



Universidade do Minho  
Escola de Engenharia

MESTRADO INTEGRADO EM ENGENHARIA DE TELECOMUNICAÇÕES E  
INFORMÁTICA

LABORATÓRIOS DE TELECOMUNICAÇÕES E INFORMÁTICA I

DESENVOLVIMENTO DE UMA APLICAÇÃO DE *chat*

RELATÓRIO FASE 2

GRUPO 4



Bruno Oliveira  
A81570



Filipe Brás  
A81307



João Cunha  
A76645



José Bravo  
A80132

Guimarães, 27 de novembro de 2020

# Índice



<b>1</b>	<b>Lista de Acrónimos e Siglas</b>	<b>4</b>
<b>2</b>	<b>Introdução</b>	<b>5</b>
<b>3</b>	<b>Fundamentos</b>	<b>6</b>
<b>4</b>	<b>Enquadramento da Fase 2</b>	<b>8</b>
<b>5</b>	<b>Planeamento de Tarefas</b>	<b>9</b>
<b>6</b>	<b>Tarefa 2.1 - Comunicação por RF Arduino-TRF-TRF-Arduino</b>	<b>10</b>
6.1	Características das interfaces de comunicação . . . . .	11
6.2	Ligações físicas necessárias entre os componentes utilizados . . . . .	13
6.3	Impacto no desempenho do sistema das configurações do <i>transciever</i> RF . . . . .	15
6.4	Configuração, transmissão e receção de dados dos <i>transcievers</i> RF . . . . .	17
<b>7</b>	<b>Tarefa 2.2 – Controlo de acesso ao meio</b>	<b>19</b>
7.1	TDMA - Time Division Multiple Access . . . . .	20
<b>8</b>	<b>Avaliação de desempenho do sistema</b>	<b>21</b>
8.1	Avaliações Teóricas . . . . .	21
8.2	Avaliações Práticas . . . . .	22
<b>9</b>	<b>Conclusão</b>	<b>24</b>
9.1	Autoavaliação . . . . .	25
<b>10</b>	<b>Referências</b>	<b>27</b>

## Lista de Figuras

1	Diagrama de Gantt relativo à Fase 2. . . . .	9
2	Tarefas atribuídas a cada membro do grupo. . . . .	9
3	Arquitetura geral do sistema para a Fase 2. . . . .	10
4	Transmissão e receção de dados com o protocolo SPI.[11] . . . . .	11
5	Transmissão de dados utilizando a modulação FSK.[12] . . . . .	12
6	Transmissão de dados utilizando a modulação GFSK.[12] . . . . .	12
7	Circuito com as ligações físicas efetuadas. . . . .	14
8	Configuração do transceiver RF. . . . .	15
9	Teste com taxa de transmissão de 250KBPS. . . . .	16
10	Teste com taxa de transmissão de 1MBPS. . . . .	16
11	Teste com taxa de transmissão de 2MBPS. . . . .	16
12	Estrutura da trama. . . . .	17
13	Estrutura da trama implementada no módulo Arduino. . . . .	17
14	Configuração da trama implementada no módulo Arduino. . . . .	18
15	Array a ser utilizado no payload. . . . .	18
16	Função implementada para cálculo do CRC. . . . .	18
17	Configuração do transceiver RF. . . . .	19
18	Frequency Division Multiple Access . . . . .	20
19	Time Division Multiple Access . . . . .	20
20	Gráfico com a evolução das tramas recebidas. . . . .	22
21	Gráfico com a evolução das tramas perdidas. . . . .	23

## Lista de Tabelas

1	Pinos do módulo <i>transciever</i> RF nRF24L01+. . . . .	13
2	Testes de alcance . . . . .	22

# 1. Lista de Acrónimos e Siglas


- API - Application Programming Interface
- CIPO - Controller-In/Peripheral-Out
- COPI - Controller-Out/Peripheral-In
- GFSK - Gaussian Frequency-Shift Keying
- ISM - Industrial, Scientific and Medical
- LLC - Logic Link Control
- MAC - Medium Access Control
- OSI - Open System Interconnection [ISO 2012]
- RF - Radiofrequência
- SCK - Serial Clock
- SPI - Serial Peripheral Interface
- WLAN - Wireless Local Area Network [IEEE 802.11]

## 2. Introdução


No âmbito da Unidade Curricular de **Laboratórios de Telecomunicações e Informática I** foi proposto a realização de um trabalho que tem como objetivo final o desenvolvimento e a implementação de uma aplicação de conversação (*chat*). Neste relatório iremos abordar apenas a Fase 2 do projeto prático.


Para que se possa executar o projeto com sucesso, anteriormente, terá que ser feita uma investigação acerca do módulo de desenvolvimento **ESP32** e do módulo *transciever* **nRF24L01+**, e o funcionamento dos mesmos. Além disso, será necessário, o desenvolvimento e compreensão de conhecimentos de *software* e a realização de um plano de tarefas do projeto.

Como o projeto envolve algumas áreas do conhecimento, como as redes de computadores, pois assenta no **Modelo OSI** (Open System Interconnection) e incide nas camadas **física** (nível 1), **de ligação de dados** (nível 2) e de **aplicação** (nível 7).

Após a aquisição do conhecimento necessário e o planeamento de tarefas, o grupo estará em condições de começar a executar o projeto. 

Uma vez que se trata de um projeto complexo e ser o primeiro contacto com um projeto com tanto detalhe como este esperamos encontrar algumas dificuldades.


A principal dificuldade que esperamos enfrentar será a nível da comunicação, uma vez que, teremos de definir um protocolo de comunicação para todas as tecnologias a serem utilizadas de forma a permitir uma transferência viável de informação. Outra dificuldade será em desenvolver o protocolo de acesso ao meio que teremos de implementar no Arduino. 

Certamente existirão mais dificuldades que serão encontradas pelo caminho como por exemplo a interface gráfica para a aplicação ou até algo tão simples como este relatório. Apesar disso esperamos ultrapassar estas dificuldades e apresentar um trabalho final completo. 

### 3. Fundamentos

Nesta secção, o relatório irá incidir na síntese dos fundamentos teóricos necessários à compreensão e execução da Fase 2.

