



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 1

GRUPO 4

Guimarães, 29 de outubro de 2020

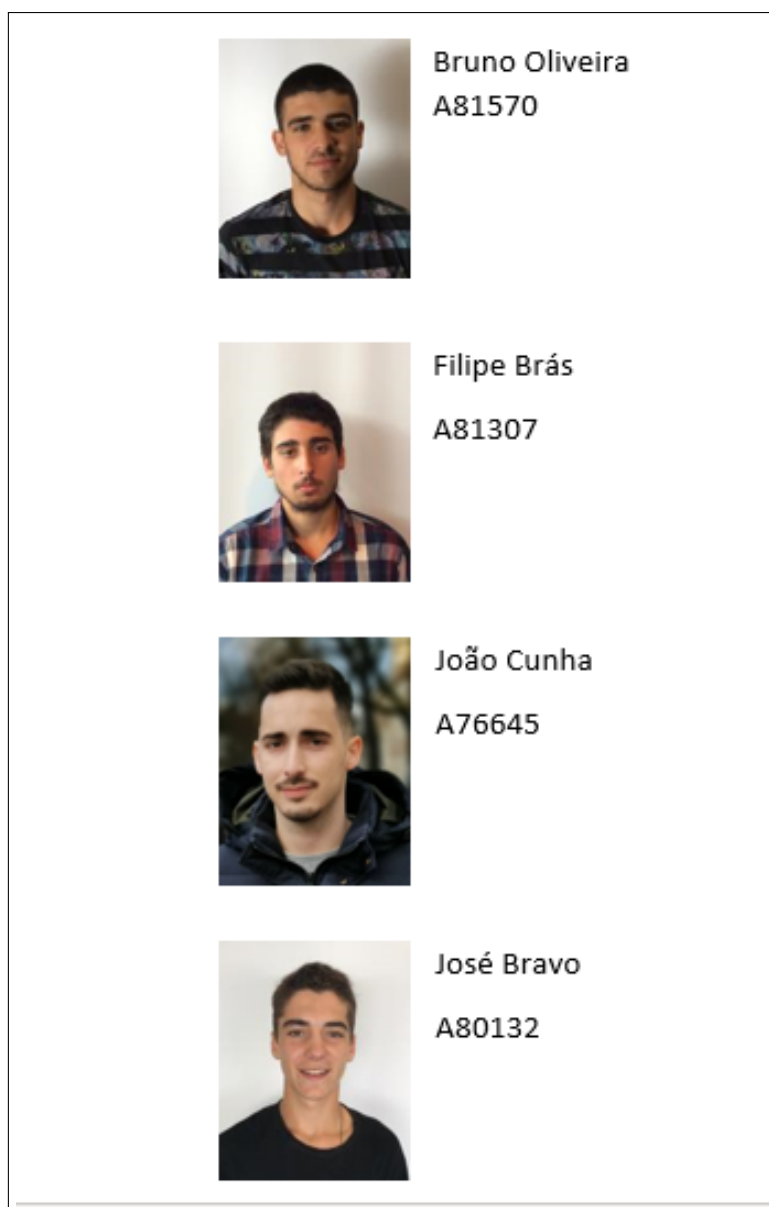
Índice

1	Elementos do Grupo	3
2	Lista de Acrónimos e Siglas	4
3	Introdução	5
4	Fundamentos	6
5	Enquadramento do Projeto	8
6	Tarefa 1.1 - Especificação do Projeto	9
7	Tarefa 1.2 - Comunicação via porta série PC-Arduino-Arduino-PC	13
7.1	Características das interfaces de comunicação série	13
7.2	Parâmetros de configuração das interfaces de comunicação série	14
7.3	Ligações físicas entre os componentes utilizados	15
7.4	Transferência de um ficheiro de texto e de um ficheiro de imagem	16
8	Teste, Resultados e Discussão	17
9	Conclusão	18
9.1	Autoavaliação	18
10	Referências	20

