Universidade do Minho
Escola de Engenharia

Laboratórios de Telecomunicações e Informática I

ENGENHARIA DE TELECOMUNICAÇÕES E INFORMÁTICA

2020/2021

(Docentes: José Augusto Afonso, Nuno Vasco Moreira Lopes)

26 de novembro de 2020

Relatório

FASE 2

Inês Barreira Marques – a84913@alunos.uminho.pt

José Pedro Fernandes Peleja - a84436@alunos.uminho.pt

Rui Filipe Ribeiro Freitas - a84121@alunos.uminho.pt

Tiago João Pereira Ferreira - a85392@alunos.uminho.pt

Índice

Índice de figuras

Índice de tabelas

Lista de abreviaturas

RF - Radiofrequência

TRF - *Transceiver* radiofrequência

SPI - Serial Peripheral Interface

CE - *Chip Enable*

CSN - *Chip Select Not*

SCK - *Serial Clock*

MOSI - *Master Out Slave In*

MISO - *Master In Slave Out*

RTT - *Round-Trip Time*

Ttrans- tempo de transmissão

L- Tamanho do pacote

R- Taxa de Transmissão

Introdução

No âmbito da Unidade Curricular de LTI I (Laboratórios de Telecomunicações e Informática I) foi-nos proposto o desenvolvimento de uma aplicação que permita a conversação em modo texto, em tempo real, entre dois computadores pessoais. Pretende-se que para além da possibilidade de ambos os utilizadores conversarem em modo texto possam enviar os mais variados ficheiros incluindo imagens.

Para a realização deste projeto iremos utilizar o modelo OSI que será devidamente explicado posteriormente. As camadas que utilizaremos neste projeto serão as camadas 1, 2 e 7 que correspondem à camada física, à camada de ligação de dados e à camada de aplicação.

Este projeto está dividido em 5 fases em que cada uma terá uma entrega de um relatório e demonstração prática da aplicação em funcionamento, com exceção da última fase que para além disso terá também uma apresentação oral.

O presente relatório corresponde à fase 2 em que os objetivos principais passam por conhecer as caraterísticas da interface de comunicação série entre os transceivers e o Arduino, conhecer e compreender o impacto no desempenho do sistema dos diferentes parâmetros configuráveis dos transceivers e desenvolvimento de código necessário para a realização de testes experimentais para avaliação de desempenho do sistema como *round-trip time*(RTT), débito máximo e alcance de transmissão.

De modo a sermos capazes de cumprir com os objetivos deste projeto semestral devemos pôr em prática conhecimentos adquiridos noutras unidades curriculares, nomeadamente Redes de Computadores I, Sistemas Operativos, Paradigmas de Programação, entre outras.

1. Comunicação por radiofrequência Arduino-TRF-TRF-Arduino

1.1. Fundamentos

1.1.1. Transceiver RF

Nesta fase do projeto foi-nos proposto a elaboração da comunicação por radiofrequência entre a placa ESP32 e as placas *transceiver* de radiofrequências (nRF24L01+). Para isso tivemos de aprofundar o estudo nos *transceivers* pois vamos precisar destes para o funcionamento do trabalho. O módulo RF consiste num circuito integrado capaz de receber e transmitir sinais a pequenas distâncias, capaz de uma ligação *full-duplex* permitindo a receção e transmissão de informação entre dois dispositivos ao mesmo tempo.

Após estas considerações gerais, passemos à especificação do nosso módulo RF, este utiliza uma banda de frequência de 2.4 GHz e opera a um *baud rate* entre os 250 kbps e os 2 Mbps, significativamente superior aos utilizados na primeira fase na transferência de ficheiros através de porta-série.

A figura 1 corresponde ao *pinout* do módulo RF.

Figura 1 - Pinout do módulo RF.

