Universidade do Porto
FEUP Faculdade de Engenharia

REDES DE COMPUTADORES
2004/2005

"Protocolo de Ligação Lógica da porta RS232"

Realizado por:

Ana Luísa Martins ee04255@fe.up.pt
Gustavo Pimentel ee01074@fe.up.pt

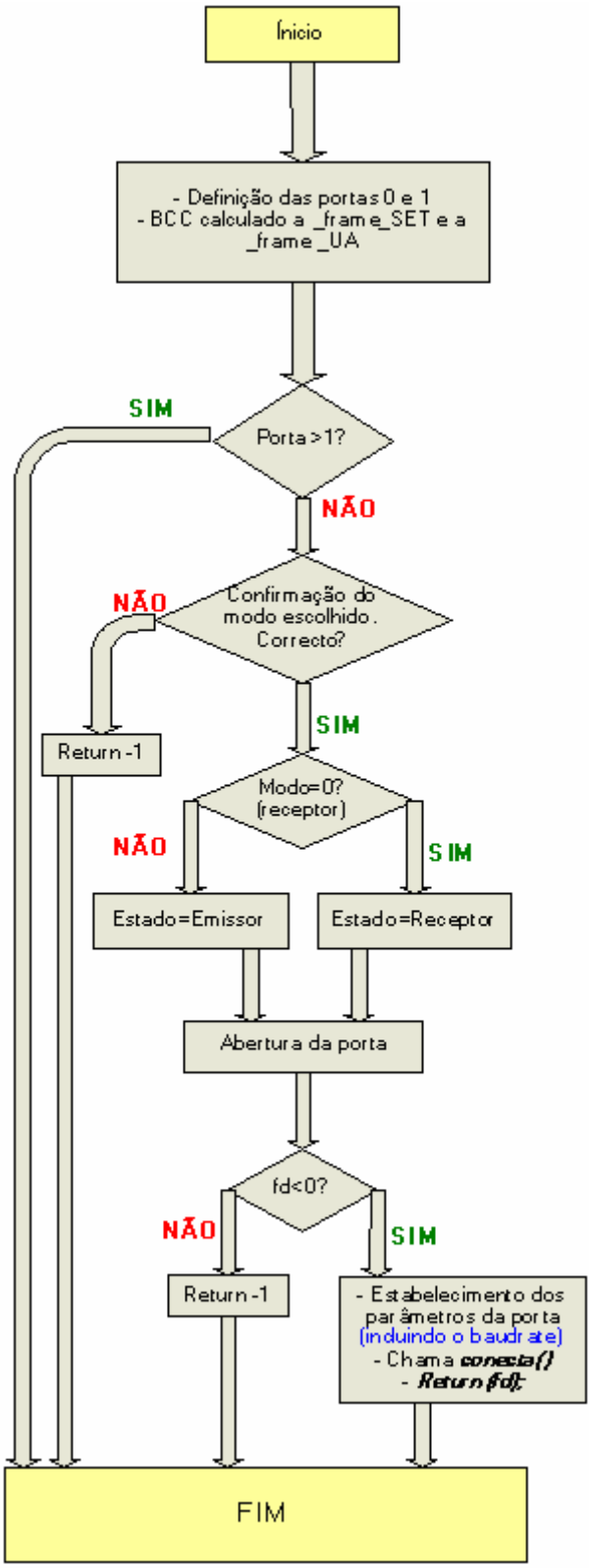
Introdução/Objectivos

Este trabalho tem como objectivos a implementação do protocolo de ligação lógica da porta RS-232 e o teste deste protocolo com uma aplicação de transferência de ficheiros.

Os diagramas que se seguem formam sumariamente o raciocínio por nós tomado na execução das várias funções que compõem o protocolo.

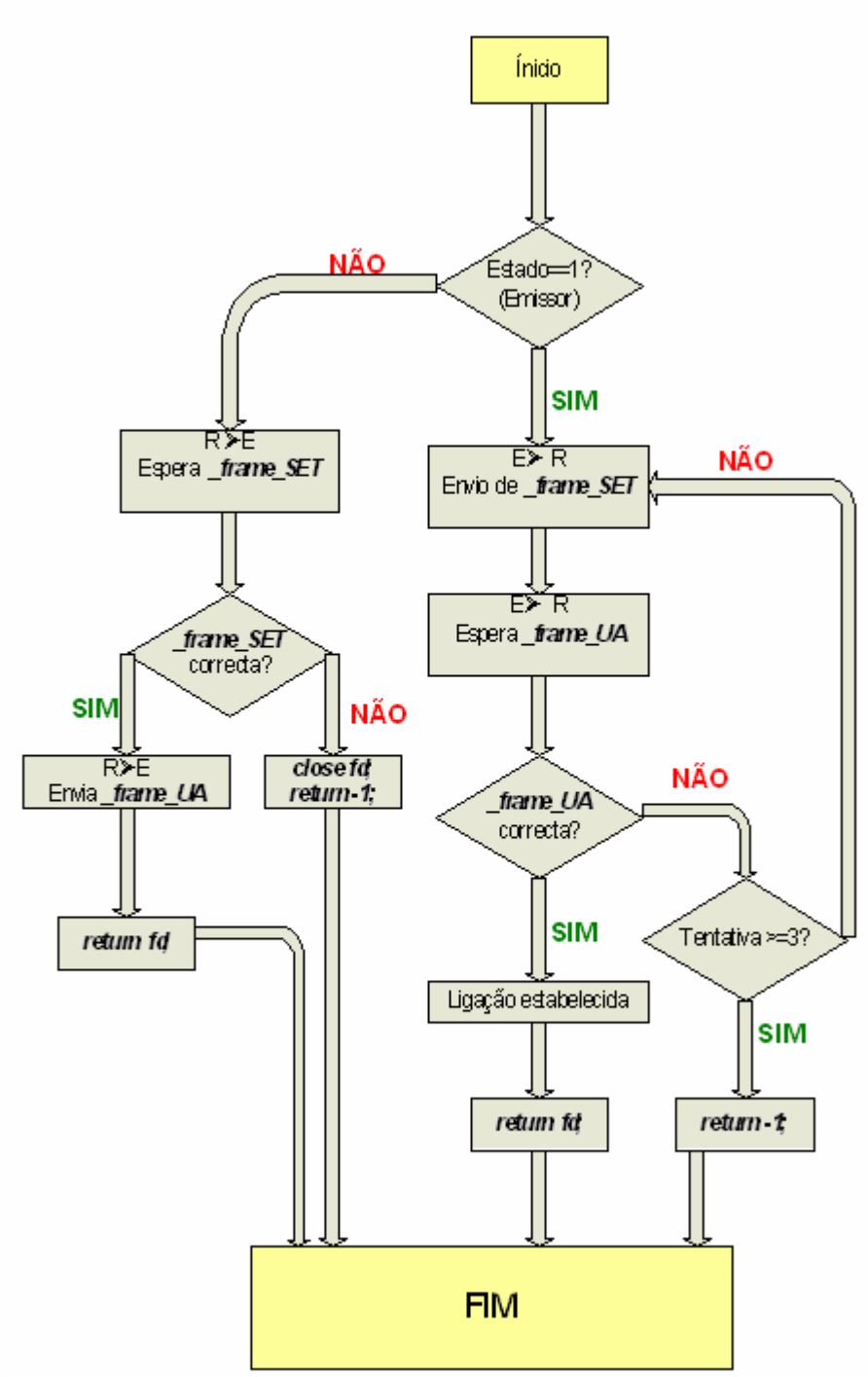
(Nota: as áreas “sombreadas a azul” encontram-se fora do protocolo, mais tarde é explicado o porquê)