Lista de Figuras

1	Funcionamento de fluxo de dados do protocolo UART.	7
2	Funcionamento de fluxo de dados do protocolo RS-232.	7
3	Placa de desenvolvimento ESP32.	8
4	Módulo <i>transciever</i> RF nRF24L01+.	8
5	Arquitetura de rede do projeto.	9
6	Diagrama de Gantt relativo à Fase 1.	10
7	Tarefas atribuídas a cada membro do grupo.	10
8	Descrição da placa de desenvolvimento ESP32.	11
9	Módulo <i>transciever</i> RF nRF24L01+.	12
10	Interfaces utilizadas no PC-Arduino-Arduino-PC na Fase 1.	13
11	Estrutura da trama.	14
12	Parâmetros utilizados na configuração da porta série.	14
13	Ligação física implementada na Fase 1.	15
14	Teste com <i>baudrate</i> a 9,600 Kbps.	17
15	Teste com <i>baudrate</i> a 115,200 Kbps.	17

1. Elementos do Grupo



2. Lista de Acrónimos e Siglas

- API - Application Programming Interface
- ISM - Industrial, Scientific and Medical
- ISO - International Organization for Standardization
- LLC - Logic Link Control
- MAC - Medium Access Control
- OSI - Open System Interconnection [ISO 2012]
- RF - Radiofrequência
- RS-232 - Recommended Standard 232
- SPI - Serial Peripheral Interface
- UART - Universal Asynchronous Receiver/Transmitter
- WLAN - Wireless Local Area Network [IEEE 802.11]

3. 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 1 do projeto.

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 e termos de o implementar no código do Arduino.

Certamente existirão mais difuldades 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.

4. Fundamentos

Nesta secção, o relatório irá incidir na síntese dos fundamentos teóricos necessários à compreensão e execução da Fase 1.

- **Modelo OSI - Open System Interconnection [ISO 2012]** - Modelo criado pela ISO (International Organization for Standardization), com o intuito de organizar as redes de computadores em 7 camadas (Camada Física, Camada de Ligação de Dados, Camada de Rede, Camada de Transporte, Camada de Sessão, Camada de Apresentação e Camada de Aplicação). Este projeto incide diretamente, apenas, nas camadas física, de ligação de dados e de aplicação.
- **Camada Física** - Camada 1 do modelo OSI, ou seja, a camada mais rudimentar, que provém o sistema de ajuda funcional, mecânica e elétrica para manter e transmitir *bits* em conexões físicas.⁴
- **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.
- **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 *transciever* RF nRF24L01+).⁷

- **Protocolo UART - Universal Asynchronous Receiver/Transmitter** - Protocolo de comunicação de dados assíncrona, que ao contrário do protocolo SPI e do protocolo I^2C (Inter-Integrated Circuit), é apenas um circuito integrado, cujo objetivo é transmitir e receber informação em série assíncrona. A figura abaixo representa o fluxo de dados do protocolo UART.⁸

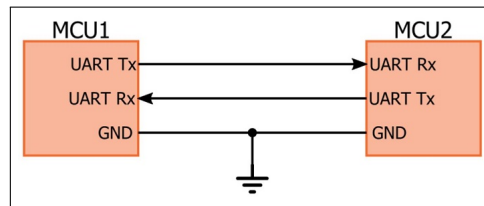


Figura 1: Funcionamento de fluxo de dados do protocolo UART.

- **Protocolo RS-232 - Recommended Standard 232** - Protocolo de comunicação de dados em série, que é utilizado para conectar o DTE (Data Transmitter Equipment) e o DCE (Data Communication Equipment), de forma, a que haja um maior fluxo de dados em série entre o PC e os respectivos periféricos. A figura abaixo representa o fluxo de dados do protocolo RS-232.⁹

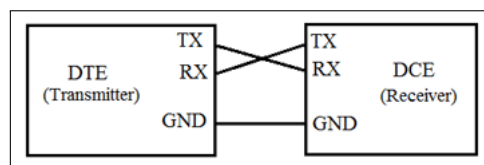


Figura 2: Funcionamento de fluxo de dados do protocolo RS-232.

- **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
- **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 descodificação 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, conseqüentemente, o comprimento de onda do sinal.

