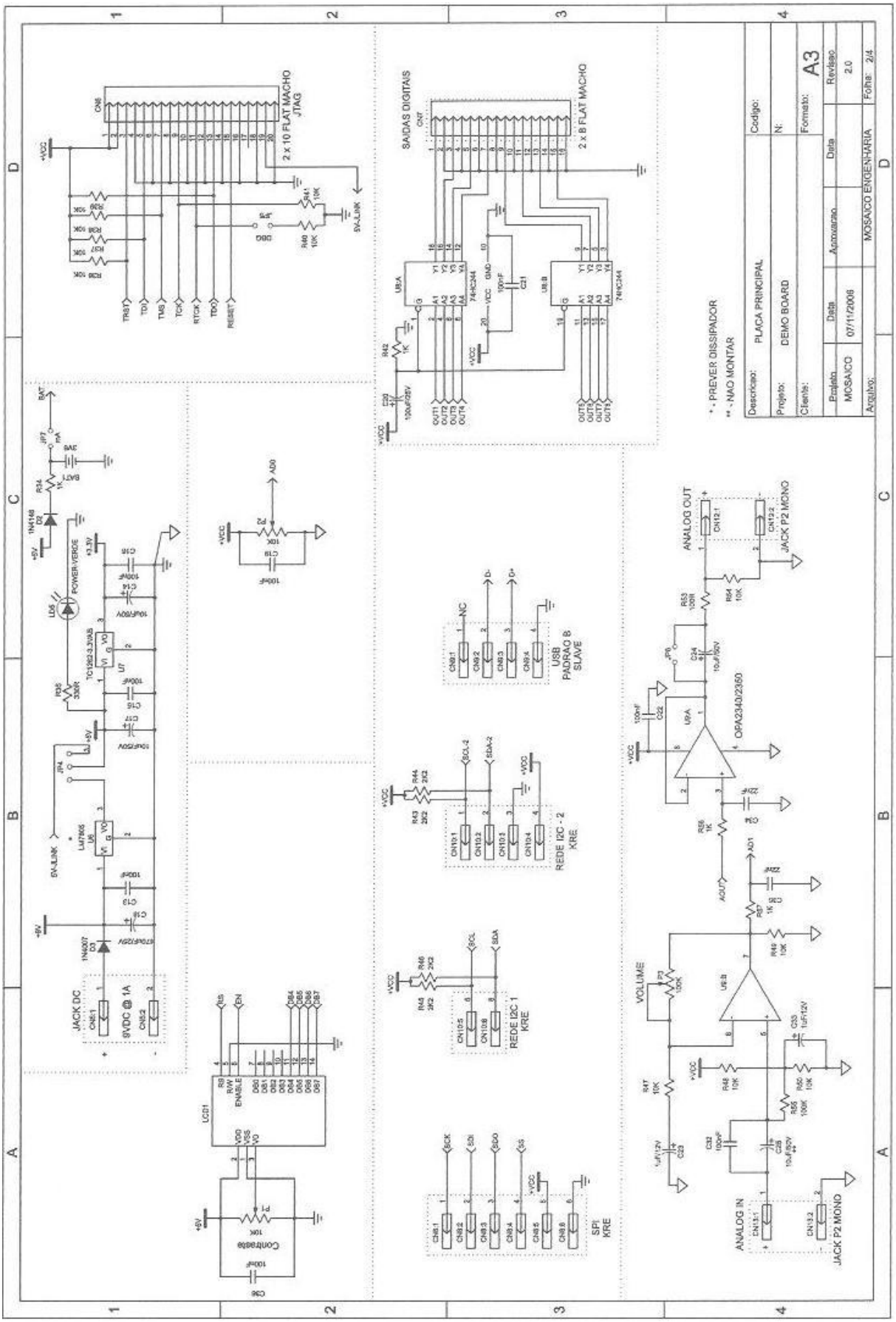
```
}
**=====================================================================
**        delay_1ms_x(unsigned char x)
**
**        Gera delay de múltiplos de 1 milisegundo
**
**        OBS: O Timer 0 deve ser configurado com Prescaler de 100
**=====================================================================*/
void delay_1ms_x(unsigned int x)
{
   T0MR0 = x * 0x000000C8;   //CARREGA T0MR0
   T0IR |= 0x00000001;       //LIMPA FLAG DO T0MR0
   MA_Start_TIMER(0,1);      //LIGA TIMER
   while(!(T0IR & 0x00000001));//SE FLAG DO T0MR0 FOR 1, SAI DA FUNÇÃO
}
**=====================================================================
**        delay_10us_x(unsigned char x)
**
**        Gera delay de múltiplos de 10 microsegundos
**
**        OBS: O Timer 0 deve ser configurado com Prescaler de 100
**=====================================================================
void delay_10us_x(unsigned int x)
{
   T0MR0 = x * 0x00000002;   //CARREGA T0MR0
   T0IR |= 0x00000001;       //LIMPA FLAG DO T0MR0
   MA_Start_TIMER(0,1);      //LIGA TIMER
   while(!(T0IR & 0x00000001));//SE FLAG DO T0MR0 FOR 1, SAI DA FUNÇÃO
}
**=====================================================================
**=====================================================================
**
**        FUNÇÕES DE CONTROLE DO LCD 16 X 2
**
**=====================================================================
**=====================================================================
**        void  pulso_enable(void)
**
**        Gera pulso no pino ENABLE do LCD
**=====================================================================
void  pulso_enable(void)  //GERA PULSO NO PINO DE ENABLE
{
   IO1SET = 0x00040000;  //SETA PINO DE ENABLE DO LCD
   delay_10us_x(1);      //DELAY DE 10us
   IO1CLR = 0x00040000;  //LIMPA PINO DE ENABLE DO LCD
}
**=====================================================================
**        void  seleciona_dado(void)
**
**        Configura o LCD para o envio de dados (caracteres)
**=====================================================================
void  seleciona_dado(void)  //SELECIONA DADO
{
   IO1SET = 0x00080000;    //SETA PINO RS DO LCD
}
**=====================================================================
**        void  seleciona_comando(void)
**
**        Configura o LCD para o envio de comandos
**=====================================================================
void  seleciona_comando(void) //SELECIONA COMANDO
{
   IO1CLR = 0x00080000;      //LIMPA PINO RS DO LCD
}
**=====================================================================
**        void  envia_byte(unsigned char dado)
**
**        Envia um byte para o LCD
**=====================================================================
void  envia_byte(unsigned char dado)
{
   union lcd_1
   {
      struct
      {
         unsigned char  NIBBLE_BAIXO:4;
```

