**Faculdade de Engenharia da Universidade do Porto**

C:\Users\-DIOGO\Downloads\Cores oficiais\logótipo com cores oficiais.tif

**Rede de Computadores**

**Trabalho 1**

**27/04/2015**

**Turma:** *3MIEIC5*

**Autores:**

Ana Catarina Amaral [@fe.up.pt](mailto:up201306082@fe.up.pt)

Pedro Miguel Vieira da Câmara [up201304073@fe.up.pt](mailto:up201304073@fe.up.pt)

Pedro Oliveira da Silva [up201306095@fe.up.pt](mailto:up201306095@fe.up.pt)

Índice

[Introdução 3](#_Toc434015549)

[Arquitetura 4](#_Toc434015550)

[Estrutura do código 4](#_Toc434015551)

[Casos de uso principais 4](#_Toc434015552)

[Protocolo de ligação lógica 4](#_Toc434015553)

[Protocolo de aplicação 4](#_Toc434015554)

[Validação 4](#_Toc434015555)

[Elementos de valorização 5](#_Toc434015556)

[Conclusão 5](#_Toc434015557)

[Bibliografia 6](#_Toc434015558)

# Sumário

O seguinte relatório serve para apresentar a implementação de um protocolo, estudado nas aulas teóricas, que permite a transmissão de ficheiros entre computadores, com o uso de um cabo de série.

Quanto à implementação em si, o código está dividido em 2 camadas, uma que representa a aplicação para comunicar com o utilizador e uma outra, mais baixo nível, a camada de link que trata do funcionamento interno e implementação do protocolo em si.

# Introdução

Como descrito no Sumário, este trabalho serviu para pôr em prática a matéria estudada nas aulas teóricas sobre o mecanismo de comunicação entre computadores com o uso de uma porta série. Este mecanismo está secionado em várias camadas de modo a ter uma estrutura organizada e de fácil interpretação. Para uma melhor clarificação, este relatório irá servir para explicar com melhor clareza todo este funcionamento.

Em seguida, irá ser apresentado um conjunto de tópicos que permite explicar como está organizado o código, desde a comunicação com o utilizador ao próprio envio dos pacotes.

# Arquitetura e Estrutura do código

Como descrito anteriormente, todo o código foi divido em camadas.

A cama da aplicação é responsável ligação entre o utilizador e o programa em si. Tem como principal função obter as informações necessárias para a configuração do protocolo, como por exemplo, o número de tentativas no caso de uma falha de leitura ou até mesmo o tempo para ativar o alarme. Para além disso, contém as funções mais alto nível para o envio e receção de pacotes de dados relativo ao ficheiro a enviar e pacotes de controlo para avisar o início e fim do envio.

A outra camada, *linkLayer*, trata de todo o funcionamento mais baixo nível do programa. Após o envio de um pacote cabe a esta processá-lo e verificar a cada leitura se a informação enviada contém erros, se é informação repetida ou se simplesmente falta informação. Esta verificação tem um tempo limite e um número máximo de tentativas que caso sejam ultrapassados dá erro.

Foi criada em cada camada algumas estruturas para melhor gerir os dados. Na primeira era essencial guardar o descritor da porta série usada e em que modo foi acedida, emissor ou recetor.

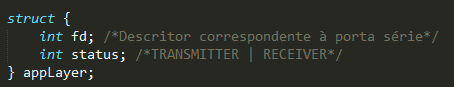


Figura : Estrutura que representa a camada da aplicação.

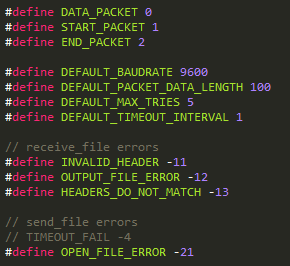
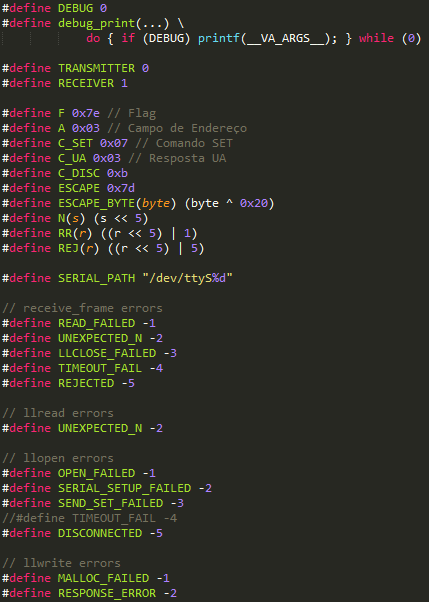
Para além desta estrutura foram guardadas diversas constantes, desde *flags* usadas a valores guardados por defeito no caso de o utilizador não os inserir ou mesmo possíveis erros.

Figura 2: Ficheiro .h da camada de aplicação

No caso da segunda camada, foi criada uma estrutura de dados para gerir a estatística, as configurações definidas e uma última para representar a informação enviada em cada trama. Mais uma vez foram guardas diversas variáveis para os diversos erros que podem ocorrer nesta camada.

Figura 3: Ficheiro .h usado na cama de linklayer

# Casos de uso principais

No começo do programa é necessário passar-lhe, no mínimo, o número da porta série e o nome do ficheiro a enviar. Para além disso, podem ser definidos o baudrate, o número máximo de tentativas, o tempo para ativar o alarme, o tamanho de cada pacote de dados e a ativação do modo de simulação de erros.

É chamada a função llopen() para gerar o descritor da porta série utilizada. Em seguida, dependendo do modo entre emissor ou recetor e chamada a sua função respetiva para gerir o ficheiro, send\_file() ou receive\_file(). No caso do emissor, é enviado um pacote de controlo através da função send\_control\_packet() e a partir daí são criadas tramas I com o uso da função create\_i\_frame() para depois serem enviadas pelo método llwrite(). Cada trama com dados do ficheiro a enviar tem o tamanho definido. Em cada trama são colocadas diversas flags e verificações de erros para garantir a fiabilidade do protocolo.

Após o envio é enviado uma verificação se o envio foi de uma trama correta ou não com o uso da função receive\_RR\_frame(). Isto repete-se num ciclo até ser enviado todos os dados, no fim é enviado novamente um novo pacote de controlo a sinalizar o fim do envio dos dados.

Para o caso do recetor o processo é semelhante, vai-se lendo os bytes recebidos na função receive\_file() e processados na função receive\_frame() ,quando chamada pela llread(), e enviado a informação relativa ao sucesso ou não da receção através da função send\_rr\_frame().

Quando todas as tramas forem enviadas e com sucesso é chamada a função llclose() para sinalizar o fim do envio com o uso da flag DISC de disconnect.

# Protocolo de ligação lógica

# Protocolo de aplicação

# Validação

Para garantir que o trabalho foi bem feito, foram feitos diversos testes.

O principal de todos foi o envio do ficheiro com sucesso (Inserir imagem). Após garantir que o envio estava funcional, foi testado o envio com a remoção do cabo durante o processo e com a introdução de informação errada com o raspar de uma chave. E o último foi testar o exceder de tentativas e de timeout.

# Elementos de valorização

Da lista apresentada no guião do trabalho foi possível implementar todas os elementos de valorização com sucesso. Para além disso foi implementado uma opção de novo reenvio no fim de todas as tentativas.

# Conclusão