

U.C. de Projeto Integrado de

Telecomunicações

Ano Letivo: **2021/2022**

**Relatório da Fase B**

**Grupo 2**

* Catarina Neves, a93088
* Eduardo Cardoso, a89627
* José Gomes, a93083
* Luís Oliveira, a89380

27/04/2022

Universidade do Minho

Mestrado Integrado em Engenharia de Telecomunicações e Informática

Índice

[1. Introdução 5](#_Toc101698500)

[2. Trabalho Relacionado 6](#_Toc101698501)

[2.1 ESP32 WebSocket Server: Control Outputs 6](#_Toc101698502)

[2.2 ESP32 WebSocket Server with Multiple Sliders: Control LEDs Brightness 7](#_Toc101698503)

[3. Etapas do trabalho desenvolvido 8](#_Toc101698504)

[3.1. Protocolo de comunicação 8](#_Toc101698505)

[3.2. Sistema Central 9](#_Toc101698506)

[3.2.1 Sample 10](#_Toc101698507)

[3.3. ETAPA 3 12](#_Toc101698508)

[3.4. ETAPA 4 13](#_Toc101698509)

[3.5. ETAPA 5 (Servidor Web maybe?) 14](#_Toc101698510)

[4. Análise de resultados e testes efetuados 15](#_Toc101698511)

[5. Conclusão 16](#_Toc101698512)

[5.1. Contribuição de cada aluno 16](#_Toc101698513)

[6. Lista de referências 17](#_Toc101698514)

Índice de figuras

Figura 1 - Tarefas propostas pela fase B. 5

Figura 2 – Aplicação e circuito implementados pelo autor. 6

Figura 3 – Página web com 3 sliders. 7

Figura 4 - Circuito eletrónico implementado pelo autor. 7

Figura 5 - Ficheiros presentes na diretoria sample. 9

Figura 6 - Ficheiros presentes na diretoria security. 9

Figura 7 - Ficheiro presente na diretoria template. 9

Figura 8 - Constituição da diretoria resources. 9

Figura 9 - Estrutura da classe WeatherSample. 10

Figura 10 - Atributos da tabela weather. 10

Figura 11 - Estrutura da classe WeatherSampleService. 11

Figura 12 - Estrutura da classe WeatherSampleService. 11

Figura 13 - Interações entre as camadas do gestor de serviço. 11

Figura - 14

Figura - Frontend da página com todas as amostras 14

Lista de siglas e acrónimos

**LED** *Light Emitting Diode*

**PWM** *Pulse Width Modulation*

**TCP** *Transmission Control Protocol*

**IP** *Internet Protocol*

# Introdução

Serve o presente relatório como síntese do trabalho desenvolvido e implementado no decorrer desta segunda fase. Este contém a descrição das estratégias e algoritmos adotados pelo grupo, tal como os respetivos testes realizados. Além dos tópicos acima citados, faz-se referência a trabalhos ou projetos similares, que se enquadram na ótica deste projeto.

O relatório abordará cada etapa desta fase de forma detalhada, ou seja, será apresentada a resposta ou proposta de solução empregue pelo grupo em junção com as ferramentas que foram necessárias para a sua construção.

A figura 1 ilustra o sumário das diferentes tarefas propostas nesta fase.

Uma imagem com texto, ecrã

Descrição gerada automaticamente

Figura 1 - Tarefas propostas pela fase B.

Nesta fase, o objetivo é a implementação de um sistema central e do protocolo de comunicação entre este sistema e os *gateways* (*Gateway* e Sistemas Simulados), que deve estar assente numa comunicação por mensagens aplicacionais sobre TCP (*Transmission Control Protocol*) /IP (*Internet Protocol*).

O sistema central recolhe a informação enviada por todos os *gateways*, que por sua vez, será armazenada na base de dados. Sempre que o serviço *web* queira atualizar os dados apresentados, este faz um pedido ao gestor de serviço que, de forma encadeada, irá buscar os dados à base de dados e enviará os mesmos para o serviço *web*.

# Trabalho Relacionado

Relativamente aos trabalhos relacionados, enquadram-se os projetos seguintes, que serviram de base à construção deste projeto.