5. Enquadramento do Projeto

Este projeto tem como objetivo o desenvolvimento de uma aplicação de comunicação que possibilite a conversação em modo texto e a transferência de ficheiros, de qualquer formato, entre dois dispositivos do sistema de comunicação. A conversação será efetuada através de uma rede local sem fios por radio-frequência (RF) permitindo assim uma comunicação *full-duplex*, ou seja, bidirecional, fiável e simultânea entre duas ou mais estações. Cada estação será constituída por um **PC (Personal Computer)**, uma **placa de desenvolvimento ESP-32** e um **módulo *transceiver* RF nRF24L01+**. Entre as diversas competências necessárias, é de realçar a importância da aprendizagem e capacidade de implementação de protocolos das camada 1, camada 2 e camada 7 do modelo OSI e as funcionalidades inerentes a cada camada, o conhecimento das diferentes interfaces de comunicação série com periféricos, aprofundar as competências do trabalho em equipa e autónomo, tornando-se assim crucial manter uma boa organização entre todos os elementos do grupo. Abaixo, encontra-se as figuras relativas à placa de desenvolvimento ESP32 e o módulo *transceiver* RF nRF24L01+, respetivamente.

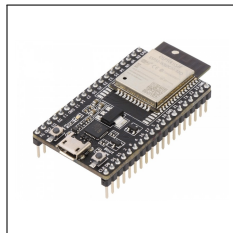


Figura 3: Placa de desenvolvimento ESP32.

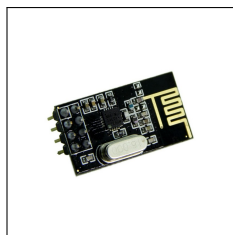


Figura 4: Módulo *transceiver* RF nRF24L01+.

6. Tarefa 1.1 - Especificação do Projeto

Quanto à arquitetura de rede, o sistema de comunicação tem como base, uma rede local sem fios por RF (Radiofrequência), possibilitando, assim, a comunicação bidirecional e fiável entre duas ou mais estações, ou seja, a rede local é do tipo P2P(*peer-to-peer*). Cada estação é composta por um PC, uma placa de desenvolvimento **ESP-32** e o um módulo *transciever* RF **nRF24L01+**.

Por outro lado, as funcionalidades a serem oferecidas serão, inicialmente, a conversação em modo texto, em tempo real, e, posteriormente, a transferência de ficheiros, como texto, imagem, etc. entre as estações.

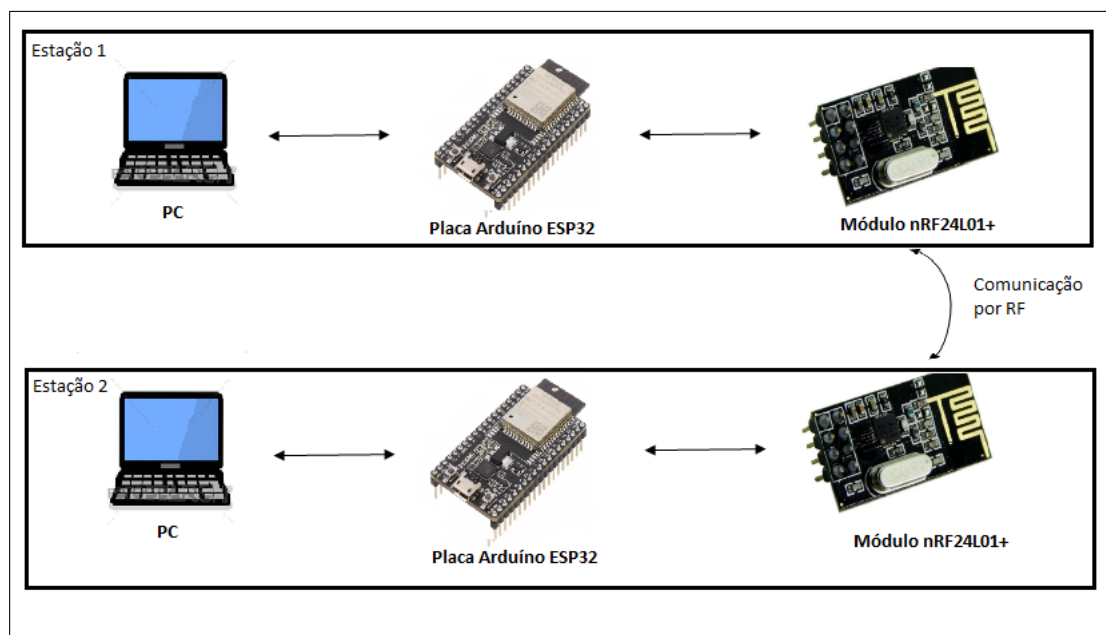


Figura 5: Arquitetura de rede do projeto.