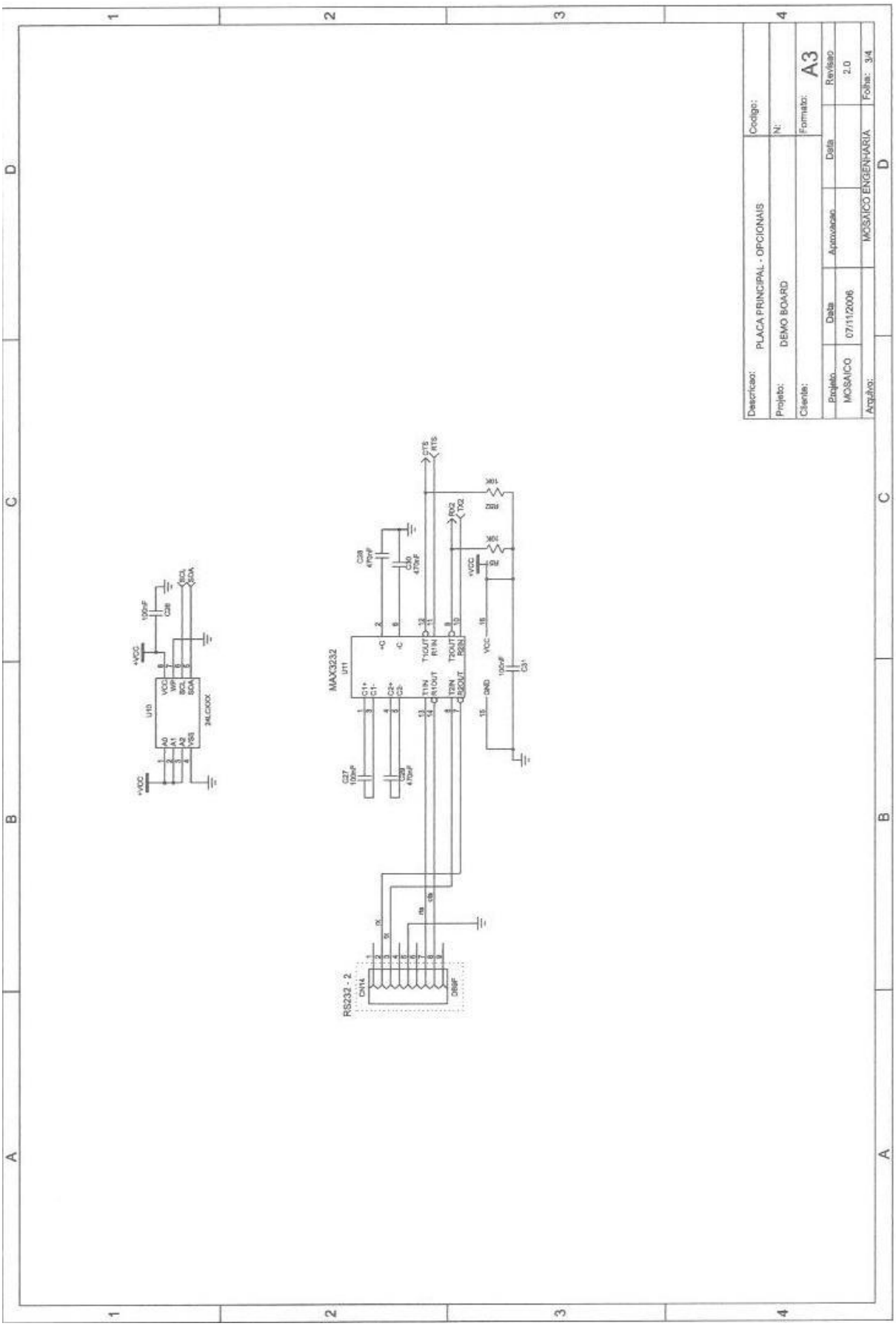
```
        unsigned char  NIBBLE_ALTO:4;
      } ;
      struct
      {
        unsigned char BITS;
      } ;
   };
   union lcd_2
   {
      struct
      {
        unsigned int  :20;
        unsigned int  DADOS_LCD:4;
        unsigned int  :8;
      } ;
      struct
      {
        unsigned int BITS;
      } ;
   };

   union lcd_1 DADObits;
   union lcd_2 TRATA_LCDbits;

   DADObits.BITS = dado;                  //CARREGA DADO
   TRATA_LCDbits.DADOS_LCD = DADObits.NIBBLE_ALTO; //SELECIONA NIBBLE
                                                    //SUPERIOR
   IO1CLR = 0x00F00000;               //LIMPA PINOS DADO DO LCD
   IO1SET = TRATA_LCDbits.BITS;  //CARREGA NIBBLE SUPERIOR NO
                                    //BARRAMENTO DE DADOS DO LCD
   pulso_enable();              //PULSO PARA O ENVIO DO DADO NO LCD
   TRATA_LCDbits.DADOS_LCD = DADObits.NIBBLE_BAIXO; //SELECIONA NIBBLE
                                    //INFERIOR
   IO1CLR = 0x00F00000;          //LIMPA PINOS DADO DO LCD
   IO1SET = TRATA_LCDbits.BITS;  //CARREGA NIBBLE INFERIOR NO
                                    //BARRAMENTO DE DADOS DO LCD

   pulso_enable();              //PULSO PARA O ENVIO DO DADO NO LCD

   delay_10us_x(4);             //DELAY DE 40us
}
**=====================================================================
**         void  limpar_LCD(void)
**
**         Limpa (apaga) as mensagens no LCD
**=====================================================================
void  limpar_LCD(void)
{
   seleciona_comando();  //SELECIONA COMANDO (BIT RS = 0)
   envia_byte(0x01);     //ENVIA COMANDO 0x01
   delay_1ms_x(2);       //DELAY DE 2ms
}
**=====================================================================
**         void  dado_LCD(unsigned char dado)
**
**         Envia um dado (caractere) para o LCD
**=====================================================================
void  dado_LCD(unsigned char dado)
{
   seleciona_dado();   //SELECIONA COMANDO (BIT RS = 1)
   envia_byte(dado);   //ENVIA DADO PARA O LCD
}
**=====================================================================
**         void  comando_LCD(unsigned char dado)
**
**         Envia um comando para o LCD
**=====================================================================*/
void  comando_LCD(unsigned char dado)
{
   seleciona_comando();  //SELECIONA DADO
   envia_byte(dado);     //ENVIA BYTE PARA O LCD
}
**=====================================================================
**         void escreve_frase_LCD(const unsigned char *frase)
**         Envia uma string para o LCD
**=====================================================================
```

