Universidade do Minho
Escola de Engenharia

METI - 2024/2025

Projeto Integrador em Telecomunicações e Informática

Relatório de Especificação - Fase B

Grupo 5

Fernando João Santos Mendes (PG55807)
Bruno Miguel Fernandes Araújo (PG55806)
Junlin Lu (A101270)

Índice

Lista de Acrónimos

LED - Light-Emitting Diode.
UART - Universal Asynchronous Receiver /Transmitter.

Lista de Figuras

1. Introdução

A Fase B do projeto visa principalmente estabelecer um protocolo de comunicação confiável para a transmissão de dados entre duas placas ESP32. O objetivo é garantir uma transmissão robusta e eficaz, de forma a manter a integridade das informações, a identificação e controlo de falhas, além do controlo de fluxo entre os dispositivos. Em prol de uma comunicação fiável será necessário a estruturação de um protocolo de comunicação e a definição de métodos de controlo e correção de erros.

2. Objetivos

A Fase B do projeto visa explorar formas de obtermos uma comunicação entre o emissor e transmissor garantindo a sua fiabilidade, com tal são necessários compreender os seguintes objetivos:

1. Conhecer os dois modos de funcionamento: normal e transparente (raw), da porta série.
2. Conhecer e configurar os parâmetros e modo de funcionamento da porta série.
3. Compreender os conceitos de trama/pacote, cabeçalho, payload e cauda.
4. Perceber como representar corretamente a estrutura de um pacote (nome, tamanho e conteúdo de cada campo).
5. Definir os tipos de trama e os campos da mesma.
6. Especificar e conceber as primitivas de serviço a oferecer à camada superior e especificar as interações entre os dois componentes.
7. Calcular de forma teórica o tempo de transmissão de um ficheiro com base nos parâmetros de transmissão.
8. Implementação de funcionalidades do protocolo de nível 2 no emissor e no recetor.
9. Desenvolvimento do código para a comunicação entre a camada 2.
10. Implementação total da API.

3.Arquitetura

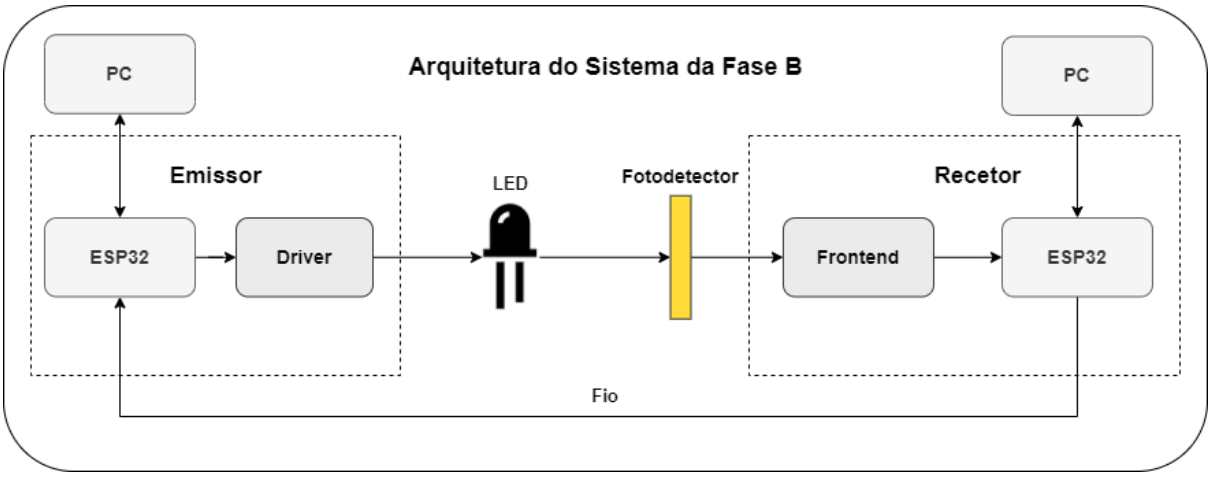

Figura 1: Arquitetura geral da Fase B

A Fase B concentra-se na execução de um protocolo de comunicação sólido na Camada 2 (Ligação de Dados). A arquitetura preserva os elementos centrais da Fase A, mas adiciona novas características para permitir comunicação em ambas as direções e correção de possíveis falhas.

