Universidade do Porto

FEUP Faculdade de Engenharia

Licenciatura em Engenharia Electrotécnica e de Computadores
Ramo de Automação, Produção e Electrónica Industrial

Tecnologias de Sistemas de Controlo e Automação
5º Ano – 1ºSemestre
2005 - 2006

Métodos de Controlo PID via PC

Trabalho Prático nº2

Elaborado por:

Ana Luísa Mendonça Amaro Martins	ee04255
Magno Alexandre Tavares Petiz dos Santos	ee00168
Ricardo Manuel da Rocha Canossa	ee00114

Índice

Índice de Figuras

Índice de Tabelas

Sumário

Este trabalho vem no seguimento do trabalho anterior em que se estudou o controlo por PID feito por um controlador PID industrial e por um Autómato. Neste trabalho o controlo PID será feito via PC através de uma aplicação realizada em Delphi.

O controlo será realizado segundo duas metodologias:

1- Controlo em malha fechada da Planta
2- Obtenção da ordem do modelo da Planta mais adequada e respectivos parâmetros através do método dos mínimos quadrados

Para ambos os casos será necessário implementar o protocolo de comunicação Hostlink, para que o PC possa comunicar com o autómato e vice-versa.

As ligações eléctricas e o programa do autómato mantêm-se do trabalho anterior tendo havido lugar a algumas alterações no programa do autómato.

Objectivos

Pretende-se controlar a temperatura de um forno eléctrico através de um controlador PID comandado por um PC, sendo feito o controlo em malha fechada da planta, desenvolvendo assim uma aplicação numa linguagem de programação alternativa ao SCADA para aquisição e controlo que comunique via porta com o Autómato.

Pretende-se também aplicar sinais de excitação ao sistema, de modo a ser possível a recolha de dados para identificar através de algoritmos de Minimos Quadrados modelos discretos de sistemas dinâmicos.

Alterações no programa do autómato

Para a execução deste programa teve que ser efectuada uma pequena alteração ao programa do autómato, na realidade esta alteração foi simplesmente a implementação de uma condição de funcionamento do PID interno deste, em que para o PID do autómato estar activo teria que ser passado um valor a uma variável (pelo Delphi) a fim de o habilitar ou desabilitar.

Protocolo Hostlink

Para a comunicação entre o autómato e o PC foi utilizado o protocolo "Host Link – FINS COMMANDS".
Este protocolo consiste basicamente no envio e recepção de tramas específicas por parte do PC e do autómato:

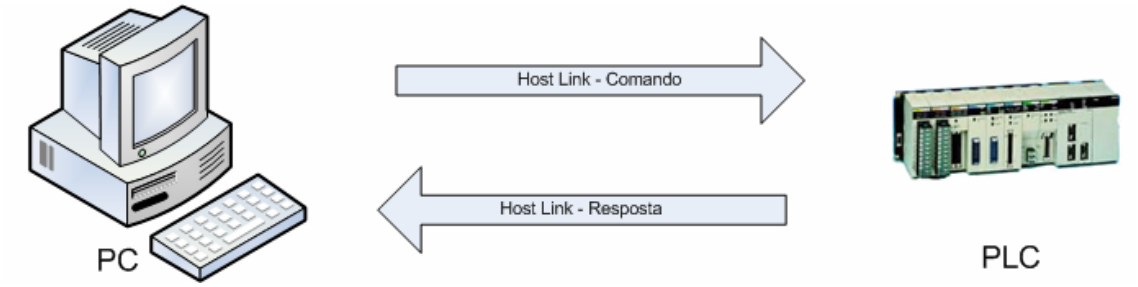

Figura 1 - Comunicação PC-PLC

Formato das tramas:

Trama de Comando (PC-PLC):

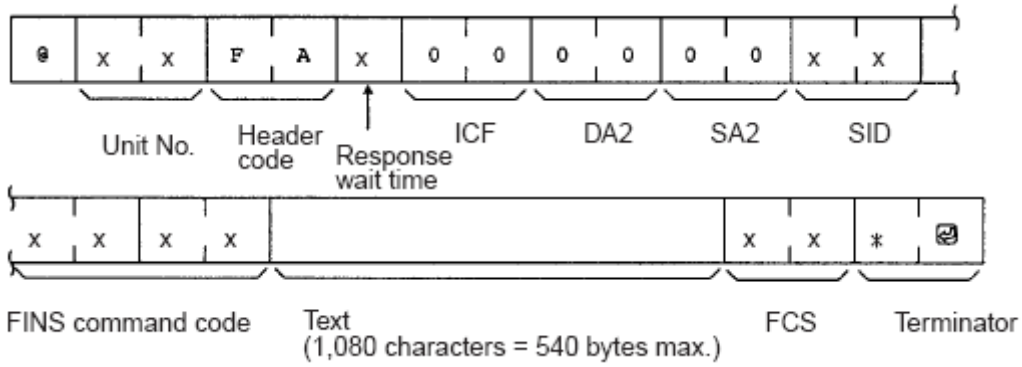

Figura 2 - Trama de comando

Trama de Resposta (PLC-PC):

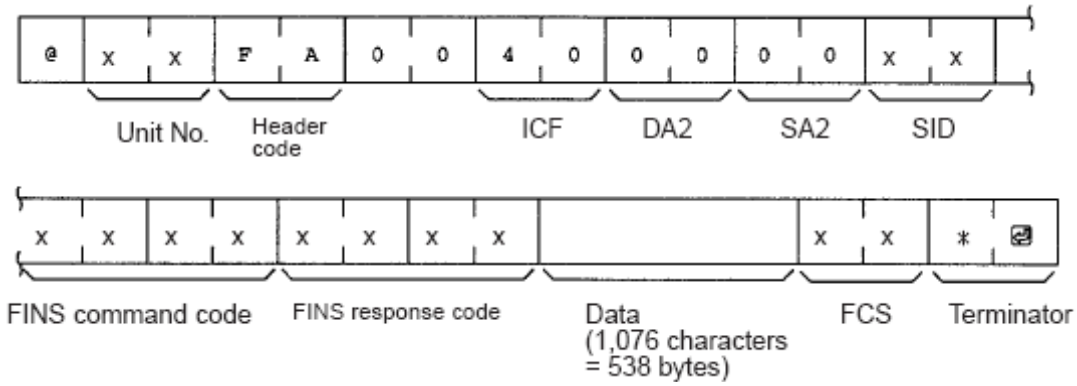

Figura 3 - Trama de Resposta

Descrição dos campos das tramas:

@	Identificador de um comando Host Link, tem que estar sempre no íncio de cada trama
Unit Number	Colocado em BCD de 0 a 31 para cada unidade Host Link
Header code	Especificado em 2 caracteres, faz a distinção entre os diversos tipos de comandos. Para comandos "FINS" coloca-se "FA"
Response wait time	Este parâmetro indica o tempo que a CPU unit demora a receber uma trama de comando antes de enviar uma trama de resposta. Pode ser colocado de 0 a F
ICF (Information Control Field)	Colocado a "00" (ASCII: 30,30) quando o PC está directamente ligado à CPU unit.
DA2 (Destination Unit Address)	Colocado a "00"
SA2 (Source Unit Address)	Colocado a "00"
SID (Source ID)	Colocado a "00"
FINS command mode DNA e GTC	Utilizados quando se utiliza a rede para enviar/receber tramas
Text	Parâmetros do Comando enviados para o PLC
FCS	Frame Check Sequence – resultado do cálculo do FCS da trama, consiste em fazer um xor entre todos os caracteres da trama
Terminator	Os caracteres '*' e ASCII:13 indicam o fim de cada trama

HostLink programado em Delphi

Para proceder à implementação do protocolo HostLink foram definidas as seguintes variáveis em Delphi:

Nome Váriavel	Valor	Definição
base_frame	'@00FA'	Constituido por '@' + Unit Number ('00') + Header Code ('FA')
response_wait_time	'0'	Tempo para resposta de 0 segundos
icf_command	'00'	ICF para o envio de uma trama
icf_response	'40'	ICF da recepção de uma trama
header_frame	'000000'	DA2 ('00') + SA2 ('00') + SID ('00')
code_write	'0102'	FINS COMMAND para escrita
code_read	'0101'	FINS COMMAND para leitura
dmword	'82'	Identificador da word do tipo DM

Tabela 1 – Variáveis globais utilizadas para gerar tramas

As tramas que são enviadas para o PLC são construídas pela aplicação, recorrendo a duas funções, frametowrite() e frametoread(), em que uma é uma trama de escrita e outra de leitura:

Trama de comando:

```
function frametowrite(address:integer; memareac:string; data:integer):string;
```

Esta função recebe como parâmetros que informação, data, que tipo de memória, memareac, e em que endereço, address, se pretende escrever. Estes dados são assim utilizados para gerar uma trama, com o formato anteriormente descrito, sendo no fim retornada.