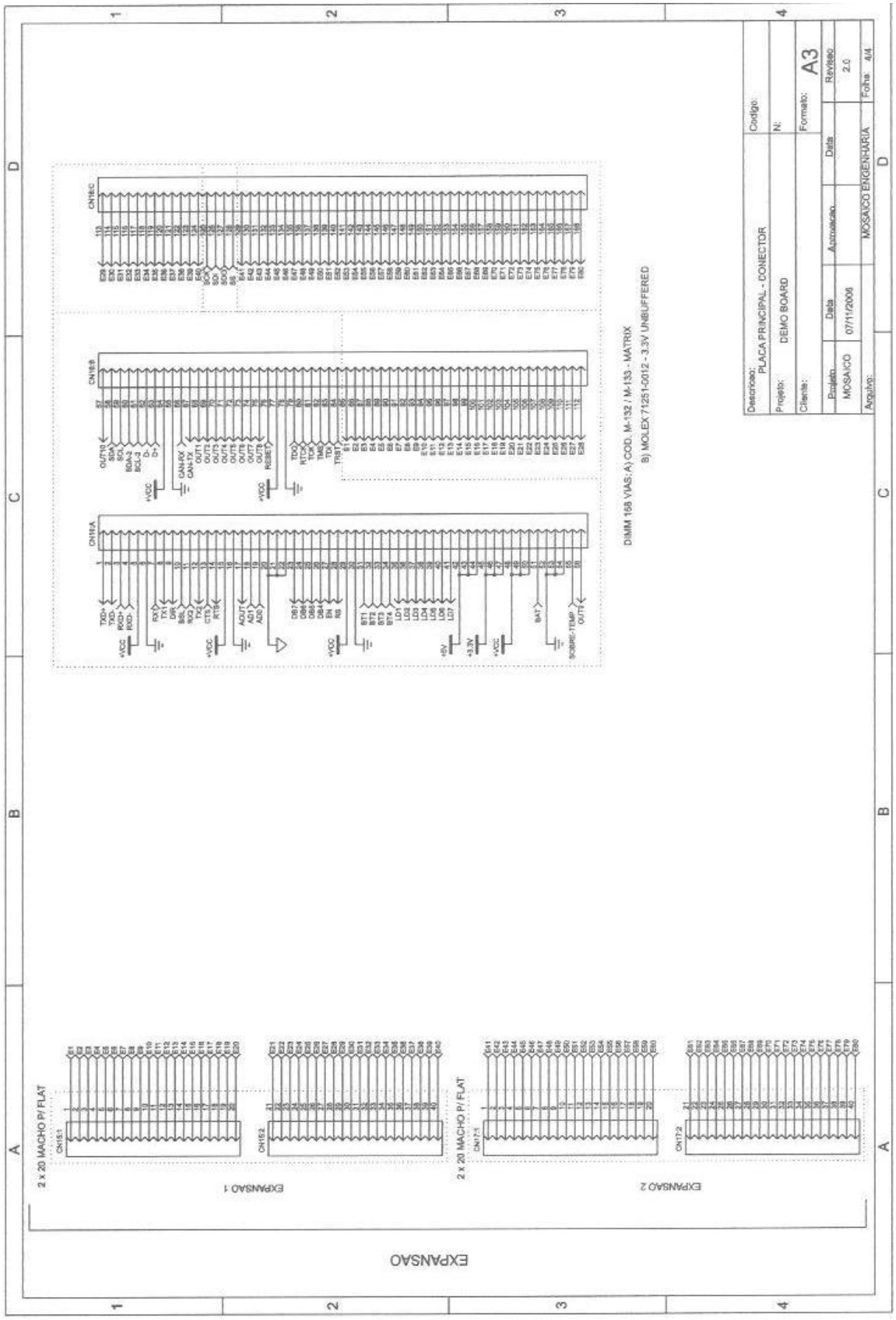
```
void escreve_frase_LCD(const unsigned char *frase)
{
   do
   {
      dado_LCD(*frase);
   }
   while(*++frase);
}
**=====================================================================
**        void escreve_frase_UART0(const unsigned char *frase)
**
**        Envia uma string para o UART0
**=====================================================================
void escreve_frase_UART0(const unsigned char *frase)
{
   do
   {
      MA_PutChar_UART0(*frase);
   }
   while(*++frase);
}
**=====================================================================
**        void  inicializa_LCD(void)
**
**        Inicializa o módulo LCD
**=====================================================================
void  inicializa_LCD(void)
{
   union lcd
   {
      struct
      {
         unsigned int  :20;
         unsigned int  DADOS_LCD:4;
         unsigned int  :8;
      } ;
      struct
      {
         unsigned int BITS;
      } ;
   };
   union lcd TRATA_LCDbits;
   TRATA_LCDbits.BITS = 0;   //INICIALIZA VARIÁVEL EM 0
   delay_1ms_x(50);              //DELAY DE 50 ms PARA A COMPLETA
                                 //ENERGIZAÇÃO DO LCD
//  ENVIO DO COMANDO 0x03 (3 X)
   seleciona_comando();      //SELECIONA COMANDO
   TRATA_LCDbits.DADOS_LCD = 0x03; //ENVIA COMANDO 0X03
   IO1CLR = 0x00F00000;         //LIMPA PINOS DADO DO LCD
