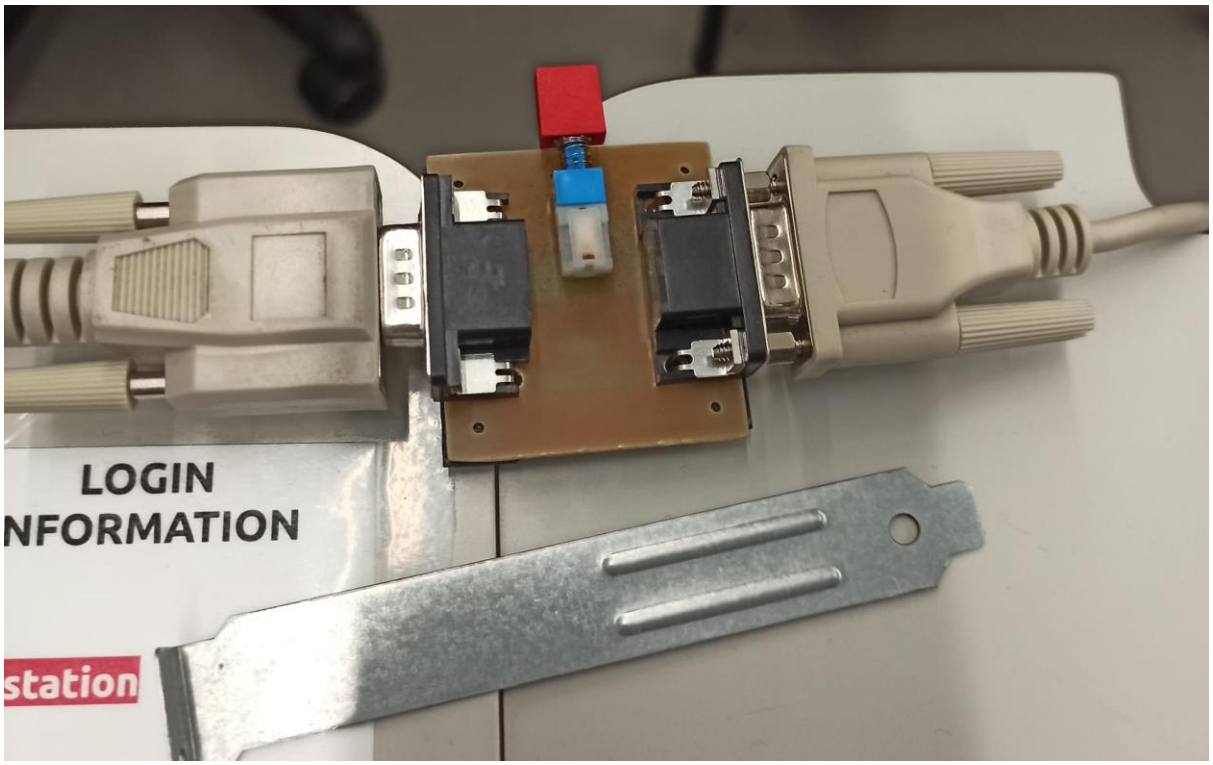
U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

1º Trabalho Laboratorial
Ligação de Dados



Trabalho realizado por:

Henrique Nunes – up201906852
Frederico Rodrigues – up201807626

Unidade Curricular: RC
Data de Entrega: 11 de dezembro de 2021

Sumário

Este relatório foi elaborado com o propósito de explicar, apresentar e resumir o projeto desenvolvido durante a primeira metade do semestre. Trata-se de uma aplicação capaz de realizar a transferência de ficheiros com sucesso através da porta de série. Os objetivos foram alcançados sendo esta uma aplicação fiável, capaz de lidar com possíveis erros e com ficheiros de vários tamanhos.

Introdução

Este trabalho laboratorial tem como objetivo principal elaborar um protocolo de ligação de dados que permita enviar informação através da porta série. Além disso é pretendido também a realização de uma aplicação que envie um ficheiro pela porta série tirando partido para isso do protocolo de ligação de dados elaborado.