Código implementado em Delphi:

```
function frametowrite(address:integer; memareac:string; data:integer):string;
begin
  result := base_frame;
  result := result + response_wait_time;
  result := result + icf_command;
  result := result + header_frame;
  result := result + code_write;
  result := result + memareac;
  result := result + IntToHex(address, 4) + '00';
  result := result + '0001';
  result := result + IntToHex(data, 4);
  result := result + frame_check_sequence(result);
  result := result + '*' + chr(13);
end;
```

Trama de resposta:

function frametoread(address:integer; memareac:string):string;

Esta função recebe como parâmetros qual o tipo de memória e qual a posição que pretende ler, contudo apesar de ser possível ler mais que um endereço ou apenas um bit desse endereço, esta função apenas permite ler um e completo endereço de memória. Uma vez mais a trama criada é retornada à função que a chama.

```
function frametoread(address:integer; memareac:string):string;
begin
  result := base_frame;
  result := result + response_wait_time;
  result := result + icf_command;
  result := result + header_frame;
  result := result + code_read;
  result := result + memareac;
  result := result + IntToHex(address, 4) + '00';
  result := result + '0001';
  result := result + frame_check_sequence(result);
  result := result + '*' + chr(13);
end;
```

Cálculo do FCS para as tramas de comando e resposta

O PLC calcula o valor de FCS para cada trama que recebe e envia. No caso de uma trama de recepção, com o FCS calculado e comparando com o campo de FCS da trama recebida o PLC detecta se a trama está incorrecta, para o PC poder detectar efectuar a mesma operação o PLC tem também que o calcular e enviar na trama. Do lado do PC também é efectuado o mesmo procedimento. Com este mecanismo é possível encontrar erros de transmissão de dados e impedir que aconteçam erros de leitura.

O FCS é calculado com um XOR entre todos os caracteres.

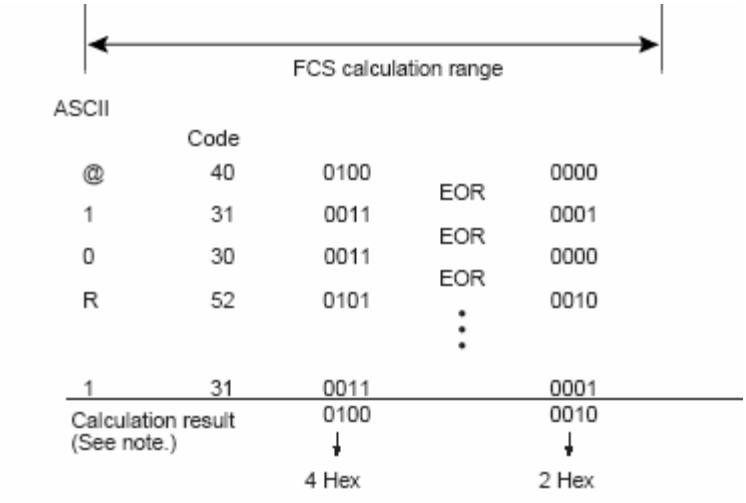

Figura 4 - Cálculo do FCS

Foi implementada uma função que quando invocada calcula o FCS da trama, sendo esta recebida como parâmetro, FrameToCheck. Devolve o resultado num valor hexadecimal de 2 dígitos.

Código em Delphi:

```
function frame_check_sequence(FrameToCheck : string) : string;
var i, fcs : integer;
begin
  fcs := 0;
  for i := 1 to length(FrameToCheck) do
     fcs := fcs xor ord(FrameToCheck[i]);

  result := IntToHex(fcs, 2);
end;
```

Recepção de Dados (tramas de resposta) pela porta série

Para a recepção de dados pela porta série foi implementada a seguinte máquina de estados:

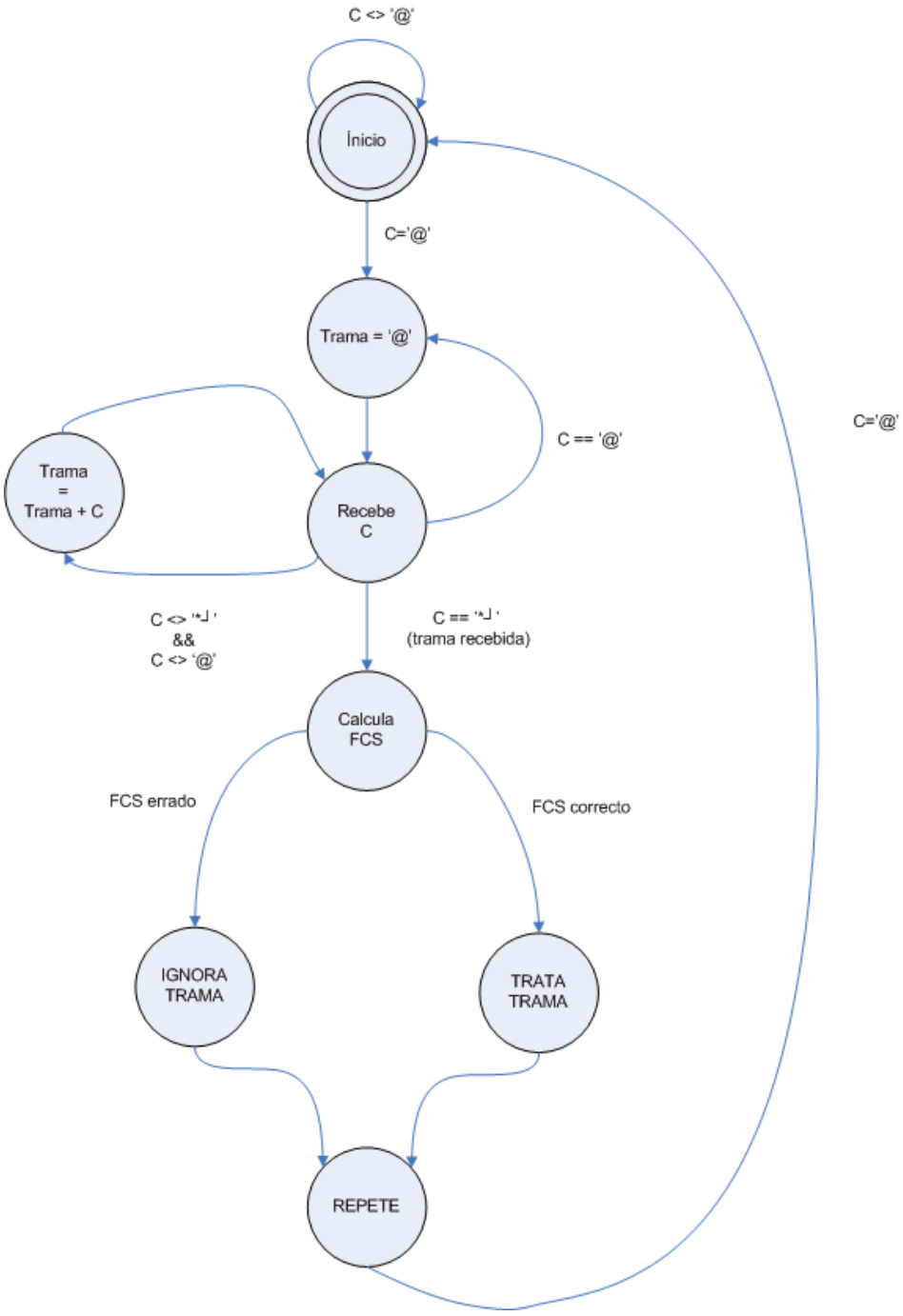

Figura 5 – Recepção de dados pela porta série

Código em Delphi:

```
procedure Tfmain.comportRxChar(Sender: TObject; Count: Integer);
var vchar : string;
  i       : integer;
begin
  comport.ReadStr(vchar, Count);

  for i := 1 to Count do
  begin
    case validation of
      vstart:
        begin
          if vchar[i] = '@' then
          begin
            validation := vstop;
            response_frame := vchar[i];
          end;
        end;
      vstop:
        begin
          if vchar[i] = chr(13) then
          begin
            validation := vstart;
            trata_response_frame();
          end
          else
          begin
            if vchar[i] = '@' then
              response_frame := vchar[i]
                                                    else
              response_frame := response_frame + vchar[i]
          end;
        end;
    end;
  end;
end;
```

Envio do Set Value (trama de comando) para o autómato

O envio do Set Value é enviado para o autómato procedendo como mostra na seguinte figura:

Figura 6 – Envio do Set Value para o autómato

Com este botão podemos assim enviar o valor neste caso 50 para o autómato e depois poder observar a evolução da temperatura pela acção do PID implementado no autómato