   IO1SET = TRATA_LCDbits.BITS;  //CARREGA DADO LCD
   pulso_enable();              //PULSO PARA O ENVIO DO COMANDO NO LCD
   delay_1ms_x(4);               //DELAY DE 4 ms
   pulso_enable();              //PULSO PARA O ENVIO DO COMANDO NO LCD
   delay_10us_x(10);             //DELAY DE 100us
   pulso_enable();              //PULSO PARA O ENVIO DO COMANDO NO LCD
   delay_10us_x(4);              //DELAY DE 40us
//  ENVIO DO COMANDO 0x02 (COMUNICAÇÃO EM 4 VIAS)
   TRATA_LCDbits.DADOS_LCD = 0x02; //ENVIA COMANDO 0X02
   IO1CLR = 0x00F00000;         //LIMPA PINOS DADO DO LCD
   IO1SET = TRATA_LCDbits.BITS;  //CARREGA DADO LCD
   pulso_enable();              //PULSO PARA O ENVIO DO COMANDO NO LCD
   delay_10us_x(4);              //DELAY DE 40us
//  ENVIO DO COMANDO 0x14 (COMUNICAÇÃO EM 4 VIAS, DISPLAY DE 2 LINHAS E
//  MATRIZ 7x5)
   envia_byte(0x14);             //ENVIA BYTE PARA O LCD
   limpar_LCD();                 //limpar LCD
//  ENVIO DO COMANDO 0x0C (DISPLAY SEM CURSOR)
   envia_byte(0x0C);             //ENVIA BYTE PARA O LCD
//  ENVIO DO COMANDO 0x06 (DESLOCAMENTO AUTOMÁTICO DU CURSOR PARA A DIREITA)
   envia_byte(0x06);             //ENVIA BYTE PARA O LCD
}
**=====================================================================
** END OF FILE
**=====================================================================
```