- **Camada de Ligação de Dados** - Camada 2 do modelo OSI, que utiliza os serviços da camada física para enviar e receber dados através de canais de comunicação, e executam as funções de controlo de ligação lógica e controlo de acesso ao meio[2].
- **LLC - Logic Link Control** - Funcionalidade que permite controlar a troca de informação desde a ligação física à camada de rede. Esta funcionalidade divide-se em 4 aplicações, que são: a delimitação das tramas, o controlo de erros, o controlo de fluxo e o endereçamento das estações envolvidas.
- **MAC - Medium Access Control** - Funcionalidade que controla os períodos em que cada estação pode transmitir, sem que haja colisões na transmissão dos sinais.
- **Protocolo SPI - Serial Peripheral Interface** - Protocolo de barramento de interfaces, que é responsável por transmitir informação entre microcontroladores (como o microcontrolador ESP-WROOM-32D) e pequenos periféricos (como o *transceiver* RF nRF24L01+)[7].
- **Interface API - Application Programming Interface** - Interface de programação, cujo objetivo é permitir que dois sistemas independentes comuniquem entre si. Desta forma, a API define funcionalidades que são independentes das suas implementações, assim, essas funcionalidades ajustam-se a qualquer cenário de programação[11].
- **Redes WLAN - Wireless Local Area Network [IEEE 802.11]** - Rede local sem fios *peer-to-peer*, ou, rede local *ad-hoc*, que ao contrário das redes locais com fios, não requer uma infraestrutura de ligação como, por exemplo, um *router*. Apenas necessita de, pelo menos, duas estações dentro do mesmo alcance e, após, a configuração das estações é possível comunicarem as duas[5].
- **Comunicação por RF** - Consiste na emissão de informação previamente codificada e modulada num sinal eletromagnético que se propaga através do espaço, sendo posteriormente captada pelo recetor, responsável pela decodificação destes sinais. O sinal eletromagnético é originado, a partir de uma oscilação de um eletrão num local, e essa oscilação determina a frequência de onda e, consequentemente, o comprimento de onda do sinal.

- **Modulação GFSK - Gaussian Frequency Shift-Keying** - Modulação que codifica os dados na forma de variações de frequência numa onda portadora em função do *bit* que é transmitido.[12] 
- **TDMA- Time Division Multiple Access** - Método de acesso por divisão de tempo, permitindo a diferentes utilizadores a comunicação através do mesmo canal de comunicação, com uma gama de frequência comum, sendo que cada utilizador dispõe de um período de tempo para efetuar a comunicação.





## 4. Enquadramento da Fase 2



A Fase 2 do projeto prático irá incidir na comunicação entre a placa de desenvolvimento ESP-32 e o módulo *transciever* RF nRF24L01+ (Arduino-TRF e TRF-Arduino), e entre os módulos *transciever* RF nRF24L01+(TRF-TRF) e a implementação de um protocolo de acesso ao meio.



A tarefa 2.1 tem como objetivos principais a comunicação por RF entre **Arduino-TRF-TRF-Arduino** e o controlo da ligação lógica. Para tal, será necessário a construção de uma arquitetura do sistema bem definida, o desenvolvimento de competências ao nível do *hardware*, como efetuar as ligações físicas necessárias, e ao nível de *software*, como o desenvolvimento do código na linguagem C++ no IDE Arduino, para a configuração e, transmissão e receção de dados dos módulos *transcievers* RF nRF24L01+.



A tarefa 2.2, sendo de realização facultativa, não é de menor importância, visto que aborda o controlo de acesso ao meio. Nesta tarefa, o objetivo será a implementação de um protocolo de acesso ao meio, de forma, a que não haja colisões na transmissão das tramas e, que cada estação possa transmitir num determinado período, usando para tal o método de acesso ao meio por divisão de tempo (**TDMA- Time Division Multiple Access**).

Relativamente ao desenvolvimento de código, nos módulos Arduino, começamos por desenvolver uma aplicação cujo objetivo é a transferência de informação entre os dois módulos *transcievers* nRF24L01+, configurando os parâmetros relativos á camada física e desabilitando os mecanismos para deteção e correção de erros disponibilizados pelo *transciever* RF. Desenvolvemos também uma aplicação de conversação em tempo real entre os mesmos.

## 5. Planeamento de Tarefas

Esta secção irá incidir no planeamento de tarefas da Fase 2. Um correto planeamento de tarefas é necessário, para que haja uma melhor distribuição de trabalho, organização e eficácia. Abaixo encontra-se o planeamento de tarefas da Fase 2.

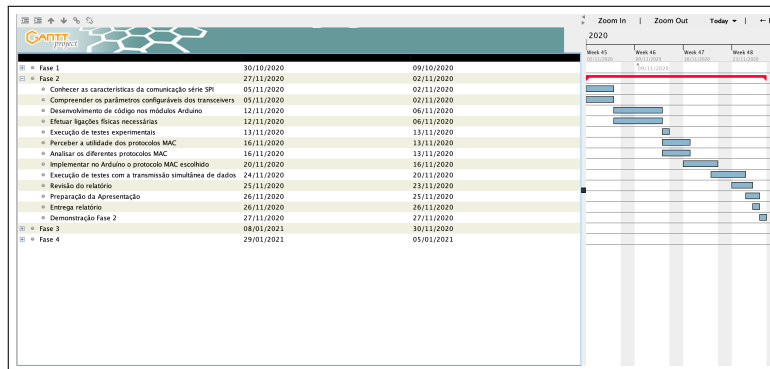


Figura 1: Diagrama de Gantt relativo à Fase 2.

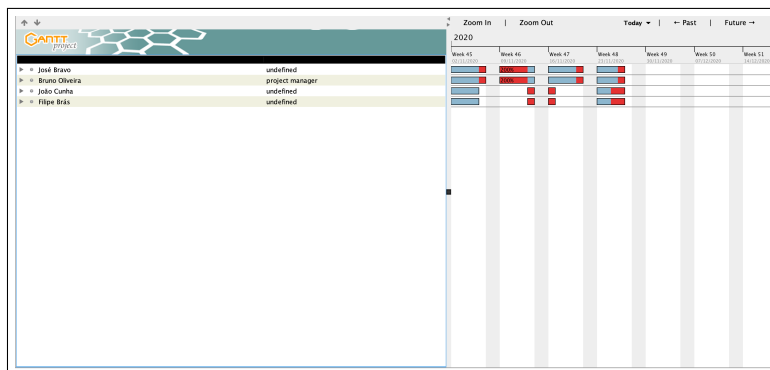


Figura 2: Tarefas atribuídas a cada membro do grupo.