O objetivo do relatório é descrever o trabalho laboratorial desenvolvido e encontra-se dividido pelas seguintes secções:

- Arquitetura - Visualização de interfaces e modelo de interação com o utilizador
- Estrutura do código - Apresentação dos vários módulos utilizados, bem como a descrição das suas principais funcionalidades
- Casos de uso principais - Identificação das sequências principais de chamadas de funções descrevendo o seu fluxo
- Protocolos - Identificação dos principais aspetos funcionais, descrição da estratégia de implementação destes aspetos com apresentação de extratos de código
    - Ligação de Dados - Ao nível da ligação de dados
    - Aplicação - Ao nível da aplicação
- Validação - Descrição de testes efetuados com resultados quantificados
- Eficiência do protocolo de Ligação de Dados - Caracterização estatística da eficiência do protocolo
- Conclusão - Reflexão sobre os objetivos alcançados

Arquitetura

O projeto desenvolvido apresenta dois blocos fundamentais: o Transmissor e o Recetor. Ambos recorrem a várias funções comuns tanto da camada de aplicação como da camada de ligação de dados, no entanto a sua implementação determina o código a executar de acordo com qual desencadeou a chamada da aplicação.

A camada de aplicação, implementada no ficheiro ***app.c***, gere o enviar e receber de ficheiros, não conhecendo os detalhes do protocolo de ligação de dados, mas apenas o modo como acede ao serviço.

A camada de ligação de dados, implementada no ficheiro ***data_link.c*** e também ***data_link_aux.c***, é responsável pelo estabelecimento e terminação da ligação, e trata da

interação com a porta série, fornecendo um serviço de comunicação de dados fiável entre dois sistemas ligados.

Existe também uma interface que, baseada num módulo de *logs* ***log.c***, revela informações úteis sobre a utilização do programa, tais como o estabelecimento ou fecho da ligação, receção ou envio de tramas, tamanho do ficheiro já transferido e se este foi terminado com ou sem sucesso. Também são impressas mensagens no caso de ocorrência de erros.

No momento da execução o utilizador deve indicar se é o transmissor ou o recetor, o número da porta série a utilizar e, no caso do transmissor, o *path* do ficheiro a transmitir ou, no caso do recetor, o *path* onde deve ser colocado o ficheiro recebido.

Estrutura de Código

app
Este módulo é a camada superior do programa e implementa o protocolo da aplicação com o objetivo de lidar com o ficheiro a ser transmitido/recebido, bem como construir *packets* que permitam transmiti-lo através da camada inferior do programa, a camada de ligação de dados. Possui uma estrutura **ApplicationLayer** que guarda o *path* e o tamanho do ficheiro a ser transferido/recebido, o papel do executante (transmissor ou recetor) e o *file descriptor* da porta série.

data_link
Este módulo implementa o protocolo de ligação de dados que se encontra na camada inferior à camada de aplicação e permite assim lidar com a escrita e leitura da porta série com recurso a um módulo auxiliar ***data_link_aux***. A separação do protocolo de ligação de dados nestes dois módulos permite abstrair a camada superior das implementações específicas do protocolo, deixando-a apenas com acesso às suas funções principais. Possui uma estrutura **LinkLayer** que guarda o número de sequência da trama, a capacidade da ligação (*baudrate*), tempo base do *timeout* e número de retransmissões máximas aquando de uma falha.

data_link_aux
Neste módulo, que serve de auxílio ao módulo ***data_link***, são implementadas funções necessárias ao protocolo de ligação de dados, mas que se pretende esconder o seu acesso das camadas superiores.

state_machine
Neste módulo é implementada uma máquina de estados que permite ler as tramas recebidas no que diz respeito ao protocolo de ligação de dados. Interiormente, consegue distinguir entre diferentes tipos de tramas, seguindo o seu caminho de acordo com o recebido.

alarm
Este módulo implementa o *setup* do alarme (comportamento do programa quando recebe um sinal *SIGALARM*) bem como a estratégia para lidar com a sua ocorrência.