Função: ***int llopen (int porta, char modo[],int baud)***

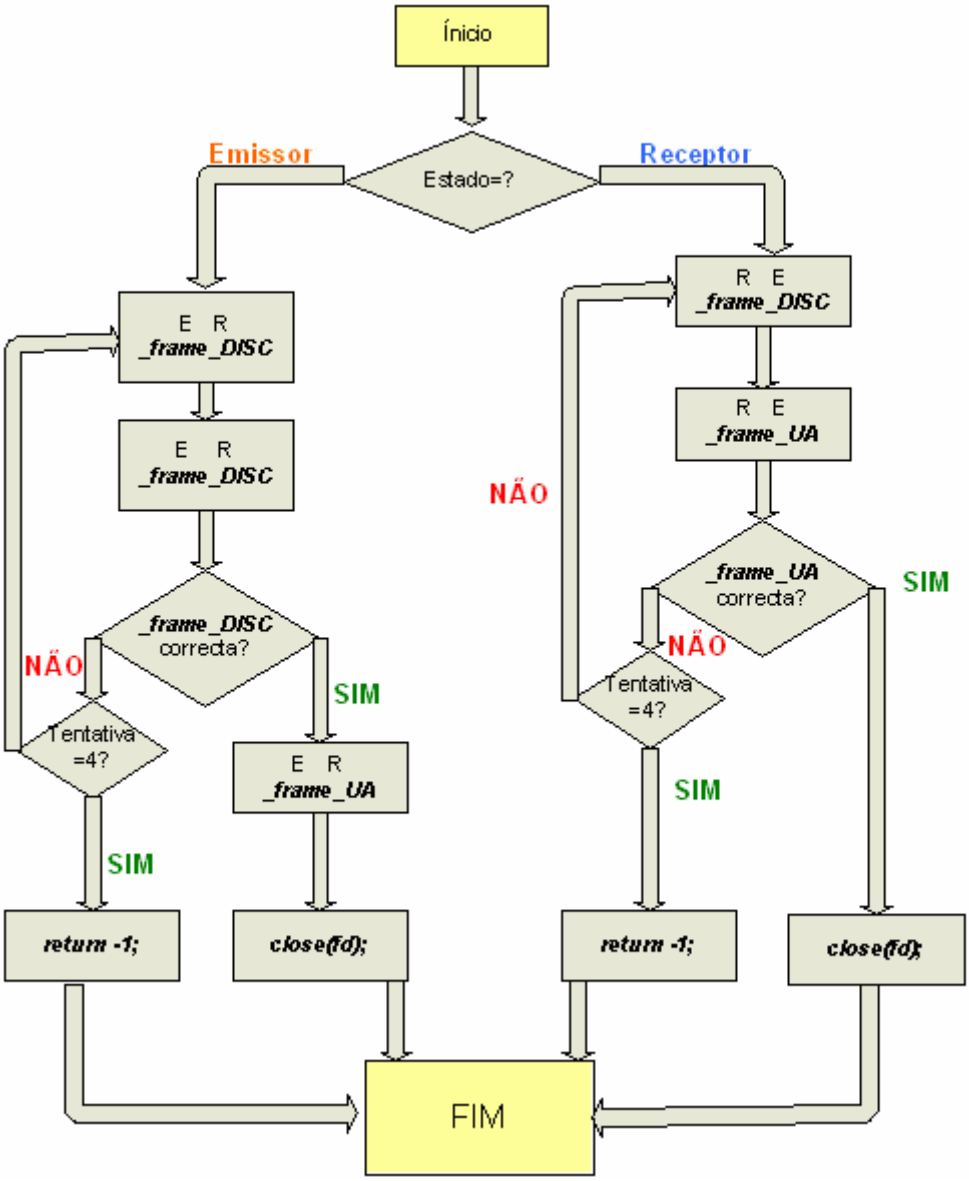


Função: ***int conecta(int fd)***

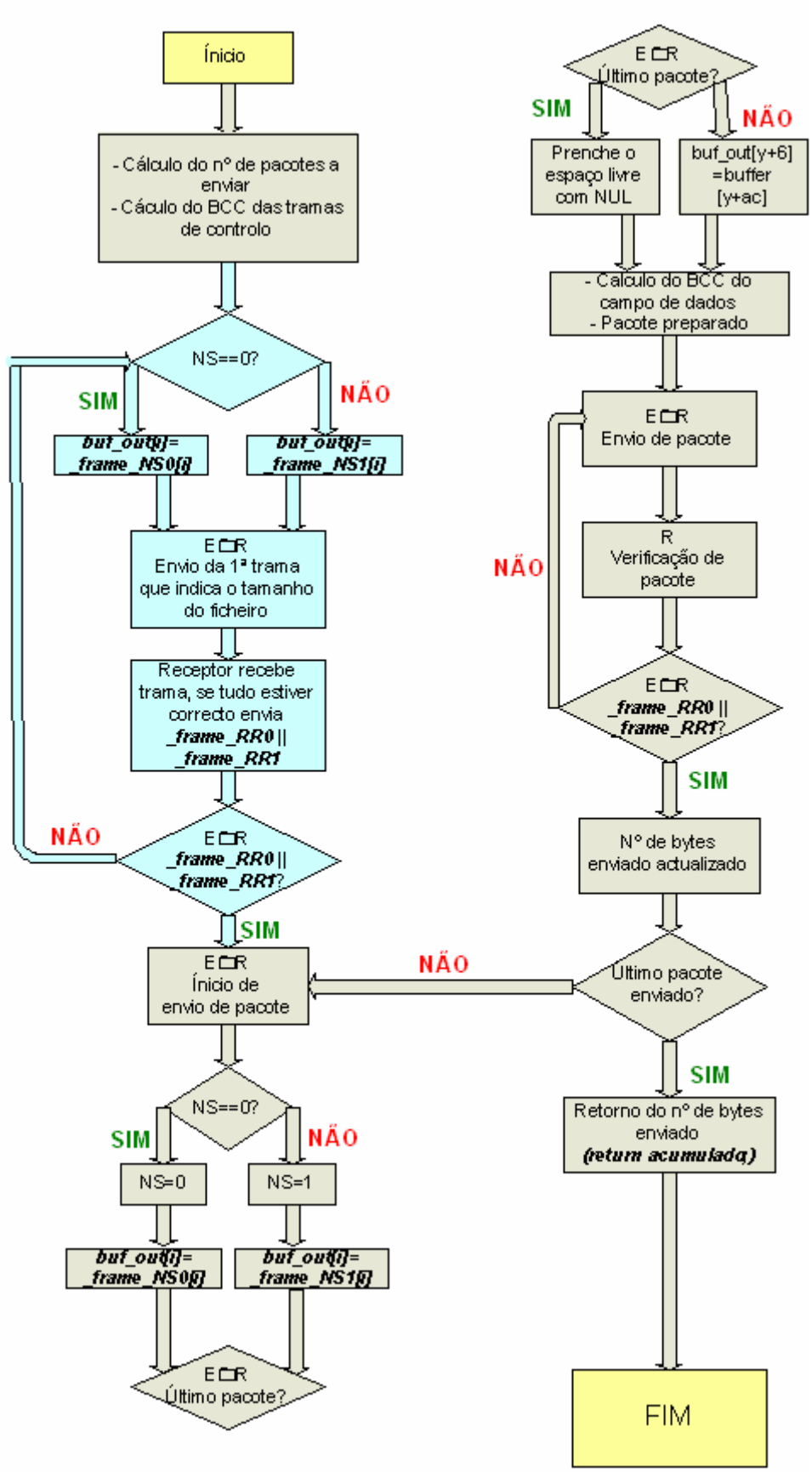
(Nota: função embebida na llopen)