Quanto ao planeamento geral de tarefas, o grupo optou por utilizar o *software* **Gantt Project** para se organizar em relação às tarefas. Abaixo encontra-se o Diagrama de Gantt relativo à Fase 1 do projeto prático.

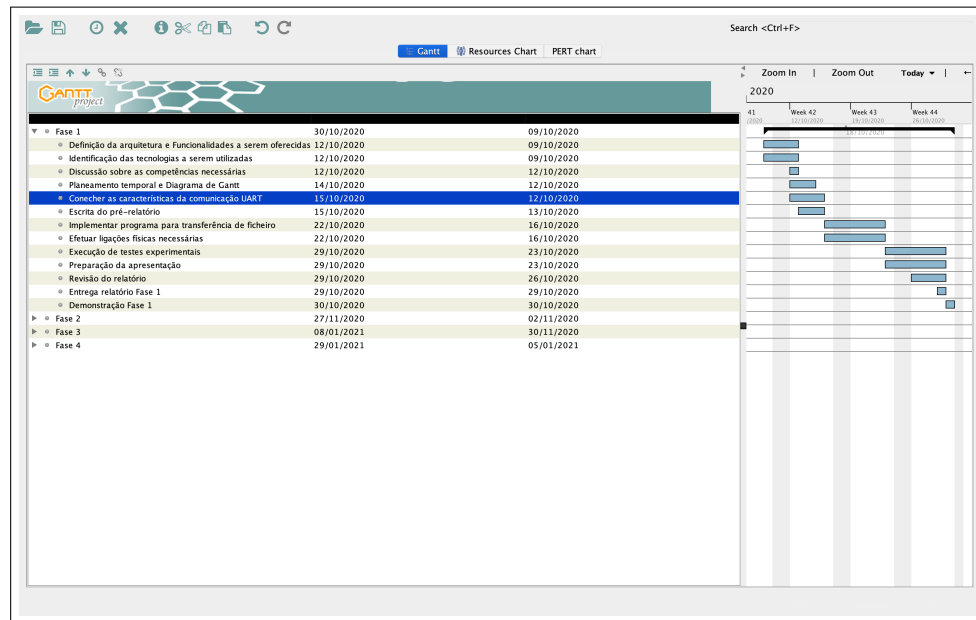


Figura 6: Diagrama de Gantt relativo à Fase 1.