ANEXO A – DESENHOS KIT MCBOARD


CAN-BUS
AQUECEDOR
NÃO AUTO-OSCILANTE
RS-485
LED RGB SUNLED
MAX3232
RS232 - 1
RJ-45 + TRAFO
PULSE J0018D21
Descrição: PLACA PRINCIPAL
Projeto: DEMO BOARD
Cliente:
Código:
N:
Formato: A3
Data
Revisão 2.0
Projeto: MOSAICO
Data 07/11/2006
MOSAICO ENGENHARIA
Folha: 1/4


2 x 10 FLAT MACHO JTAG
SAIDAS DIGITAIS
2 x 8 FLAT MACHO
* - PREVER DISSIPADOR
** - NAO MONTAR
Descricao: PLACA PRINCIPAL
Projeto: DEMO BOARD
Cliente:
Codigo:
N:
Formato: A3
Data
Revisao 2.0
Folha 2/4
MOSAICO ENGENHARIA
Aprovacao
07/11/2008
MOSAICO
JACK DC
9VDC @ 1A
ANALOG OUT
JACK P2 MONO
ANALOG IN
VOLUME
OPA2340/2350
REDE I2C - 2 KRE
REDE I2C 1 KRE
USB PADRAO B SLAVE
SPI KRE
Contraste


Descricao: PLACA PRINCIPAL - OPCIONAIS
Projeto: DEMO BOARD
Cliente:
Codigo:
N:
Formato: A3
Data
Revisao
2.0
Projeto
MOSAICO
Data
07/11/2006
Aprovacao
MOSAICO ENGENHARIA
Folha: 3/4
Arquivo:
MAX3232
U11
24LC00/X
U10
RS232 - 2
CN14
DB9F
+VCC
GND
SCL
SDA
RTS
CTS
TX2
RX2


DIMM 168 VIAS: A) COD. M-132 / M-133 - MATRIX
B) MOLEX 71251-0012 - 3.3V UNBUFFERED
Descricao: PLACA PRINCIPAL - CONECTOR
Projeto: DEMO BOARD
Cliente:
Codigo:
N:
Formato: A3
Revisao: 2.0
Folha: 4/4
Projeto: MOSAICO
Data: 07/11/2006
Aprovacao:
Arquivo:
MOSAICO ENGENHARIA
2 x 20 MACHO P/ FLAT
EXPANSAO 1
EXPANSAO 2
EXPANSAO

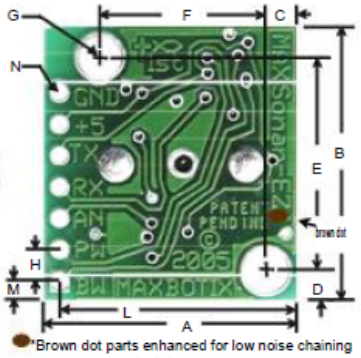
ANEXO B – DATASHEET SENSOR ULTRA-SOM

LV-MaxSonar®-EZ1™
Data Sheet

LV-MaxSonar ®-EZ1™ High Performance Sonar Range Finder

Alimentado com 2.5V - 5.5V o LV-MaxSonar®-EZ1™ permite e detecção e medição de obstáculos em uma placa de pequeno tamanho. O LV-MaxSonar®-EZ1™ detecta objetos antre 0 e 254 polegadas (6.45 metros) e permite a medição de distância de 6 polegadas até 254 polegadas com uma resolução de 1 polegada. Objetos entre 0 e 6 polegadas são medidos como 6 polegadas. Os sinais de saída são: Largura de pulso, voltagem analógica e saída serial digital.

Tamanho



Brown dot parts enhanced for low noise chaining

A	0.785"	19.9 mm	H	0.100"	2.54 mm
B	0.870"	22.1 mm	J	0.645"	16.4 mm
C	0.100"	2.54 mm	K	0.610"	15.5 mm
D	0.100"	2.54 mm	L	0.735"	18.7 mm
E	0.670"	17.0 mm	M	0.065"	1.7 mm
F	0.510"	12.6 mm	N	0.038" dia.	1.0 mm dia.
G	0.124" dia.	3.1 mm dia.	weight, 4.3 grams		

valores nominais

Características

- Ganho variável ajustado continuamente para controle preciso do feixe de ultrasom.
- A detecção de objetos inclui objetos encostados no sensor.
- Alimentação de 2.5V a 5.5V com consumo de 2mA.
- Medição a cada 50mS, (20-Hz rate)
- Operação automática, mede e retorna os valores continuamente.
- Operação manual permite receber os valores apenas quando desejado.
- Todas as saídas são ativas simultaneamente.
  - Serial, 0 a Vcc
  - 9600Baud, 81N
  - Analógica, (Vcc/512) / pol
  - Pulse width, (147uS/pol)
  - Cancela os obstáculos no início da medição.
- Desenhado para proteção de ambientes internos
- Sensor opera a 42KHz
- Excitação do sensor de alta voltagem (dobro de Vcc)

Benefícios

- Baixo custo
- Dados de saída precisos e estáveis
- Sem "zona morta" de medição
- baixo consumo
- Feixe de qualidade
- Furos de montagem na placa.
- Extremo baixo consumo, ideal para operar a baterias.
- Pode ser disparado externamente ou internamente.
- O sensor retorna o valor da medição, não necessitando nenhum cálculo.
- Ciclo de leitura rápido.
- O usuário pode escolher qualquer uma das três saídas.