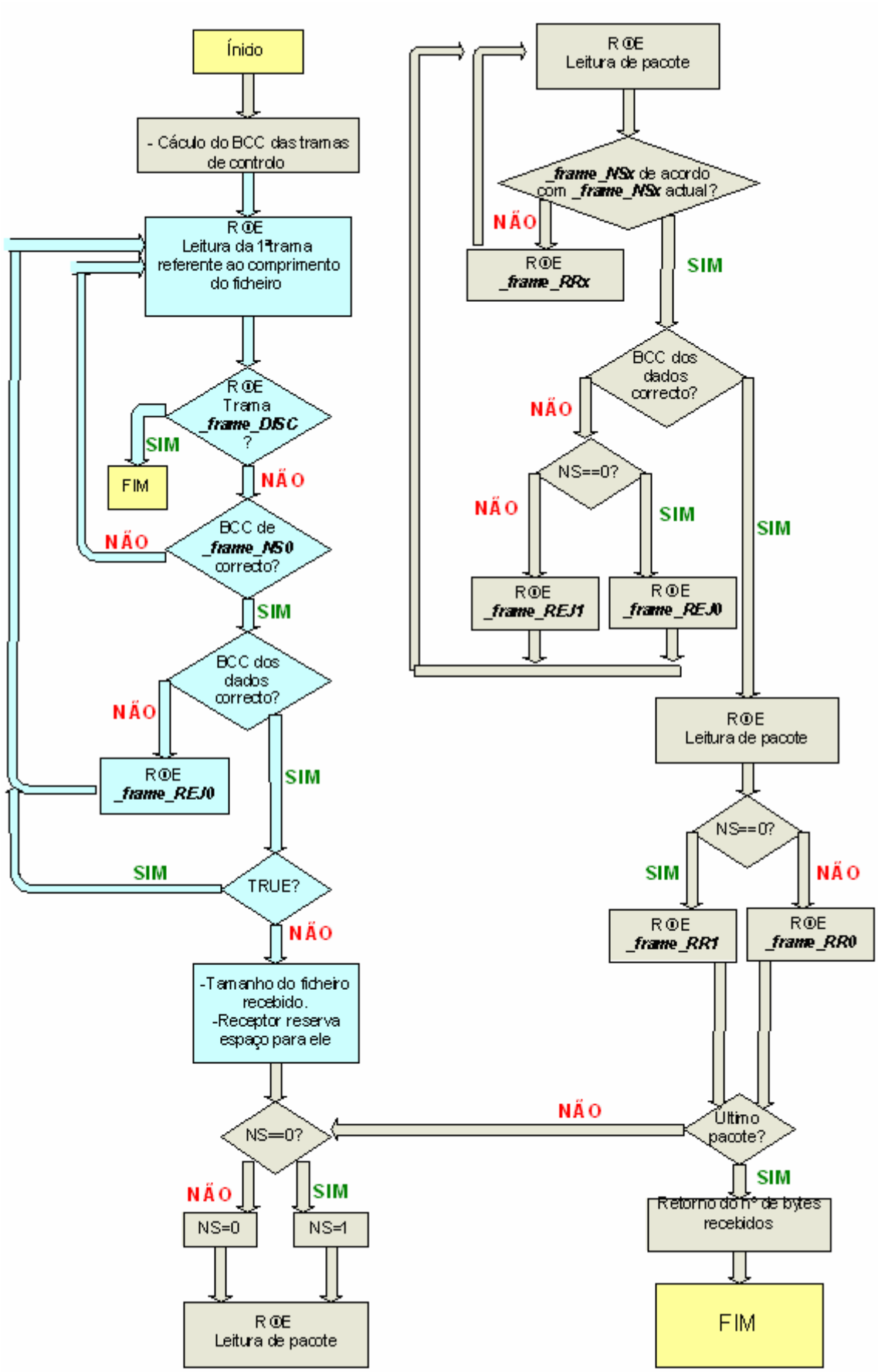
Função: ***int llclose (int fd)***



Função: ***int llwrite(int fd, unsigned char *buffer, int length)***



Função: ***int llread(int fd, unsigned char *buffer)***



Comentários

Este protocolo devia ser mais robusto devendo fazer parte dele:

- O envio do tamanho do ficheiro, de modo o receptor poder atempadamente reservar espaço para este e em caso de não haver espaço emitir um aviso
- O nome do ficheiro a enviar, pois pode haver confusão da extensão dos ficheiros por parte do sistema operativo
- A velocidade desejada da porta poder ser alterada facilmente pelo utilizador na aplicação sem entrar em pormenores de programação, por exemplo ser mais um parâmetro na função llopen.

Foi a pensar nestes pormenores que decidimos inclui-los no "nosso protocolo" (assim deixa de ser protocolo pois perde a sua universalidade). Estes pormenores estão "sombreados a azul" nos diagramas acima realizados.

Função llopen:

A nossa função recebe mais um parâmetro que os estipulados no protocolo, o ***baudrate*** da porta. À parte isto,o procedimento é o indicado no protocolo.

Função llclose:

De acordo com o protocolo.