## 6. Tarefa 2.1 - Comunicação por RF Arduino-TRF-TRF-Arduino

Antes da implementação e execução da tarefa 2.1 é necessário a definição de uma arquitetura de sistema, de forma, a conhecer as características das interfaces de comunicação entre cada *transceiver* RF nRF24L01+ e a respetiva placa de desenvolvimento ESP32. Na figura abaixo está a perspetiva geral da arquitetura do sistema para a Fase 2.

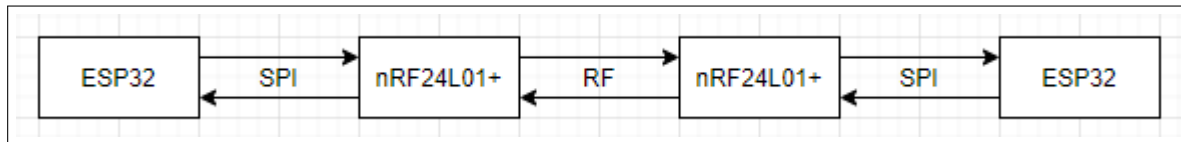



Figura 3: Arquitetura geral do sistema para a Fase 2.

Nesta fase, a arquitetura do sistema é composta por duas placas de desenvolvimento ESP32 e dois módulos *transceiver* RF nRF24L01+, e as interfaces de comunicação são o protocolo SPI e a comunicação por RF.

A comunicação de dados *será* feita de forma *wireless* e no modo *full-duplex*.

## 6.1. Características das interfaces de comunicação

Esta secção pode ser dividida em duas partes, visto que nesta fase do projeto, o grupo implementou duas interfaces de comunicação diferentes, para as ligações **ESP32 - nRF24L01+** e **nRF24L01+ - nRF24L01+**.

Para a ligação **ESP32 - nRF24L01+**, a interface de comunicação implementada é o protocolo SPI, que é um protocolo de **barramento de interfaces** síncrono. Por oposição ao protocolo UART (utilizado na Fase 1), o protocolo SPI faz a transmissão de dados através de um sinal de *clock* (SCK - Serial Clock) gerado pela placa ESP32, um sinal de envio de dados (**COPI - Controller-Out/Peripheral-In**), e para a receção de dados, a placa ESP32 continua a gerar ciclos do sinal SCK para que o módulo nRF24L01+ possa enviar informação pelo sinal de dados **CIPO (Controller-In/Peripheral-Out)**, como se pode comprovar com a figura abaixo. [10] 

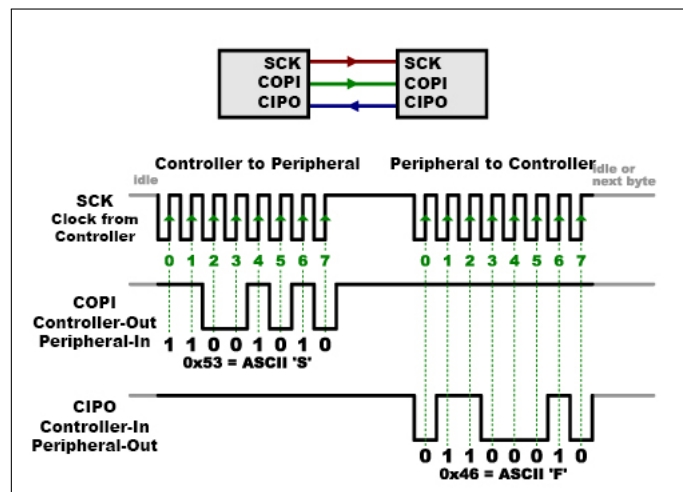




Figura 4: Transmissão e receção de dados com o protocolo SPI. [11] 

De notar que, apesar de acima **estar** documentado **os** sinais de dados como COPI e CIPO, no *software* Arduino IDE e no módulo nRF24L01+ estes termos são descritos como MOSI (Master-Out/Slave-In) e MISO (Master-In/Slave-Out).

O protocolo SPI **utiliza o sinal de *clock* para modular** os sinais de MISO e MOSI. Para a implementação do protocolo SPI foi utilizada a *library* **SPI.h**, disponibilizada no *software* Arduino IDE. 

Para a ligação **nRF24L01+** - **nRF24L01+**, a interface de comunicação é a comunicação por radiofrequência, que consiste na emissão e receção de informação codificada e modulada num sinal eletromagnético. De forma, a implementar a comunicação por RF foi utilizada a *library* **RF24.h**, disponibilizada no *software* Arduino IDE.

Para que possa transmitir, o módulo nRF24L01+ foi desenhado para operar na banda de frequências ISM de 2,4 GHz e utiliza a modulação GFSK, que é uma extensão da modulação FSK (Frequency Shift-Keying). Abaixo encontram-se a figura que exemplifica a transmissão de dados com a modulação FSK e a figura que exemplifica a transmissão de dados com a modulação GFSK, respetivamente.

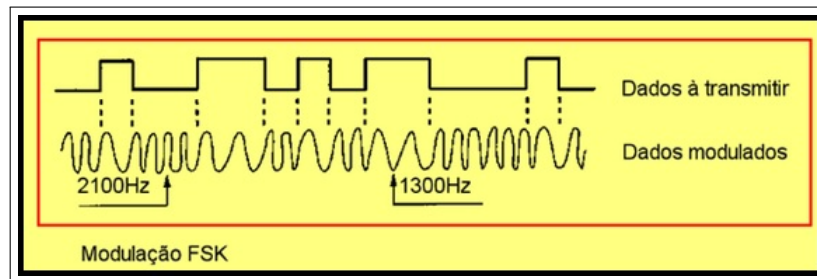



Figura 5: Transmissão de dados utilizando a modulação FSK.[12]



Figura 6: Transmissão de dados utilizando a modulação GFSK.[12]

## 6.2. Ligações físicas necessárias entre os componentes utilizados

De forma, a que fosse possível o envio de tramas, com uma ligação *peer-to-peer* foi necessário a compreensão e análise do *pinout* da placa de desenvolvimento ESP32 e o módulo *transciever* RF nRF24L01+, bem como, os respetivos *datasheets*.