log
Neste módulo são implementadas funções que permitem revelar informações de ocorrências no programa - erros, tramas enviadas ou recebidas, progresso da transferência do ficheiro e apresentações de mensagens -, quando invocadas, de forma uniforme.

efficiency
Este módulo permitiu realizar testes de eficiência no protocolo de ligação de dados, contendo implementações de geração de erros, estratégias de verificação de desempenho do programa e apresentação dos resultados obtidos em *stderr* de modo a facilitar a extração da sua informação para um ficheiro, aquando da não ocorrência de erros na sua execução.

Casos de uso principais

Do lado do **transmissor**:

O ficheiro a transmitir é escolhido e a ligação é estabelecida após a chamada da função ***llopen***, que por sua vez chama a função ***ll_open_transmitter***. É chamada a função ***transmitter*** que tem a capacidade de construir *packets* através de chamadas às funções ***build_control_packet*** e ***build_data_packet*** e inicia e termina a transferência do mesmo com chamadas sucessivas a ***llwrite***. No fim da transmissão é chamada a função ***llclose***, que por sua vez chama a função ***ll_close_transmitter***, resultando na terminação da ligação.

Do lado do **emissor**:

É escolhido o *path* da localização do novo ficheiro e a ligação é estabelecida após a chamada da função ***llopen*** que, por sua vez, chama a função ***ll_open_receiver***. É chamada a função ***receiver***, que através de sucessivas chamadas à função ***llread*** efetua a leitura das tramas de informação enviadas pelo emissor. Por fim é chamada a função ***llclose*** que por sua vez chama a função ***ll_close_receiver***, resultando na terminação da ligação.

Protocolo de ligação lógica

O protocolo de ligação de dados é o principal módulo deste programa uma vez que permite atingir o objetivo principal proposto: estabelecer uma ligação e trocar dados de informação através da porta série.

Este protocolo é baseado na transmissão e receção de tramas codificadas que são iniciadas e terminadas com uma *FLAG* para identificação do início e do término da mesma. Além disso ainda possuem um campo de Endereço (*A*) e um campo de Controlo (*C*), bem como mecanismos de deteção de erros (BCC_1 *e* BCC_2). As tramas de informação ainda possuem o campo de informação que contem os dados a transmitir.

```
void create_su_frame(uchar* frame, uchar address, uchar control) {
  frame[0] = FLAG;
  frame[1] = address;
  frame[2] = control;
  frame[3] = BCC(frame[1], frame[2]);
  frame[4] = FLAG;
}
```

Assim, tal como pedido no enunciado, foram implementadas quatro funções principais (***llopen***, ***llwrite***, ***llread***, ***llclose***) que são disponibilizadas à camada superior - camada da Aplicação - de forma a esta utilizar o protocolo de ligação de dados.

A função ***llopen*** tem um comportamento diferente consoante seja chamada pelo transmissor ou pelo recetor. Em ambos os casos, primeiramente é aberto o ficheiro correspondente à porta série indicada pelo utilizador em modo não canónico. De seguida, há uma tentativa de estabelecer a ligação entre as duas partes na qual o recetor espera que uma trama de controlo *SET* seja enviada por parte do transmissor para de seguida enviar a confirmação da sua receção com uma resposta *UA*. Já o transmissor envia primeiro o comando *SET* e espera de seguida pela confirmação de receção do lado do recetor.

A função ***llwrite*** deverá ser utilizada por parte do transmissor para enviar tramas de informação onde são incluídos os dados de informação que são pretendidos transmitir pela camada superior. Antes de ser enviada esta trama passa por um mecanismo de *byte stuffing* que impede que a *FLAG* seja reproduzida no interior da trama. É tentado enviar a trama até que seja recebida uma resposta de confirmação positiva (*RR*) por parte do recetor. Caso a resposta recebida seja uma resposta de confirmação negativa (*REJ*) então volta a enviar a mesma trama.