Como podemos observar na figura 1 o módulo RF contém 8 pinos, cujas funções vão ser explicadas a seguir **[1]**:

- **VCC** -> Fornece energia ao módulo que pode variar entre 1.9 e 3.9 volts;

- **GND** -> Corresponde ao *Ground*;

- **CE** -> Corresponde ao *Chip Enable,* quando selecionado o módulo transmite ou recebe dependendo do modo que estiver ativo;

- **CSN** -> Corresponde ao *Chip Select Not,* quando o pino está *LOW*, o módulo começa a receber dados e processa-os na sua porta SPI;

- **SCK** -> Corresponde ao *Serial Clock,* que recebe impulsos de relógio fornecidos pelo SPI *Master*;

- **MOSI** -> Corresponde ao *Master Out Slave In,* que é a entrada SPI do módulo;

- **MISO** -> Corresponde ao *Master In Slave Out,* que é a saída SPI do módulo RF.

- **IRQ** -> É um pino de interrupção que alerta o *Master* quando há dados novos para processar.

1.1.2. Interface SPI

A interface SPI (*Serial Peripheral Interface*) é a utilizada para o estabelecimento da conexão entre a placa ESP32 e o módulo RF. Nesta interface, quer o *Master* quer o *Slave*, possuem internamente um registo de deslocamento que vão ser ligados entre si por 2 pinos previamente explicitados, o MOSI e o MISO como podemos ver na figura 2. A transferência dos bits é feita ao ritmo do sinal de relógio (SCK) gerado pelo *Master*.

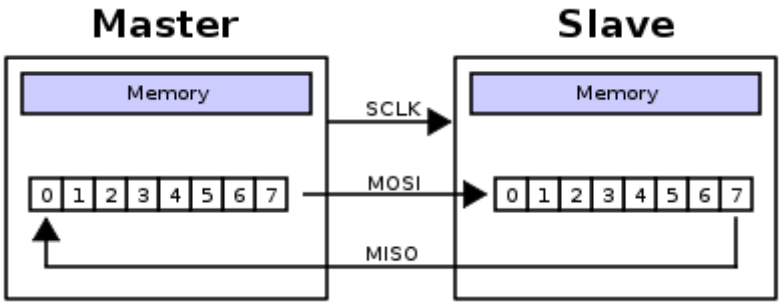

Figura 2 - Transferência de dados entre Master e Slave.

No caso da placa ESP32 as ligações da interface necessárias à implementação do protocolo SPI estão presentes na figura 3:

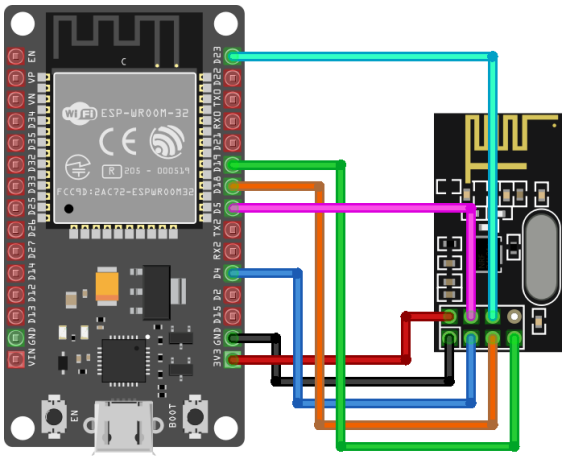

Figura 3 - Ligações ESP32-Módulo RF.

VCC -> **3.3V** na placa ESP32

GND -> **GND** na placa ESP32

CE -> **D4** na placa ESP32

CSN -> **D5** na placa ESP32

MOSI -> **D23** na placa ESP32

SCK -> **D18** na placa ESP32

MISO -> **D19** na placa ESP32

1.1.3. Parâmetros para a realização de testes

Na realização de testes vamos ter de levar em conta o atraso (*delay*), o *round-trip time* (RTT), o debito máximo (*throughput*) e o alcance de transmissão de modo a que não haja perda de informação ou erros na execução. *Delay* e *round-trip time* são atrasos que acontecem na emissão de dados, em que ambos são afetados diretamente pela distância da comunicação:

- ***Delay*** poderá ser provocado por questões de hardware, interferências na transmissão, por vezes provocados por software caso seja necessário e outras vezes devido ao processamento que os componentes tem de realizar.

