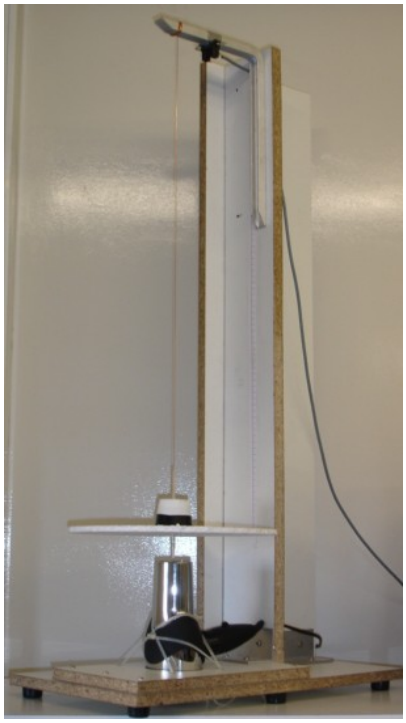
Universidade do Porto
FEUP Faculdade de Engenharia

Sistemas Baseados em Microprocessadores

2004/2005

"Levitação de um disco por variação de velocidade de um Ventilador"



Intervenientes:

Ana Luísa Martins ee04255@fe.up.pt

Fernando Manuel Gonçalves Santos ee01136@fe.up.pt

Índice

Pag:

Introdução

Este mini projecto de SBM faz parte de um projecto global, que surgiu no âmbito da cadeira de Lógica difusa em que pretendemos controlar a posição de um "disco" sobre um eixo através do disparo de um triac, este que por sua vez controla a potência de um ventilador, potência esta que é transmitida ao disco sobre a forma de caudal. Por fim o caudal vai impor uma posição ao disco. O controlo deste sistema é realizado de duas formas (através de um controlador PD e um controlador baseado em lógica difusa) que no final irão ser comparadas.

Objectivos

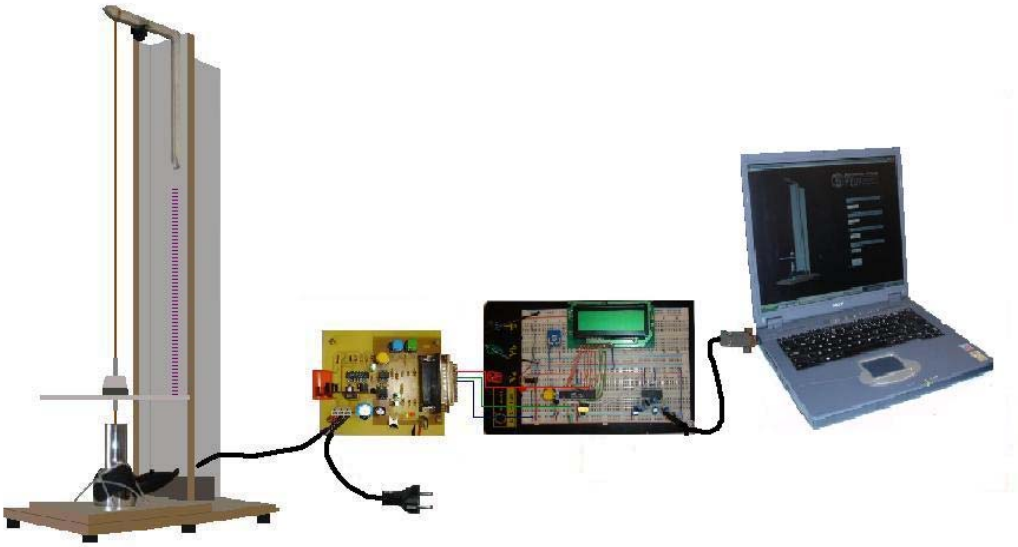
Em relação a SBM os objectivos são projectar um circuito que acondicione correctamente os sinais recebidos pela placa (sinal de passagem por zero) e enviados (disparo do triac), e ainda o sinal de entrada analógico proveniente do sensor.

Foi necessário também "criar" um "mini" protocolo de comunicação para interligar o micro com o PC. PC este que estava a utilizar o Matlab como objecto de controlo.

Foi-nos proposto ainda a implementação de um controlador PD no próprio micro, sem necessitar de intervenção do PC.

Por fim pretendia-se também integrar um display que indique a posição actual do disco e a potência consumida pelo ventilador.

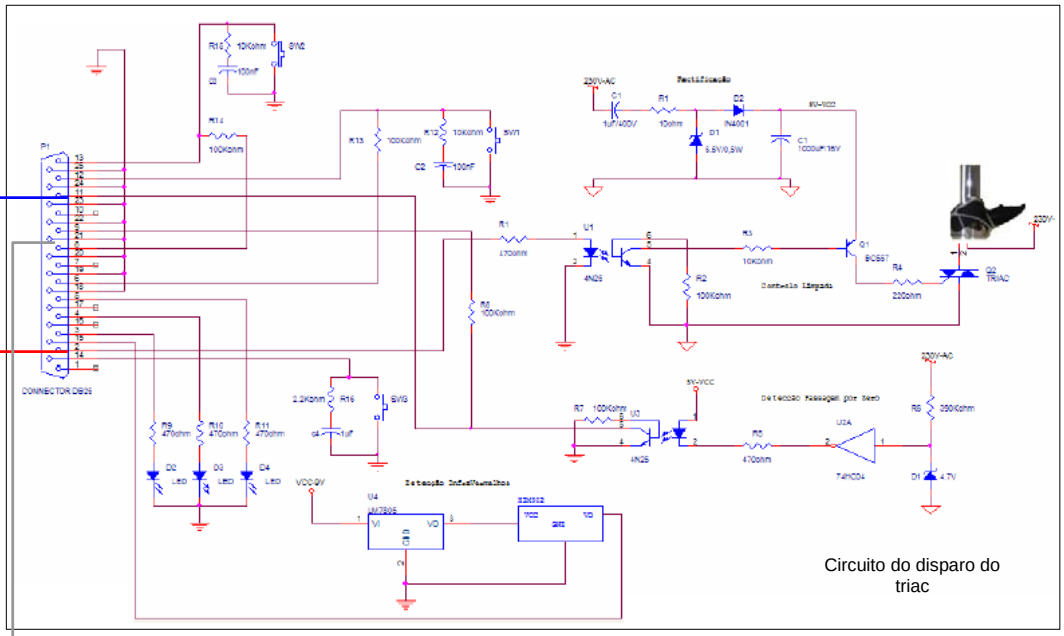
Composição do sistema:



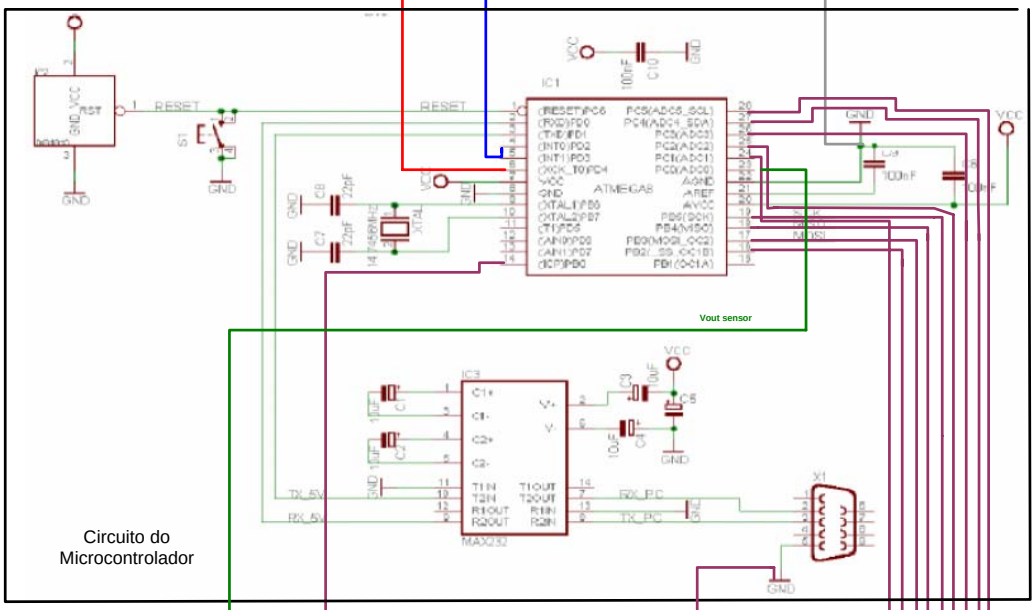
Neste pequeno ponto quisemos demonstrar de certa forma a interligação entre os vários dispositivos, para posteriormente ser mais fácil visualizar a mesma interligação.