O módulo *transciever* RF nRF24L01+ é um módulo *transciever* implementado com o protocolo *ShockBurst*, adequado para aplicações wireless. Vem equipado com uma antena embutida que opera numa gama de frequências ISM de 2.4 GHz a 2.525GHz, com uma velocidade máxima de operação de 2 Mbps, um alcance que pode chegar aos 10 metros em ambiente interiores e 50 metros em ambientes exteriores, respetivamente. O módulo vem equipado com 10 pinos de interligação. Caracteriza-se pelo baixo consumo de energia e pela alta velocidade de comunicação, que pode chegar a 2 Mbps. Devido ao uso da interface SPI, é possível interligar este módulo com a maioria dos microcontroladores. Uma outra grande vantagem, é a capacidade, de poder atuar como emissor, recetor ou transceiver, dependendo apenas da configuração de software. A sua tensão de alimentação pode variar entre 1,9 V a 3,6 V e os pinos de sinal podem trabalhar normalmente com nível de sinal de 5 V, utiliza modulação GFSK (Gaussian Frequency-Shift Keying), capacidade anti-interferência, verificação de erros, comunicação *peer-to-peer* de 126 canais e controlo de fluxo. 

Pino	Nome	Função	Ligação ESP32
1	Vcc	Alimentação	3.3 V
2	Vcc	Alimentação	3.3 V
3	CE	Chip Enable RX/TX	Pino 0
4	CSN	SPI Chip Select	Pino 5
5	SCK	SPI Clock	Pino 18
6	MOSI	SPI Slava Data Input	Pino 23
7	MISO	SPI Slave Data Output	Pino 19
8	IRQ	Interrupção	Não utilizado
9	GND	Terra	GND
10	GND	Terra	GND

Tabela 1: Pinos do módulo *transciever* RF nRF24L01+.



Em relação à ligação TRF-TRF, não existem ligações físicas a executar, visto que, a comunicação é feita sem fios. Abaixo encontra-se a figura relativa às ligações físicas necessárias.

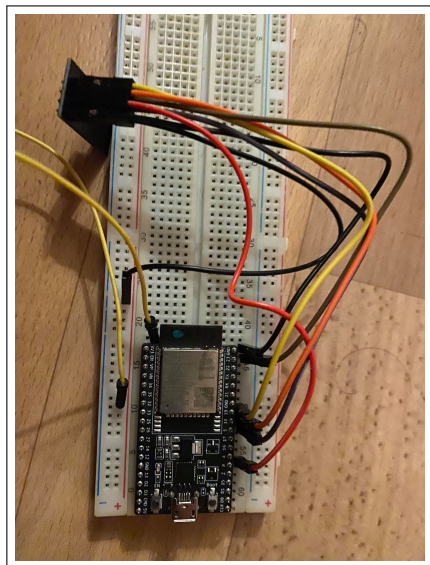


Figura 7: Circuito com as ligações físicas efetuadas.

### 6.3. Impacto no desempenho do sistema das configurações do *transciever* RF

O módulo *transciever* nRF24L01+ opera na banda de frequência de 2.4GHz e utiliza modulação GFSK ( Gaussian Frequency Shift-Keying) para a transmissão de dados, que pode ter velocidades de transmissão de 250kbps, 1Mbps ou 2Mbps. A transmissão e receção de dados é efetuada para uma determinada gama de frequência (canal), sendo que para dois módulos *transceivers* RF comunicarem, necessitam de utilizar o mesmo canal. Os canais possíveis correspondem às frequências dentro da gama de 2.4GHz - 2.525GHz, pelo que existem 125 canais possíveis.

De modo a efetuar a configuração dos módulos *transceivers* RF bem como a sua interação, recorreremos à *library* **RF24.h** que disponibiliza diferentes métodos que permitem definir os parâmetros acima mencionados. No entanto, para a implementação do controlo de acesso ao meio recorreremos também à *library* **RF24Network.h**, explicado no decorrer do presente relatório.

Nesta fase, decidimos utilizar o canal 115, com frequência de 2515 MHz, uma velocidade de transmissão de dados de 2 Mbps e definimos o nível do amplificador de potência máximo, conforme ilustrado na figura seguinte, onde *myRadio* representa o módulo *transciever* RF.

```
myRadio.begin();  
myRadio.setChannel(115);  
myRadio.setPALevel(RF24_PA_MAX);  
myRadio.setAutoAck(false);  
myRadio.setDataRate( RF24_2MBPS );
```

Figura 8: Configuração do transciever RF.

A seguir são apresentados diferentes testes experimentais realizados para diferentes taxas de transmissão de dados possíveis de configurar através da função *setDataRate()* disponível a partir da *library* **RF.h**.



```

void setup() {
  Serial.begin(115200);
  myRadio.begin();
  myRadio.setChannel(115);
  myRadio.setPALevel(RF24_PA_MAX);
  myRadio.setAutoAck(false);
  myRadio.setDataRate( RF24_250KBPS );
}

```

```

20:16:50.655 -> Recebido
20:16:50.655 -> Round-Trip Time: 1974 microsegundos
20:16:50.692 ->
20:16:50.692 -> Enviado
20:16:50.692 -> Pacote:997
20:16:50.692 -> CRC: 80
20:16:50.692 -> Recebido
20:16:50.692 -> Round-Trip Time: 1974 microsegundos

```

Figura 9: Teste com taxa de transmissão de 250KBPS.



```

void setup() {
  Serial.begin(115200);
  myRadio.begin();
  myRadio.setChannel(115);
  myRadio.setPALevel(RF24_PA_MAX);
  myRadio.setAutoAck(false);
  myRadio.setDataRate( RF24_1MBPS );
}

```

```

20:13:31.290 -> Recebido
20:13:31.290 -> Round-Trip Time: 1013 microsegundos
20:13:31.290 ->
20:13:31.290 -> Enviado
20:13:31.290 -> Pacote:997
20:13:31.290 -> CRC: 80
20:13:31.290 -> Recebido
20:13:31.290 -> Round-Trip Time: 1017 microsegundos