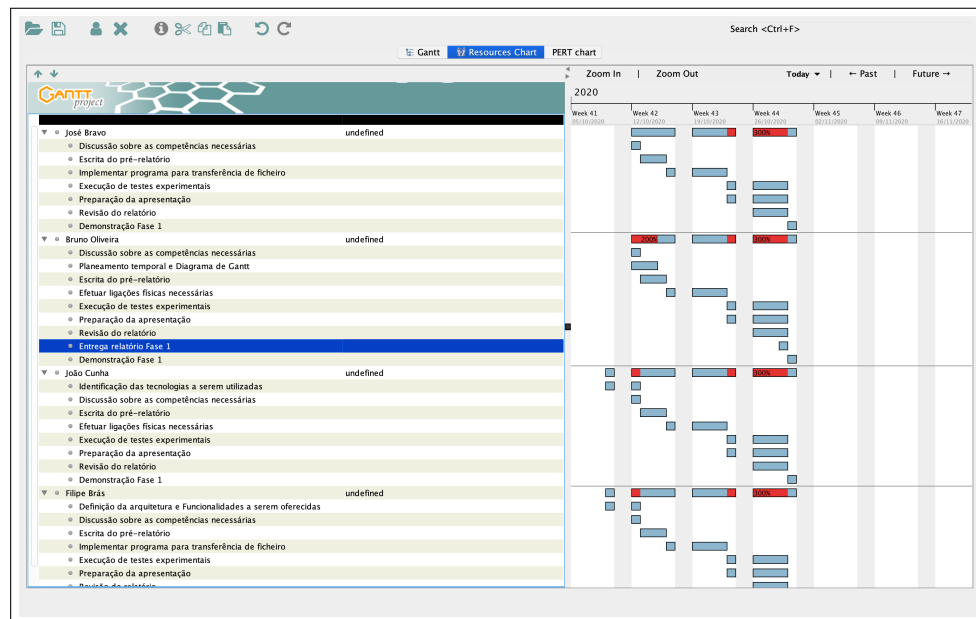


Figura 7: Tarefas atribuídas a cada membro do grupo.

As tecnologias a utilizar são a placa de desenvolvimento ESP32 e o módulo *transciever* RF nRF24L01+.

- **Placa de Desenvolvimento ESP32** - A placa de desenvolvimento ESP32 permite a leitura de *inputs* de dispositivos tais como o *transciever* transformando num *output* útil. A sua programação é feita em C++ através do IDE(Integrated Development Environment) de programação "Arduino". A sua construção é baseado no microcontrolador **ESP-WROOM-32D**.

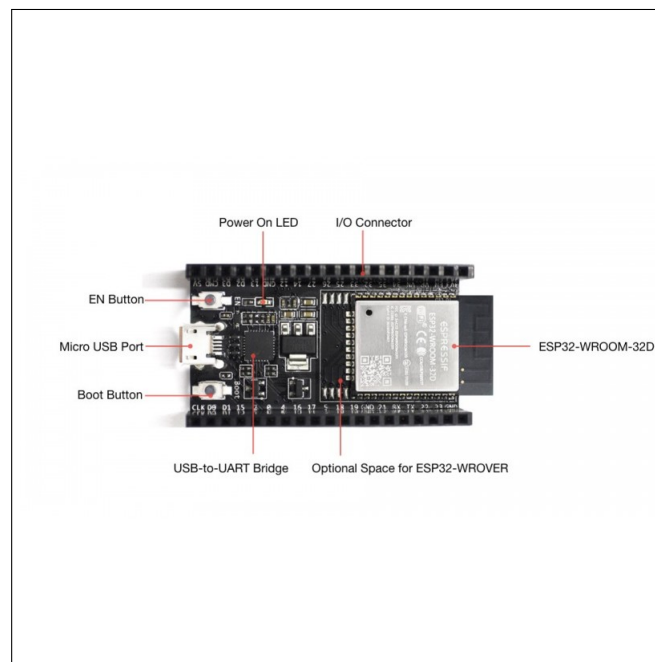


Figura 8: Descrição da placa de desenvolvimento ESP32.

- **Módulo *transciever* de RF nRF24L01+** - *Transciever* de radiofrequência que opera na banda de frequências ISM (Industrial, Scientific and Medical) 2,4 GHz, e é configurado e executa as suas funções através da sua interface SPI (Serial Peripheral Interface), com o auxílio de um microcontrolador.



Figura 9: Módulo *transciever* RF nRF24L01+.

7. Tarefa 1.2 - Comunicação via porta série

PC-Arduino-Arduino-PC

7.1. Características das interfaces de comunicação série

Nesta matéria, o grupo optou pela interface de comunicação série para comunicação com periféricos UART/RS-232.

O protocolo UART é um protocolo, que consiste numa interface responsável por implementar comunicação série, ou seja, faz a conversão de comunicação paralela para a comunicação série.¹² Abaixo encontram-se as figuras relativas às interfaces de comunicação utilizadas para a Fase 1 e à interface UART.