- ***Round-trip time*** pela definição é o tempo que demora o emissor enviar uma mensagem ou algum tipo de pacotes de dados para um recetor e este mandar uma mensagem de confirmação de que recebeu o pacote. O valor do RTT varia consoante o tempo de transmissão, que depende do tamanho do pacote e da velocidade de transmissão. A confirmação da receção do pacote deverá demorar aproximadamente o mesmo tempo que o tempo de envio do pacote. **[2]**

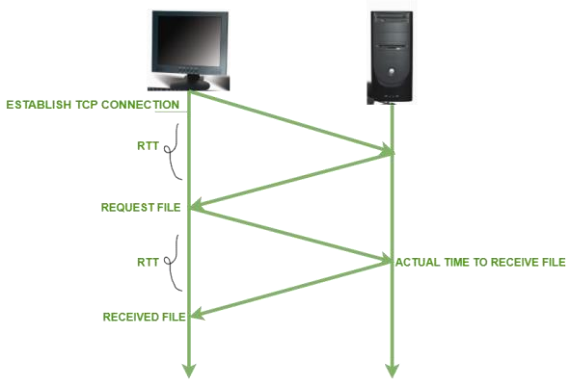

Figura 4 - Round-Trip Time.

- **Débito máximo(*throughput*)** refere-se à quantidade de dados que pode ser transferido entre dois dispositivos num intervalo de tempo. Este tempo depende do tipo e da qualidade de ligação realizada, para tal é necessária uma configuração correta dos ESP32 para que esta transferência possa ser a maior e mais rápida possível. **[3]**

- **Alcance de transmissão** é o limite de distância que a conexão permite uma ligação segura sem erros e sem perda de informação, reforçando que a configuração correta dos ESP32 e dos transceivers é bastante importante para que este alcance possa ser o melhor possível.

1.2. Desenvolvimento

Para esta fase do projeto foi necessária a elaboração de código para que o módulo ESP32 comunicasse eficientemente com os transceivers de radiofrequência. Para isso tivemos de utilizar a libraria RF24-master de modo a facilitar a configuração e a escrita do código. Em relação à configuração dos transceivers algumas linhas de código foram relevantes para a execução desta segunda fase do projeto, salientando as que usamos para desativar as funcionalidades pedidas pelos docentes.

Tabela 1 - Linhas de código relevantes

Instrução	Referência
RF24 radio(4, 5);	1
byte address[2][6] = {"1Node","2Node"};	2
radio.setChannel(60);	3
radio.openWritingPipe(address[0]);	4
radio.openReadingPipe(1, address[1]);	5
radio.disableCRC();	6
radio.setAutoAck(false);	7
radio.setPALevel(RF24_PA_LOW);	8
radio.setDataRate(RF24_250KBPS) ;	9
radio.startListening();	10
radio.write(&c, sizeof(c));	11
radio.read(&text,sizeof(text));	12

No que toca à configuração manual do *transceiver* começamos por colocar no código fora das funções loop() e setup() todas as instruções necessárias para o seu funcionamento começando por configurar as funções CE e CSN do transceiver (instrução 1) explicadas na página 9. Após isso definimos os endereços que nos permitem a comunicação entre os transceivers (instrução 2). Já dentro da função setup(), com o auxílio do *datasheet* do *transceiver* de RF **[5]** retiramos que para o RF24 a frequência dos canais vai de 2.4 GHz até 2.525 GHz sendo que a frequência utilizada pelo nosso grupo pode ser calculada através da expressão (1.1) .

$$f = 2400 + RF\_CH[MHz] \quad (1.1)$$

Como o RF que estamos a utilizar possui 126 canais (0-125), o nosso grupo optou por escolher o canal 60 em que através da fórmula 1.1 corresponde a uma frequência de 2460 MHz. Esta informação é dada ao transceiver através da instrução 3.

De modo a fazer a conversação entre os transceivers tivemos de abrir os canais para ler e escrever através das instruções 4 e 5 respetivamente.

Para esta fase do projeto necessitamos de desativar certas funcionalidades do módulo transceiver a pedido dos docentes, as quais são executadas nas instruções 6 e 7 onde desabilitamos as funções de controlo de erros. Na instrução 8 definimos a potência de transmissão que quanto maior for melhor será a comunicação a longas distâncias, no entanto de acordo com o *datasheet* é aconselhado usar menores valores de potência.