```

Figura 10: Teste com taxa de transmissão de 1MBPS.

```

void setup() {
  Serial.begin(115200);
  myRadio.begin();
  myRadio.setChannel(115);
  myRadio.setPALevel(RF24_PA_MAX);
  myRadio.setAutoAck(false);
  myRadio.setDataRate( RF24_2MBPS );
}

```

```

20:09:26.377 -> Recebido
20:09:26.377 -> Round-Trip Time: 866 microsegundos
20:09:26.410 ->
20:09:26.410 -> Enviado
20:09:26.410 -> Pacote:997
20:09:26.410 -> CRC: 79
20:09:26.410 -> Recebido
20:09:26.410 -> Round-Trip Time: 866 microsegundos

```

Figura 11: Teste com taxa de transmissão de 2MBPS.



Como podemos observar para diferentes taxas de transmissão de dados obtemos **temos** de diferentes **de round-trip time**, ou seja, **devemos levar em conta esta configuração para obtermos o melhor desempenho possível**. Outra configuração que pode ter impacto no desempenho do sistema é a amplificação da potência através da função `setPALevel()`, disponível a partir da biblioteca **RF24.h**, sendo que para transmissões de longo alcance é sugerido configurar esta para o valor máximo possível.



## 6.4. Configuração, transmissão e receção de dados dos *transcievers* RF

Os dados a transmitir têm por base uma estrutura de 32 *bytes*, constituída por três campos: - *número de sequência* cujo tamanho é de 2 *bytes* e representa o número do pacote a enviar; - *payload* cujo tamanho é de 29 *bytes* e corresponde ao campo dos dados; - *CRC* cujo tamanho é de 1 *byte*, e servirá como mecanismo de deteção de erros na transmissão.

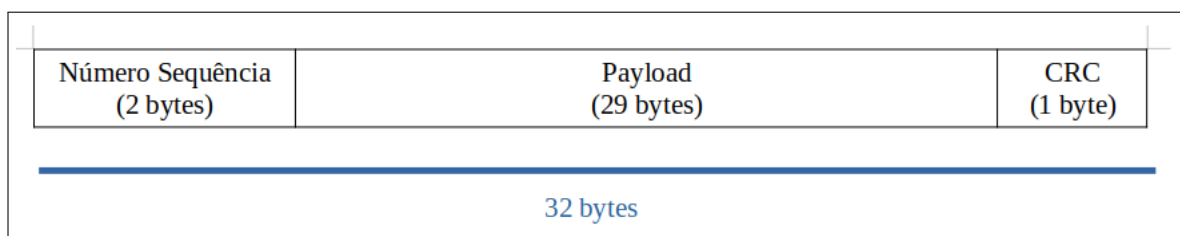


Figura 12: Estrutura da trama.

A seguinte estrutura foi implementada e configurada da seguinte forma no módulo **Arduino**.

```
struct package {  
    uint16_t seq;  
    byte payload[29];  
    uint8_t crc;  
};
```

Figura 13: Estrutura da trama implementada no módulo Arduino.

```

for (counter = 1; counter <= 1000; counter++){
  if(myRadio.available() > 0 ){
    while(myRadio.available()){
      myRadio.read(&aux_ack, sizeof(aux_ack));
      Serial.print(aux_ack);
    }
    elapsedss_time = micros() - start_time;
    Serial.print("\nRound-Trip Time: ");
    Serial.print(elapsedss_time);
    Serial.println(" microssegundos");
  }

  delay(10);
  myRadio.stopListening();
  data.seq = counter;
  memcpy(data.payload, array_file, sizeof(char)*29);
  strncpy(array_file, array_file + 29, sizeof(array_file) - 29);
  data.crc = genCRC((uint8_t*)&data.payload, sizeof(data.payload));
  Serial.print("\nEnviado");
  Serial.print("\nPacote:");
  Serial.print(data.seq);
  Serial.print("\n");
  //Serial.println((char*)data.payload);
  Serial.print("CRC: ");
  Serial.println(data.crc);

  start_time = micros();
  myRadio.openWritingPipe(addresses[0]);
  myRadio.write(&data, sizeof(data));
  myRadio.openReadingPipe(1, addresses[1]);
  myRadio.startListening();
}

```

Figura 14: Configuração da trama implementada no módulo Arduino.

Nesta estrutura o número de sequência irá ser o valor do pacote a enviar, sendo que o payload é obtido através de uma sequência aleatórias de letras que foram usadas para substituir um ficheiro, ou seja, obtemos 29000 bytes de letras aleatórias.

```

//Preencher Array p/ enviar --> Na proxima fase este array vai ser o ficheiro
for (int i = 0; i <= 29000; i++){
  array_file[i] = 'a' + (rand()%26);
}

```

Figura 15: Array a ser utilizado no payload.

Por fim obtemos o CRC através da função *genCRC()*, também implementada no arduino, para ocupar o último byte da trama.

```

uint8_t genCRC(uint8_t *data, size_t len){
  uint8_t crc = 0xff;
  size_t i, j;
  for (i = 0; i < len; i++) {
    crc ^= data[i];
    for (j = 0; j < 8; j++) {
      if ((crc & 0x80) != 0){
        crc = (uint8_t)((crc << 1) ^ 0x31);
      }else{
        crc <<= 1;
      }
    }
  }
  return crc;
}

```

Figura 16: Função implementada para cálculo do CRC.

## 7. Tarefa 2.2 – Controlo de acesso ao meio

Tendo em conta que o canal de comunicação é partilhado, caso exista a necessidade de criar uma rede entre diferentes módulos *transciever* RF nRF24L01+, torna-se necessário estabelecer regras para coordenar o acesso ao meio efetuado pelos diferentes módulos. Para tal utilizamos o mecanismo (**TDMA- Time Division Multiple Access**) que permite que cada ponto desta rede transmita informação durante um determinado período de tempo. Cada módulo *transciever* tem a capacidade de comunicar com módulos diferentes, pelo que, para a criação de uma rede entre diversos módulos, é gerada uma descendência entre os diferentes módulos, conforme ilustrado na figura seguinte, podendo-se obter uma rede entre 3125 módulos *transciever* diferentes. Todos os módulos possuem um endereço próprio que irá ser utilizado para definir o destino de qualquer transmissão de dados efetuada.