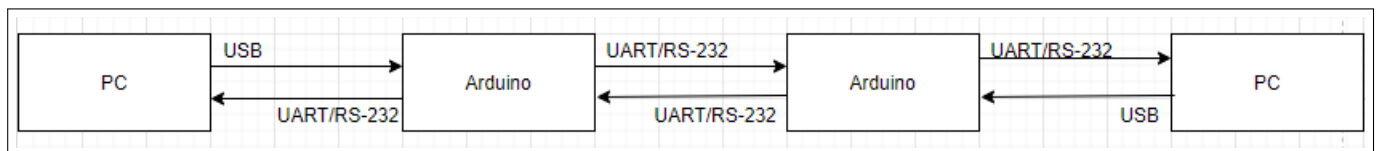


Figura 10: Interfaces utilizadas no PC-Arduino-Arduino-PC na Fase 1.

7.2. Parâmetros de configuração das interfaces de comunicação série

Relativamente à configuração das interfaces de comunicação série, nesta fase, não implementamos mecanismos de controlo de erros e controlo de fluxo, definimos uma *baudrate* de '115200 bps', um *stopbit* e oito bits de informação, conforme ilustrado na Figura 11.

Em termos de níveis de tensão, o nível lógico "0" corresponde a uma tensão de 0 V e o nível lógico "1" corresponde a uma tensão de 3,3 V ou 5 V.

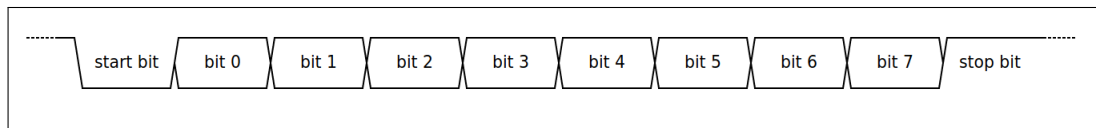


Figura 11: Estrutura da trama.

```
port.openPort();  
port.setParams(  
    SerialPort.BAUDRATE_115200,  
    SerialPort.DATABITS_8,  
    SerialPort.STOPBITS_1,  
    SerialPort.PARITY_NONE  
);
```

Figura 12: Parâmetros utilizados na configuração da porta série.

7.3. Ligações físicas entre os componentes utilizados

Para que fosse possível o envio de tramas de um ponto a outro implementamos ligações físicas nos módulos ESP32 de forma a utilizarmos a bridge USB-to-UART, esta comunicação é efetuada através dos pinos de TX, RX de ambos os módulos, sendo que, o pino TX irá ligar ao pino RX e vice-versa, sendo também necessária uma ligação GND-GND. Estas ligações permitem uma comunicação entre os módulos ESP-32 sendo que a ligação PC-Arduino é efetuada através de USB.

Na figura 11 apresentada em baixo podemos verificar a ligação física efetuada.

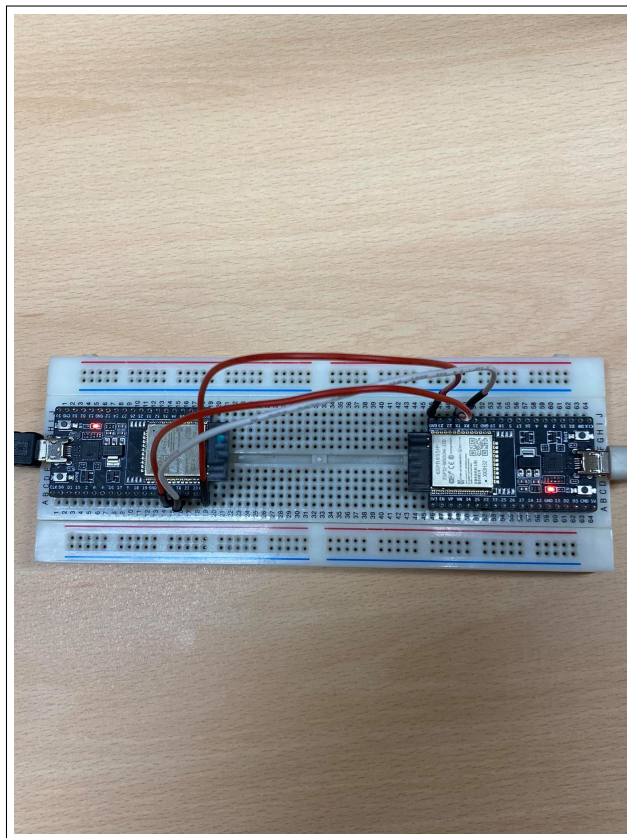


Figura 13: Ligação física implementada na Fase 1.