A função ***llread*** deverá ser utilizada por parte do recetor para receber tramas de informação. Estas tramas passam por um mecanismo de *byte destuffing* para voltarem à sua forma original, são verificados os mecanismos de deteção de erros e se a trama recebida é repetida de forma a escolher que resposta deverá enviar ao recetor (*RR* caso pretenda receber a trama seguinte ou *REJ* caso pretenda receber a mesma trama novamente, terminando ou permanecendo à espera da nova receção da mesma trama, respetivamente).

A função ***llclose***, que pretende terminar a ligação estabelecida, tem um comportamento diferente consoante seja chamada pelo transmissor ou pelo recetor. No caso do transmissor, envia um comando *DISC*, espera para receber um comando *DISC* vindo do recetor e de seguida envia uma resposta *UA*. Já o recetor espera receber um comando *DISC*, envia um novo comando *DISC* e espera pela resposta de confirmação *UA* do transmissor. No fim, em ambos os casos, a porta série aberta no início da ligação é fechada.

Nota$_1$: Em todos os comandos enviados, quer pelo transmissor quer pelo recetor, há um mecanismo de reenvio caso não seja recebido uma resposta do lado oposto ao fim de um tempo máximo predefinido. Caso o limite de retransmissões seja atingido, o comando não é enviado novamente e cabe assim à camada superior decidir o passo seguinte.

Nota$_2$: A leitura das tramas é realizada byte a byte e tira partido de uma máquina de estados de forma a confirmar que o byte recebido é o esperado para o estado em que se encontra no que toca ao cabeçalho das tramas (*FLAG, A, C, BCC$_1$*). Caso a trama recebida

```c
int read_info_frame(int fd, uchar address, uchar* wanted_controls, int n_controls, uchar* frame) {
  State_machine* sm = create_sm(address, wanted_controls, n_controls);
  uchar byte;
  while (sm->state != STOP ) {
    if (read(fd, &byte, sizeof(char)) > 0) {
      event_handler_sm(sm, byte, frame, INFORMATION);
    }
  }

  int frame_size = sm->frame_size;
  destroy_sm(sm);

  return frame_size;
}
```

não esteja de acordo com o esperado a máquina de estados não atinge o estado final, voltando ao estado inicial e ficando, assim, à espera de uma nova trama.

Protocolo de aplicação

O protocolo de aplicação é o módulo que permite tirar partido do protocolo de ligação de dados para enviar um ficheiro do transmissor para o recetor.

Este protocolo é baseado na passagem de pacotes codificados à camada inferior (*Camada de Ligação de Dados*). Os pacotes de dados possuem um campo de Controlo (*C*), um número de sequência (*N*), dois octetos que indicam o tamanho dos dados a enviar (L_2 e L_1) e o campo de dados que possui *K* octetos com dados ($K = L_2*256 + L_1$). Os pacotes de controlo possuem também um campo de Controlo (*C*) e parâmetros codificados na forma *TLV* (*Type, Length, Value*).

Este módulo possui duas funções principais (*transmitter*, *receiver*) que são invocadas dependendo se o programa foi executado em modo transmissor ou recetor. Antes da sua invocação é estabelecida a ligação entre as partes com a chamada à camada inferior *llopen* e após a sua invocação é terminada a ligação com a chamada novamente à camada inferior *llclose*, à qual se segue o término do processo.