Código em Delphi para o envio do Set Value:

```
procedure Tfmain.setvalue_frameClick(Sender: TObject);
var
  command_frame : string;
  data          : integer;
begin
  if comport.Connected then
  begin
    senviar := 1;
    data := strtoint(ed_setvalue.text)*40;
    command_frame := frametowrite(set_value, dmword, data);
    comport.WriteStr(command_frame);
  end
  else
    showmessage('COMPORT is closed! Open it!!');
end;
```

Controlo em malha fechada da planta

Depois de se ter o protocolo Hostlink a funcionar, construiu-se a seguinte interface para o utilizador em que este pode enviar com facilidade a temperatura desejada (Set Value) para o forno e o valor do PWM:

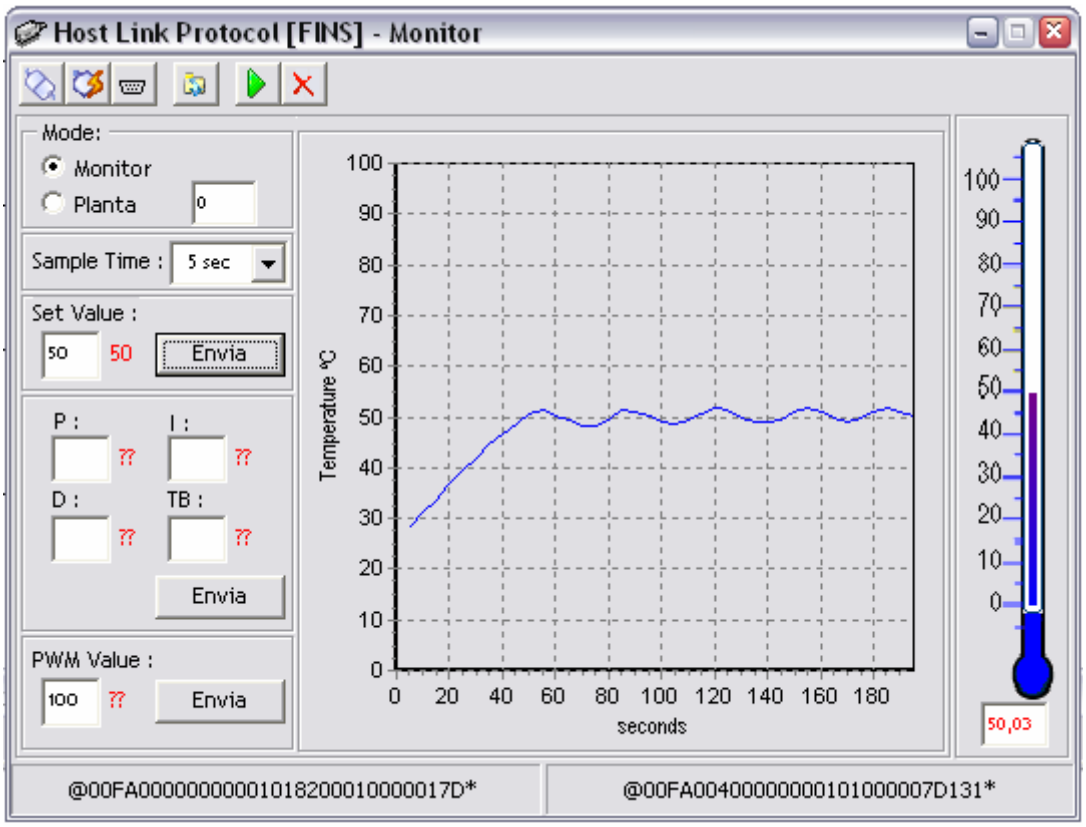

Figura 7 – Interface Construída em Delphi em que a temperatura desejada é 50ºC

	Abrir a porta série
	Fecha a porta série
	Configurar os parâmetros da porta série
	Abrir ficheiro de ruído
	run
	stop

Tabela 2 – Funcionalidades básicas da interface

Nesta interface o utilizador pode escolher entre os modos “Monitor” e “Planta”.

No modo ‘Monitor’, o utilizador pode enviar o valor de temperatura desejado (setvalue) através do botão envia do Set Value , estando a funcionar o PID implementado no autómato. Já quando clica sobre envia em pwm, este desabilita o PID do autómato, passando a funcionar o valor para pwm que foi enviado.

Neste modo é possível ao utilizador ver de que modo se encontra a temperatura a evoluir de acordo com o Set Value, através do gráfico presente na interface (Temperatura em função do tempo) e também pode ver o valor preciso actual de temperatura a que o forno se encontra.

No modo ‘Planta’, o operador escolhe um ficheiro, com dados de entrada de PWM para enviar para o autómato, sendo também nesta situação desabilitado o PID do autómato. Neste modo, depois de obter os parâmetros PID através da aplicação do método dos mínimos quadrados, estes são introduzidos nos respectivos locais da interface obtendo-se a resposta do sistema para estes parâmetros.

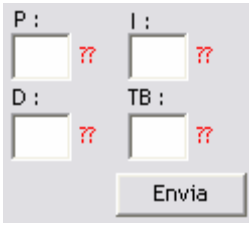

Figura 8 – Locais da interface para introduzir os valores do PID

Em qualquer dos modos é possível visualizar quais as tramas que estão a ser recebidas e enviadas em baixo na interface:

Figura 9 – Exemplo de tramas trocadas entre o PC e o PLC

Comentários a esta parte

Foi de todo o interesse poder trabalhar com uma ferramenta deste género, pois revelou-se muito útil e poderosa para a comunicação do PC via porta série com o PLC. Adquiriu-se o conhecimento do protocolo de comunicação hostlink que poderá ser uma referência no futuro para outras aplicações em que se utilize um autómato Omrom.

Esta ferramenta permite em tempo real percepcionar o que se passa no forno de uma maneira simples e objectiva para o utilizador, pois permite ver a qualquer instante a temperatura actual do forno.

Obtenção do modelo da Planta e respectivos parâmetros através do método dos mínimos quadrados

De modo a obter a planta do sistema utilizou-se o método dos mínimos quadrados.

Para poder aplicar este método procedeu-se da seguinte maneira:

Na referência do PWM foi adicionado um sinal de ruído. Este sinal foi gerado com 128 pontos aleatórios em scilab que varre uma gama de valores entre 0 e 100 através da seguinte função:

```
noise = nivel + largura*rand([1:128]','normal')

//noise -  sinal de ruído que se pretende obter
//nível  - valor de referência do pwm
//largura - variação do ruído
```

Aplicou-se este sinal à entrada da planta u(t) e retirou-se a resposta do sistema a este sinal y(t), sendo o sistema em malha aberta:

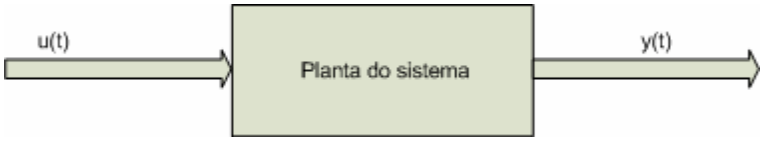

Figura 10 – Sistema em Malha Aberta

sendo a função transferência genérica do sistema:

$$\frac{y(z)}{u(z)} = \frac{b_1 z^{n-1} + b_2 z^{n-2} + ... + b_n}{z^n + a_1 z^{n-1} + ... + a_n}$$

Aplicação do método dos mínimos quadrados:

Para aplicar este método é necessário inicialmente escolher o ponto de funcionamento e subtrai-lo em todas as amostras.

De modo a obter os parâmetros da planta aplica-se a expressão:

$\theta = \left(X^T X\right)^{-1} X^T y$.

Exemplo da aplicação do método para um sistema de 1ª ordem:

Sendo a função de transferência:

$$\frac{y(z)}{u(z)} = \frac{b_1}{z + a_1}$$

$$y(z) \times (z + a_1) = b_1 \times u(z) \Rightarrow y[k] = -a_1 \times y[k-1] + b_1 \times u[k-1]$$

$$\theta_1 = a1$$
$$\varphi_{11} = -y[k-1]$$
$$\theta_2 = b_1$$
$$\varphi_{12} = u[k-1]$$

$$y[k] = \theta_1 \times \varphi_1 + \theta_2 \times \varphi_2$$

$$\theta = \begin{bmatrix} a_1 \\ b_1 \end{bmatrix} \qquad \varphi_1 = \begin{bmatrix} -y[k-1] \\ u[k-1] \end{bmatrix}$$

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_N \end{bmatrix} \qquad X = \begin{bmatrix} \varphi_1^T \\ \varphi_2^T \\ \dots \\ \varphi_N^T \end{bmatrix}$$

$Sabendo \quad X \quad e \quad Y \quad obtém-se: \quad \theta = (X^T X)^{-1} X^T Y \quad , \quad obtendo \quad a_1 \quad e \quad b_1$

$$Posto \quad isto \quad obtém-se \quad o \quad erro: \quad E = (Y - X\theta) = \begin{bmatrix} e_1 \\ e_2 \\ \dots \\ e_n \end{bmatrix}$$

$e \quad o \quad erro \quad quadrático \quad da \quad estimativa: \quad S = (Y - X\theta)^T (Y - X\theta)$

Para as outras ordens procede-se da mesma maneira. O erro quadrático obtido entre o valor estimado e o valor medido deve ser o menor possível, por isso deve-se escolher a ordem do sistema que apresenta o menor erro quadrático.