7.4. Transferência de um ficheiro de texto e de um ficheiro de imagem

A transferência de ficheiros tanto de texto como de imagem foi implementada através da elaboração de um programa em linguagem JAVA com recurso a diferentes bibliotecas existentes, com principal foco na biblioteca **jssc** que nos facilitou a interação e manipulação das portas série.

Inicialmente, o software conecta-se a uma determinada porta série, neste caso, o USB-to-UART Bridge presente no módulo ESP32, configurando todos os parâmetros mencionados anteriormente, de modo a efetuar o envio de ficheiros.

Após efetuada a ligação e a configuração da porta série, o programa lê os *bytes* presentes no ficheiro, armazenando-os num *bytearray* que irá ser enviado através da porta série. A receção deste é um processo idêntico (no sentido inverso), isto é, o programa recebe o *bytearray* presente na porta série, e verifica o tipo de ficheiro representado no mesmo (nesta fase apenas validámos ficheiros com formatos *.png* e *.txt*) criando o ficheiro respetivo.

Nesta primeira fase, conforme exposto no enunciado, o tamanho dos ficheiros a transmitir é conhecido á partida, sendo *hard-coded* no programa.

8. Teste, Resultados e Discussão

Os testes realizados foram a nível do *baudrate* pois quanto maior o valor deste maior será a transmissão de *bits* por segundo.

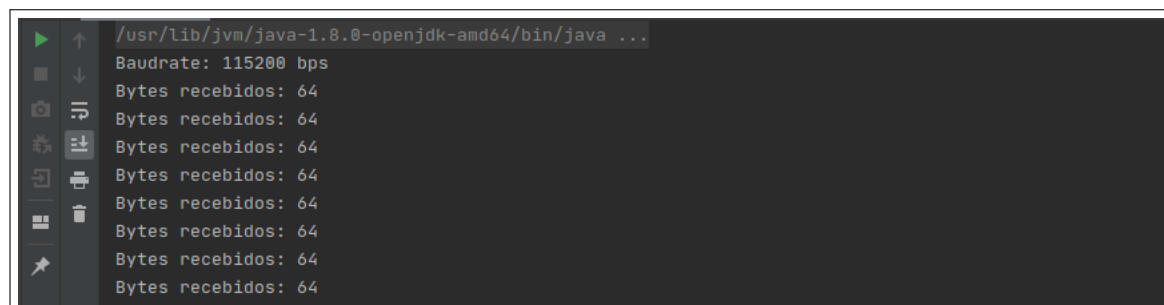
Nestes testes observamos que para um *baudrate* de 9,600 Kbps, a porta série através do método **readbytes()** retorna apenas 3 *bytes* de cada vez o mesmo acontece para um *baudrate* de 4,800 Kbps mas, para um *baudrate* de 115,200 Kbps, o mesmo método retorna 64 *bytes* de cada vez, o que faz com que a transferência de imagem seja mais rápida. Nas figuras apresentadas abaixo podemos observar o que foi descrito acima.

Consoante estes resultados o grupo decidiu utilizar um *baudrate* de 115,200 Kbps.

A terminal window with a dark background and light text. The command prompt shows the path to the Java executable. The output displays the baudrate as 9600 bps and shows eight consecutive lines of 'Bytes recebidos: 3'.

```
/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...  
Baudrate: 9600 bps  
Bytes recebidos: 3  
Bytes recebidos: 3  
Bytes recebidos: 3  
Bytes recebidos: 3  
Bytes recebidos: 3  
Bytes recebidos: 3  
Bytes recebidos: 3  
Bytes recebidos: 3
```

Figura 14: Teste com *baudrate* a 9,600 Kbps.