Projecto do circuito:

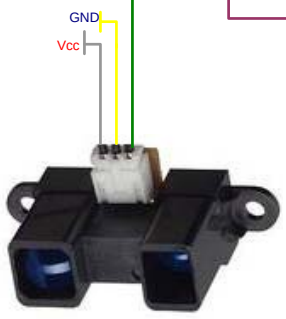
Para este trabalho, decidimos utilizar o atmega8 para o controlo do nosso sistema, isto devido ao facto de ser muito fácil a sua utilização, mas também devido à facilidade de programação. Agora preenchendo os nossos requisitos efectuamos a seguinte montagem:



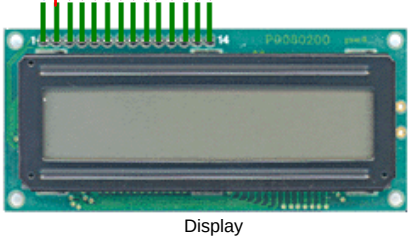
Circuito do disparo do triac



Circuito do Microcontrolador



Sensor



Display

Aspectos relevantes:

Note-se que a nível do circuito de disparo do triac, o pino 9 da porta paralela não está ligado. Este pino era necessário para efectuar a função de "pull-up" no circuito da mesma placa. Para solucionar esta "falha", colocamos uma resistência de "pull-up" de 100 KΩ entre os pinos 3 e 2 do Atmega e VCC.

É ainda importante salientar que curto-circuitamos a massa do circuito do Atmega, com a do circuito de Controlo da placa, para um funcionamento correcto do mesmo.

Para o sensor tivemos o cuidado de verificar se era possível a ligação directa do mesmo ao Atmega, e ao ver a "datasheet" do sensor verificamos que este sensor nos dava esta possibilidade.

A nível de Display fizemos a montagem proposta pelo "datasheet" do mesmo, mudando as ligações de modo a estar compatível com o resto do circuito.

De resto tivemos as mesmas precauções a nível de acondicionamento do Atmega que tivemos para o segundo trabalho.

Lógica de Controlo do Disparo do Triac

Neste ponto vamos apresentar a lógica de controlo. Esta baseia-se na variação do ângulo de disparo do triac. Este ângulo é definido ou através da interacção com o PC (Controlo via Lógica difusa), ou via única e exclusivamente pelo Atmega (Controlo PD).

A placa de disparo do triac gera uma onda quadrada de período 20 (ms) (período da onda da rede), os ciclos (positivo e negativo) sendo estes os correspondentes à sinusóide da rede. O momento em que a onda quadrada muda de Tensão 5 (V) -> 0 (V) ou 0 (V) -> 5 (V), corresponde á passagem por zero na rede.

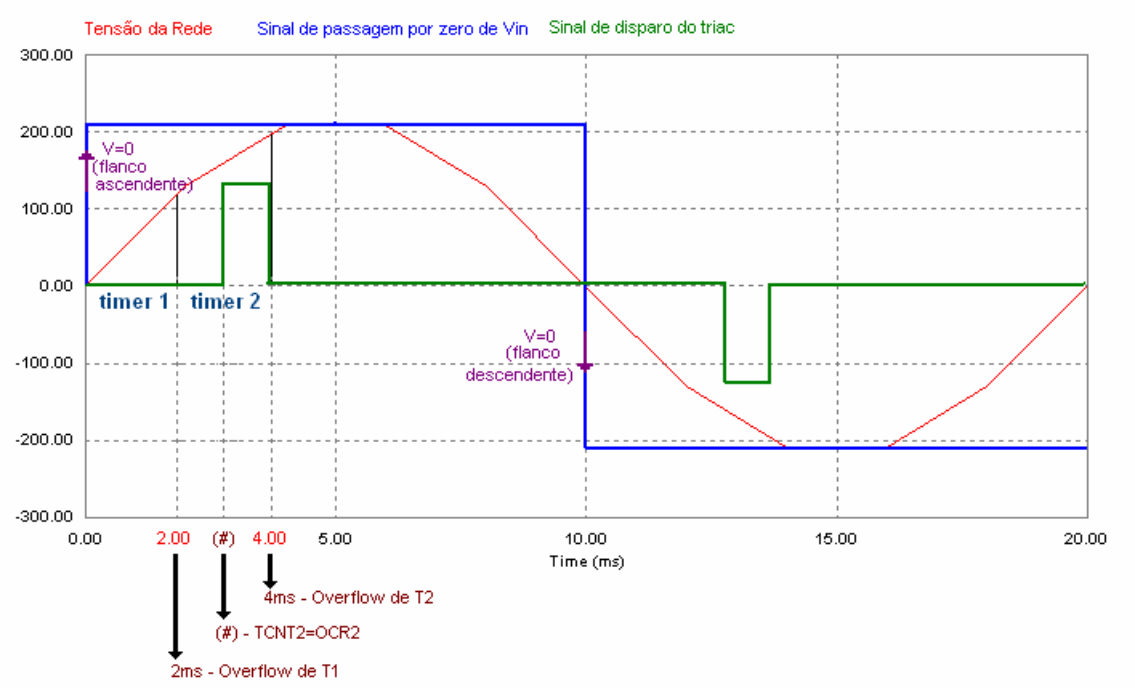
A nível de Atmega o disparo é criado com a interacção de dois timers, e das duas interrupções externas, o timer 1 em modo normal e o timer 2 em modo "fast-PWM", sendo este último o responsável pelo disparo do triac.

Juntando estes dois últimos parágrafos podemos englobar na totalidade a lógica de controlo. Assim e num diagrama temporal, ocorrem os seguintes passos:

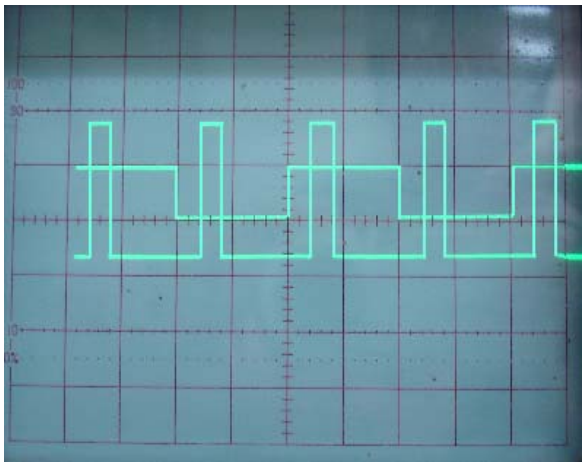
1) Detecção da passagem por zero (INT0/INT1) → Activa Timer 1
2) "Overflow" do timer 1→ desactiva-se a sí próprio e activa timer 2
3) Comparação do valor do timer 2 com "OCR2" → Activa a saída do disparo (coloca a 1 (5 V))
4) "Overflow" do timer 2 → Desactiva-se a sí próprio.

Note-se ainda a importância de se habilitar as interrupções externas (INT0 e INT1), ao flanco ascendente e descendente para garantir o sincronismo do micro com a rede.

Agora vamos apresentar uma imagem ilustrativa deste processo:



Visto no osciloscópio obtivémos a seguinte imagem:



Esta parte do programa durante o semestre teve duas fases, uma em que para o controlo do disparo era única e exclusivamente feito pelo timer 2 (em modo “fast-PWM”) de 0 a 7ms. Porém depois de algumas experiências vimos que a variação de “OCR2” em uma unidade, criava uma variação muito grande na saída. Desta forma resolvemos o problema com a solução apresentada anteriormente, variando o disparo de 2 a 4 ms. Note-se que este valor é mais que suficiente para levitar o disco nos 55 cm de altura que temos para controlar.