A função ***transmitter*** começa por abrir o ficheiro que pretende enviar. De seguida, envia, através da chamada à camada inferior *llwrite*, um pacote de controlo com o campo de Controlo *START* para informar o recetor que vai começar o envio da informação do ficheiro e o tamanho e nome do ficheiro como parâmetros do código *TLV*. Seguidamente, dá-se início ao envio da transmissão do ficheiro. São lidos do ficheiro um número predefinido de bytes de informação e enviado um pacote de dados com o campo de Controlo *DATA* seguido do número de sequência que é gerado sequencialmente e dos dados (bytes lidos do ficheiro). Esta sequência de operações é realizada até que todos os dados lidos do ficheiro tenham sido enviados ou até que haja um erro no envio da informação que comprometa a transmissão do ficheiro. Após, é enviado um novo pacote de controlo, semelhante ao inicial, mas com o campo de Controlo *END* para informar o recetor que terminou o envio de informação do ficheiro.

```
int build_control_packet(uchar* packet, uchar type) {
  uchar offsize = (uchar)sizeof(off_t);
  packet[0] = type;
  packet[1] = FILE_SIZE;
  packet[2] = offsize;
  memcpy(&packet[3], &al.filesize, offsize);
  uchar filename_len = (uchar)strlen(al.filename);
  packet[3+offsize] = FILE_NAME;
  packet[4+offsize] = filename_len;
  memcpy(&packet[5+offsize], &al.filename, filename_len);

  return 5 + offsize + filename_len;
}
```

```
int build_data_packet(uchar* packet, uchar n, uchar* data, int data_size) {
  packet[0] = PACK_DATA;
  packet[1] = n % 256;
  packet[2] = data_size / 256;
  packet[3] = data_size % 256;
  memcpy(&packet[4], data, data_size);

  return data_size+4;
}
```

A função ***receiver*** lê, ciclicamente, os pacotes recebidos através da chamada à camada inferior *llread* e, a cada iteração, toma uma ação de acordo com o seu conteúdo. Caso o pacote recebido seja identificado como um pacote de controlo para o início da transmissão de dados, é tentado ler o parâmetro do tamanho do ficheiro para no final verificar se a transmissão foi bem sucedida e o parâmetro do nome do ficheiro que é utilizado para, mal

recebido, abrir o ficheiro no qual vai ser escrita a informação recebida por parte do transmissor. Caso o pacote recebido seja identificado como um pacote de dados e já tenha recebido ordem para começar a escrita da informação, é verificado se o número de sequência do pacote de dados é o esperado e em caso afirmativo escreve no ficheiro aberto os dados recebidos. Caso o pacote recebido seja identificado como um pacote de controlo para o fim da transmissão de dados, é verificado se o tamanho do ficheiro indicado nos pacotes de controlo coincide com o tamanho do novo ficheiro que foi escrito, agindo em conformidade e indicando o utilizador do sucesso da operação.

Validação

Foram efetuados diversos testes de validação e o programa mostrou-se bastante robusto em todos, não apresentando qualquer comportamento indesejado e, sempre que possível, transferindo o ficheiro com sucesso.

Estes testes consistiram na provocação de interferência entre a ligação da porta série - faz alterar os bits enviados/recebidos -, no bloqueio temporário da porta série - impedia qualquer informação de ser transmitida momentaneamente - e na transmissão de diferentes ficheiros de diferentes tamanhos.

Eficiência do protocolo de ligação de dados

Foram realizados testes de eficiência onde se fez alterar a probabilidade de erro numa trama (**FER**), tempo de propagação (**T_prop**), capacidade (**C/Baudrate**) e o tamanho dos dados sem *stuffing* numa trama de informação (**Size**), de forma individual para medir a eficiência do protocolo. A eficiência foi calculada como sendo $S = R/C$ onde **C** é a capacidade da porta série e **R** é a taxa (bits/s) a que foi transmitido ficheiro que é obtida da relação entre o tamanho do ficheiro transmitido e o tempo que demorou transmitir todas as tramas de Informação. Foram realizadas três amostras para cada teste das quais os valores utilizados são os valores médios. E usou-se como base de teste FER = 6 %, T_prop = 0 ms, C = 38400 bits/s e Size = 256 B que se mantiveram constantes exceto nos casos de teste do próprio parâmetro.