Funções llread e llwrite:

Na função llwrite no primeiro envio é enviado o tamanho do ficheiro ao receptor, sendo a sua implementação idêntica ao envio de uma trama normal de dados (pacote). Por seu turno a função llread na sua primeira aquisição recebe este tamanho e reserva espaço para o ficheiro.

Dificuldades na implementação do protocolo

As dificuldades maiores surgiram nos acertos dos timeouts, nos vários ciclos de espera mediante as perguntas/respostas do emissor/receptor e na introdução de "multi-janelas" na interface feita na aplicação

Apreciação global

Este trabalho em nosso entender é um trabalho muito rico ao nível académico, pois a porta série embora seja muito utilizada na nossa área (APEL, entre outras), encontra-se actualmente em desuso. Na maior parte dos PC's actuais, nomeadamente portáteis já nem vem incluída, tendo sido radicalmente substituída pela porta USB. Dois dos factores que deverão ter contribuído para isto são a fraca velocidade de transmissão da porta série no envio/recepção de ficheiros e o pouco alcance desta, sendo o seu uso justificado apenas para envio/recepção de ficheiros de pequena dimensão (na ordem dos Bytes) e curtas distâncias.

Assim a realização deste trabalho permitiu-nos o contacto com o esquema para enviar dados sobre uma rede (uso de tramas, detecção de erros) o que nos permitirá uma maior facilidade em aprender outros protocolos mais complexos e/ou desenvolver um novo protocolo (por exemplo para comunicação RS485).

ANEXOS (código desenvolvido)