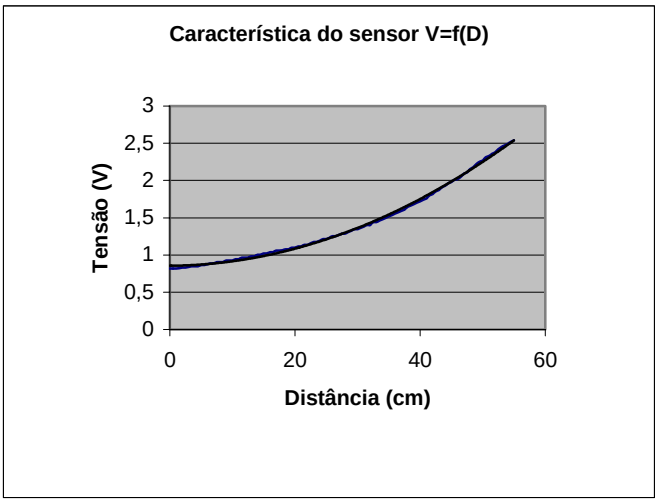
Conversão do sinal analógico do sensor para sinal digital

O sinal obtido no sensor é analógico, logo temos que converter esse sinal para formato digital através de um conversor analógico digital presente no Atmega8.

O ADC utilizado foi o ADC0 que nos proporciona uma aproximação sucessiva de 10-bit. Uma conversão é iniciada colocando ADSC=1, esta flag permanece neste estado enquanto a conversão é efectuada e é colocada a 0 por hardware quando a conversão estiver completa. O nosso ADC encontra-se a funcionar no modo "Free Running", isto significa que o ADC está constantemente em amostragem e a actualizar a informação do ADC. Este modo é seleccionado com ADFR=1. Para obtermos um valor digital mais preciso livre de ruído de cada valor analógico do sensor, realizamos 100 leituras e calculamos a média.

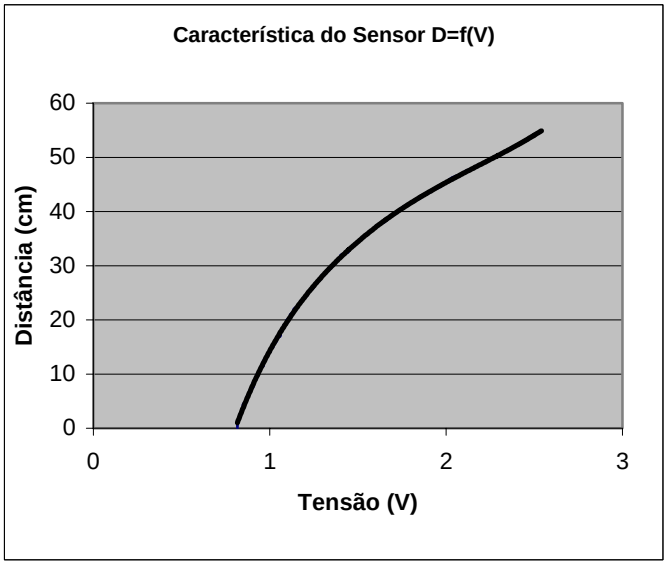
Note-se ainda que usamos como comparação a tensão de referência interna (2.56 V).

E ainda de salientar que o sensor tem uma característica não linear, e para o controlo é necessário inverter essa característica. Sendo assim esta é a característica do sensor:



Sendo a formula apresentada uma linha de tendência obtida através da ferramenta "Excel".

Com o intuito de obter uma fórmula que correlacionasse a distância do disco em função da tensão do sensor revertemos este gráfico, tomando o seguinte aspecto:



As expressões utilizadas para o controlo por lógica difusa e para o controlo PD foram obtidas a partir da característica anterior através da aproximação desta por uma linha de tendência. Escolhemos uma função polinomial de ordem 6 para o controlo por lógica difusa, visto o Matlab ter uma grande resolução matemática. Já para o controlo PD escolhemos uma função polinomial de ordem 3 para não sobrecarregar o atmega com este cálculo e perder demasiado tempo. É de notar a extrema importância do número de casas decimais colocadas, pois inicialmente colocámos poucas e verificamos que os valores davam bastante diferentes dos medidos.

MATLAB	**Altura** = -2.008377743419*((**ADC**/380)^6) + 25.856920825050*((**ADC**/380)^5) - 135.300139807327*((**ADC**/380)^4) + 376.958350602915*(**ADC**/380)^3 - 606.010587307718*((**ADC**/380)^2) + 564.866278844587*(**ADC**/380)- 210.060393336037
ATMEGA	**Altura =** 13.040354902516*pow((**adc_medf**/380),3)- 80.016034290549*pow((**adc_medf**/380),2)+179.952974341045*(**adc_medf**/380)- 99.058946926390;

É necessário dividir ADC por 380, pois esta é a razão entre o valor dado pelo ADC e o seu valor em tensão. Depois este valor da altura vai entrar para os cálculos do nosso controlo em que temos como entradas o erro, o derro e como saída o disparo:

Funcionamento do display

O display por nós utilizado é constituído por duas linhas e 20 colunas sendo as suas características físicas:

Driver do display KS0066U e KS0063B
Dutty Cycle 1/16

	Pino	Símbolo	Descrição
PINAGEM	1	Vss	Ground
	2	Vdd	Terminal para tensão de alimentação
	3	Vo	Fonte de potência para o Driver do cristal líquido
	4	RS	Selecção de registo RS = 0 …. Instrução de Registo RS = 1 …. Informação do Registo
	5	R/W	Leitura/Escrita: Nível alto = Leitura Nível baixo = Escrita
	6	E	Pino de activação
	7 a 14	D0 a D7	Barramento de informação bidireccional.
	15	BL -	Terminais de alimentação do LED
	16	BL +	

No display pretendemos colocar o valor da altura actual do disco (2ª linha do display) e a potência consumida pelo secador (1ª linha do display).

Para o cálculo da potência medimos para todos os valores de disparo a tensão e corrente na entrada, obtendo os seguintes valores:

```
int potencia[68]={14,16,17,18,20,21,23,25,27,29,31,33,34,36,39,41,45,46,49,52,53,56,59,
61,64,67,70,72,75,78,81,84,87,90,94,97,99,103,106,109,112,116,119,123,126,130,134,137,
140,143,147,150,153,156,160,162,167,170,173,176,180,183,186,189,192,195,199,202};
```

Assim para cada valor de disparo a tabela anterior é consultada e o respectivo índice indica a potência consumida pelo secador. Em relação à altura, esta é recebida do PC.

Para enviarmos estes resultados para o display usamos a função *linhas(linha1,linha2);*

Um ciclo de leitura/escrita para o display, processa-se da seguinte maneira:

	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Colocação do display a ON	0	0	0	0	0	0	1	0	0	0
Limpa Display	0	0	0	0	0	0	0	0	0	1
Modo de edição de texto	0	0	0	0	0	0	0	1	1	0
Escreve 1ª linha	Faz a leitura de cada caractere com a função caractere( ) e a função LCD_write( ) adquire os caracteres 1 por 1, de notar a necessidade de um tempo de espera entre cada aquisição de novo caractere (cerca de 39 us)									
Mudança de linha	0	0	1	1	0	0	0	0	0	0
Escreve 2ª linha	Faz a leitura de cada caractere com a função caractere( ) e a função LCD_write( ) adquire os carecteres 1 por 1, de notar a necessidade de um tempo de espera entre cada aquisição de novo caractere (cerca de 39 us)									
Colocação do display a OFF	0	0	0	0	0	0	0	1	0	0

Durante a implementação deste dispositivo, tivemos alguns problemas, esses problemas tinham origem no facto de possivelmente ocorrerem interrupções durante a escrita no display, a técnica usada foi inibir interrupções no momento em que escreviamos no display. Porém este tempo em que as interrupções estão inibidas, pois é crítico que estas estejam para que não se perca o sincronismo com a rede.