Foi implementado em Scilab um script que calcula os parâmetros da planta para qualquer ordem desejada através do método dos mínimos quadrados, tendo sido implementado seguindo os passos anteriores da descrição do método.

Resultados Obtidos

O método dos mínimos quadrados implementado em Scilab permitiu obter os seguintes resultados:

Para um sistema de 1ª ordem obteve-se:

Figura 11 – Sistema de 1ªordem (y-valores reais, ye- valores estimados)

Erro quadrático	76
Função de transferência	$\frac{Y(Z)}{U(Z)} = \frac{(0.0332)}{z^1 + (-0.9357)}$

Tabela 3 – Parâmetros para um sistema de 1ª ordem

Para um sistema de 2ª ordem obteve-se:

Figura 12 – Sistema de 2ªordem (y-valores reais, ye- valores estimados)

Erro quadrático	27
Função de transferência	$\frac{Y(Z)}{U(Z)} = \frac{(0.0328)Z^1 + (0.0275)}{z^2 + (-0.8263)Z^1 + (-0.0562)}$

Tabela 4 – Parâmetros para um sistema de 2ª ordem

Para um sistema de 3ª ordem obteve-se:

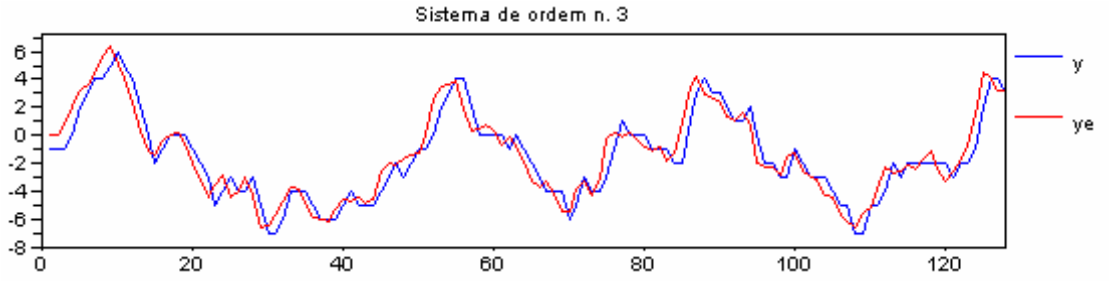

Figura 13 – Sistema de 3ªordem (y-valores reais, ye- valores estimados)

Erro quadrático	26
Função de transferência	$\frac{Y(Z)}{U(Z)} = \frac{(0.0328)Z^2 + (0.0348)Z^1 + (0.0078)}{z^3 + (-0.6035)Z^2 + (-0.2758)Z^1 + (0.0293)}$

Tabela 5 – Parâmetros para um sistema de 3ª ordem

Para um sistema de 4ª ordem obteve-se:

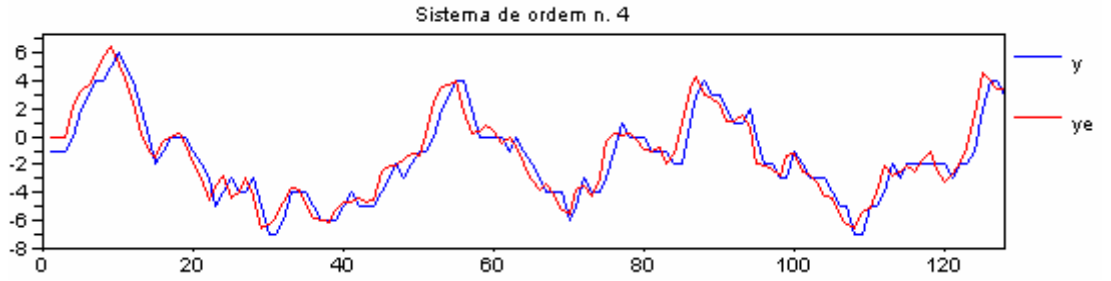

Figura 14 – Sistema de 4ªordem (y-valores reais, ye- valores estimados)

Erro quadrático	24
Função de transferência	$\frac{Y(Z)}{U(Z)} = \frac{(0.0328)Z^3 + (0.0368)Z^2 + (0.0137)Z^1 + (0.0051)}{z^4 + (-0.5458)Z^3 + (-0.1849)Z^2 + (-0.1114)Z^1 + (0.0199)}$

Tabela 6 – Parâmetros para um sistema de 4ª ordem

Para um sistema de 5ª ordem obteve-se:

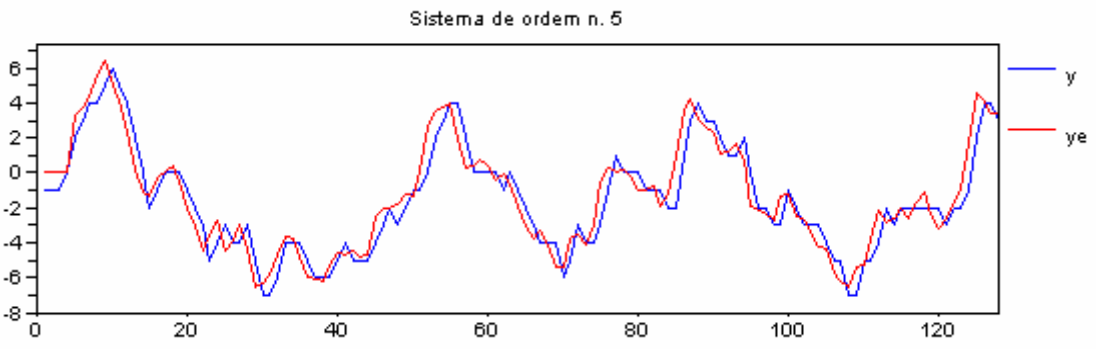

Figura 15 – Sistema de 5ªordem (y-valores reais, ye- valores estimados)

Erro quadrático	28
Função de transferência	$\frac{Y(Z)}{U(Z)} = \frac{(0.0327)Z^4 + (0.0366)Z^3 + (0.0130)Z^2 + (0.0036)Z^1 + (0.0014)}{z^5 + (-0.5562)Z^4 + (-0.1950)Z^3 + (-0.1164)Z^2 + (0.1102)Z^1 + (-0.0707)}$

Tabela 7 – Parâmetros para um sistema de 5ª ordem

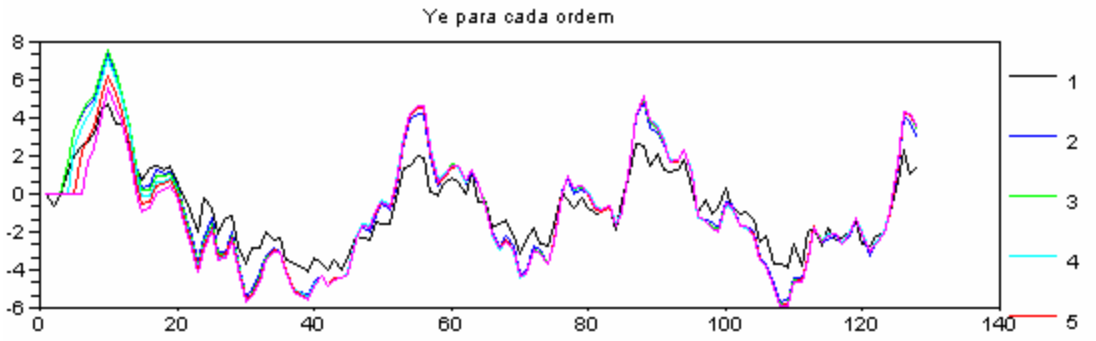

Figura 16 – Valores estimados (ye) para cada ordem

Erro quadrático em função da ordem do sistema:

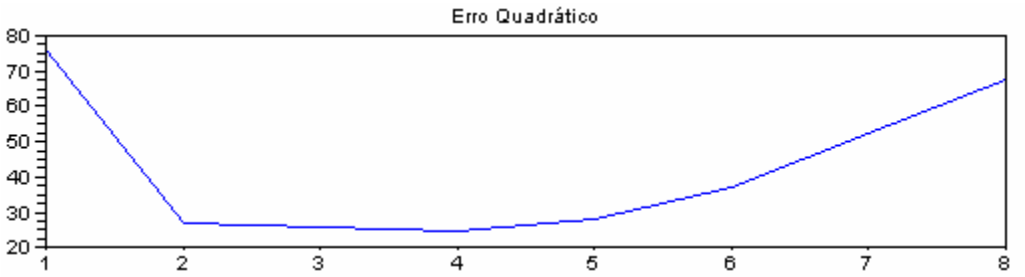

Figura 17 – Erro quadrático em função da ordem do sistema

Como é possível verificar a partir do gráfico anterior, uma maior ordem do sistema não implica um menor erro quadrático, a ordem do sistema que apresenta um menor erro quadrático é o sistema de ordem 4, sendo por isso o que aproxima melhor o sistema.