Para completar a arquitetura do hardware necessário para o frontend seguimos o seguinte diagrama de blocos:

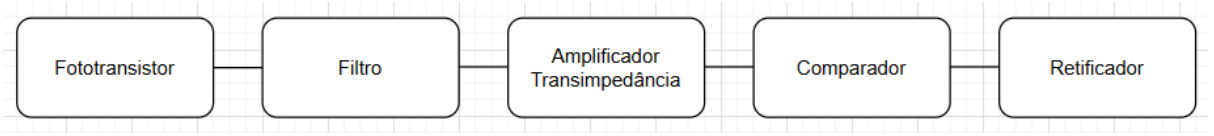

Figura 2: Diagrama de blocos do Frontend

A fase B distingue-se da fase A, devido à implementação de novos blocos necessários para que o esp32 consiga receber o devido sinal, sendo estes:

O bloco do comparador para conseguirmos obter um sinal entre os níveis +5 V e -5 V, sendo que utilizaremos o valor de referência ,0 V, para realizar a devida comparação.

O bloco do retificador para ajustar o sinal de saída do comparador aos valores de funcionamento do ESP32, e como tal, retificar o valor +5 V para 3.3 V e -5 V para 0 V.

3. Protocolo UART

A porta série UART (Universal Asynchronous Receiver/Transmitter) é uma interface de comunicação que permite a transmissão e receção de dados de forma assíncrona (sem clock compartilhado) entre dispositivos. Utiliza dois fios principais: TX (transmissão) e RX (recepção), enviando dados bit a bit em formato serial. A UART permite escolher o modo de funcionamento desejado sendo eles:

1. O modo RAW na comunicação UART (Universal Asynchronous Receiver/Transmitter) refere-se a uma forma de transmissão de dados onde os dados são enviados e recebidos diretamente, sem a aplicação de protocolos de alto nível ou formatação adicional. Isso significa que os dados são transmitidos em seu formato bruto, sem controle de fluxo, encapsulamento ou manipulação especial.

2. Modo normal, que ao contrário do modo "raw", pode incluir características como controlo de fluxo, tratamento de caracteres especiais (como novas linhas ou retornos de carro) e, em alguns casos, o uso de protocolos de camada superior (como Modbus, ASCII, etc.). Podemos ver a definição de uma trama no modo normal na figura 3.

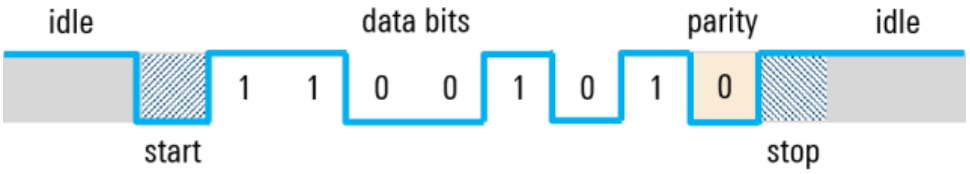

Figura 3: Exemplo da estrutura da trama em modo normal da UART

4. Protocolo de comunicação

A. Definição das tramas

Após o estudo sobre os modos de operação da UART, foi definido que seria usado o modo “raw” para que seja possível a definição da estrutura da nossa trama dado que neste modo os dados são enviados no seu formato bruto. Desta forma a estrutura definida caracteriza-se por:

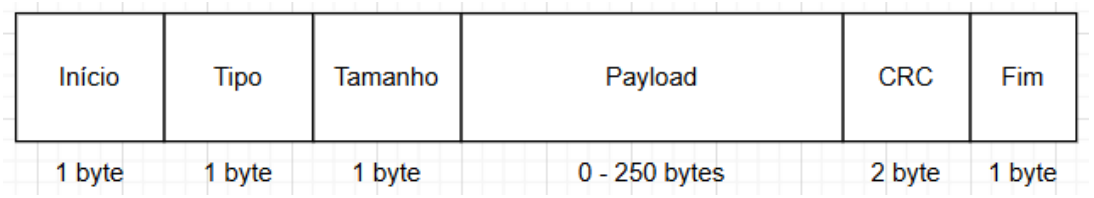

Figura 4: Estrutura da trama

Os seguintes campos da trama são definidos por:

- **Bloco de início e fim:**
  Dado que a comunicação UART é assíncrona é necessário criar sinalizações nas tramas que nos permite identificar o início e fim das mesmas. Para tal, usamos os campos definidos como início e fim, ambos definidos por 8 bits (1 byte). Utilizamos 1 byte ao invés de 1 bit para obtermos uma sincronização mais robusta, visto que desta forma é possível identificar ambos os campos com mais clareza e exatidão.