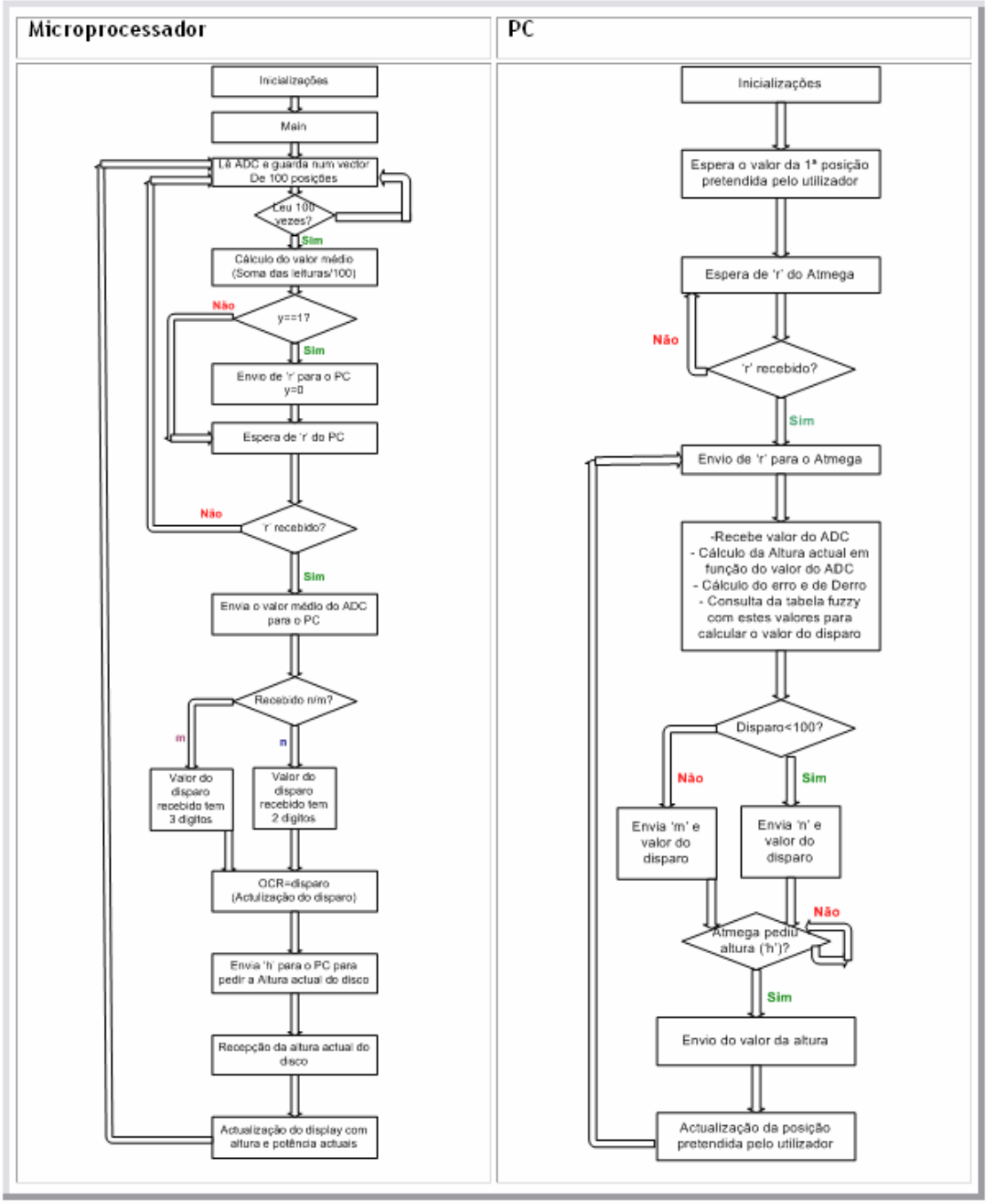
No entanto ainda alguns problemas resistiram a esta solução, pensamos que sejam ruídos de massas. Tivemos pena de não termos tempo para resolver este problema.

Técnicas de Controlo

Esquema de raciocínio para o controlo por lógica difusa:

Neste ponto vamos confrontar as duas técnicas de controlo utilizadas, apresentando primeiro o controlo por lógica difusa e depois o controlo via PD.

Agora vamos apresentar um fluxo grama mostrando o raciocínio do mesmo, visto ser de mais fácil compreensão.



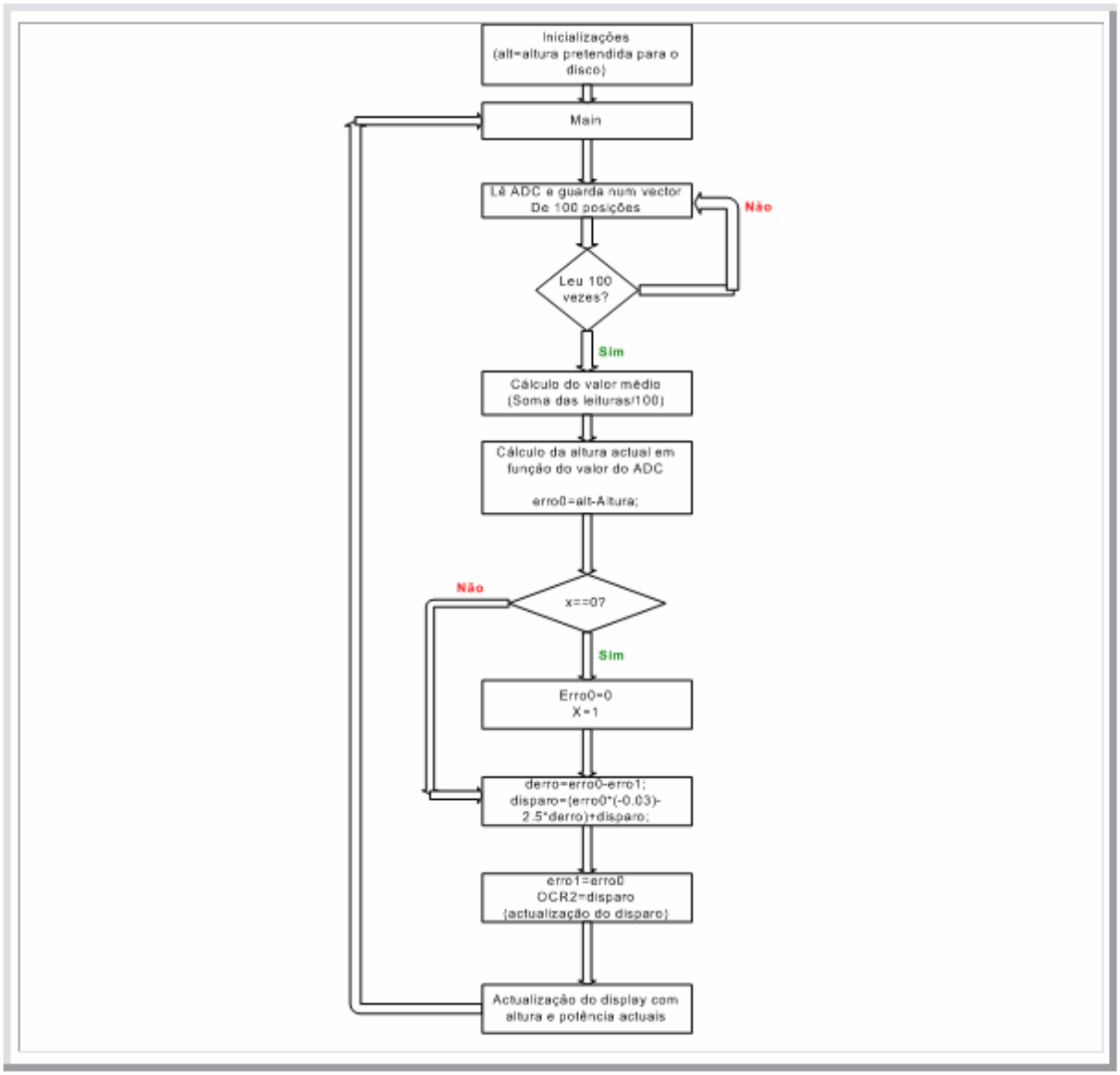
É importante salientar que quando o PC e o micro estão sincronizados, no nosso caso é no momento quando é enviado o "r" de resposta do PC→ Atmega, são desligadas as interrupções porta série para que a transmissão de controlo seja exclusiva.

De certa forma podemos dizer que o PC se comporta como Master e o Atmega como Slave, residindo assim o processo efectivo de controlo no PC.

Para uma vista mais pormenorizada deste processo propomos a leitura do ANEXO 1.2

Esquema de raciocínio para o controlo através de um controlador do tipo PD:

Neste momento vamos avançar para uma explicação do controlo com o PD, apresentando da mesma forma um fluxo grama para uma melhor compreensão.