“protocolo.h”

```
#include <stdio.h>
#include <termios.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <strings.h>
#include <string.h>
#include <stdlib.h>

#define SYN 22
#define SOH 01
#define SET 03
#define DISC 11
#define UA 07
#define NUL 00
#define NS_0 00
#define NS_1 16
#define RR_0 01
#define RR_1 17
#define REJ_0 05
#define REJ_1 21
#define L 248
#define timeout 30
#define timeout2 60

struct termios _newtio, _oldtio;
int estado, NS=0;
unsigned char _frame_SET[6]={SYN,SYN,SOH,SET,NUL,NUL};
unsigned char _frame_UA[6]={SYN,SYN,SOH,UA,NUL,NUL};
unsigned char _frame_DISC[6]={SYN,SYN,SOH,DISC,NUL,NUL};
unsigned char _frame_RR0[6]={SYN,SYN,SOH,RR_0,NUL,NUL};
unsigned char _frame_RR1[6]={SYN,SYN,SOH,RR_1,NUL,NUL};
unsigned char _frame_REJ0[6]={SYN,SYN,SOH,REJ_0,NUL,NUL};
unsigned char _frame_REJ1[6]={SYN,SYN,SOH,REJ_1,NUL,NUL};
unsigned char _frame_NS0[6]={SYN,SYN,SOH,NS_0,L,NUL};
unsigned char _frame_NS1[6]={SYN,SYN,SOH,NS_1,L,NUL};
unsigned const char
_BAUDRATE[8]={B110,B300,B1200,B2400,B4800,B9600,B19200,B38400};

void limpa_buffer(unsigned char *buffer, int length)
{
int i;
for(i=0;i<length;i++) buffer[i]=(unsigned char) NUL;
}

void set_paridade_controlo(unsigned char *trama)
{
      trama[5]=(unsigned char)((((int)trama[3])+((int)trama[4]))%2);
}
int cmp_paridade_controlo(const unsigned char *trama, const unsigned
char *comp)
{
// >0 Verdadeiro  ou  <0 Falso
if ((int)trama[5]==(int)comp[5]) return 1;
else return -1;
```

```
};
void set_paridade_informacao(unsigned char *trama)
{
int i, soma=0;
for (i=6; i<(L+6);i++) soma=soma+((int)trama[i]);
trama[(L+6)]=(unsigned char)(soma%2);
};
int cmp_paridade_informacao(const unsigned char *trama)
{
// >0 Verdadeiro  ou  <0 Falso
int i, soma=0;
for (i=6; i<(L+6);i++) soma=soma+((int)trama[i]);
if (trama[L+6]==(soma%2)) return 1;
else return -1;
};

 unsigned long int expoente(int n)
 {
int i, resul=1;
if (n==0) return 1;
for(i=1; i<=n;i++)resul=10*resul;
return resul;
};

int conecta(int fd)
{
int escrita,leitura,tentativa=1;
unsigned char buf_in[6];

if (estado==1){
_newtio.c_cc[VTIME]=timeout;
_newtio.c_cc[VMIN]=0;
if (tcsetattr(fd,TCSANOW,&_newtio)==-1)
{
      close (fd);
      return -1;
}
do{
      tcflush(fd, TCIFLUSH);
      escrita = write(fd, &_frame_SET, 6);
      tcflush(fd, TCIFLUSH);
      limpa_buffer(buf_in,6);
      leitura = read(fd, &buf_in,6);
      if (memcmp(_frame_UA, buf_in, 6)==0)
      {
            _newtio.c_cc[VTIME]=0;
            if (tcsetattr(fd,TCSANOW,&_newtio)==-1)
            {
                  close (fd);
                  return -1;
            }
                  return fd;
      }
      else
      {
      if (tentativa>=3)
      {
            tcsetattr(fd,TCSANOW,&_oldtio);
            close(fd);
            return -1;
      }
```

```
      }
      tentativa++;
      tcflush(fd, TCIFLUSH);
}while(tentativa<4);
return -1;
}
else{
_newtio.c_cc[VTIME]=0;
_newtio.c_cc[VMIN]=6;
if (tcsetattr(fd,TCSANOW,&_newtio)==-1)
{
      close (fd);
      return -1;
}
limpa_buffer(buf_in,6);
leitura = read(fd,&buf_in,6);
if (memcmp(_frame_SET, buf_in, 6)==0)
{
      escrita = write(fd, &_frame_UA, 6);
      return fd;
}
else
{
      tcsetattr(fd,TCSANOW,&_oldtio);
      close(fd);
      return -1;
}
}
return -1;
};



int llopen(int porta, char modo[], int baud)
{
//Baud                              //Porta
//00 -> 110    bits/sec             //00 -> Para a porta serie 0
//01 -> 300    bits/sec             //01 -> Para a porta serie 1
//02 -> 1200   bits/sec
//03 -> 2400   bits/sec             //Estado
//04 -> 4800   bits/sec             //00 -> Para a receptor
//05 -> 9600   bits/sec             //01 -> Para o emissor
//06 -> 19200  bits/sec
//07 -> 38400  bits/sec
//08 -> 57600  bits/sec
//09 -> 115200 bits/sec
//10 -> 230400 bits/sec
//11 -> 460800 bits/sec
//12 -> 921600 bits/sec

char caminho[2][15]={"/dev/ttyS0","/dev/ttyS1"};
char opcao[2][11]={"RECEIVER","TRANSMITTER"};
int fd;

set_paridade_controlo(_frame_SET);
set_paridade_controlo(_frame_UA);


if (porta>1) return-1;
if (strncmp(modo,opcao[0],8)!=0 && strncmp(modo,opcao[1],11)) return -
1;
```

```
if (strncmp(modo,opcao[0],8)==0) estado=0;
else estado=1;
fd=open(caminho[porta], O_RDWR | O_NOCTTY);
if (fd<0) return fd;
if (tcgetattr(fd,&_oldtio)==-1)
{
      close(fd);
      return -1;
}
tcflush(fd, TCIFLUSH);
bzero(&_newtio, sizeof(_newtio));
_newtio.c_cflag= _BAUDRATE[baud] | CS8 | CLOCAL | CREAD;
_newtio.c_iflag= IGNPAR;
_newtio.c_lflag= 0;
_newtio.c_oflag= 0;
return conecta(fd);
};


int llclose(int fd)
{
int escrita,leitura, tentativa=1, i;
unsigned char buf_in[6];
unsigned char buf_first[L+7];
if (estado==1)
{
//emissor
_newtio.c_cc[VTIME]=timeout;
_newtio.c_cc[VMIN]=6;
if (tcsetattr(fd,TCSANOW,&_newtio)==-1)
{
      close (fd);
      return -1;
}
do{
tcflush(fd, TCIFLUSH);
for(i=0;i<6;i++) buf_first[i]=_frame_DISC[i];
for(;i<(L+7);i++) buf_first[i]=NUL;
set_paridade_informacao(buf_first);
escrita = write(fd, &buf_first, L+7);
limpa_buffer(buf_in,6);
leitura = read(fd, &buf_in,6);
if (memcmp(_frame_DISC, buf_in, 6)==0) break;
else tentativa++;
}while(tentativa<4);
if (tentativa==4) return -1;
escrita = write(fd, &_frame_UA, 6);
if (tcsetattr(fd,TCSANOW,&_oldtio)==-1)
{
      close(fd);
      return -1;
}
close(fd);
return 0;
}
else
{
//Receptor
_newtio.c_cc[VTIME]=0;
_newtio.c_cc[VMIN]=6;
if (tcsetattr(fd,TCSANOW,&_newtio)==-1)
```

```
{
      close (fd);
      return -1;
}
do{
tcflush(fd, TCIFLUSH);
escrita=write(fd,&_frame_DISC,6);
limpa_buffer(buf_in,6);
leitura = read(fd, &buf_in,6);
if (memcmp(_frame_UA, buf_in, 6)==0)break;
else tentativa++;
}while(tentativa<4);
if (tentativa==4) return -1;
if (tcsetattr(fd,TCSANOW,&_oldtio)==-1)
{
      close(fd);
      return -1;
}
close(fd);
return 0;
}
};

int llwrite(int fd, unsigned char *buffer, int length)
{
unsigned char buf_out[(L+7)], buf_in[(L+7)];
double tamanho=0;
int i,y,pac=0,z=0;
int leitura=0, escrita=0;
long int temp1, temp2;
unsigned long ac=0, acumulado=0;
pac=length/L;
if((length%L)!=0) pac++;

_newtio.c_cc[VTIME]=timeout;
_newtio.c_cc[VMIN]=6;
if (tcsetattr(fd,TCSANOW,&_newtio)==-1)
{
      close (fd);
      return -1;
}

set_paridade_controlo(_frame_RR0);
set_paridade_controlo(_frame_RR1);
set_paridade_controlo(_frame_REJ0);
set_paridade_controlo(_frame_REJ1);
set_paridade_controlo(_frame_NS0);
set_paridade_controlo(_frame_NS1);
do{
if(NS==0) for(i=0;i<6;i++) buf_out[i]=_frame_NS0[i];
else for(i=0;i<6;i++) buf_out[i]=_frame_NS1[i];
tamanho=length;
for(;(int)tamanho>0;i++)
{
      tamanho=tamanho/10.0;
      temp1=(int)tamanho;
      temp2=((tamanho-temp1)*10);
      buf_out[i]=(unsigned char)temp2;
}
for(;i<(L+7);i++) buf_out[i]=NUL;
set_paridade_informacao(buf_out);
```

```
do{
      printf("\nSeccao do tamanho");
      leitura=0;
      tcflush(fd, TCIFLUSH);
      write(fd, &buf_out, (L+7));
      limpa_buffer(buf_in,6);
      leitura = read(fd, &buf_in,6);
      printf("\nLeitura=%d",leitura);
      if(leitura!=0) break;
      printf("\nTimeout!!!");
}while(1);
}while(!((memcmp(_frame_RR1, buf_in, 6)==0)||(memcmp(_frame_RR0,
buf_in, 6)==0)));


for(i=0;i<pac;i++)
{
z=0;
if(NS==0) {NS=1;printf("\nEnviada trama de NS1");}
else {NS=0;printf("\nEnviada trama de NS0");}
  if(NS==0)for(y=0;y<6;y++) buf_out[y]=_frame_NS0[y];
  else for(y=0;y<6;y++) buf_out[y]=_frame_NS1[y];
for(y=0;y<L;y++)
{
      if((length-(i*L))<=y){z++; buf_out[y+6]=NUL;}
      else buf_out[y+6]=buffer[y+ac];
}
set_paridade_informacao(buf_out);
do{
do{
      printf("\nSeccao da informacao");
      leitura=0;
      tcflush(fd, TCIFLUSH);
      escrita=write(fd, &buf_out, (L+7));
      printf("\nEscrita=%d",escrita);
      limpa_buffer(buf_in,L+7);
      leitura = read(fd, &buf_in,(L+7));
      printf("\nLeitura=%d",leitura);
      if(leitura!=0) break;
      printf("\nTimeout!!!");
}while(1);
}while(!((memcmp(_frame_RR0, buf_in, 6)==0) || (memcmp(_frame_RR1,
buf_in, 6)==0)));
acumulado=acumulado+escrita-7-z;
ac=(L*(i+1));
}
return acumulado;
};



int llread(int fd, unsigned char *buffer)
{
unsigned char buf_in[(L+7)];
int leitura, escrita;
int i, pac=1, x=0, j,y=0;
unsigned long acc=0;
_newtio.c_cc[VTIME]=0;
_newtio.c_cc[VMIN]=(L+7);
if (tcsetattr(fd,TCSANOW,&_newtio)==-1)
```

```
{
      close (fd);
      return -1;
}
set_paridade_controlo(_frame_RR0);
set_paridade_controlo(_frame_RR1);
set_paridade_controlo(_frame_REJ0);
set_paridade_controlo(_frame_REJ1);
set_paridade_controlo(_frame_NS0);
set_paridade_controlo(_frame_NS1);
tcflush(fd, TCIFLUSH);
do{
do{
      limpa_buffer(buf_in,L+7);
      leitura=read(fd, &buf_in,(L+7));
      if (memcmp(_frame_DISC, buf_in,6)==0) {
                        printf("\nDesconectado!\n");
                        return 0;
}
}while(cmp_paridade_controlo(buf_in, _frame_NS0)<0);
if(cmp_paridade_informacao(buf_in)>0) break;
else escrita = write(fd, &_frame_REJ0, 6);
}while(cmp_paridade_controlo(buf_in, _frame_NS0)<0);
for(i=6;i<(L+6);i++) acc=(buf_in[i]*expoente(i-6))+acc;
// printf("\nacc do protocolo=%ld",acc);
buffer=(unsigned char *) realloc(buffer,acc);

escrita = write(fd, &_frame_RR1, 6);

do{
if (NS==0) NS=1;
else NS=0;
do{
do{
      limpa_buffer(buf_in,L+7);
      leitura = read(fd, &buf_in,(L+7));
      printf("\nLeitura=%d",leitura);
      if ((memcmp(_frame_NS0, buf_in, 6)==0) && (NS==0)) break;
      if ((memcmp(_frame_NS1, buf_in, 6)==0) && (NS==1)) break;
//    if ((memcmp(_frame_NS0, buf_in, 6)==0) && (NS==1)) {escrita =
write(fd, &_frame_RR1, 6); printf("\nEnviada trama de correcao de
RR1");}//para caso de falhar a transmissao de receptor para o emissor
e corrigir a variavel NS de ambos
//    if ((memcmp(_frame_NS1, buf_in, 6)==0) && (NS==0)) {escrita =
write(fd, &_frame_RR0, 6); printf("\nEnviada trama de correcao de
RR0");}
}while(1);

if(cmp_paridade_informacao(buf_in)>0) break;
else{
      if (NS==0) {
                  tcflush(fd, TCIFLUSH);
                  escrita = write(fd, &_frame_REJ0,
6);printf("\nEnviada trama de REJ0");
            }
      else {
            tcflush(fd, TCIFLUSH);
            escrita = write(fd, &_frame_REJ1, 6);printf("\nEnviada
trama de REJ1");
            }
 }
```

```
}while(1);
for(j=6; j<(L+6) && (y<acc);j++)
{
      x=L*(pac-1);
      y=x+(j-6);
      buffer[y]=buf_in[j];
}
pac++;
if (NS==0) {escrita = write(fd, &_frame_RR1, 6);printf("\nEnviada
trama de RR1");}
else {escrita = write(fd, &_frame_RR0, 6); printf("\nEnviada trama de
RR0");}
}while(y<(acc-1));
return y;
};
```