De modo a definir a velocidade de transmissão utilizamos a instrução 8 da tabela onde colocamos se queremos uma das 3 velocidades aceites pelo transceiver, 250Kbps, 1Mbps e 2 Mbps. As instruções 11 e 12 correspondem à escrita e leitura de pacote pelo emissor e recetor respetivamente. Na instrução 10 é onde o módulo RF começa a ouvir dos endereços de transmissão à espera de receber informação.

Na tabela 1 estão então representadas algumas linhas de código relevantes para a execução de 3 programas previamente pedidos pelos docentes, um de chat, um para o cálculo do RTT e outro para o cálculo do débito máximo.

Em seguida vamos apresentar os fluxogramas que permitem a compreensão do código elaborado. Em primeiro lugar o fluxograma do código relativo ao *chat* onde o emissor manda uma mensagem que aparece no *Serial Monitor* do recetor (Figura 5).

1.2.1. Fluxograma “Chat”

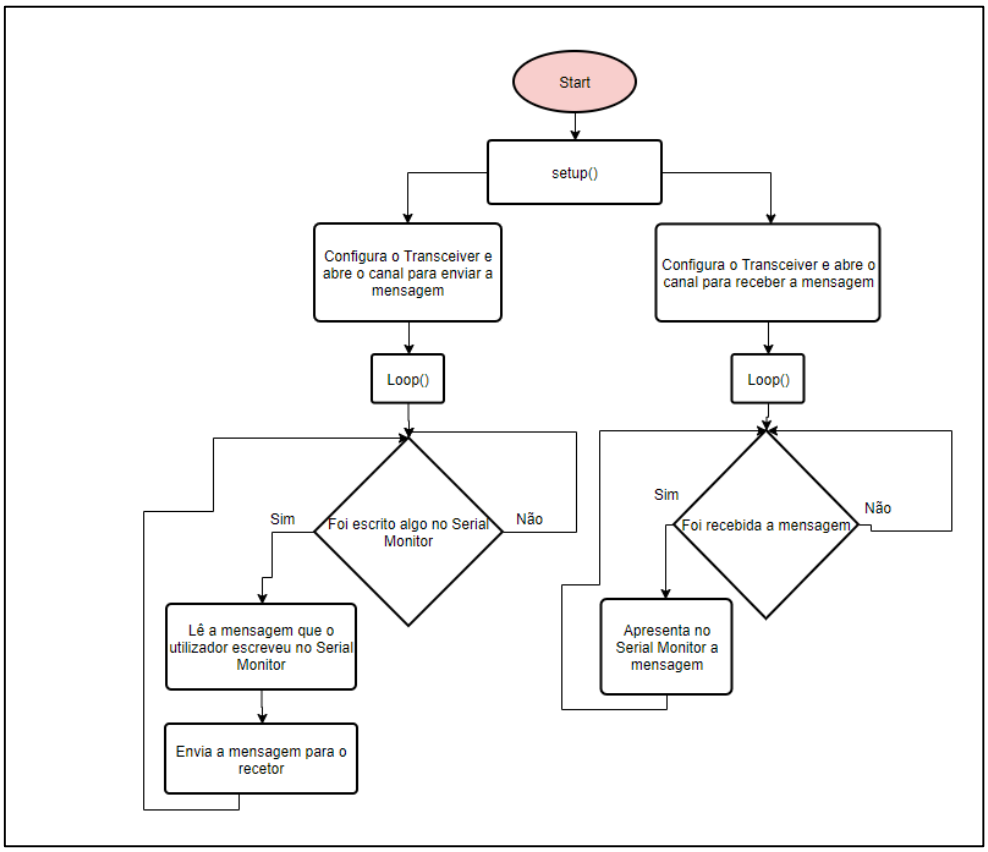

Figura 5 - Fluxograma "Chat".

1.2.2. Fluxograma “Round-Trip Time”

Os próximos 2 fluxogramas dizem respeito ao programa do cálculo do *round-trip time* em que o emissor envia uma mensagem ao recetor, este devolve uma mensagem de confirmação para que o emissor consiga calcular o tempo que leva a mandar a mensagem e receber a confirmação. Na figura 6 temos o fluxograma do emissor e na 7 do recetor.