Neste processo para acertarmos a expressão que rege o controlo PD, tivemos que proceder a sucessivos ajustes obtidos pela visualização do controlo do nosso sistema. Desta forma após várias tentativas chegamos à seguinte expressão:

disparo = (erro0*(-0.03)-2.5*derro)+disparo

Sendo:

Erro0 = Alt-Altura; Alt = altura pretendida e Altura=posição actual

Derro=Erro0 – Erro1; Erro0 = erro actual; Erro1 = erro anterior

Podemos dizer assim, da mesma forma que no método de controlo anterior, que temos uma saída relativa, isto quer dizer que, a variável de disparo depende obrigatoriamente do valor imediatamente anterior.

Mais uma vez propomos a leitura do ANEXO 1.3 para uma melhor compreensão do processo.

Interrupções da porta série

Aqui proporcionamos ao utilizador alguns privilégios de controlo e também a obtenção de certos parâmetros do sistema.

Para isto foram criadas as rotinas *recebe();* e *transmite();* e *testa_byte();* sendo respectivamente rotinas de recepção e envio de dados e teste de recepção.

A nível de interrupção, o caractere lido é comparado com os seguintes caracteres:

'h' → pedido de altura actual; o Atmega responde com o valor da mesma
'w' → pedido do valor do disparo actual; o Atmega responde com o respectivo valor
'+' → incremento de uma unidade da altura pretendida para o disco (válido para o controlo PD)
'-' → decremento de uma unidade da altura pretendida para o disco (válido para o controlo PD)

E dependendo do seu valor realiza a respectiva operação. Note-se que esta interrupção só existe no controlo por PD, visto não ser necessário no controlo por Lógica Difusa.

Conclusões

Este trabalho possibilitou-nos a interligação de diversos processos, o que foi bastante interessante, tanto a nível de controlo como também no processo de transmissão de informação entre eles.

Pudemos comparar também dois tipos de controlo. Um controlo feito por um controlador PD implementado no micro, e outro implementado no PC baseado em Lógica Difusa. Neste ponto reparamos várias coisas. Entre elas o controlo via PD, é muito mais rápido e preciso relativamente ao controlo baseado em Lógica Difusa. Quanto ao tempo, tentamos ver o porquê de tanta diferença e vimos que o problema residia no facto do programa Matlab demorar muito tempo (quase 1 segundo) para comunicar com o Atmega. Desta forma concluímos que o Matlab não é uma boa ferramenta de controlo, porém não tivemos tempo para implementar este controlo noutro programa ou até mesmo no micro. Note-se ainda que tivemos em conta que a velocidade de transmissão da porta série e por isso mesmo escolhemos um "baudrate" de 38400 bps, anulando assim as dúvidas quanto a possíveis atrasos devido a este pormenor.

Mesmo com este "incidente" o controlo foi possível em ambos os casos, e obtivemos uma resolução de cerca de 1 cm através de Lógica difusa e cerca de 0,5 cm com o PD.

Verificamos que o Atmega é uma óptima ferramenta de controlo, durante este semestre fomo-nos apercebendo gradualmente desta característica, para além de fiável é robusto desde que convenientemente programado.

Nós tivemos um cuidado especial na escolha dos tempos utilizados nos timers pois para termos uma correcta sincronização com a rede, a soma dos tempos contados pelos timers não pode ser maior do que 10 ms (próxima passagem por zero). Como se mostrou anteriormente este limite nunca será ultrapassado pois 2 ms (timer 1) + 2 ms (timer 2) é bem mais pequeno que 10 ms, podendo assim desprezadas algumas variações na contagem dos timers.

Por fim podemos dizer que este trabalho ajudou-nos a consolidar os conhecimentos adquiridos durante este semestre, e de certa forma colmatar algumas falhas que cometemos nos outros trabalhos.

É sempre interessante poder controlar um processo físico, e visto que estamos num ramo muito "virado" para o controlo de processos industriais, este tipo de trabalhos dá-nos uma "leve" ideia de como serão controlados alguns processos a nível industrial e de certa forma vamos amadurecendo neste sentido com a realização destes pequenos projectos.

ANEXOS

ANEXO 1 - Código em WinAvr para a programação do Atmega

//----------

//ANEXO 1.1 → "LCD.c" código do Display

```
#include "sbld.h"

#define sbi(port,bit)  (port |=  ( 1 << bit ))

#define cbi(port,bit)  (port &= ~( 1 << bit ))

void caracter(unsigned char letra)
{
        unsigned char MSB=0x00;
        unsigned char LSB=0x00;
        unsigned char A=0x00;
        MSB=(letra & 0x0F0); //Seleciona os 4 bits mais significativos
        LSB=(letra & 0x00F); //Seleciona os 4 bits menos significativos
        MSB=MSB>>2;          //Faz um rotate de 2 posicoes para a direita
        LSB=LSB<<2;          //Faz um rotate de 2 posicoes para a esquerda
        A=(PORTC & 0x0C3);   //Faz uma AND logica bit a bit com o que esta na porta e com
1100 0011
        PORTC=(MSB | A);     //Faz um OR logico bit a bit com o que esta no A com os bits
mais significativos
        A=(PORTB & 0x0C3);   //Faz uma mascara com o que esta na porta e com 1100 0011
        PORTB=(LSB | A);     //Faz um OR logico bit a bit com o que esta no A com os bits
menos significativos
}

void wait(void)
{
        unsigned long int t=3632;//Espera o tempo necessário para a proxima funçao
        do{
                t--;
        }while(t>1);
}
void lcd_tool(unsigned char valor)
{
        cbi(PORTC,1);   //Enable        LCD
        cbi(PORTB,0);   // RS           LCD
        caracter(valor);
        wait();
        sbi(PORTC,1);   //Enable
        wait();
        cbi(PORTC,1); //Enable

}
void lcd_write(unsigned char letra)
{

        cbi(PORTC,1); //Enable         LCD
        sbi(PORTB,0); // RS            LCD

        caracter(letra);//acondiciona letra
        wait();
```

```
        sbi(PORTC,1);//Enable
        wait();
        cbi(PORTC,1);//Enable

}
void lcd_control(int op)
{//cli();
        cbi(PORTC,1);  //Enable
        cbi(PORTD,6); // RW
        cbi(PORTB,0); // RS
        if (op==000) caracter(0x08);//display ON
        if (op==100) caracter(0x0C);//display OFF


        wait();
        sbi(PORTC,1);   //Enable
        wait();
        cbi(PORTC,1); //Enable
        //sei();
}
void linhas(unsigned char linha1[],unsigned char linha2[])
{

        int i;
        lcd_control(000);//coloca display a ON
        lcd_tool(0x01);//Limpa Display
        lcd_tool(0x06);//modo de edição


        for (i=0; i<=10;i++)
        {
        lcd_write(linha1[i]);//escreve letra
        }
                lcd_tool(0x0C0);//manda o cursor p a 2ª linha

        for (i=0; i<=11;i++)
        {
        lcd_write(linha2[i]);//escreve letra
        }

                lcd_control(100);//coloca display a OFF
        }
//-------------------------------------------------------------------------------------------
```