“evaristo.c”

```
#include <stdio.h>
#include "menus.h"
#include "protocolo.h"

int menu_opcoes(int *fd,int *m, int *p, int *b, int *fich, unsigned
char *ficheiro);
int menu_definicoes(int *m,int *p, int *b);
void menu_erro(void);

int main(int argc, char **argv)
{
int p, m, b, fd=0, pass=1, temp0, temp1, temp, flag=1,fich=-1;;
unsigned char *ficheiro;
ficheiro=(unsigned char *)malloc(500);
temp0=open("/dev/ttyS0", O_RDWR |O_NOCTTY);
if (temp0<0){
      close(temp0);
      menu_erro();
      return -1;
}
temp1=open("/dev/ttyS1", O_RDWR |O_NOCTTY);
if (temp1<0){
      close(temp1);
      temp1=1;
}
if (argc==4){
      flag=0;
      if ((atoi(argv[1])==0) || (atoi(argv[1])==1)) m=atoi(argv[1]);
      else flag=1;
      switch(atoi(argv[2])) {
            case 0: p=0; break;
            case 1: if(temp1==1) p=1; break;
            default: flag=1;
            }
```

```
      if ((atoi(argv[3])>=0) && (atoi(argv[3])<=7)) b=atoi(argv[3]);
      else flag=1;
}
if(flag==1)
{
      temp=menu_definicoes(&m,&p, &b);
      if(temp<=0) return -1;
}
do{
pass=1;
switch (menu_opcoes(&fd,&m,&p,&b,&fich,ficheiro))
{
      case 100: pass=1;return 0;
      case 99: pass=0;

}
}while(pass==0);
printf("\nPrograma terminado!\n");
return 0;
}
```