Para testar a veracidade da planta foi feito um teste semelhante mas com outros valores de entrada, tendo-se obtido para as duas entradas com um sistema 2ª ordem um erro de 262 e 238 respectivamente:

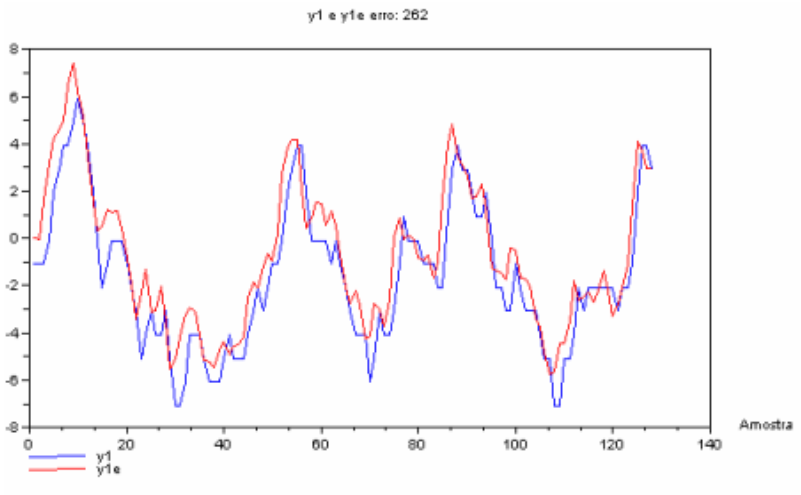

Figura 18 – Sistema de 2ª ordem para uma dada saída y1, sendo y1e os valores estimados

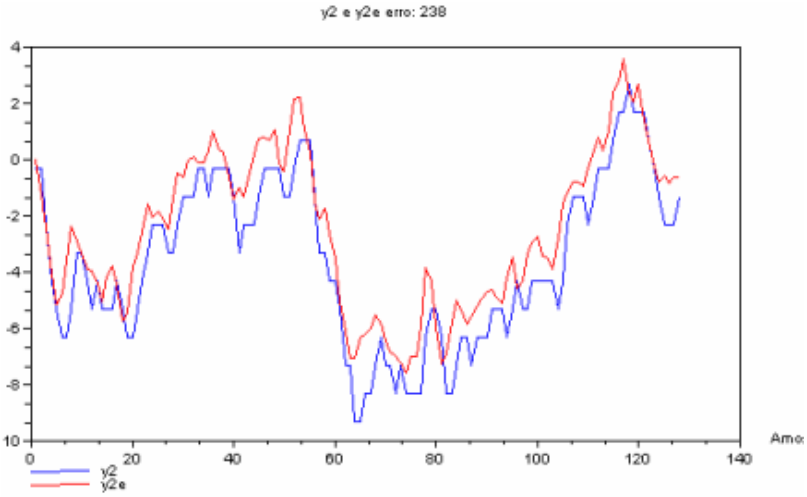

Figura 19 – Sistema de 2ª ordem para uma dada saída y2, sendo y2e os valores estimados

Como o erro obtido apresenta uma pequena variação sendo semelhante para as duas entradas conclui-se que a aproximação do sistema é suficientemente credível, sendo a planta obtida satisfatória.

Comentários a esta parte

Como foi possível verificar uma maior ordem do sistema não implica uma melhor aproximação do sistema à realidade, tornou-se por isso desnecessário calcular os parâmetros da planta para uma ordem superior a 4, pois através do gráfico do erro quadrático é possível ver que é um sistema de ordem 4 que melhor o aproxima.

O método dos mínimos quadrados revelou-se uma ferramenta poderosíssima para determinar o modelo de um sistema, uma vez que através do método experimental, não sendo por isso complexo, permite determinar com uma grande fiabilidade um modelo para um dado sistema.

Conclusão

A realização deste trabalho foi de todo o interesse académico pois permitiu uma consolidação de conhecimentos entre a teórica e a prática, pois sem uma abordagem prática por vezes torna-se complicado e abstracto perceber os conceitos teóricos que estão envolvidos e que muitas vezes passam ao lado. E como neste caso revelam-se de extrema importância e utilidade na área de automação.

Seria de todo o interesse a partir do método dos mínimos quadrados implementado obter a planta do forno, através por exemplo do método Ziegler Nichols obter os parâmetros do controlador PID e comparar os resultados obtidos com outros métodos de controlo PID para dai tirar uma percepção de qual o método de controlo PID a aplicar. Por motivos de escassez de tempo isso não foi possível.