// ANEXO 1.2 → "sbld.c" código do Programa principal (Controlo por Lógica Difusa)

```
#include "sbld.h"
#include "sbldADC.h"
#include "sbldUART.h"
#include "sbldtimer.h"
#include "sbldExtINT.h"
#include "bit_tools.h"
#include "lcd.h"
#include <ctype.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#define sbi(port,bit)  (port |=  ( 1 << bit ))
```

```
#define cbi(port,bit)  (port &= ~( 1 << bit ))


float disparo=100;
unsigned char pot[3]; //potencia absorvida pelo motor
extern unsigned int alt; // Altura pretendida
double  Altura; // Altura Actual
float erro0=0; // erro actual
float erro1=0; // erro anterior
float derro=0; // derivada do erro
unsigned int u; // variável auxiliar
unsigned int y=1; //variável auxiliar
unsigned int i; //variável auxiliar
unsigned char j; //variável auxiliar
double soma;
unsigned int adc_value[100]; // vector de valores do ADC
float adc_med; //valor médio do ADC
int l=0; //variável auxiliar
int x=0; //variável auxiliar
unsigned char numero[10]="         ";//string auxiliar
unsigned char mudolinha[4]="\n\r"; //string para mudar de linha
unsigned int disp[5];// String auxiliar para recepcao do disparo
unsigned int altu[2]; // String auxiliar para recepcao da alura
unsigned char pot[3]; // String auxiliar para recepcao da potência
unsigned char linha1[11]="Pot->  0(w)"; // string (modificável) para enviar para o Display
unsigned char linha2[12]="Alt->  0(cm)"; // string (modificável) para enviar para o Display
/////////////////Tabela da potencia///////////////
Int potencia[68]=
{14,16,17,18,20,21,23,25,27,29,31,33,34,36,39,41,45,46,49,52,53,56,59,61,64,67,70,72,75,78
,81,
84,87,90,94,97,99,103,106,109,112,116,119,123,126,130,134,137,140,143,147,150,153,156,1
60,162,
167,170,173,176,180,183,186,189,192,195,199,202};
////////////////////////////////////////////////////////////////////////////
static void io_init(void)
{
        // Configuracao do porto B
        PORTB = 0x0;
        DDRB = 0xFF;
        // Configuracao do porto C
        PORTC = 0x0;
        DDRC=0xFE;
        // Configuracao do porto D
        PORTD = 0x0;
        DDRD=0x13;
}

        while(1)
        {
                adc_value[l]=adc_data[0];
                if(l==100)// faz 100 leituras
                {
                        for(u=0;u<100++)
                        {
                                soma=soma+adc_value[u]; // Calculo da média
                        }

                        adc_med=soma/100; // Calculo da média
```

```
		soma=0; //faz "reset" á variável soma
		l=-1; // artifício de calculo para a seguir ter o valor 0
	}
	l++;
	if(y==1)
	{
		transmite('r'); // pedido de transmissão de controlo
		y=0;//inviabilisa próximo pedido
	}

	if(testa_byte()!=0)
		j=recebe();
	else
	{
		if(j=='r')//se recebeu um 'r' inicia transmissão de controlo
		{
			itoa(adc_med,n1,10);
			for(i=0;i<=10;i++)
		{
		transmite(n1[i]); // transmite o valor do ADC
		}


		while(testa_byte()==0)
		{
			;//espera por um digito
		}

		j=recebe();
		if(j=='n') //Disparo com 3 digitos
		{
			for(u=0;u<=1;u++)
			{
				while(testa_byte()==0)
				{
					;
				}

				j=recebe();
				disp[u]=j; // recebe disparo
			}
			disparo=disp[0]*10+disp[1]-528;//Actualiza disparo
			OCR2=disparo;//actualiza OCR2
			transmite('h');//espera pelo valor da altura
			for(u=0;u<=1;u++)
			{
				while(testa_byte()==0)
				{
					;//espera por um digito
				}
				j=recebe();
				altu[u]=j; //Guarda valor da altura
			}
			lcd_tool(0x3c);
			linha2[11]=altu[0]; // Muda o valor da altura
			linha2[12]=altu[1]; // Muda o valor da altura
			itoa(potencia[254-disparo],pot,10);
```

```
                        if(potencia[254-disparo]>99)//Decide se Potencia tem 3 ou 2
digitos
                        {
                                linha1[11]=pot[0]; //Muda valor d Potência
                                linha1[12]=pot[1]; //Muda valor d Potência
                                linha1[13]=pot[2]; //Muda valor d Potência
                        }
                        else //Potencia com 2 digitos
                        {
                                linha1[11]=pot[0];//Muda valor d Potência
                                linha1[12]=pot[1];//Muda valor d Potência
                                linha1[13]=' '; // Muda o valor de Potência
                        }
                        linhas(linha1,linha2);//Actualiza Display
                        y=1; // Viabiliza a próxima transmissão de controlo
                }

                if(j=='m') // Disparo com 3 digitos
                {
                        for(u=0;u<=2;u++)
                        {
                                while(testa_byte()==0)
                                {
                                        ;//espera por um digito
                                }

                                j=recebe();
                                disp[u]=j; // recebe disparo
                        }
                        disparo=disp[0]*100+disp[1]*10+disp[2]-5328; //Calcula o valor
de disparo
                        OCR2=disparo;//actualiza OCR2
                        transmite('h');//espera pelo valor da altura
                        for(u=0;u<=1;u++)
                        {
                                while(testa_byte()==0)
                                {
                                        ;// espera por um digito
                                }
                                j=recebe();
                                altu[u]=j;// recebe altura
                        }
                        lcd_tool(0x3c);
                        linha2[6]=altu[0]; // Muda o valor da altura
                        linha2[7]=altu[1]; // Muda o valor da altura

                        itoa(potencia[254-disparo],pot,10);
                        if(potencia[254-disparo]>99) // decide se 3 ou 2 digitos
                        {
                                linha1[5]=pot[0]; //Muda o valor da potencia
                                linha1[6]=pot[1]; //Muda o valor da potencia
                                linha1[7]=pot[2]; //Muda o valor da potencia
                        }
                        else
                        {
                                linha1[6]=pot[0]; //Muda o valor da potencia
                                linha1[7]=pot[1]; //Muda o valor da potencia
                                linha1[5]=' ';} //Muda o valor da potencia
```

```
                                        linhas(linha1,linha2); // Actualiza Display
                                }
                                y=1;//Viabiliza a próxima transmissão de controlo
                        }
                }
        }
}
//-------------------------------------------------------------------------------------------
```

// ANEXO 1.3 → "sbld.c" código do Programa principal (Controlo PD)

```
#include "sbld.h"
#include "sbldADC.h"
#include "sbldUART.h"
#include "sbldtimer.h"
#include "sbldExtINT.h"
#include "bit_tools.h"
#include "lcd.h"
#include <ctype.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

#define sbi(port,bit)  (port |=  ( 1 << bit ))

#define cbi(port,bit)  (port &= ~( 1 << bit ))



float disparo=100;
unsigned char pot[3]; //potencia absorvida pelo motor
extern unsigned int alt; // Altura pretendida
double  Altura; // Altura Actual
float erro0=0; // erro actual
float erro1=0; // erro anterior
float derro=0; // derivada do erro
unsigned int u; // variável auxiliar
unsigned int y=1; //variável auxiliar
unsigned int i; //variável auxiliar
unsigned char j; //variável auxiliar
double soma;
unsigned int adc_value[100]; // vector de valores do ADC
float adc_med; //valor médio do ADC
int l=0; //variável auxiliar
int x=0; //variável auxiliar
unsigned char numero[10]="         ";//string auxiliar
unsigned char mudolinha[4]="\n\r"; //string para mudar de linha
unsigned char linha1[11]="Pot->  0(w)"; // string (modificável) para enviar para o Display
unsigned char linha2[12]="Alt->  0(cm)"; // string (modificável) para enviar para o Display
/////////////////Tabela da potencia///////////////
Int potencia[68]=
{14,16,17,18,20,21,23,25,27,29,31,33,34,36,39,41,45,46,49,52,53,56,59,61,64,67,70,72,75,78
,81,
84,87,90,94,97,99,103,106,109,112,116,119,123,126,130,134,137,140,143,147,150,153,156,1
60,162,
167,170,173,176,180,183,186,189,192,195,199,202};
////////////////////////////////////////////////////////////////////////
static void io_init(void)
{
```

```
	// Configuracao do porto B
	PORTB = 0x0;
	DDRB = 0xFF;
	// Configuracao do porto C
	PORTC = 0x0;
	DDRC=0xFE;
	// Configuracao do porto D
	PORTD = 0x0;
	DDRD=0x13;
}
int main(void)
{
	// Incializacoes
	io_init();
	adc_init();
	extint_init();
	timers_init();
	uart_init();

	sei();//activação de todas as interrupções

	cbi(PORTB,0);  //Enable		LCD
	cbi(PORTD,6); // RW		LCD
	cbi(PORTB,1); // RS		LCD

	while(1)
	{
		// TODO: Add your code here
		adc_value[l]=adc_data[0];// guarda o valor do adc
		if(l==100)// faz 100 leituras
		{
			for(u=0;u<100++)
			{
				soma=soma+adc_value[u]; // Calculo da média
			}

			adc_med=soma/100; // Calculo da média
			/////PID//////////////////////
			Altura = 13.040354902516*pow((adc_medf/380),3)-
80.016034290549*pow((adc_medf/380),2)+179.952974341045*(adc_medf/380)-
99.058946926390; //Calculo da altura
			erro0=alt-Altura; /calculo do erro
			if(x==0) //Excepcao para o primeiro caso (lixo)
			{
				erro0=0;
				x=1;
			}

			derro=erro0-erro1;//calculo de derro
			disparo=(erro0*(-0.03)-2.5*derro)+disparo;
// calculo do disparo

			if(disparo<=27) // limite inferior do disparo
				disparo=27;
			if(disparo>=255) // limite superior do disparo
				disparo=255;
```

```
                        itoa(erro0,numero,10);
                        for(i=0;i<9;i++){
                                transmite(numero[i]);} //envia erro (usado para debug)
                        for(i=0;i<4;i++){
                                transmite(mudolinha[i]);}

                                for(i=0;i<10;i++)
                                {
                                numero[i]='\0'; //limpa a string
                                }

                        itoa(derro,numero,10);
                        for(i=0;i<9;i++){
                                transmite(numero[i]);} //envia derro (usado para debug)
                        for(i=0;i<4;i++){
                                transmite(mudolinha[i]);}

                                for(i=0;i<10;i++)
                                {
                                numero[i]='\0'; //limpa a string
                                }

                        itoa(disparo,numero,10);
                        for(i=0;i<9;i++){
                                transmite(numero[i]);}//envia disparo (usado para debug)
                        for(i=0;i<4;i++){
                                transmite(mudolinha[i]);}
                        for(i=0;i<4;i++){
                                transmite(mudolinha[i]);}

                                for(i=0;i<10;i++)
                                {
                                numero[i]='\0';
                                }


                        erro1=erro0;//mudamos para o erro anterior o valor do erro actual
                        OCR2=(int)disparo; // actualização de OCR2
                        ///////FIM DE PID//////////
                        soma=0; //faz "reset" á variável soma
                        l=-1; // artifício de calculo para a seguir ter o valor 0
                }
                l++;
        }
}
//-------------------------------------------------------------------------------------------
```