Nesta fase, implementamos uma rede entre os dois módulos *transciever*, configurando os parâmetros necessários como o canal de comunicação a utilizar e o intervalo entre transmissões de dados. Esta implementação permite que, apenas módulos já definidos possam comunicar entre si.

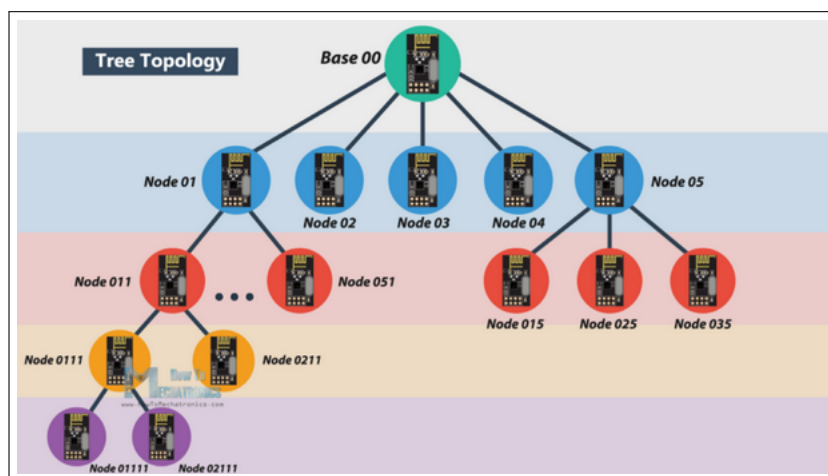


Figura 17: Configuração do transciever RF.



## 7.1. TDMA - Time Division Multiple Access

O TDMA ocupa um canal do módulo *transciever nRF24L01+* que permite a dois utilizadores diferentes ocuparem o mesmo canal em períodos de tempo diferentes, isto difere do método FDMA (Frequency Division Multiple Access) que apenas permite dois utilizadores em canais diferentes cada um com uma frequência, e apenas consegue enviar ou receber nesse tempo em que o canal está a ser utilizado.

Depois deste estudo optamos por utilizar o TDMA pois como o processo é rápido, alguns milissegundos, os utilizadores não perceberão esta noção de tempo, para haver mais utilizadores é necessário existir outros canais.[13]

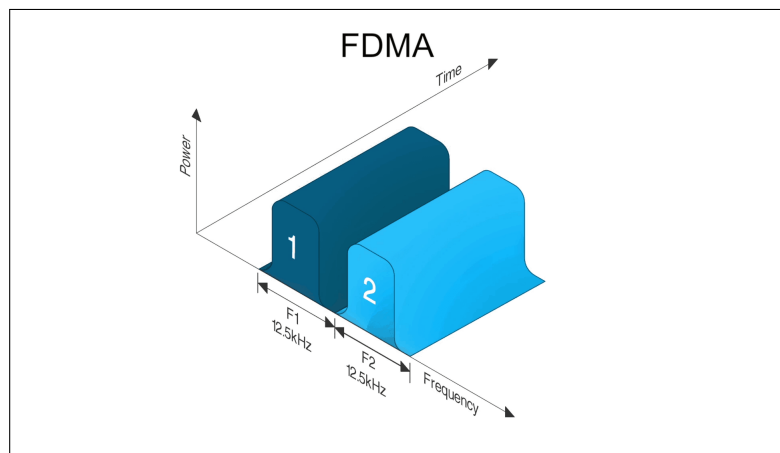


Figura 18: Frequency Division Multiple Access

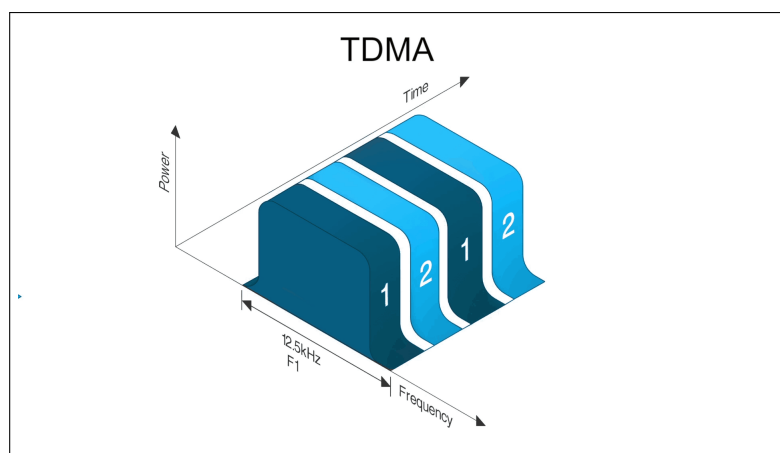


Figura 19: Time Division Multiple Access

## 8. Avaliação de desempenho do sistema

### 8.1. Avaliações Teóricas

Esta subsecção destina-se à demonstração dos cálculos teóricos para as condições ideais.

Os parâmetros que podem afetar o desempenho do sistema são o tamanho do ficheiro ( $L_f$ ) em *bits*, a taxa de transmissão ( $R$ ) em bps, o tamanho da trama ( $L$ ) em *bits*, o tamanho do *payload* ( $L_p$ ) em *bits*, o débito útil ( $S_g$ ) em bps, o tempo de transmissão de dados ( $T_{dados}$ ) em segundos, a distância ( $d$ ) em metros, a velocidade da luz ( $c$ ), o tempo de processamento ( $T_{proc}$ ) com o valor de  $58 \mu s$ , o RTT(Round Trip Time) ( $T_{rtt}$ ) em segundos, o tempo total ( $T_t$ ), o tamanho do ACK ( $T_{ack}$ ) e o tempo de propagação ( $T_p$ ) em segundos, que não vai ser considerado, visto que tem um valor teórico bastante pequeno.