Características do Feixe

A detecção de pessoas requer alta sensibilidade, Ainda que um feixe estreito requeira baixa sensibilidade. O LV-MaxSonar®-EZ1™ combina de detecção de pessoas com um feixe de largura estreita.

Exemplos de medição são mostrados abaixo em um grid de 12 polegadas. O padrão de detecção é mostrado para:

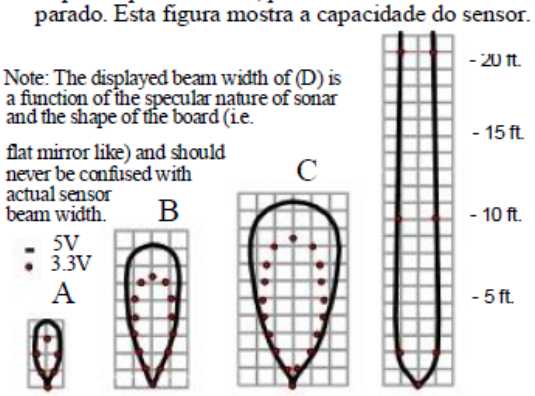
(A) Pino de 0.25 polegadas de diâmetro, veja o ângulo estreito para objetos pequenos.

(B) Pino de 1 polegada de diâmetro, veja o padrão longo de detecção.

(C) Tubo de 3.25 polegadas de diâmetro, veja o padrão longo e controlado de detecção.

(D) Placa de 11 polegadas de largura movimentada de esquerda para a direita, paralela ao sensor e o sensor parado. Esta figura mostra a capacidade do sensor.

Note: The displayed beam width of (D) is a function of the specular nature of sonar and the shape of the board (i.e. flat mirror like) and should never be confused with actual sensor beam width.



beam characteristics are approximate

MaxBotix® Inc.

MaxBotix, MaxSonar & EZ1 are trademarks of MaxBotix Inc.
LV-EZ1™ • v3.0c • 07/2007 • Copyright 2005 – 2007
Distribuidor no Brasil: Tato Equipamentos Eletrônicos (11) 5506-5335– www.tato.ind.br

LV-MaxSonar®-EZ1™ Pin Out

GND - Terra da alimentação. Os pinos GND e Vcc devem estar livres de ruído para uma ótima operação.

+5V - Vcc - Opera de 2.5V a 5.5V. Capacidade de corrente recomendada: 3mA para 5V, e 2mA para 3V.

TX – Quando o pino *BW está aberto ou em nível baixo, o pino TX envia o sinal serial no formato RS232, exceto pela voltagem que é 0-Vcc. A saída é o ASCII "R", seguido por 3 digitos ASCII representando a distância em polegadas até o máximo de 255, seguido por um ENTER(ASCII 13). O baud rate 9600, 8 bits, sem paridade, com um stop bit. Apesar da voltagem de 0-Vcc estar fora do padrão RS232, a maioria dosdispositivos conseguem ler o dado serial. Se o nível padrão for necessário, inverta o sinal e conecte a um conversor como o MAX232.

- ***Com ponto marrom:** Quando o pino BW estiver em nível alto, o pino TX envia um único pulso, (e não dados seriais).

RX – Este pino é mantido em alto internamente. O EZ1™ mede continuamente se este pino for deixado aberto ou em alto. Se levado a nível baixo o EZ1™ irá parar a medição. Leve a nível alto por 20uS ou mais para iniciar uma medição.

AN – Saída analógica com um fator de (Vcc/512) por polegada. Com uma alimentação de 5V temos ~9.8mV/pol. E com 3.3V temos ~6.4mV/pol.