O **FER** foi utilizado para configurar dois parâmetros, um que indica a probabilidade de ocorrer um erro no BCC_1 e outro que indica a probabilidade de ocorrer um erro no BCC_2, a partir de uma relação entre o número de bytes da trama que poderiam gerar o erro do respetivo BCC e o número total de bytes da trama, a multiplicar pelo FER. Este erro é gerado após a leitura de cada BCC, mas antes da sua validação, o BCC_1 na máquina de estados e o BCC_2 na função *llread*. O **T_prop** foi simulado com recurso à invocação da função *usleep* no momento imediatamente anterior ao envio de uma trama. Já **Size** e **Baudrate** foram variados alterando as macros *MAX_SIZE* e *BAUDRATE*, respetivamente.

Com análise dos gráficos obtidos (Anexo I) dos vários testes, pode concluir-se o seguinte: a eficiência do protocolo diminui com o aumento do **FER**, pois, como mais tramas vão falhar, será necessário mais tempo para transmitir o mesmo número de bytes com sucesso (Gráfico

1); com o aumento de **T_prop**, a eficiência também parece diminuir embora seja algo impreciso devido à reduzida quantidade de amostras (Gráfico 2); o aumento de **Size** provoca um ligeiro aumento na eficiência do protocolo (Gráfico 3); já o aumento de **C** aparenta manter a eficiência do protocolo constante uma vez que faz aumentar a taxa de receção de tramas (**R**) na mesma proporção (Gráfico 4);

Assim, conclui-se que diferentes parâmetros provocam diferentes eficiências do protocolo e que é importante escolher bem estes parâmetros de forma a maximizar o seu resultado. Além disso, também foi notado que o aumento do número de amostras por testes, bem como o aumento do número de testes utilizados melhoraria a qualidade dos resultados obtidos, tornando-os mais precisos e com um nível de confiança maior, de forma ser possível retirar conclusões mais concisas e transmitir uma ideia mais exata das curvas de eficiência com as variações provocadas.

Conclusões

Em suma, este trabalho laboratorial permitiu conceber um protocolo de ligação de dados entre duas máquinas pela porta série. Desta forma, foi colocado em prática os conhecimentos teóricos e implementado um protocolo *STOP & WAIT* de raiz, o que fez entender a importância de uma estruturação por camadas, bem como da independência entre as mesmas. Além disso, foi possível entender os métodos de transmissão de dados através de camadas utilizados, bem como o encapsulamento deles nas tramas e nos pacotes.

Assim, tanto o protocolo de ligação de dados como o protocolo de aplicação, foram corretamente implementados, respondendo muito positivamente aos possíveis erros que possam surgir ao longo da transmissão dos dados, tal como esperado, e, simultaneamente, com uma eficiência elevada nos casos de uso correntes.

Por fim, dado o sucesso na transmissão de ficheiros e o conhecimento obtido na realização do trabalho, entende-se que todos os objetivos foram cumpridos.