$$T_{rtt} = T_{dados} + T_{ack} + T_{proc} \quad \text{🗨️} \quad (1)$$

$$T_{dados} = T_{trans_{spi1}} + T_{trans_{ar12}} + T_{trans_{ar21}} + T_{trans_{spi2}} \quad (2)$$

$$T_{dados} = \frac{(32 * 8)bits}{10Mbps} + \frac{32 * 8bits}{10Mbps} + \frac{32 * 8bits}{10Mbps} + \frac{(32 * 8)bits}{10Mbps} = 563,2\mu s \quad \text{🗨️} \quad (3)$$

$$T_{ack} = \frac{(10 * 8)bits}{10Mbps} + 10 * 8bits + 10 * 8bits + \frac{(10 * 8)bits}{10Mbps} = 176\mu s \quad (4)$$

$$T_{rtt} = 563,2\mu s + 176\mu s + 58\mu s = 797,2\mu s \quad (5)$$

Sendo que a taxa útil de transmissão é:

$$U = \frac{T_{rtt}}{T_{total}} = \frac{797,2\mu s}{860\mu s} = 92,7\% \quad \text{🗨️} \quad (6)$$

$$Throughput = \frac{L_{pacote}}{T_{rtt}} = \frac{32 * 8bits}{860\mu s} = 297.674Kbps \quad \text{🗨️} \quad (7)$$

## 8.2. Avaliações Práticas

Esta subsecção incide na demonstração e exposição dos diferentes testes e avaliações ao sistema, variando os diversos fatores que o podem alterar o resultado final.

A distância enquanto, grandeza física, tem a capacidade de alterar o desempenho do sistema e criar perdas de informação e possíveis erros. Abaixo encontra-se a tabela de testes de alcance dependendo da distância entre cada estação.

Distância	Tramas enviadas	Tramas recebidas	Tramas perdidas
10 metros	1000	1000	0
20 metros	1000	1000	0
30 metros	1000	998	2
40 metros	1000	986	14
50 metros	1000	973	27
100 metros	1000	928	72

Tabela 2: Testes de alcance

Visto que, a distância faz variar o sistema, à medida que a distância aumenta o número de tramas recebidas diminui de forma, exponencial, como se pode comprovar com o gráfico seguinte.

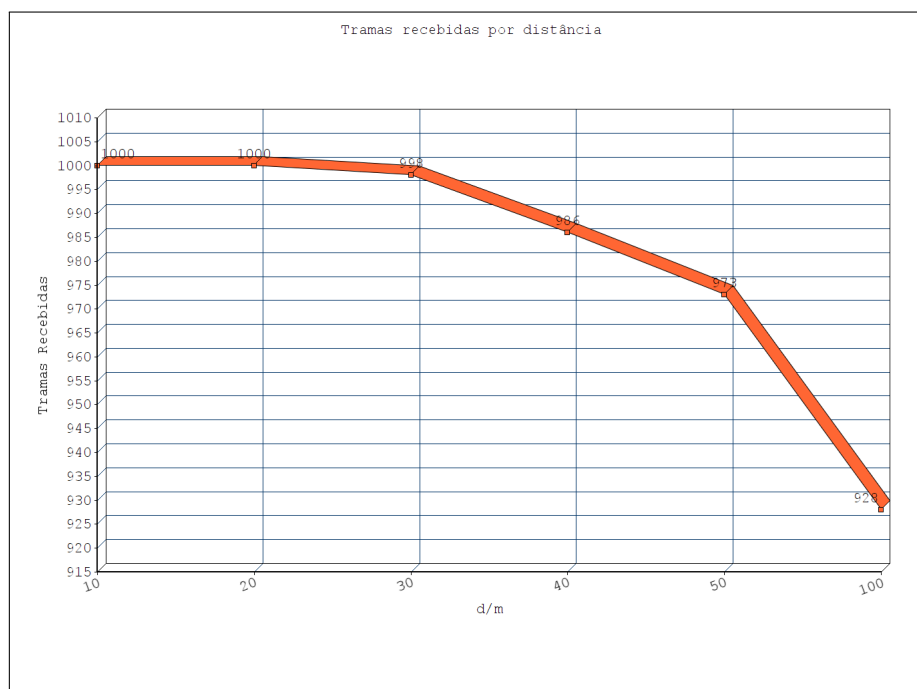


Figura 20: Gráfico com a evolução das tramas recebidas.

Por outro lado, o número de tramas perdidas aumenta de forma exponencial, à medida que a distância aumenta, como se pode comprovar com o gráfico seguinte.

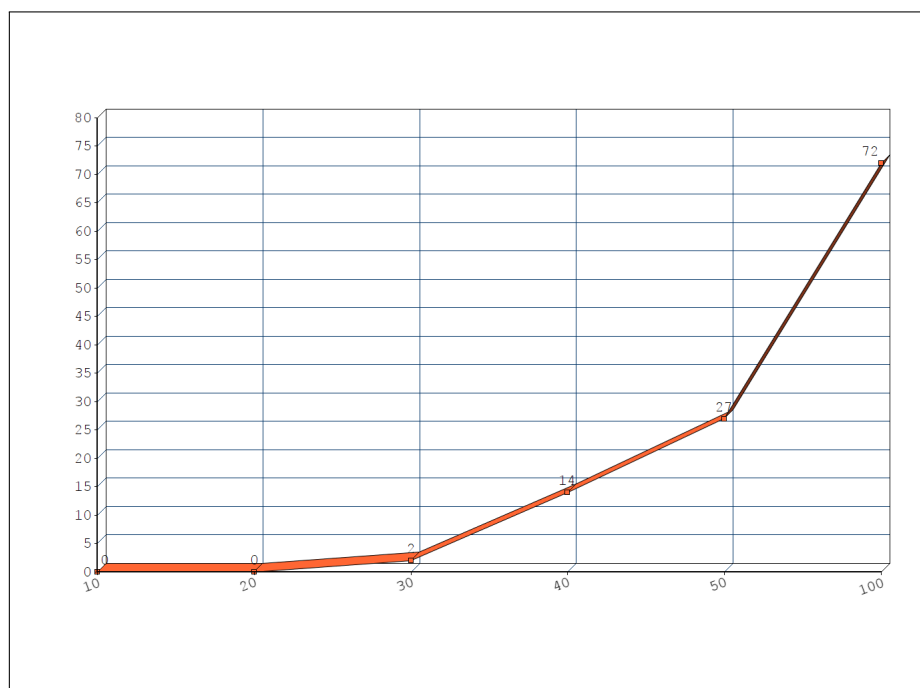


Figura 21: Gráfico com a evolução das tramas perdidas.