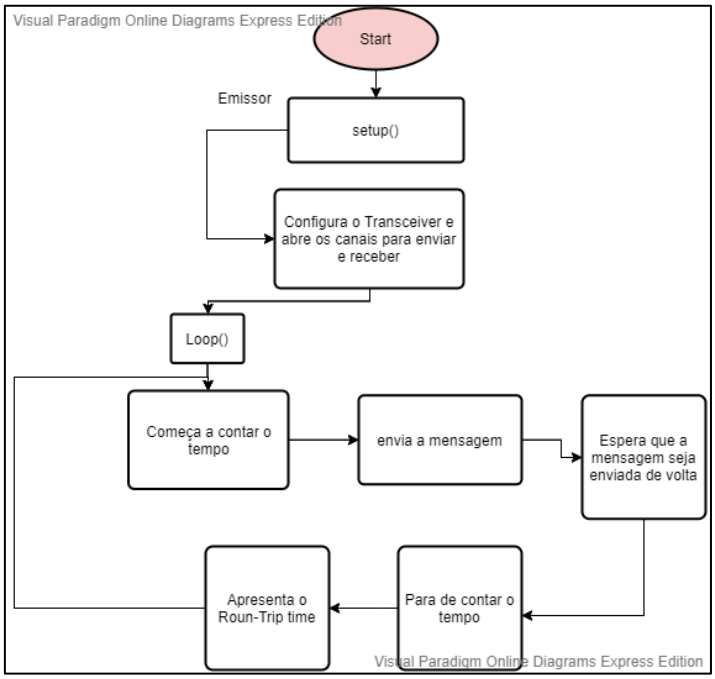

Figura 6 - Fluxograma do cálculo do RTT (Emissor).

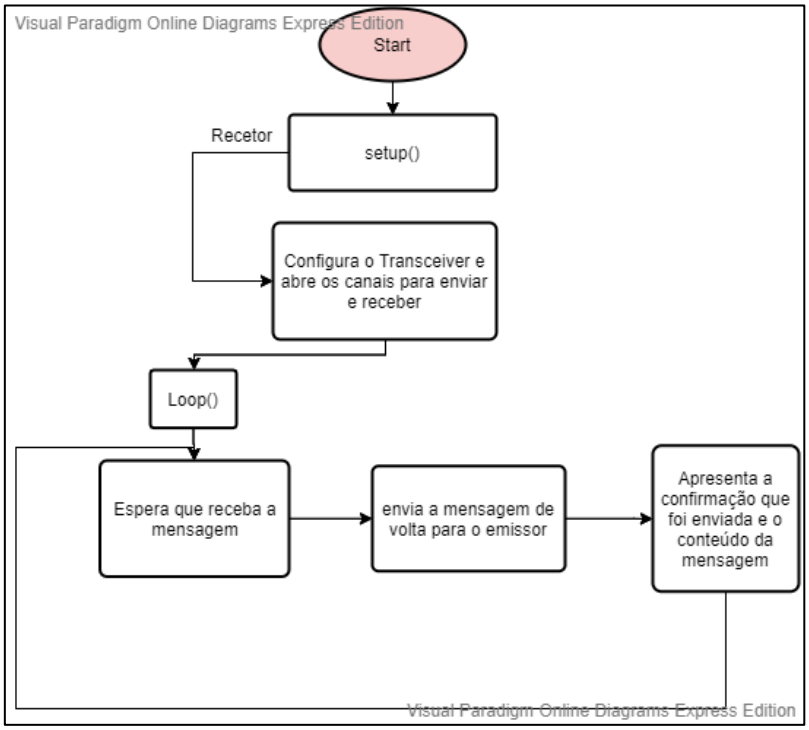

Figura 7 - Fluxograma do cálculo do RTT (Recetor).

1.2.3. Fluxograma “Débito e perdas”

Os últimos 2 fluxogramas que vão ser apresentados a seguir correspondem ao programa mais completo dos 3 que contém a informação sobre o número de pacotes que um recetor recebe de um dado emissor, a qualidade com que os pacotes chegam e o cálculo do débito e do PER. Na figura 9 está presente o fluxograma do código emissor e na 8 do recetor.