Anexo 1 – Código desenvolvido em Delphi

```
unit hostlink;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Menus, StdCtrls, ExtCtrls, TeeProcs, TeEngine, Chart, Series,
  Buttons, Config, CPort, ComCtrls, QProgBar, ToolWin, ImgList, OleServer,
  ExcelXP;

const
  vstart = 0;       // DETECTAR O INICIO DA TRAMA //
  vstop  = 1;       // DETECTAR O FIM DA TRAMA    //

type
  Tfmain = class(TForm)
    gbMain: TGroupBox;
    gbSetvalue: TGroupBox;
    grafic: TGroupBox;
    setvalue_frame: TButton;
    chart: TChart;
    temperatura: TFastLineSeries;
    ed_setvalue: TEdit;
    Label7: TLabel;
    request_timer: TTimer;
    comport: TComPort;
    GroupBox3: TGroupBox;
    Image1: TImage;
    pb_temperatura: TQProgressBar;
    rgMode: TRadioGroup;
    mmonitor: TRadioButton;
    mplanta: TRadioButton;
    gbTime: TGroupBox;
    Label9: TLabel;
    request_time: TComboBox;
    lsetvalue: TLabel;
    estado: TStatusBar;
    ToolBar1: TToolBar;
    tbconnect: TToolButton;
    ToolButton2: TToolButton;
    tbdisconnect: TToolButton;
    tbconfig: TToolButton;
    ToolButton5: TToolButton;
    tbstart: TToolButton;
    ImageList1: TImageList;
    tbstop: TToolButton;
    tbfile: TToolButton;
    ToolButton9: TToolButton;
    gbPID: TGroupBox;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    pid_p: TEdit;
    pid_i: TEdit;
    pid_d: TEdit;
    tbase: TEdit;
    setvalue_pid: TButton;
    Label2: TLabel;
    Label8: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    temp: TEdit;
    GroupBox1: TGroupBox;
    Label12: TLabel;
    ed_pwmvalue: TEdit;
    lpwmvalue: TLabel;
    pwmvalue: TButton;
    procedure setvalue_frameClick(Sender: TObject);
    procedure request_timerTimer(Sender: TObject);
```

```
    procedure comportRxChar(Sender: TObject; Count: Integer);
    procedure ModeClick(Sender: TObject);
    procedure tbconnectClick(Sender: TObject);
    procedure tbdisconnectClick(Sender: TObject);
    procedure tbconfigClick(Sender: TObject);
    procedure tbstartClick(Sender: TObject);
    procedure tbstopClick(Sender: TObject);
    procedure tbfileClick(Sender: TObject);
    procedure pidKeyPress(Sender: TObject; var Key: Char);
    procedure pidChange(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure pwmvalueClick(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var fmain : Tfmain;
var pidchanged  : bool;
var tempo       : double;
var pidvalue    : string;
var option      : integer = 1;
var planta_file : string = '';
var ifile       : integer = 0;
var noiseArray  : Array[1..128] of integer;   // ARRAY COM OS DADOS DO RUIDO //
var noiseArrayr : Array[1..128] of integer;   // ARRAY COM OS RESULTADOS //
var senviar     : integer = 0;                // Nº DE DADOS DO RUIDO ENVIADOS//
var response_frame      : string = '';
var base_frame          : string = '@00FA';
var response_wait_time  : string = '0';
var icf_command         : string = '00';
var icf_response        : string = '40';
var header_frame        : string = '000000';
var code_read           : string = '0101';
var code_write          : string = '0102';
var dmword              : string = '82';
var actual_value        : integer = 1;        // ACTUAL VALUE - D00001 //
var pwm_value           : integer = 2;        // SAIDA DO PID - D00002 //
var set_value           : integer = 10;       // SET VALUE    - D00010 //
var p_value             : integer = 11;       // PROPORCIONAL - D00011 //
var d_value             : integer = 12;       // DERIVATIVO   - D00012 //
var i_value             : integer = 13;       // INTEGRAL     - D00013 //
var st_value            : integer = 14;       // T AMOSTRAGEM - D00014 //

var validation : integer = 0;

implementation

{$R *.dfm}

//*****************************************************************************//
//                                                                             //
//                DETERMINAR O FCS DE UMA TRAMA                                //
//                                                                             //
//*****************************************************************************//

function frame_check_sequence(FrameToCheck : string) : string;
var i, fcs : integer;
begin
  fcs := 0;
  for i := 1 to length(FrameToCheck) do
     fcs := fcs xor ord(FrameToCheck[i]);

  result := IntToHex(fcs, 2);
end;

//*****************************************************************************//
//                                                                             //
//                CRIAR UMA TRAMA PARA LER DADOS                               //
//                                                                             //
//*****************************************************************************//

function frametoread(address:integer; memareac:string):string;
begin
  result := base_frame;
```

```
  result := result + response_wait_time;
  result := result + icf_command;
  result := result + header_frame;
  result := result + code_read;
  result := result + memareac;
  result := result + IntToHex(address, 4) + '00';
  result := result + '0001';
  result := result + frame_check_sequence(result);
  result := result + '*' + chr(13);

  fmain.estado.panels[0].Text := result;
end;

//*****************************************************************************//
//                                                                             //
//                CRIAR UMA TRAMA PARA ESCREVER DADOS                          //
//                                                                             //
//*****************************************************************************//

function frametowrite(address:integer; memareac:string; data:integer):string;
begin
  result := base_frame;
  result := result + response_wait_time;
  result := result + icf_command;
  result := result + header_frame;
  result := result + code_write;
  result := result + memareac;
  result := result + IntToHex(address, 4) + '00';
  result := result + '0001';
  result := result + IntToHex(data, 4);
  result := result + frame_check_sequence(result);
  result := result + '*' + chr(13);

  fmain.estado.panels[0].Text := result;
end;

//*****************************************************************************//
//                                                                             //
//                GUARDAR OS RESULTADOS NUM FICHEIRO                           //
//                                                                             //
//*****************************************************************************//

procedure write2file();
var
  FileToWrite : Textfile;
  line        : string;
  i           : integer;
begin

  Assignfile(FileToWrite, planta_file + '.bak');
  ReWrite(FileToWrite);

  i := 1;
  while i <= 128 do
  begin
    line := IntToStr(noiseArray[i]) + ' ' + IntToStr(noiseArrayr[i]);
//    line := IntToStr(noiseArrayr[i]);
    WriteLn(FileToWrite, line);
    inc(i);
  end;
  CloseFile(FileToWrite);
  showmessage('file saved: ' + planta_file + '.bak')
end;

//*****************************************************************************//
//                                                                             //
//         PROCESSA OS DADOS RECEBIDOS [GRÁFICO e/ou noiseArrayr]              //
//                                                                             //
//*****************************************************************************//

procedure process_data(data : string);
var temperatura : double;
var temperaturai : integer;
begin
  if data <> '' then
  begin
    temperatura := StrToInt('$' + data) / 40;
```

```
    tempo := tempo + fmain.request_timer.Interval / 1000;
    fmain.temp.Text := format('%f', [temperatura]);
    temperaturai := StrToInt('$' + data) div 40;
    fmain.pb_temperatura.position := temperaturai;
    fmain.temperatura.AddXY(tempo, temperatura);
    if option = 2 then
    begin
      noiseArrayr[ifile] := temperaturai;
      fmain.Caption := 'Host Link Protocol [FINS] - Planta [' + IntToStr(ifile) + ']';
      if ifile = 128 then
      begin
        fmain.request_timer.Enabled := false;
        ifile := 0;
        write2file();
      end;
    end;
//    fmain.edit1.Text := IntToStr(ifile);
  end;
end;

//*****************************************************************************//
//                                                                             //
//                TRATA A TRAMA DE RESPOSTA RECEBIDA                           //
//                                                                             //
//*****************************************************************************//

procedure trata_response_frame();
var fcs, error, frame_text, check_fcs : string;
    frame_length  : integer;
begin
  fmain.estado.Panels[1].Text := response_frame;
  frame_length := length(response_frame);
  check_fcs := copy(response_frame, 1, frame_length - 3);
  fcs := response_frame[frame_length - 2] + response_frame[frame_length - 1];
  error := copy(response_frame, 20, 4);

  if fcs = frame_check_sequence(check_fcs) then
  begin
    if error = '0000' then
    begin
      if senviar = 1 then
        fmain.lsetvalue.caption := fmain.ed_setvalue.Text;
      if senviar = 2 then
        fmain.lpwmvalue.Caption := fmain.ed_pwmvalue.Text;
      if senviar <> 0 then
        senviar := 0;

      frame_text := copy(response_frame, 24, frame_length - 3 - 23);
      process_data(frame_text);
    end
    else
    begin
      fmain.request_timer.Enabled := false;
      messagedlg('Error in PLC! error - [' + error + ']', mterror, [mbOk], 0);
    end;
  end
  else
  begin
    fmain.request_timer.Enabled := false;
    messagedlg('Error in FCS!', mterror, [mbOk], 0);
  end;
end;

//*****************************************************************************//
//                                                                             //
//                ENVIA O SET VALUE PARA O AUTOMATO                            //
//                                                                             //
//*****************************************************************************//

procedure Tfmain.setvalue_frameClick(Sender: TObject);
var
  command_frame : string;
  data          : integer;
begin
  if comport.Connected then
  begin
    senviar := 1;
```

```
    data := strtoint(ed_setvalue.text)*40;
    command_frame := frametowrite(set_value, dmword, data);
    comport.WriteStr(command_frame);
    command_frame := frametowrite(7, dmword, 65535);
    comport.WriteStr(command_frame);
  end
  else
    showmessage('COMPORT is closed! Open it!!');
end;

//*****************************************************************************//
//                                                                             //
//                        INICIALIZA UM TIMER                                  //
//                                                                             //
//*****************************************************************************//

procedure Tfmain.request_timerTimer(Sender: TObject);
var command_frame : string;
var data : integer;
begin
  if option = 2 then
  begin
    ifile := ifile + 1;
    data := noiseArray[ifile]*40;
    command_frame := frametowrite(pwm_value, dmword, data);
    comport.WriteStr(command_frame);
    estado.Panels[0].Text := command_frame;
  end;

  command_frame := frametoread(actual_value, dmword);
  comport.WriteStr(command_frame);
  estado.Panels[0].Text := command_frame;
end;

//*****************************************************************************//
//                                                                             //
//              RECEPÇÂO DE INFORMAÇÂO NA PORTA SE'RIE                         //
//                                                                             //
//*****************************************************************************//

procedure Tfmain.comportRxChar(Sender: TObject; Count: Integer);
var vchar : string;
  i       : integer;
begin
  comport.ReadStr(vchar, Count);

  for i := 1 to Count do
  begin
    case validation of
      vstart:
        begin
          if vchar[i] = '@' then
          begin
            validation := vstop;
            response_frame := vchar[i];
          end;
        end;
      vstop:
        begin
          if vchar[i] = chr(13) then
          begin
            validation := vstart;
            trata_response_frame();
          end
          else
          begin
            if vchar[i] = '@' then
              response_frame := vchar[i]
            else
              response_frame := response_frame + vchar[i]
          end;
        end;
    end;
  end;
end;

//*****************************************************************************//
```

```
//                                                                      //
//                    ABRIR A PORTA SE'RIE                              //
//                                                                      //
//**********************************************************************//

procedure Tfmain.tbconnectClick(Sender: TObject);
begin
  if comport.Connected = FALSE then
  begin
    comport.Open;
    showmessage('The ' + comport.Port + ' is Open!');
  end
  else
    showmessage('The ' + comport.Port + ' is already in use! Try again!');
end;

//**********************************************************************//
//                                                                      //
//                    FECHAR A PORTA SE'RIE                             //
//                                                                      //
//**********************************************************************//

procedure Tfmain.tbdisconnectClick(Sender: TObject);
begin
  comport.Close;
end;

//**********************************************************************//
//                                                                      //
//                    CONFIGURAR A PORTA SE'RIE                         //
//                                                                      //
//**********************************************************************//

procedure Tfmain.tbconfigClick(Sender: TObject);
begin
  comport.ShowSetupDialog;
end;

//**********************************************************************//
//                                                                      //
//                    INICIAR A AQUISICAO DE DADOS                      //
//                                                                      //
//**********************************************************************//

procedure Tfmain.tbstartClick(Sender: TObject);
var
  command_frame : string;
begin
  if comport.Connected = TRUE then
  begin
    tbstop.Enabled := TRUE;
    tempo := 0;
    temperatura.Clear;
    if option = 2 then
    begin
      ifile := 0;
      command_frame := frametowrite(7, dmword, 0);
      comport.WriteStr(command_frame);
      request_timer.Interval := 10000;
    end
    else
    begin
      command_frame := frametowrite(7, dmword, 65535);
      comport.WriteStr(command_frame);
      request_timer.Interval := (1 + request_time.ItemIndex) * 5000;
    end;
    request_timer.Enabled := TRUE;
  end
  else
    showmessage('The ' + comport.Port + ' isn''t open!');
end;

//**********************************************************************//
//                                                                      //
//                    PARAR A AQUISICAO DE DADOS                        //
//                                                                      //
//**********************************************************************//
```

```
procedure Tfmain.tbstopClick(Sender: TObject);
begin
  request_timer.Enabled := FALSE;
end;

//*****************************************************************************//
//                                                                             //
//            INFROMACAO PARA UTILIZADOR SOBRE O MODO ACTUAL                   //
//                                                                             //
//*****************************************************************************//

procedure Tfmain.ModeClick(Sender: TObject);
begin
  if sender = mmonitor then
  begin
    option := 1;
    request_timer.Interval := (1 + request_time.ItemIndex) * 5000;
    fmain.Caption := 'Host Link Protocol [FINS] - ' + mmonitor.Caption;
//    fmain.edit1.Text := '0';
  end;
  if sender = mplanta then
  begin
    option := 2;
    fmain.Caption := 'Host Link Protocol [FINS] - ' + mplanta.Caption;
    fmain.tbfileClick(Sender);
//    fmain.edit1.Text := '0';
  end;
  request_timer.Enabled := FALSE;
end;

//*****************************************************************************//
//                                                                             //
//                  LER DADOS DO RUIDO DO FICHEIRO                             //
//                                                                             //
//*****************************************************************************//

procedure Tfmain.tbfileClick(Sender: TObject);
var
  FileToRead  : Textfile;
  line        : string;
  OpenDialog  : TOpenDialog;
  i           : integer;
begin

  ifile := 0;
  openDialog := TOpenDialog.Create(self);
  openDialog.InitialDir   := GetCurrentDir + '\files\';
  openDialog.Options      := [ofFileMustExist];
  openDialog.Filter       := 'Host Link Files|*.hlf';
  openDialog.FilterIndex  := 1;

  if openDialog.Execute then
  begin
    Assignfile(FileToRead, openDialog.FileName);
    reset(FileToRead);
    planta_file := openDialog.FileName;

    i := 1;
    while not Eof(FileToRead) do
    begin
      ReadLn(FileToRead, line);
      noiseArray[i] := StrToInt(line);
      inc(i);
    end;
    CloseFile(FileToRead);
    mplanta.Checked := TRUE;
    option := 2;
  end;

  openDialog.Free;
end;

procedure Tfmain.pidKeyPress(Sender: TObject; var Key: Char);
begin
  if not(Key in ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9',
     chr(vk_Escape), chr(vk_Return), chr(vk_Back), chr(vk_Delete)]) then
```

```
    begin
      if sender = pid_p then pidvalue := pid_p.Text;
      if sender = pid_i then pidvalue := pid_i.Text;
      if sender = pid_d then pidvalue := pid_d.Text;
      if sender = tbase then pidvalue := tbase.Text;
      pidchanged := true;
    end;
end;

procedure Tfmain.pidChange(Sender: TObject);
begin
  if pidchanged then
  begin
    if sender = pid_p then pid_p.Text := pidvalue;
    if sender = pid_i then pid_i.Text := pidvalue;
    if sender = pid_d then pid_d.Text := pidvalue;
    if sender = tbase then tbase.Text := pidvalue;
    pidchanged := false;
  end;
end;

procedure Tfmain.FormCreate(Sender: TObject);
begin
  if mmonitor.Checked then
    fmain.Caption := 'Host Link Protocol [FINS] - ' + mmonitor.Caption;
  if mplanta.Checked then
    fmain.Caption := 'Host Link Protocol [FINS] - ' + mplanta.Caption;
end;

//*****************************************************************************//
//                                                                             //
//                ENVIAR O VALOR DE PWM PRO AUTOMATO                           //
//                                                                             //
//*****************************************************************************//

procedure Tfmain.pwmvalueClick(Sender: TObject);
var
  command_frame : string;
  data          : integer;
begin
  if comport.Connected then
  begin
    senviar := 2;
    data := strtoint(ed_pwmvalue.text)*40;
    command_frame := frametowrite(pwm_value, dmword, data);
    comport.WriteStr(command_frame);
    command_frame := frametowrite(7, dmword, 0);
    comport.WriteStr(command_frame);
  end
  else
    showmessage('COMPORT is closed! Open it!!');

end;

end.
```