“menus.h”

```
#include <stdio.h>
#include "string.h"
#include "fgrafico.h"
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <wait.h>


#define pacotes 1024

void clrscr(void);
void cor(int a,int b,int c);
int llopen(int porta, char modo[], int baud);
int llread(int fd, unsigned char *buffer);
int llwrite(int fd, unsigned char *buffer, int length);
int llclose(int fd);
unsigned long int expoente(int n);
void limpa_buffer(unsigned char *buffer, int length);

extern char **environ;
int sair=0;

int menu_ficheiro(unsigned char *ficheiro)
{

const char *filename={"/bin/ls"};
char caminho[600]="./outfiles/";
char *const argv []={"/bin/ls","outfiles",NULL};
pid_t pid;
int status, i, fd_fich;
clrscr();
printf("\e[%dm",00);
printf("\e[%dm",00);
printf("\e[%dm",01);
```

```
printf("Escreva o nome do ficheiro que deseja enviar\n\n");
printf("\e[%dm",00);
pid=fork();
if (pid==0) return execve(filename, argv, environ);
wait(&status);
printf("\n_____
_____\n");
printf("Ficheiro: ");
scanf (" %s",ficheiro);
if (strlen(ficheiro)==0) return -1;
strcat(caminho, ficheiro);
ficheiro=(unsigned char *) realloc(ficheiro,strlen(caminho)+1);
for(i=0;i<=(strlen(caminho)+1);i++)ficheiro[i]=caminho[i];
fd_fich=open(ficheiro, O_RDONLY);
if(fd_fich<0) return -1;
close(fd_fich);
return 1;
}

int menu_modo(void)
{
char op;
clrscr();
printf("\e[%dm",00);
printf("\e[%dm",00);
printf("\e[%dm",01);
printf("Escolha o modo como deseja trabalhar:\n\n");
printf("\e[%dm",00);
printf("0 - Receptor\n");
printf("1 - Emissor\n");
printf("_____
_____\n");
printf("S - Sair\n");
printf("\nOpcao: ");
fflush(stdin);
scanf(" %c",&op);
fflush(stdin);
printf("\e[%dm",00);
switch(op)
{

      case '0': return 0;
      case '1': return 1;
      case 'S':
      case 's': return 100;
      default:    {
                        printf ("\nOpcao Invalida!!! Terminando o
programa!\n");
                        return -1;
                  }
}
return -1;
}

int menu_baudrate(void)
{
char op;
clrscr();
printf("\e[%dm",01);
printf("Escolha o baudrate que deseja trabalhar:\n\n");
printf("\e[%dm",00);
```

```
printf("0 - 110    bits/s\n");
printf("1 - 300    bits/s\n");
printf("2 - 1200   bits/s\n");
printf("3 - 2400   bits/s\n");
printf("4 - 4800   bits/s\n");
printf("5 - 9600   bits/s\n");
printf("6 - 19200  bits/s\n");
printf("7 - 38400  bits/s\n");
printf("_____________________________\n");
printf("S - Sair\n");
printf("\nOpcao: ");
fflush(stdin);
scanf(" %c",&op);
fflush(stdin);
switch(op)
{

      case '0': return 0;
      case '1': return 1;
      case '2': return 2;
      case '3': return 3;
      case '4': return 4;
      case '5': return 5;
      case '6': return 6;
      case '7': return 7;
      case 'S':
      case 's': return 100;
      default:    {
                        printf ("\nOpcao Invalida!!! Terminando o
programa!\n");
                        return -1;
                  }
}
return -1;
}

int menu_porta(void)
{
char op, retorno;
clrscr();
printf("\e[%dm",01);
printf("Escolha a porta serie na qual deseja trabalhar:\n\n");
printf("\e[%dm",00);
printf("0 - Porta 1\n");
if(open("/dev/ttyS1", O_RDWR |O_NOCTTY)>0) printf("1 - Porta 2\n");
else return 0;
printf("_____________________________\n");
printf("S - Sair\n");
printf("\nOpcao: ");
fflush(stdin);
scanf(" %c",&op);
fflush(stdin);
switch(op)
{

      case '0': retorno=0; break;
      case '1': retorno=1; break;
      case 'S':
      case 's': retorno=100; break;
```

```
      default:    {
                        printf ("\nOpcao Invalida!!! Terminando o
programa!\n");
                        retorno=-1;
                  }
}
return retorno;
}

void menu_cabecalho(int *fd, int *m, int *p, int *b)
{
clrscr();
printf("\n");
printf("\e[%dm",1);
printf("Estado -> ");
if (*(fd)<=0){
      printf("\e[%dm",31);
      printf("Inactivo");
}
if (*(fd)>0) {
      printf("\e[%dm",32);
      printf("Activo");
}
printf("\e[%dm",0);
printf("\e[%dm",1);
printf("\tModo -> ");
printf("\e[%dm",34);
if (*(m)==0) printf("Receptor");
else printf("Emissor\t");
printf("\e[%dm",0);
printf("\e[%dm",1);
printf("\tPorta n. ");
printf("\e[%dm",34);
printf("%d",*(p)+1);
printf("\e[%dm",0);
printf("\e[%dm",1);
printf("\tBaudrate -> ");
printf("\e[%dm",34);
switch(*(b)){
      case 0: printf("110 bits/s"); break;
      case 1: printf("300 bits/s"); break;
      case 2: printf("1200 bits/s"); break;
      case 3: printf("2400 bits/s"); break;
      case 4: printf("4800 bits/s"); break;
      case 5: printf("9600 bits/s"); break;
      case 6: printf("19200 bits/s"); break;
      case 7: printf("38400 bits/s"); break;
}
printf("\e[%dm",0);
printf("\n_____________________________________________________________
_____________________________\n\n");
}

int menu_liga()
{
      char op;
      sair=1;
      printf("0 - Connectar...\n");
      printf("1 - Alterar definicoes\n");
      printf("_____________________________________________________________
_____________________________\n");
```

```
      printf("S - Sair\n");
      printf("\nOpcao: ");
      fflush(stdin);
      scanf(" %c",&op);
      fflush(stdin);
      switch(op)
      {
            case '0': return 0;
            case '1': return 1;
            case 's':
            case 'S': return 100;
            default: return 99;
      }

return 99;
}

int menu_desliga(int *m, unsigned char *ficheiro, int *fich)
{
      char op;
      printf("0 - Desconnectar...\n");
      if (*(m)==1) {
            printf("1 - Definir o ficheiro\n");
      }
      if (*(fich)==1){
            printf("2 - Enviar ficheiro localizado em %s\n",ficheiro);

      }
      printf("_____
_____\n");
      if (sair==0)printf("S - Sair\n");
      printf("\nOpcao: ");
      fflush(stdin);
      scanf(" %c",&op);
      fflush(stdin);
      switch(op)
      {
            case '0': return 0;
            case '1': if (*(m)==1) return 1;
            case '2': if (fich>0) return 2;
            case 's':
            case 'S': if (sair==0) return 100;
            default: return 99;
      }
return 99;
}

int menu_definicoes(int *m,int *p, int *b)
{
switch(menu_modo())
{
      case 0: *(m)=0; break;
      case 1: *(m)=1; break;
      case 100: printf("\nPrograma terminado!\n"); return 0;
      default: printf("\nPrograma terminado!\n"); return -1;
}
switch(menu_porta())
{
      case 0: *(p)=0; break;
      case 1: *(p)=1; break;
      case 100: printf("\nPrograma terminado!\n"); return 0;
```

```
      default: printf("\nPrograma terminado!\n"); return -1;
}
switch(menu_baudrate())
{
      case 0: *(b)=0; break;
      case 1: *(b)=1; break;
      case 2: *(b)=2; break;
      case 3: *(b)=3; break;
      case 4: *(b)=4; break;
      case 5: *(b)=5; break;
      case 6: *(b)=6; break;
      case 7: *(b)=7; break;
      case 8: *(b)=8; break;
      case 100: printf("\nPrograma terminado!\n"); return 0;
      default: printf("\nPrograma terminado!\n"); return -1;
}
return 1;
}

void menu_erro(void){
      char op;
      clrscr();
      printf("\e[%dm",31);
      printf("\a\v\v\v\v\v\vNao foi possivel realizar o seu pedido neste
momento. Verifique os parametros da ligacao!!!\v\v\v\v");
      printf("\e[%dm",0);
      printf("\n_____________________________________________________
_____________________________\n");
      printf("\nPrima qualquer tecla para prosseguir...\n");