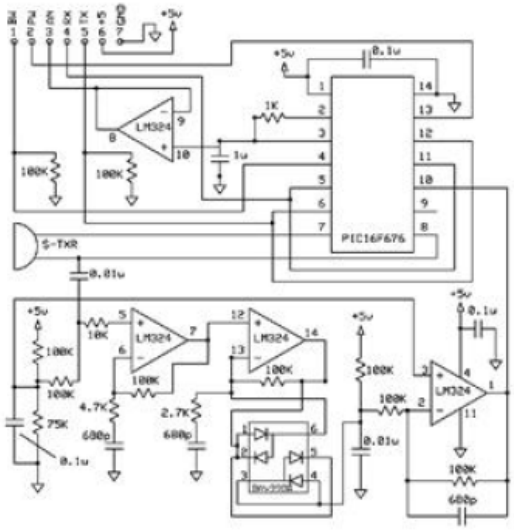
PW - Esta saída é um pulso cuja largura representa a distância. Para calcular use um fator de 147uS por polegada.

BW - *Deixe aberto ou em nível baixo para ter saída serial em TX.

- ***Com ponto marrom:** Quando o pino BW é mantido alto, o pino TX envia um pulso (e não dados seriais).

Circuito do LV-MaxSonar®-EZ1™

O sensor LV-MaxSonar®-EZ1™ utiliza para seu funcionamento um LM324, uma matriz de diodos, um PIC16F676, junto com uma variedade de componentes passivos.



LV-MaxSonar®-EZ1™ Descrição da Temporização

250mS após alimentado, o LV-MaxSonar®-EZ1™ está pronto para aceitar comandos no pinoRX. Se o pino RX for deixado aberto ou mantido em nível alto, o sensor irá inicialmente fazer um ciclo de calibração (49mS), depois irá fazer uma medição (49mS). Então, a primeira medição leva ~100mS. As próximas leituras levarão 49mS. O LV-MaxSonar®-EZ1™ checa o pino RX no fim de cada ciclo. O resultado pode ser lido a cada 49mS.

Cada ciclo de 49mS começa com o pino RX estando aberto ou em alto, depois o LV-MaxSonar®-EZ1™ envia 30 pulsos de 42KHz, e o pino PW é levado a nível alto. Quando um obstáculo é detectado, o pino PW é levado a nível baixo. O pino PW fica alto por até 37.5mS se nenhum obstáculo for detectado. O restante do ciclo de 49mS (less 4.7mS) é usado para ajustar a saída analógica para o nível correto. Quando uma longa distância for lida imediatamente após uma pequena distância, a saída analógica pode não refletir o valor correto em um ciclo. Durante os últimos 4.7mS, o valor serial é enviado. A temporização do LV-MaxSonar®-EZ1™ calibrado na fábrica para 1% em 5V, e em uso sua precisão é melhor que 2%. Além disto, a operação em 3.3V normalmente faz com que o resultado seja um ou dois porcento maior que o real.

LV-MaxSonar®-EZ1™ Instruções de Uso

Cada vez que o LV-MaxSonar®-EZ1™ é ligado, ele irá se calibrar no primeiro ciclo. O sensor usa esta informação para medir os objetor próximos. É importante que não haja obstáculos próximos ao sensor nesta fase. A melhor sensibilidade é obtida quando não há objetos no raio de 14 polegadas, mas bons resultados são comuns com distâncias de pelo menos 7 polegadas. Se um objeto estiver muito próximo durante a calibração, o sensor pode ignorar objetos naquela distância.

O LV-MaxSonar®-EZ1™ não usa esta calibração para compensação de temperatura, mas para cancelar o padrão de ressonância do sensor. Se a temperatura, umidade ou tensão mudarem durante a operação, o sensor pode precisar de uma recalibração. Se ele não for recalibrado e a temperatura aumentar, a leitura informada será menor que a real. Se a temperatura diminuir o sensor terá uma redução na sensibilidade de objetos próximos. Para recelibrar o LV-MaxSonar®-EZ1™, desligue e ligue a alimentação.

Produto/especificações sujeitas a mudança sem aviso prévio. Para mais informações visite www.maxbotix.com/MaxSonar-EZ1_FAQ

MaxBotix® Inc.

MaxBotix, MaxSonar & EZ1 are trademarks of MaxBotix Inc.
LV-EZ1™ • v3.0c • 07/2007 • Copyright 2005 – 2007
Distribuidor no Brasil: Tato Equipamentos Eletrônicos (11) 5506-5335– www.tato.ind.br