- **Bloco Tipo**:
  Este bloco é responsável por definir o tipo das tramas que estão a ser enviadas, sejam elas do tipo:
  - “M”, tramas correspondentes ao envio da informação de ficheiros (resposta do Servidor);
  - “P”, envio de informação respetivo aos pedidos do cliente (pedido);
  - “ACK”, resposta/confirmação positiva correspondente à devida recepção sem erros das tramas;
  - “NACK”, resposta de confirmação negativa correspondente a tramas com erros;

- **Bloco do tamanho e payload:**
  O bloco do tamanho compreende o tamanho da mensagem do bloco do payload e o bloco de payload contém a informação útil.

- **Bloco CRC:**
  Este bloco será especificado no ponto “Deteção de erros” descrito mais abaixo;

B. Controlo de fluxo e sincronização

Para garantir a sincronização no envio das mensagens e um bom controlo do fluxo, foram consideradas algumas técnicas como *Stop-And-Wait* e o *Sliding-Window*. Das técnicas referidas foi optado pelo *Stop-And-Wait* visto que além de ser mais simples de implementar , este é bastante fiável. A figura 5 ilustra o processo desta técnica, demonstra o envio e receção de frames entre dois dispositivos. Sempre que um ACK não é recebido dentro do tempo limite definido ou se um frame se perd**e**, este é retransmitido. Caso um frame duplicado seja recebido, o recetor simplesmente descarta-o, evitando inconsistências na transmissão. Este método é de longe o mais eficiente, uma vez que a transmissão fica parada enquanto é feita a espera de um ACK.

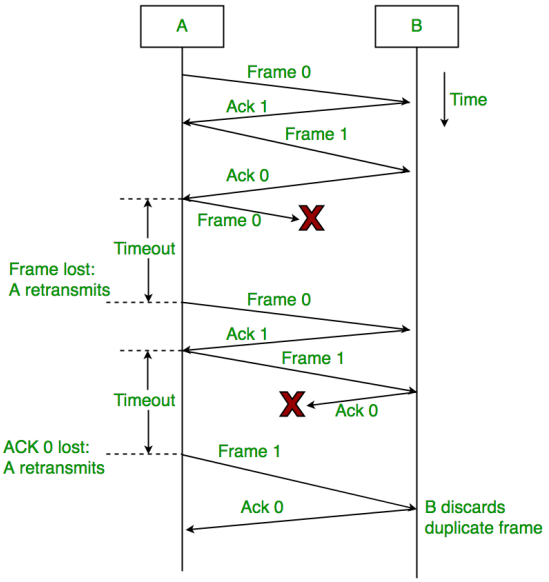

Figura 5: Ilustração da técnica de *Stop-And-Wait*

A implementação seguirá o seguinte raciocínio: O intervalo definido para os timeouts será de 3 tempos de transmissão ,e, caso sejam detectados 3 timeouts seguidos, será concluído que a comunicação entre os dispositivos não é mais viável.

C. Deteção de erros

Relativamente à deteção de erros, quer seja quando há uma receção de um payload com erros ou quando não é recebida a mensagem mesmo após o envio de acks, temos duas opções de detecção, o uso de *Checksum* ou CRC.

Checksum - Um método simples de deteção de erros que consiste na soma dos bytes do payload, onde é posteriormente comparado no destino com o valor esperado. Este método apresenta algumas limitações, uma vez que não consegue detetar todos os tipos de erros, especialmente *burst errors,* tornando-se menos fiável em canais de comunicação sujeitos a ruído.

CRC - Utiliza operações de divisão polinomial para gerar um valor de verificação, proporcionando uma deteção de erros muito mais eficaz do que o checksum. Devido à sua capacidade de detetar erros únicos, múltiplos e em rajada, o CRC é amplamente utilizado em comunicações fiáveis, de forma a proporcionar uma maior integridade nos dados transmitidos.

Considerando os aspetos anteriormente mencionados, optámos pelo método *CRC* e escolhemos o *CRC16* em vez do *CRC8*, uma vez que, para uma comunicação sujeita a ruído, é essencial garantir uma deteção de erros fortíssima. Apesar de o *CRC16* usar um overhead maior, este oferece uma deteção de erros superior em comparação com o *CRC8*.

Quando se processa as tramas e são detectados erros, o processo de recuperação de informação íntegra é dado pela técnica referida na secção B (controlo de fluxo e sincronização).

D. Cálculos Teóricos

Considerando o circuito receptor previamente desenvolvido e as suas características, foi definido que a frequência do sinal transmitido seria de 7 kHz.

Para calcularmos o Bit Rate (taxa de transmissão) a ser usado no nosso sistema de comunicação utilizamos a frequência do sinal como referência e como tal obtemos um valor de 7 Kbps.

Dado que vamos transmitir em cada símbolo 1 bit consideramos o valor do Baud Rate igual ao Bit Rate.