      fflush(stdin);
      scanf(" %c",&op);
      fflush(stdin);
}

int envia(int *fd, unsigned char *ficheiro)
{
int fd_fich, leitura=0, escrita=0, i, pac=0, temp1, temp2,
percentagem, tamanho=0, pacotes2, acc=0, percentagem_back=-1;
unsigned char *buffer, *tam;
unsigned char *ficheiro2;
float pac2;

ficheiro2=(unsigned char *)malloc(600);
tam=(unsigned char *)malloc(600);
for(i=11;i<strlen(ficheiro);i++) ficheiro2[i-11]=ficheiro[i];
realloc(ficheiro2,strlen(ficheiro2)+1);
buffer=(unsigned char *)malloc(2);
struct stat buf;
fd_fich=open(ficheiro, O_RDONLY);
if (fd_fich<0) return -1;
// Descobrimos o tamanho do ficheiro
fstat(fd_fich, &buf);
tamanho=buf.st_size;
printf("\nA enviar o ficheiro \"%s\" com o tamanho de %d
bytes",ficheiro2 ,tamanho);
pac=tamanho/pacotes;
if ((tamanho%pacotes)!=0)pac++;

pac2=(float)pac;
for(i=0;(int)pac2>0;i++)
{
```

```
      pac2=pac2/10.0;
      temp1=(int)pac2;
      temp2=((pac2-temp1)*10);
      tam[i]=(char)temp2;
}

realloc(tam,strlen(tam)+1);
realloc(buffer,buf.st_size+1);
// manda o nome do ficheiro
llwrite(*(fd),ficheiro2,strlen(ficheiro2)+1);
// manda o numero de pacotes
llwrite(*(fd),tam,strlen(tam)+1);
for(i=1;i<=pac;i++)
{
      percentagem_back=percentagem;
      percentagem=(i*100)/pac;
      if (percentagem_back!=percentagem){
            if ((percentagem>=0)&&(percentagem<10)) printf("\n%d%%
concluido",percentagem);
            if ((percentagem>=10)&&(percentagem<100)) printf("\n%d%%
concluido",percentagem);
            if (percentagem==100) printf("\n%d%%
concluido",percentagem);
      }
      if (i==pac) {
                  pacotes2=tamanho-((i-1)*pacotes);
                  printf("\npacotes2=%d",pacotes2);
                  limpa_buffer(buffer, pacotes2);
                  }
      else{
            pacotes2=pacotes;
            limpa_buffer(buffer, pacotes2);
            }
      leitura=read(fd_fich,buffer,pacotes2);
      if (leitura<0) return -1;
      // envia o ficheiro
      escrita=llwrite(*(fd),buffer,pacotes2);
      if (escrita<0) return -1;
      acc=acc+escrita;
}
close(fd_fich);
return 1;
}

int recebe(int *fd)
{
int back=99999, i, fd_fich, leitura=0, escrita,pac=0, acc=0,
percentagem=0, percentagem_back=-1;
unsigned char *ficheiro, *buffer,*tam;
unsigned char caminho[600]="./infiles/";
ficheiro=(unsigned char *)malloc(600);
buffer=(unsigned char *)malloc(pacotes);
tam=(unsigned char *)malloc(600);
printf("\nPronto a receber...");
printf("\n");
// vai receber o ficheiro
limpa_buffer(buffer,pacotes);
limpa_buffer(ficheiro,600);
limpa_buffer(tam,600);
back=99999;
back=llread(*(fd), ficheiro);
```

```
if (back==0)
{
      if (llclose (*(fd))==0){ *(fd)=-
1;free(buffer);free(ficheiro);free(tam); return 100;}
      else menu_erro();
}
else
if (back<0) return back;
for(i=10;(i-10)<(strlen(ficheiro)+1);i++) caminho[i]=ficheiro[i-10];
fd_fich=open(caminho,O_WRONLY|O_CREAT|O_TRUNC,S_IRUSR|S_IWUSR|S_IXUSR|
S_IRGRP|S_IWGRP|S_IXGRP|S_IROTH|S_IWOTH|S_IXOTH);

printf("\nA receber o ficheiro \"%s\".",ficheiro);
llread(*(fd),tam);

for(i=0;i<(strlen(tam)+1);i++) pac=(tam[i]*expoente(i))+pac;


for(i=1;i<=(pac+1);i++){
      printf("\npac=%d",i);
      percentagem_back=percentagem;
      percentagem=(i*100)/pac;
      if (percentagem!=percentagem_back){
            if ((percentagem>=0)&&(percentagem<10)) printf("\n%d%%
concluido",percentagem);
            if ((percentagem>=10)&&(percentagem<100)) printf("\n%d%%
concluido",percentagem);
            if (percentagem==100) printf("\n%d%%
concluido",percentagem);
      }
      limpa_buffer(buffer,pacotes);
      leitura=llread(*(fd),buffer);
      escrita=write(fd_fich, buffer, leitura);
      acc=escrita+acc;
      printf("\nacc=%d",acc);
}
printf("\n\nRecebido ficheiro \"%s\" de tamanho %d
bytes.\n\n",ficheiro,acc);
close(fd_fich);
llclose(*(fd));
return 100;
}

int menu_opcoes(int *fd,int *m, int *p, int *b, int *fich, unsigned
char *ficheiro)
{

int temp=0;
char modo[2][13]={"RECEIVER","TRANSMITTER"};


      menu_cabecalho(fd,m,p,b);
      if (*(fd)<=0){
            switch (menu_liga())
                  {
                        case 0: {
                              // Conectar
                              *(fd)=llopen(*(p),modo[*(m)],*(b));
                              if (*(fd)<0) menu_erro();
                              menu_cabecalho(fd,m,p,b);
                              if (*(m)==0) return recebe(fd);
```

```
                              return 99;
                              }
                        case 1: {
                              //Alterar definições
                              do{
                                    temp=menu_definicoes(m,p,b);
                              }while(temp<0);
                              return 99;
                              }
                        case 100: return 100;
                        case 99: return 99;
                  }
       }
      else {
            switch (menu_desliga(m, ficheiro, fich))
            {
                        case 0: {
                              //Desconecta
                              temp=llclose(*(fd));
                              if (temp<0) menu_erro();
                              else {
                                    *(fd)=0;
                                    *(fich)=-1;
                                    }
                              return 100;
                              }
                        case 1: {
                              //Definir ficheiro
                              *(fich)=-1;
                              if (menu_ficheiro(ficheiro)<0) *(fich)=-
1;
                              else *(fich)=1;
                              return 99;
                              }
                        case 2: {
                              if (envia(fd,ficheiro)>0) {
                                          temp=llclose(*(fd));
                                          *(fd)=0;
                                          *(fich)=-1;
                                          printf("\nFicheiro enviado
com exito!\n\n");
                                          return 100;
                                          }
                              else menu_erro();
                              return 99;
                              }
                        case 100: return 100;
                        case 99: return 99;
            }
      }
return 99;
}
```

“fgrafico.h”

```
#include <stdio.h>

void clrscr(void)
{
      printf("\033[2J");
      printf("\033[0;0f");
};


void cor(int a,int b,int c)
{
/*
      Atributos do ecran

            00-nada
            01-negrito
            04-sublinhado
            05-piscar
            07-inverter
            08-escondido

      Cores para o texto

            30-preto
            31-vermelho
            32-verde
            33-amarelo
            34-azul
            35-magenta
            36-azul_bebe
            37-branco

      Cores para o fundo do texto

            40-preto
            41-vermelho
            42-verde
            43-amarelo
            44-azul
            45-magenta
            46-azul_bebe
            47-branco

      Para voltar ao normal

            00-normal
*/
      /*printf("\33[%dm",x);*/
      printf("\e[%d;%d;%dm",a,b,c);
};
```