Anexo 2 – Código desenvolvido em Scilab para a implementação do método dos mínimos quadrados para uma entrada U

```
function [] = tsca_noise(nivel, largura)

  current_path = 'C:/Documents and Settings/Ana Luísa/Ambiente de trabalho/parte2/delphi/files'
  chdir(current_path)
  path = tk_savefile('*.hlf')

  noise = nivel + largura*rand([1:128]','normal')

  for i = 1:128,
     if noise(i) < 0 then
       noise(i) = 0;
     else if noise(i) > 100 then
       noise(i) = 100;
       end
     end,
  end

  [fd, err] = mopen(path, 'wb');

  if err == 0 then
    mfprintf(fd, '%d\n', noise),
    mclose(fd),
    mprintf('\n\tFicheiro gerado e gravado com sucesso!\n\t\n\t\t%s', path);
  else
    mprintf('\n\tErro ao abrir o ficheiro!\n');
  end

endfunction

function [] = tsca_values(filex, referenciau, referenciay, ordem)

  xdel(0),
  current_path = 'C:/Documents and Settings/Ana Luísa/Ambiente de trabalho/parte2/delphi/files'
  chdir(current_path)

  [fdx,err] = mopen(filex, 'rb')

  u = [];
  y = [];

  for i = 1:128,
    [n,a,b] = mfscanf(fdx, '%d %d'),
    u = [u a],
    y = [y b],
  end

  y = y - referenciay;
  u = u - referenciau;

  // i - ordem
//  ordem = 4;

  for i = 1:ordem,
    clear x,
    // GERAR O VALOR INICIAL PARA A MATRIZ [X]
    for j = 1:i,
      for m = 1:2*i,
        x(j,m) = 0,
      end,
    end,
    // PREENCHER OS RESTANTES VALORES DA MATRIZ [X]
    for k = (i+1):128,
      for m = 1:i,
        x(k,m) = -y(k-m),
      end,
      for m = (i+1):2*i,
        x(k,m) = u(k-(m-i)),
      end,
```

```
  end,
  // DETERMINAR OS VALORES DE TETA E DO ERRO
  xt = x',
  yt = y',
  teta = inv(xt*x)*xt*yt,
  v(i) = (yt-x*teta)'*(yt-x*teta),
  ye = xt'*teta,

  mprintf('\n\t[ ORDEM nº %d ]\n', i),
  // PARTE SUPERIOR
  mprintf('\n\t\t  Y(Z)    '),
  for j = i+1:2*i,
    if j == 2*i then
      mprintf('(%2.4f)', teta(j));
    else
      mprintf('(%2.4f)Z^%d + ', teta(j), 2*i-j);
    end,
  end,
  // SEPARADOR
  mprintf('\n\t\t ---- = '),
  for j = 1:i*16,
    mprintf('-'),
  end,
  // PARTE INFERIOR
  mprintf('\n\t\t  U(Z)    z^%d + ', i),
  for j = 1:i,
    if j == i then
      mprintf('(%2.4f)\n', teta(j));
    else
      mprintf('(%2.4f)Z^%d + ', teta(j), i-j);
    end,
    mprintf('\n\tErro =%d', v( i));
  end,



  //*************************************************************************************************//
  //                                IMPRIMIR GRÁFICOS                                                //
  //*************************************************************************************************//

  if min(yt) < min(ye) then
    ymin = min(yt);
  else
    ymin = min(ye);
  end,
  if max(yt) < max(ye) then
    ymax = max(ye);
  else
    ymax = max(yt);
  end,


  //*************************************************************************************************//
  //                                ATRASAR O Ye                                                     //
  //*************************************************************************************************//

  for j = 1:127,
     ye(j) = ye(j + 1),
  end

  //*************************************************************************************************//
  //                    DETERMINAR A RESPOSTA A UM U DIFERENTE                                       //
  //*************************************************************************************************//

//   un = referencia + 20*rand([1:128]','normal')
//
//   for j = 1:128,
//      if un(j) < 0 then
//        un(j) = 0;
//      else if un(j) > 100 then
//        un(j) = 100;
//        end
//      end,
//   end
//
```

```
//    for k = 1:128,
//      yn(k) = 0,
//      if k <= i then
//        yn(k) = 0;
//      else
//        for l = 1:i,
//          yn(k) = yn(k) - teta(l)*yn(k-l) + teta(i+l)*un(k-l);
//        end
//      end
//    end

    for k = 1:128,
      yn(k) = 0,
      if k <= i then
        yn(k) = 0;
      else
        for l = 1:i,
          yn(k) = yn(k) - teta(l)*yn(k-l) + teta(i+l)*u(k-l);
        end
      end
    end

    ve(i) =  (yt-yn)'*(yt-yn),

    subplot(fix(ordem/2) + 1 + (ordem - fix(ordem/2)*2), 2, i),
    plot2d(1:128, [yt ye], [2 5], rect = [0, ymin - 1, 128, ymax + 1]),
    legends(['y', 'ye'], [2 5],  with_box = %f, opt = [128, ymax + 1]),
//    newu = u'/8
//    plot2d(1:128, [yt ye newu], [2 5 1], rect = [0, ymin - 1, 128, max(newu) + 1]),
//    legends(['y', 'ye', 'u'], [2 5 1],  with_box = %f, opt = [130, max(newu) + 1]),
    xtit = msprintf('Sistema de ordem n. %d', i),
    xtitle(xtit),

    //*****************************************************************************************//
    //                    IMPRIMIR A RESPOSTA PARA Ye DIFERENTE                                //
    //*****************************************************************************************//

    subplot(fix(ordem/2) + 1 + (ordem - fix(ordem/2)*2), 2, ordem + 2 + (ordem - fix(ordem/2)*2)),
    plot2d(1:128, [yn], i),
    xtitle('Ye para cada ordem'),

    ynmax(i) = max(yn);

  end

  for j = 1:i,
    legenda(j) = msprintf('%d', j),
    cor(j) = j,
  end,

  legends(legenda, cor, with_box = %f, opt = [140, max(ynmax)]),

  //*****************************************************************************************//
  //                            IMPRIMIR GRÁFICO DE ERROS                                    //
  //*****************************************************************************************//

  subplot(fix(ordem/2) + 1 + (ordem - fix(ordem/2)*2), 2, ordem + 1 + (ordem - fix(ordem/2)*2)),
//  plot2d(1:ordem, [v ve], [2 5]);
  plot2d(1:ordem, [v], [2]);
  xtitle('Erro Quadrático');

endfunction
```