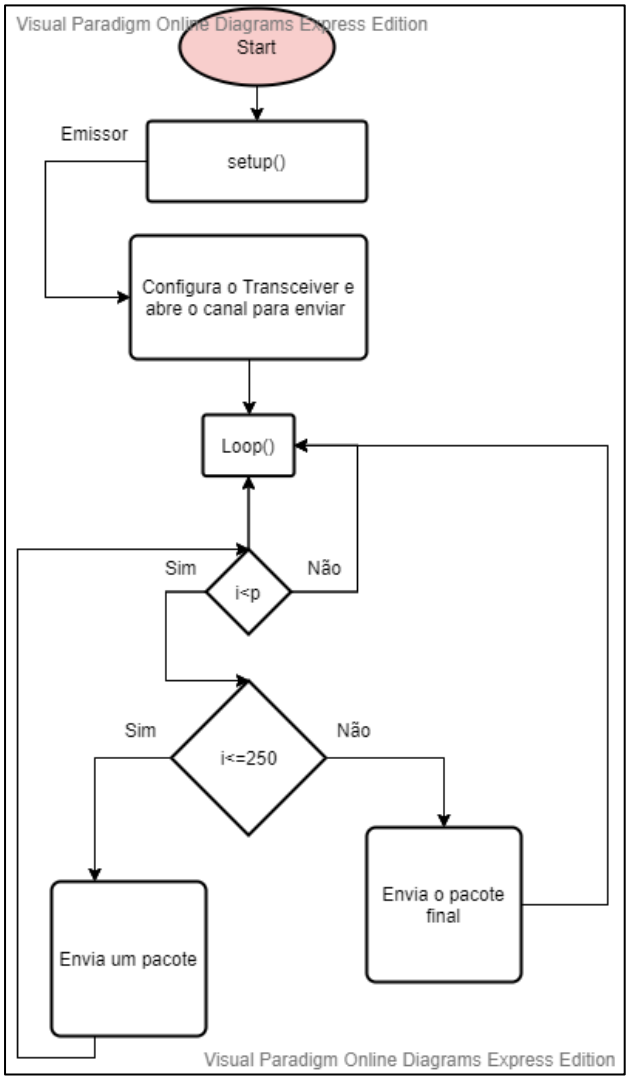

Figura 9 - Fluxograma "Débito" (Emissor).

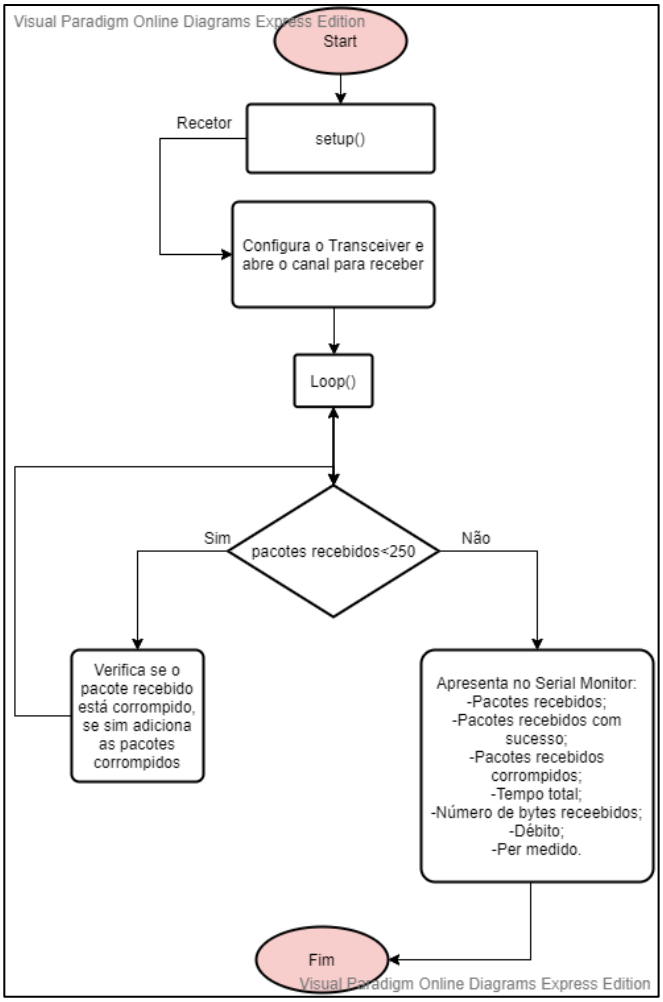

Figura 8 - Fluxograma "Débito" (Recetor).