## 9. Conclusão

A realização desta fase permitiu ao grupo obter mais conhecimento relativo às comunicações por radiofrequência e como podem ser estruturados os dados para envio/receção, permitiu uma maior noção do impacto no desempenho do sistema causado por diferentes configurações dos módulos **nRF24L01+**, como também compreender as situações em que se justifica a implementação de mecanismos de controlo de acesso ao meio. É de realçar que a execução de testes experimentais, foi fundamental para uma análise crítica dos resultados obtidos.

O grupo acabou por ter algumas dificuldades no desenvolvimento do código, mais em concreto na definição das tramas, também acresceu mais a dificuldade devido ao IDE utilizado não ser tão habitual mas acreditamos que o grupo ultrapassou as suas dificuldades.

## 9.1. Autoavaliação

- Bruno Oliveira

Nesta fase estive envolvido na elaboração do código de envio e recepção de dados aleatórios e também na elaboração do código para o chat. Realizei os testes experimentais e ajudei na escrita do relatório e pré-relatório desta fase. A maior dificuldade encontrada foi no desenvolvimento de código para a estrutura das tramas.

- Filipe Brás

Nesta fase estive envolvido na elaboração e edição do relatório de especificação desta fase e, deste mesmo relatório. Também estive envolvido no cálculo dos testes teóricos de avaliação do sistema. Apesar, de algumas dificuldades, penso que o resultado obtido pelo grupo foi o esperado com todas as tarefas propostas realizadas.

- João Cunha

Nesta fase estive envolvido no trabalho de pesquisa e na projeção do trabalho. Ajudei também na edição do relatório e nos problemas que iam aparecendo no código. Esta fase já não foi tão fácil como a primeira, apareceram muitos mais problemas mas acho que conseguimos um bom resultado tendo realizado todas as tarefas.

- José Bravo

Nesta fase estive envolvido na elaboração do código implementado de envio e receção de dados, na elaboração do código para o chat e na implementação do controlo de acesso ao meio. Realizei testes experimentais e ajudei na elaboração do relatório. A maior dificuldade foi no desenvolvimento do código de transmissão de dados.

## 10. Referências

- [1] Computer Networking: A Top-Down Approach; 6th Edition; Kurose, James F.; Ross, Keith W.  
[online] Disponível em: [https://eclass.teicrete.gr/modules/document/file.php/TP326/%CE%98%CE%B5%CF%89%CF%81%CE%AF%CE%B1%20\(Lectures\)/Computer\\_Networking\\_A\\_Top-Down\\_Approach.pdf](https://eclass.teicrete.gr/modules/document/file.php/TP326/%CE%98%CE%B5%CF%89%CF%81%CE%AF%CE%B1%20(Lectures)/Computer_Networking_A_Top-Down_Approach.pdf)  
[Acedido a 27 de novembro de 2020].
- [2] Computer Networks; 5th Edition; Tanenbaum, Andrew S.; Wetherall, David J.  
[online] Disponível em: <http://index-of.es/Varios-2/Computer%20Networks%205th%20Edition.pdf>  
[Acedido a 27 de novembro de 2020]
- [3] Mobile Communications; 2nd Edition; Schiller, Jochen  
[online] Disponível em: [https://www.academia.edu/7693415/Mobile\\_Communication](https://www.academia.edu/7693415/Mobile_Communication)  
[Acedido a 27 de novembro de 2020]
- [4] Computer Engineering: A Dec View of Hardware Systems Design; Bell, C. Gordon; Mudge, J. Craig; McNamara, John E.  
[online] Disponível em: [http://bitsavers.org/pdf/dec/\\_Books/Bell-ComputerEngineering.pdf](http://bitsavers.org/pdf/dec/_Books/Bell-ComputerEngineering.pdf)  
[Acedido a 27 de novembro de 2020]
- [5] Wireless Communications and Networks; 2nd Edition; Stallings, William  
[online] Disponível em: [https://vulms.vu.edu.pk/Courses/CS431/Downloads/Wireless\\_Communications\\_-\\_Networking\\_Stallings\\_2nd.pdf](https://vulms.vu.edu.pk/Courses/CS431/Downloads/Wireless_Communications_-_Networking_Stallings_2nd.pdf)  
[Acedido a 27 de novembro de 2020]
- [6] Informação acerca da interface SPI  
[online] Disponível em <https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all>  
[Acedido a 27 de novembro de 2020]
- [7] Informação acerca da interface API

[online] Disponível em: <https://blogs.mulesoft.com/biz/tech-ramblings-biz/what-are-apis-how-do-apis-work/>

[Acedido a: 27 de novembro de 2020]

[8] Placa de Desenvolvimento ESP32

[online] Disponível em: <https://www.botnroll.com/en/esp/3131-esp32-devkitc-32d-f-development-kit-esp32.html>

[Acedido a 27 de novembro de 2020]

[9] Módulo *transciever* de RF nRF24L01+

[online] Disponível em: <https://www.botnroll.com/pt/rf-lora/833-modulo-nrf24l01.html>

[Acedido a 27 de novembro de 2020]

[10] Informação acerca da interface SPI

[online] Disponível em: <https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all>

[Acedido a 24 de novembro de 2020]

[11] Imagem acerca da interface SPI



[online] Disponível em: [https://cdn.sparkfun.com/assets/learn\\_tutorials/1/6/BasicSPI\\_Updated.jpg](https://cdn.sparkfun.com/assets/learn_tutorials/1/6/BasicSPI_Updated.jpg)

[Acedido a 24 de novembro de 2020]

[12] Informação acerca da modulação GFSK

[online] Disponível em: <https://sites.google.com/site/nearcommunications/modulacao-gfsk>

[Acedido a 25 de novembro de 2020]

[13] Informação acerca dos mecanismos FDMA e TDMA

[online] Disponível em: <https://www.taitradioacademy.com/topic/the-difference-between-fdma-and-tdma-1/>

[Acedido a 26 de novembro de 2020]