Anexo 3 – Código desenvolvido em Scilab para a implementação do método dos mínimos quadrados para duas entradas distintas U1 e U2

```
function [] = tsca_values_2(file1, file2, referenciau, referenciay1, referenciay2, ordem)

 xdel(0),
 current_path = 'D:\feup\TSCA\parte2\delphi\files'
 chdir(current_path)

 //*********************************************************************//
 //*                  ABRIR AMOSTRA 1                                  //
 //*********************************************************************//

 [fd1,err] = mopen(file1, 'rb')

 u1 = [];
 y1 = [];

 for i = 1:128,
   [n,a,b] = mfscanf(fd1, '%d %d'),
   u1 = [u1 a],
   y1 = [y1 b],
 end

 y1 = y1 - referenciay1;
 u1 = u1 - referenciau;

 //*********************************************************************//
 //*                  ABRIR AMOSTRA 2                                  //
 //*********************************************************************//

 [fd2,err] = mopen(file2, 'rb')

 u2 = [];
 y2 = [];

 for i = 1:128,
   [n,a,b] = mfscanf(fd2, '%d %d'),
   u2 = [u2 a],
   y2 = [y2 b],
 end

 y2 = y2 - referenciay2;
 u2 = u2 - referenciau;

 //*********************************************************************//
 //*             DETERMINAR SISTEMA DE ORDEM                           //
 //*********************************************************************//

 clear x,
   // GERAR O VALOR INICIAL PARA A MATRIZ [X]
   for j = 1:ordem,
     for m = 1:2*ordem,
       x(j,m) = 0,
     end,
   end,
   // PREENCHER OS RESTANTES VALORES DA MATRIZ [X]
   for k = (ordem+1):128,
     for m = 1:ordem,
       x1(k,m) = -y1(k-m),
     end,
     for m = (ordem+1):2*ordem,
       x1(k,m) = u1(k-(m-ordem)),
     end,
   end,
   // DETERMINAR OS VALORES DE TETA E DO ERRO
   x1t = x1',
   y1t = y1',
   teta1 = inv(x1t*x1)*(x1t*y1t),
   v1 = (y1t-x1*teta1)'*(y1t-x1*teta1),
   y1e = x1t'*teta1,
```

```
    mprintf('\n\t[ ORDEM nº %d ]\n', ordem),
  // PARTE SUPERIOR
  mprintf('\n\t\t  Y(Z)    '),
  for j = ordem+1:2*ordem,
    if j == 2*ordem then
      mprintf('(%2.4f)', teta1(j));
    else
      mprintf('(%2.4f)Z^%d + ', teta1(j), 2*ordem-j);
    end,
  end,
  // SEPARADOR
  mprintf('\n\t\t ---- = '),
  for j = 1:ordem*16,
    mprintf('-'),
  end,
  // PARTE INFERIOR
  mprintf('\n\t\t  U(Z)    z^%d + ', ordem),
  for j = 1:ordem,
    if j == ordem then
      mprintf('(%2.4f)\n', teta1(j));
    else
      mprintf('(%2.4f)Z^%d + ', teta1(j), ordem-j);
    end,
  end,
//*********************************************************************//
//*                  APRESENTAR GRÁFICOS                              //
//*********************************************************************//

u1t = u1',
u2t = u2',

subplot(2, 2, 3),

set("figure_style","new");
grap = get("current_axes");

plot2d(1:128, [u1t u2t], [2 5]),
xit = msprintf('ENTRADAS'),
xtitle(xit),

grap.x_label.text = 'Amostra';

//*************************************************************************************//
//                                ESTIMAR O y1e                                        //
//*************************************************************************************//
for k = 1:128,
  y1e(k) = 0,
  if k <= ordem then
    y1e(k) = 0;
  else
    for l = 1:ordem,
      y1e(k) = y1e(k) - teta1(l)*y1e(k-l) + teta1(ordem+l)*u1(k-l);
    end
  end
end

//*************************************************************************************//
//                                ATRASAR O y1e                                        //
//*************************************************************************************//

for j = 1:127,
  y1e(j) = y1e(j + 1),
end

y1t = y1',

//*************************************************************************************//
//                                ESTIMAR O y2e                                        //
//*************************************************************************************//

for k = 1:128,
  y2e(k) = 0,
  if k <= ordem then
    y2e(k) = 0;
  else
    for l = 1:ordem,
```

```
    y2e(k) = y2e(k) - teta1(l)*y2e(k-l) + teta1(ordem+l)*u2(k-l);
   end
  end
end


//*********************************************************************************************//
//                                ATRASAR O y2e                                                //
//*********************************************************************************************//
for j = 1:127,
  y2e(j) = y2e(j + 1),
end

y2t = y2',

//*********************************************************************************************//
//                                  ERROS                                                      //
//*********************************************************************************************//
erro1 = (y1t-y1e)'*(y1t-y1e),
erro2 = (y2t-y2e)'*(y2t-y2e),

subplot(2, 2, 1),

set("figure_style","new");
grap = get("current_axes");

plot2d(1:128, [y1t y1e], [2 5], leg='y1@y1e'),
xtit = msprintf('y1 e y1e erro: %d', erro1),
xtitle(xtit),

grap.x_label.text = 'Amostra';

subplot(2, 2, 2),

set("figure_style","new");
grap = get("current_axes");

plot2d(1:128, [y2t y2e], [2 5], leg='y2@y2e'),
xtit = msprintf('y2 e y2e erro: %d', erro2),
xtitle(xtit),

grap.x_label.text = 'Amostra';

//*********************************************************************************************//
//                            DETERMINAR SAIDA A UM DEGRAU                                     //
//*********************************************************************************************//

for k = 1:128,
  yd(k) = 0,
  ud(k) = 1,
  if k <= ordem then
    yd(k) = 0;
  else
    for m = 1:ordem,
      yd(k) = yd(k) - teta1(m)*yd(k-m) + teta1(ordem+m)*ud(k-m);
    end
  end
end

//*********************************************************************************************//
//                                ATRASAR O y2e                                                //
//*********************************************************************************************/
for j = 1:127,
  yd(j) = yd(j + 1),
end

ydt = yd',
udt = ud',
subplot(2, 2, 4),

set("figure_style","new");
grap = get("current_axes");

plot2d(1:128, [ydt], [2], leg='y'),
xtit = msprintf('Resposta ao Degrau'),
xtitle(xtit)
```

```
grap.x_label.text = 't';
//*************************************************************************************************//
//                          DETERMINAR VALORES DE PID                                              //
//*************************************************************************************************//

endfunction
```