A terminal window with a dark background and light text. The command prompt shows the path to the Java executable. The output displays the baudrate as 115200 bps and shows eight consecutive lines of 'Bytes recebidos: 64'.

```
/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...  
Baudrate: 115200 bps  
Bytes recebidos: 64  
Bytes recebidos: 64  
Bytes recebidos: 64  
Bytes recebidos: 64  
Bytes recebidos: 64  
Bytes recebidos: 64  
Bytes recebidos: 64  
Bytes recebidos: 64
```

Figura 15: Teste com *baudrate* a 115,200 Kbps.

9. Conclusão

Nesta fase o objetivo traçado foi realizar uma comunicação PC-Arduino-Arduino-PC realizando uma transferência de ficheiros de texto e de imagem.

Todo o processo para chegarmos a este objetivo foi descrito neste relatório e acreditamos que o objetivo foi alcançado devido a um forte trabalho de cooperação entre os membros do grupo. O facto de também termos delineado todas as tarefas e os tempos das mesmas ajudou a uma melhor organização que levou posteriormente ao sucesso desta fase.

9.1. Autoavaliação

Filipe Brás - Nesta fase, estive envolvido no planeamento das tarefas a realizar durante a Fase 1, na pesquisa dos fundamentos teóricos necessários à implementação da Fase 1, e, posteriormente, na construção e edição do pré-relatório e relatório.

Bruno Oliveira - Tive dificuldade no início em perceber como iríamos realizar uma transferência de ficheiro através do Arduino sem utilizarmos o RS-232 pois não sabia da existência da bridge USB-to-UART nem desta comunicação série. Neste projeto tive envolvido no desenvolvimento do programa para a transferência do ficheiro e também na construção do diagrama de Gantt, ajudando também na realização do relatório.

José Bravo - Nesta fase estive envolvido no desenvolvimento do programa JAVA bem como na elaboração deste relatório. Inicialmente, senti algumas dificuldades na receção dos dados na porta série, visto que, consoante o *baudrate* utilizado, o número de bytes recebidos varia consideravelmente, dificultando o armazenamento dos mesmos. A maior dificuldade foi a identificação do formato do ficheiro recebido, especificamente para ficheiros de imagem (extensão *.png*)

João Cunha - Nesta fase 1 fiz um trabalho de pesquisa teórico sobre o que iríamos precisar, principalmente sobre como funcionam os protocolos UART, RS232 e sobre a comunicação série. Estive também envolvido no desenvolvimento do relatório.

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 29 de outubro 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 29 de outubro de 2020]
- [3] Mobile Communications; 2nd Edition; Schiller, Jochen
[online] Disponível em: https://www.academia.edu/7693415/Mobile_Communication
[Acedido a 29 de outubro 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
[Acedido a 29 de outubro 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
[Acedido a 29 de outubro de 2020]
- [6] [online] Disponível em: <https://osi-model.com/physical-layer/>
[Acedido a 29 de outubro de 2020]
- [7] [online] Disponível em <https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all>
[Acedido a 29 de outubro de 2020]

- [8] [online] Disponível em: <https://www.circuitbasics.com/basics-uart-communication/>
[Acedido a 29 de outubro de 2020]
- [9] [online] Disponível em: <https://circuitdigest.com/article/rs232-serial-communication-protocol-basics>
[Acedido a: 29 de outubro de 2020]
- [10] [online] Disponível em: <https://learn.sparkfun.com/tutorials/serial-communication/uarts>
[Acedido a: 29 de outubro de 2020]
- [11] [online] Disponível em: <https://blogs.mulesoft.com/biz/tech-ramblings-biz/what-are-apis-how-do-apis-work/>
[Acedido a: 29 de outubro de 2020]
- [12] [online] Disponível em: <https://learn.sparkfun.com/tutorials/serial-communication/uarts>
[Acedido a: 29 de outubro de 2020]
- [13] Placa de Desenvolvimento ESP32
[online] Disponível em: <https://www.botnroll.com/en/esp/3131-esp32-devkitc-32d-f-development-kit-esp32.html>
[Acedido a 29 de outubro de 2020]
- [14] Módulo *transciever* de RF nRF24L01+
[online] Disponível em: <https://www.botnroll.com/pt/rf-lora/833-modulo-nrf24l01.html>
[Acedido a 29 de outubro de 2020]