// ANEXO 1.4 → "sbldExtINT.c", interrupções externas para detecção da passagem por zero

```
#include "sbld.h"
#include "sbldExtINT.h"
#define sbi(port,bit)  (port |=  ( 1 << bit ))

#define cbi(port,bit)  (port &= ~( 1 << bit ))


void extint_init(void)
{
```

```
        // INT0 activo, Modo: flanco ascendente
        // INT1 avtivo, Modo: flanco descendente
        MCUCR |= 0x0b;
        GICR = 0xc0;
}

SIGNAL(SIG_INTERRUPT0)
{
  TCCR1B = 0x01;

}

SIGNAL(SIG_INTERRUPT1)
{
  TCCR1B = 0x01;

}
//---------------------------------------------------------------------------------------
```

// ANEXO 1.5 → "sbldTimer.c", configuração dos timers

```
#include "sbld.h"
#include "sbldUART.h"
#include "lcd.h"
#define sbi(port,bit)  (port |=  ( 1 << bit ))

#define cbi(port,bit)  (port &= ~( 1 << bit ))
extern unsigned char linha1[20];
extern unsigned char linha2[20];

int tim=1;
void timers_init(void)
{
        // Timer/Counter0 fonte de clock: clock de sistema
        // Timer/Counter0 valor do clock: por paragem
        TCNT0 = 0x00;
        TCCR0 = 0x00;

        // Timer/Counter1 fonte de clock: clock de sistema
        // Timer/Counter1 valor do Clock: 14745.600kHz
        // Timer/Counter1 Modo: Normal
        // Timer/Counter1 Saída: A: Disconectado, B: Disconectado
        OCR1A = 0x7333;
        OCR1B = 0x2d00;
        TCNT1 = 0x8ccd;
        TCCR1A = 0x00;
        TCCR1B = 0x01;

        // Timer/Counter2 Fonte de relógio: clock de sistema
        // Timer/Counter2 valor do clock: 14.400kHz
        // Timer/Counter2 Modo: Fast PWM
        // Timer/Counter2 Saída: Invertida
        sbi(DDRB, 3);
        ASSR = 0x00;
        OCR2 = 0x89;
        TCNT2 =0x1a;
        TCCR2 =0x7d;

        TIMSK = 0xc4;
```

```
}

SIGNAL(SIG_OVERFLOW1)
{

        // Reinicia o valor do timer1
        TCNT1 = 0x8ccd;
        TCCR1B = 0x00;
        TCCR2 =0x7d;
}

SIGNAL(SIG_OUTPUT_COMPARE2)
{

        sbi(PORTD,4);//activa o pino PD4 dando o disparo ao triac
}

SIGNAL(SIG_OVERFLOW2)
{
        // Reinicia o valor do timer2
        TCNT2 =0x1a;
        cbi(PORTD,4);// desliga disparo
        TCCR2 = 0x00;
}
//-------------------------------------------------------------------------------------------
```

"ANEXO 1.6 → sbldADC.c", configuração do conversor analógico digital para tratamento do sinal recebido do sensor

```
#include "sbld.h"
#include "sbldADC.h"

#define sbi(port,bit)  (port |=  ( 1 << bit ))

#define cbi(port,bit)  (port &= ~( 1 << bit ))


void adc_init(void)
{

        // Clock do ADC: 460.800kHz
        // ADC Tensão de referência: Interna 2.56V
        // Modo do ADC: Free Running
        SFIOR |= 0x0;
        ADMUX = 0xc0;
        ADCSR = 0xed;

}

SIGNAL(SIG_ADC)
{
        static unsigned char input_index = 0;
        //Leitura do resultado da conversão do ADC
        adc_data[input_index] = ADCW;
        // Selecção da próxima entrada ADC
        if (++input_index > (ADC_LAST_INPUT - ADC_FIRST_INPUT))
                input_index = 0;
        ADMUX &= 0xE0;
```

```
        ADMUX |= input_index + ADC_FIRST_INPUT;
        //Começa a próxima conversão ADC
        sbi(ADCSR, ADSC);
}
//-------------------------------------------------------------------------------------------
```

//ANEXO 1.7 → "sbldUART.c" código das interrupções de porta série (presente em controlo com PD)1

```
#include "sbld.h"
#include "sbldUART.h"
#include <stdlib.h>
unsigned int i;
unsigned char g;
unsigned char j;
unsigned int k=160;
unsigned int u;
volatile unsigned int alt=25;
unsigned char altpretend[2]="  ";
unsigned int altura[5];
unsigned char numero[10];
unsigned char mudalinha[4]="\n\r";
unsigned char string[20]="Codigo nao valido";

void uart_init(void)
{
        UBRRL = 0x17;// Baud Rate 38400
        UBRRH = 0x00;// 8 bits de dados
        UCSRA = 0x00;// Modo assíncrono
        UCSRC = 0x86;// Sem paridade
        UCSRB = 0x98;// 1 stop bit
}

void transmite( unsigned char data )//função retirada do manual do Atmega
{
/* Wait for empty transmit buffer */
while ( !( UCSRA & (1<<UDRE)) )
        ;
        /* Copy ninth bit to TXB8 */
        UCSRB &= ~(1<<TXB8);
        if ( data & 0x0100 )
        UCSRB |= (1<<TXB8);
        /* Put data into buffer, sends the data */
        UDR = data;
}

unsigned char recebe( void ) //função retirada do manual do Atmega
{
        /* Wait for data to be received */
        while ( !(UCSRA & (1<<RXC)) )
        ;
        /* Get and return received data from buffer */
        return UDR;
}

int testa_byte(void) // verifica se tem alguma coisa no buffer
{
        if(!(UCSRA & (1<<RXC) ))
```

```
                return 0;
        return 1;
}



SIGNAL(SIG_UART_RECV)// rotina de interrupção da porta série
{
        j=recebe();
        k=OCR2;
        if(j=='w'||j=='+'||j=='-'||j=='h')//Casos possiveis
        {
                if(j=='w')// transmite valor de disparo
                {
                        itoa(k,numero,10);
                        for(i=0;i<9;i++){
                        transmite(numero[i]);}
                        for(i=0;i<4;i++){
                        transmite(mudalinha[i]);}
                }
                if(j=='-' && alt>0) // incrementa em uma unidade (cm) a altura pretendida
                {
                        alt=alt-1;
                }
                if(j=='+' &&  alt<55) // decrementa em uma unidade (cm) a altura pretendida
                {
                        alt=alt+1;
                }
                if(j=='h')// insere directamente uma altura escolhida
                {
                        UCSRB = 0x18;//desliga as interrupções de porta série
                        for(u=0;u<=1;u++)
                        {
                                while(testa_byte()==0)
                                {
                                        ;espera por um digito
                                }

                                j=recebe();
                                altura[u]=j;// guarda valor da altura

                        }
                        alt=altura[0]*100+altura[2]*10+altura[1]-5328; // converte para
inteiro

                                itoa(alt,altpretend,10);
                                for(i=0;i<=5;i++)
                                {
                                        transmite(altpretend[i]);//transmite o valor
anteriormente inserido (So para questões de debug)
                                }
                                for(i=0;i<4;i++)
                                {
                                        transmite(mudalinha[i]);
                                }
                                for(i=0;i<5;i++)
```

```
                                {
                                altpretend[i]='\0';
                                }

                        UCSRB = 0x98;//liga interrucoes da porta serie
                }

        }
        else
        {
                for(i=0;i<=20;i++)
                {
                        transmite(string[i]);//transmite que recebeu um carácter inválido
                }
        }
}
```