Dado os 256 bytes (2048 bits) que definem o tamanho máximo do pacote (L), ou seja, um pacote que utiliza os 250 bytes no payload, e uma taxa de transmissão de 7 Kbps (R) é possível deduzir o seu tempo de transmissão através da seguinte expressão:

$$T_{\text{transmissão}} = \frac{L}{R}$$

$$T_{\text{transmissão}} = \frac{2048}{7 \times 10^3}$$

$$T_{\text{transmissão}} \approx 292,57 \times 10^{-3} s$$

5. Ferramentas utilizadas

As ferramentas a níıvel do software que usamos as seguintes:

- Programa **Arduino IDE** , para o desenvolvimento de código para os **ESP32**.
- Programa **Discord**, para a comunicação e partilha de ficheiros entre os membros do grupo.
- Programa **Tina**, para a simulação dos circuitos do emissor e do recetor.
- Programa **Excel**, para a criação do diagrama de Gantt.
- Plataforma **Draw.io**, para a criação do diagrama da arquitetura do sistema desta fase.
- Plataforma **Google Docs**, para o desenvolvimento deste relatório.
- Plataforma **Overleaf**, para a construção das fórmulas matemáticas.

6. Planeamento Temporal

Apresentaremos a planificação temporal da Fase B do projeto através de um Diagrama de Gantt, que contém a lista completa de todas as tarefas e subtarefas desta fase, este encontra-se ilustrado na Figura 6.

TAREFAS	Início	Fim
Conhecer os dois modos de funcionamento da porta série e configuração desta.	3/14/25	3/14/25
Representar corretamente a estrutura do pacote.	3/14/25	3/15/25
Definir os tipos de trama e campos da mesma.	3/15/25	3/17/25
Especificação das interações entre as duas componentes.	3/17/25	3/20/25
Calculo teórico do tempo de transmissão de um ficheiro.	3/20/25	3/21/25
Relatório de especificação da Fase B	3/14/25	3/21/25
Implementação de funcionalidades do protocolo de nível 2 no emissor e no recetor.	3/21/25	3/28/25
Desenvolvimento do código para a comunicação entre a camada 2.	3/21/25	3/28/25
Implementação total da API.	3/28/25	4/4/25
Implementação da correção e tratamento de erros.	3/28/25	4/6/25
Demo da Fase A	4/7/25	4/7/25

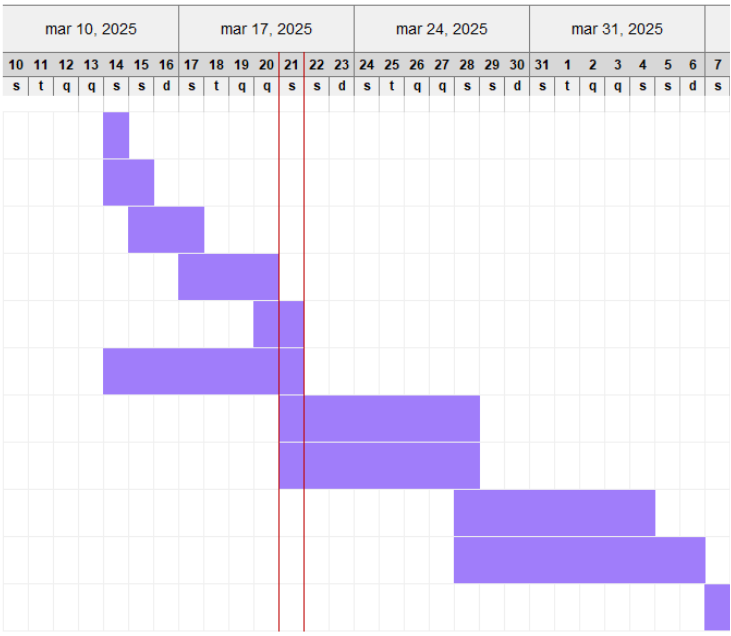

Figura 6: Diagrama de Gantt.

7. Referências

- Rohde & Schwarz. (n.d.). *Compreender UART*. Rohde & Schwarz. https://www.rohde-schwarz.com/br/produtos/teste-e-medicao/essentials-test-equipment/digital-oscilloscopes/compreender-uart_254524.html

- Copperhill Technologies. (2021). *ESP32/ESP32S2 Serial Port (Native USB) Access Using Arduino IDE*. Copperhill Technologies. https://copperhilltech.com/blog/esp32-esp32s2-serial-port-native-usb-access-using-arduino-ide/

- GeeksforGeeks**.** (n.d.). *Stop and Wait ARQ*. GeeksforGeeks. https://www.geeksforgeeks.org/stop-and-wait-arq/