Anexos I

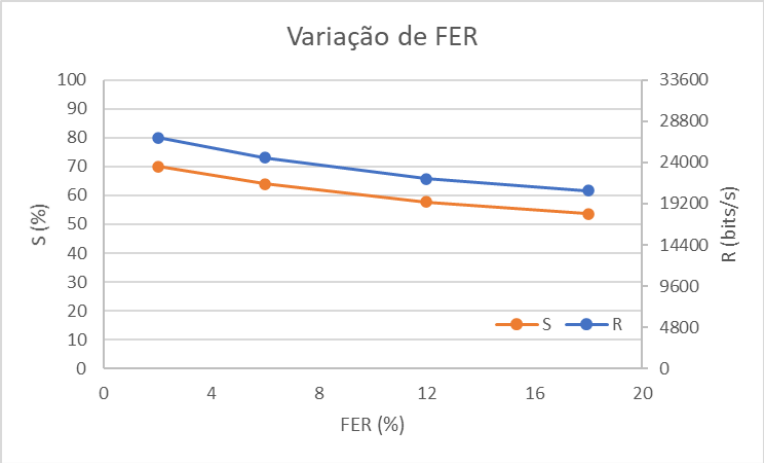


Gráfico 1: Variação de FER

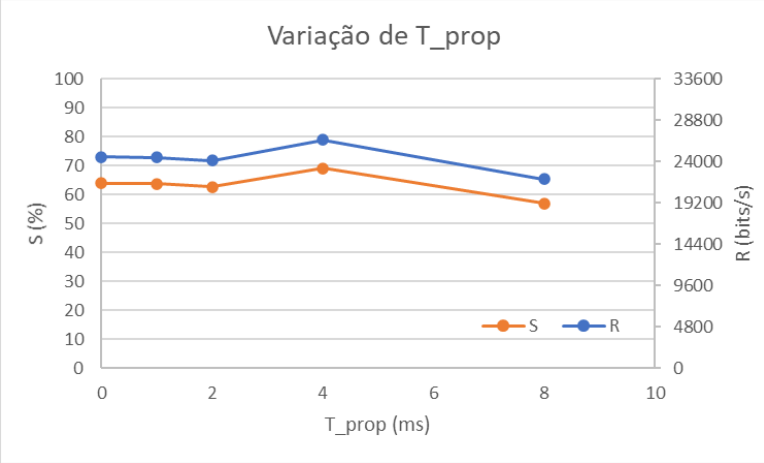


Gráfico 2: Variação de T_prop

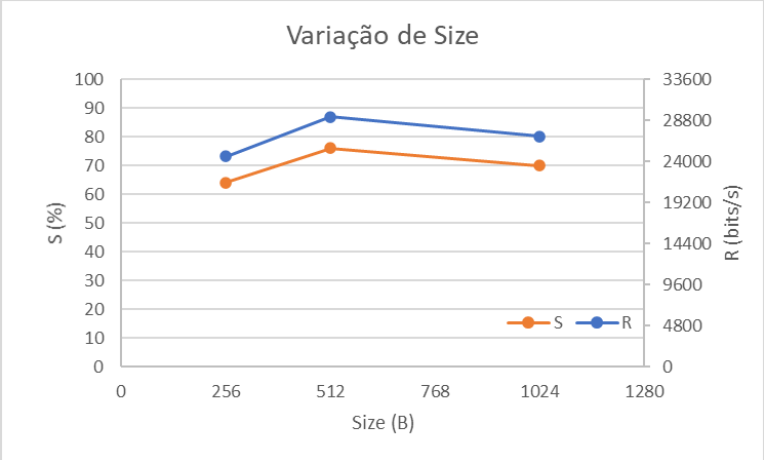


Gráfico 3: Variação de Size

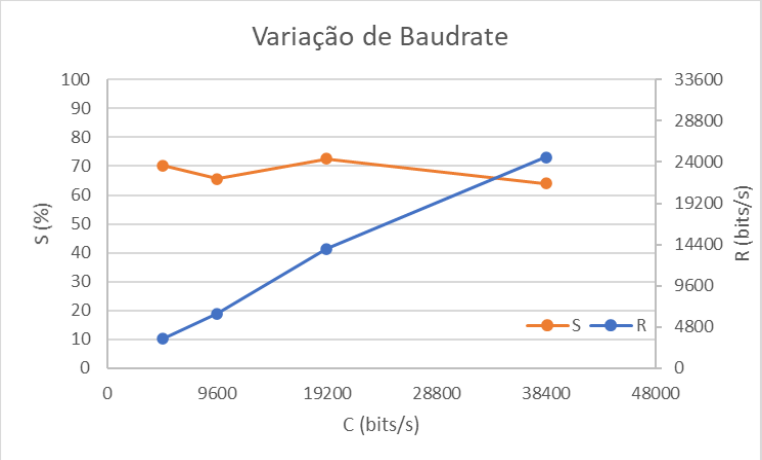


Gráfico 4: Variação de Baudrate

Anexos II

O código fonte do trabalho laboratorial encontra-se no diretório *src* no mesmo ficheiro *zip* deste relatório dado que a cópia do mesmo para este relatório não seria de leitura e interpretação tão acessível. Todo os *header files* encontram-se devidamente comentados com comentários *doxygen* de forma a facilitar a interpretação das funções utilizadas.