ANEXO 2 - Código em Matlab para o controlo por lógica difusa:

```
//-------------------------------------------------------------------------
-


close;clc;
g75=[.95 .75 .75];g8=[.8 .8 .8];i=0;op=1;

fig=figure('menubar','none','units','normalized','position',[0 0 1 .95],'name','Controlo da posição
de um Disco',...
        'numbertitle','off','color','black');
set(fig,'DoubleBuffer','ON');

axes('drawmode','fast','userdata','membrane','units','normalized','Position',[0.4 0.4 1
1],'box','on','visible','off');

%PUSH's-----------------------------------------------------------------
p1=uicontrol('Style','push', 'string','Regras Fuzzy', 'Units','normalized', 'Position',[0.62 0.29 0.08
0.03], 'callback','[op]=1', ...
          'backgroundcolor',[.75 .75 .75]);

p2=uicontrol('Style','push', 'string','SAIR', 'Units','normalized', 'Position',[0.62 0.18 0.08 0.06],
'callback','[op]=5,fclose(s1),close all', ...
          'backgroundcolor',[.75 .75 .75]);

%TEXT's-----------------------------------------------------------------
t1=uicontrol('Style','text', 'string','Posição a alcançar (cm)', 'FontSize',12,'Units','normalized',
'Position',[0.62 0.7 0.25 0.03],...
         'backgroundcolor',[.75 .75 .75]);
t2=uicontrol('Style','text', 'string','Valor actual do Disparo (graus)',
'FontSize',12,'Units','normalized', 'Position',[0.62 0.46 0.25 0.03],...
         'backgroundcolor',[.75 .75 .75]);
t3=uicontrol('Style','text', 'string','Posição actual (cm)', 'FontSize',12,'Units','normalized',
'Position',[0.62 0.58 0.25 0.03],...
         'backgroundcolor',[0.75 .75 .75]);
t4=uicontrol('Style','text', 'string','Visualização das regras fuzzy',
'FontSize',12,'Units','normalized', 'Position',[0.62 0.34 0.25 0.03],...
         'backgroundcolor',[0.75 .75 .75]);

%EDIT's-----------------------------------------------------------------
e1=uicontrol('Style','edit', 'FontSize',12,'Units','normalized', 'Position',[0.62 0.65 0.08 0.03],...
          'backgroundcolor',[.15 .61 .95]);

e2=uicontrol('Style','edit', 'FontSize',12,'Units','normalized', 'Position',[0.62 0.41 0.08 0.03],...
          'backgroundcolor',[.15 .61 .95]);

e3=uicontrol('Style','edit', 'FontSize',12,'Units','normalized', 'Position',[0.62 0.53 0.08 0.03],...
          'backgroundcolor',[.15 .61 .95]);

%%-----------------------------------------------------------------------
%%-----------------------------------------------------------------------
   hold on;
   subplot('position',[0.1,0.1,0.3,0.88]);
   B0= IMREAD('kit.bmp');
   fig0=image(B0);
   axis off;
```

```
    drawnow;

%-----------------Abertura da Porta Série--------------------------------
 s1=serial('COM3','Baudrate',38400); %define porta
 fopen(s1); %abre porta

%-------------Inicialização das variáveis-------------------------------
     erro1=0
     pos=25
     disparo3=180
%--------------------Controlo-------------------------------------------
    while op~=5


    switch op;

    case 1


    hold on;
    subplot('position',[0.51,0.8,0.37,0.17]);
    B1= IMREAD('feup.bmp');
    fig1=image(B1);
    axis off;
    drawnow;

    if pos~=str2num(get(e1,'String'))
        pos=str2num(get(e1,'String'))
          if pos<1 | pos>55

         errordlg('Posições Válidas de 1 a 55!!!','Erro');
         set(e1,'String',' ')
         op=1
         end
    else
    end
    pos
    w=s1.BytesAvailable

    if w~=0
    op=3
    end



    case 2
    ruleview(fuzzy_sets)
    op=1

    case 3

    hold on;
    subplot('position',[0.14,0.22,0.06,0.62]);%x,y tamx, tamy
    B2= IMREAD('regua.bmp');

    fig2=image(B2);
    axis off;
    drawnow;
```

```
%calculo da saida do sensor em funçao da altura pretendida


        rec=fscanf(s1,'%c',w) %le o valor actual do ADC
        k=s1.BytesAvailable


        if rec=='r'|rec=='>r'

             fprintf(s1,'%c','r') %pede ao avr o valor actual do ADC
             pause(1);
             q=s1.BytesAvailable
            while(q==0)
               q=s1.BytesAvailable
            end
                if q~=0

                ADC2=fscanf(s1,'%c',q) %le o valor actual do ADC
                k=s1.BytesAvailable
                end
                if k~=0
                karamelo=fscanf(s1,'%c',k) %limpa eventual lixo da porta série
                end
             ADC=str2num(ADC2)
             Altura = -2.008377743419*((ADC/380)^6) + 25.856920825050*((ADC/380)^5) -
135.300139807327*((ADC/380)^4) + 376.958350602915*(ADC/380)^3 -
606.010587307718*((ADC/380)^2) + 564.866278844587*(ADC/380)- 210.060393336037
             if Altura<0
               Altura=0
              end
             set(e3,'String',int16(Altura))


             hold on;

             subplot('position',[0.2,(Altura*0.01),0.2,0.6]);%x,y tamx, tamy
             B3= IMREAD('disco.bmp');

             fig3=image(B3);
             axis off;
             drawnow


             erro0=pos-Altura %calculo do erro

             derro=erro1-erro0 %calculo da derivada do erro

             disparo1=evalfis([erro0 derro],fuzzy_sets)+disparo3 %consulta da tabela fuzzy
             disparo3=disparo1
             disparo2=int16(disparo1) %converte p int o valor do disparo

             set(e2,'String',disparo2*0.16)
             disparo=num2str(disparo2) %converte p string o valor do disparo
               if(disparo2<100)
```

```
                fprintf(s1,'%c','n')

                fprintf(s1,'%c',disparo(1)) %envio do valor do disparo
                fprintf(s1,'%c',disparo(2))
            else
                fprintf(s1,'%c','m')
                fprintf(s1,'%c',disparo(1)) %envio do valor do disparo
                fprintf(s1,'%c',disparo(2))
                fprintf(s1,'%c',disparo(3))
            end

            alt=num2str(int16(Altura))
            t=s1.BytesAvailable
            while(t==0)
              t=s1.BytesAvailable
            end
              var=fscanf(s1,'%c',t)
            if(var=='h')
              if(int16(Altura)>9)
                fprintf(s1,'%c',alt(1))
                fprintf(s1,'%c',alt(2))
              else
                fprintf(s1,'%c','0')
                fprintf(s1,'%c',alt(1))
              end
            else
            end
            erro1=erro0 %actualizaçao do erro

        end %fecha if rec='r'

        if k~=0
           karamelo=fscanf(s1,'%c',k) %limpa eventual lixo da porta série
        end

        op=1



otherwise
end
end
```