1.3. Testes

1.3.1. RTT (Round Trip Time)

Como referido na página 11, o RTT é o tempo que demora o envio de um pacote de uma estação emissora até uma estação recetora mais o envio de uma mensagem da estação que recebeu o pacote. O tempo de envio do pacote e da mensagem de confirmação é aproximadamente igual.

Todos os testes para esta fase foram realizados no LAP3, no Campus de Azurém da Universidade do Minho pelo que algumas interferências relativas a aparelhos eletrónicos possam existir.

Para o cálculo do valor teórico do RTT utilizamos as seguintes expressões (1.2) onde é mostrada a forma de cálculo do tempo de transmissão e (1.3) onde calculamos o *round-trip time*:

$$T_{trans} = \frac{L}{R} \quad (1.2)$$

$$RTT = 2 \times T_{trans} \quad (1.3)$$

Tabela 2 - Resultados do Round-Trip Time.

RTT	Teórico (microssegundos)	Prático (microssegundos)
250Kbps	1024	1630
1Mbps	256	693
2Mbps	128	514

1.3.2. Alcance

Para o teste de alcance definimos um número máximo de 250 pacotes e fizeram-se vários testes para 3 distâncias (10 centímetros, 1 metro e 10 metros) de modo a determinar máximo e mínimo de pacotes recebidos. Para além disso conseguimos obter a informação da quantidade de pacotes que não foram recebidos devido a perdas e também os pacotes que chegaram, mas com defeito (corrompidos). Na tabela seguinte encontram-se o número de pacotes recebidos pelo recetor tendo em conta o fator distância e taxa de transmissão.

Tabela 3 - Resultados alcance.

Alcance	10 centímetros		1 metro		10 metros	
	Mínimo	Máximo	Mínimo	Máximo	Mínimo	Máximo
250 Kbps	250	250	249	250	229	231
1 Mbps	250	250	248	249	203	211
2 Mbps	250	250	237	239	190	192

1.4. Discussão de resultados

Em relação ao *Round-Trip Time* realizamos testes para 3 taxas de transmissão distintas, sendo estas de 250k, 1M e 2M bits por segundo. No código realizado, a parte do emissor calcula por métodos complementares à biblioteca do RF o tempo que demora a enviar uma mensagem e a receber a confirmação por parte do recetor. Começamos por calcular os tempos teóricos, sem haver interferências, que para 250Kbps deu um tempo de 1024 microssegundos, quando fomos a testar o nosso código os valores deram-nos bastante mais elevados, na ordem dos 1600 microssegundos, observamos que estas diferenças de tempo também acontecem para as outras taxas de transmissão e ocorreram devido às mais variadas interferências, por exemplo ruído, outros computadores e outros aparelhos tecnológicos presentes na sala onde realizamos os testes, mas também existem atrasos de software e hardware introduzidos pelos computadores e pela própria placa ESP. Estes valores de pacotes perdidos podem ser comparados na tabela 2.

Nos testes do alcance decidimos utilizar 3 distâncias para podermos comparar resultados. Na tabela 3 verificamos que quanta maior a distância menos pacotes são recebidos com sucesso, para obter os resultados tabelados decidimos realizar vários testes com diferentes taxas de transmissão e colocar na tabela os valores máximos e mínimos. O motivo que levou a que tais resultados não fossem os desejados foi que aumentando a distância existia um maior número de interferências fazendo com que o número de pacotes a chegar ao recetor não fosse o número de pacotes que saíram do emissor.

Os resultados obtidos para ambos os casos foram o expectável, no entanto no caso do RTT não estávamos à espera de obter uma diferença tão grande muito devido ao número de interferências que possamos ter encontrado ou então a um possível erro nos cálculos.

2. Controlo de Acesso ao meio

2.1. Introdução

Sabemos que o **MAC** (***Media Access Control***) pertence a segunda camada do modelo de OSI (camada de ligação), esta acede aos dados e controla as permissões dos mesmos para que não existam colisões de tramas ou pacotes. Isto torna-se necessário pois não existindo colisões de tramas a transmissão de dados poderá funcionar de maneira mais correta e fiável, como também com menos erros e ficheiros corrompidos.

2.2. Subprotocolos

Este protocolo possui uma variedade de subprotocolos tais como *Carrier Sense Multiple Access (CSMA), Carrier Sense Multiple Access with Collision Detection (CSMA/CD)* e *Token Passing.***[6]**

O *CSMA (Carrier Sense Multiple Access)* emite um sinal antes do envio para que este não tenha colisões no envio de dados. Este necessita de dois computadores para poder funcionar. Para realizar a comunicação entre computadores, um envia dados e o outro espera uma duração de tempo especifica para enviar os seus dados. Como tal esta regulação é lenta e adiciona dificuldades devido a necessidade de envio de sinal antes de envio de dados.

O *CSMA/CD (Carrier Sense Multiple Access with Collision Detection)* ao contrário do anterior não emite sinal para evitar colisões. *Collision detection* significa que quando há uma colisão detetada pelo MAC este interrompe por completo a transmissão de dados por um tempo aleatório antes de voltar a transmitir.

O *Token Passing* é utilizado pelo MAC para prevenir colisões. Apenas um computador que possua um espaço livre para a *Token*, este espaço é pequeno e esta autorizado a transmitir. Este é transmitido através de pequenos dados bastantes vezes. Este método é bastante eficiente em evitar colisões. Cada terminal terá um espaço reservado para o *token* se não existir uma grande prioridade de enviar dados.

Conclusão

Nesta fase concluímos que já nos é exigido mais conhecimento e mais pesquisa relativamente ao *hardware* utilizado e aos protocolos necessários. Foi necessária também uma pesquisa mais aprofundada para o desenvolvimento do código e dos *transceivers* visto que na fase anterior não foi necessária a sua utilização.

Com esta nova pesquisa entendemos a importância de protocolos e a ligação com a cadeira de Redes de Computadores. Foi necessário o conhecimento geral do grupo sobre a matéria em causa. Com a cooperação de todos conseguimos nos ajudar e ultrapassar as dificuldades que nos foram aparecendo.

A utilização dos *transceivers* foi realizada com muito sucesso e conseguimos atingir todos os objetivos da primeira parte, as comunicações foram bem realizadas e com uma distância satisfatória.

Numa fase final ainda tentamos realizar a segunda parte, no entanto não conseguimos terminar. Apesar de acharmos que foi bem conseguida a parte obrigatória da segunda fase, não conseguimos terminar nem implementar a cem porcento a fase opcional.

Para concluir, achamos que fizemos um bom trabalho apesar das advertências atuais e conseguimos uma boa coordenação dentro do grupo, o que nos deixa com mais vontade e mais empenhados em continuar com o bom trabalho.

Referências

1. https://lastminuteengineers.com/nrf24l01-arduino-wireless-communication/
2. https://searchnetworking.techtarget.com/definition/round-trip-time
3. https://techterms.com/definition/throughput
4. https://maniacbug.github.io/RF24/classRF24.html
5. https://www.sparkfun.com/datasheets/Components/SMD/nRF24L01Pluss_Preliminary_Product_Specification_v1_0.pdf
6. https://www.getkisi.com/blog/media-access-control

Autoavaliação

Inês Barreira Marques: Nesta fase do projeto ajudei na pesquisa da parte teórica, escrita do relatório, na execução dos testes e na correção de alguns erros de código.

José Pedro Fernandes Peleja: Na fase 2 do projeto fiz pesquisa teórica, ajudei na escrita de relatório e na revisão do mesmo. Participei na realização de testes.

Rui Filipe Ribeiro Freitas: Para esta fase contribui na elaboração do código assim como na realização dos testes. Ajudei também na pesquisa e escrita do relatório bem como na sua revisão.

Tiago João Pereira Ferreira: Nesta fase contribui para a elaboração dos diferentes códigos e relatório. Participei também nos testes.