## 2.1 ESP32 WebSocket Server: Control Outputs

Este projeto, da autoria de Rui Santos [1], consiste numa aplicação que varia o estado de um determinado LED (*Light Emitting Diode*)*,* sendo baseada no modelo cliente-servidor: a placa ESP32 desempenha o papel de servidor e o *browser*, que contém uma página *web* (configurada no código do servidor), atua como cliente, capaz de controlar a placa referida de forma remota.

A página *web* apresenta o estado do LED e possui um botão, que é responsável pela mudança de estado do mesmo (ligar ou desligar). É de salientar, que o LED é conectado à placa referida, através de fios de ligação e com o auxílio de uma *breadboard*.

O cliente estabelece uma ligação via *websocket* com o servidor; quando esta é estabelecida, o cliente e o servidor podem enviar dados com recurso a *sockets*, de forma *full-duplex*.

A utilização do protocolo baseado em *websockets,* permite que o servidor envie informação para o(s) seu(s) cliente(s) sem necessitar de ser requisitado pelos mesmos, logo, sempre que o estado do LED varie, essa informação é enviada para o *browser*, tal como é visível na figura 2.

Uma imagem com texto, interior

Descrição gerada automaticamente

Figura 2 – Aplicação e circuito implementados pelo autor.

A aplicação funciona de acordo com os passos seguintes:

1. Clicar no botão “*Toggle*”;
2. O cliente (*browser*) envia a mensagem “*toggle*” via *socket*;
3. O servidor (placa ESP32) recebe essa mensagem e muda o estado do LED;
4. O servidor envia o novo estado do LED para o(s) seu(s) clientes;
5. O(s) cliente(s) recebe(m) a mensagem e atualiza(m) o estado do LED, na *webpage* associada;

## 2.2 ESP32 WebSocket Server with Multiple Sliders: Control LEDs Brightness

Este projeto, da autoria de Rui Santos [2], consiste numa aplicação que permite variar a luminosidade de vários LEDs, sendo baseada no modelo cliente-servidor: a placa ESP32 desempenha o papel de servidor e o browser, que contém uma página web (configurada no código do servidor), atua como cliente, capaz de variar a luminosidade dos respetivos LEDs. A comunicação entre cliente(s) e servidor é realizada através de um protocolo assente em websockets.

A variação da luminosidade dos LEDs é feita através de um conjunto de *sliders,* que controlam o *duty cycle* dos respetivos sinais PWM (*Pulse Width Modulation*).

A figura seguinte ilustra a página web que contém os *sliders* referidos.

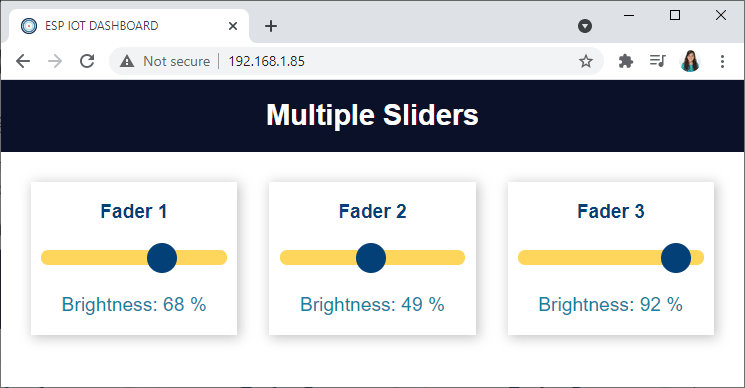


Figura 3 – Página web com 3 sliders.

A página web encontra-se organizada da seguinte forma:

* Um *fader* por LED, num total de 3 LEDs;
* Cada *fader* possui um *slider,* que é responsável pela variação da luminosidade do LED associado;
* A luminosidade do LED (em percentagem) varia entre 0 e 100;
* Sempre que um *slider* varie a luminosidade, esta é atualizada em todos os clientes de forma simultânea;

O servidor coloca na rede local uma página web, que é constituída por 3 *sliders*. Sempre que um *slider* é alterado (luminosidade alterada), o cliente envia o número relativo desse *slider* para o servidor, de acordo com o protocolo referido anteriormente. Por exemplo, se um utilizador colocar a posição do *slider* 3 em 50%, o cliente enviará a mensagem “**3s50**” para o servidor. Após o servidor ter recebido a mensagem mencionada, ajusta o *duty cycle* do sinal PWM, de acordo com os valores incorporados na mensagem e notifica os restantes clientes, com os novos valores.

A figura seguinte ilustra o circuito eletrónico efetuado pelo autor.

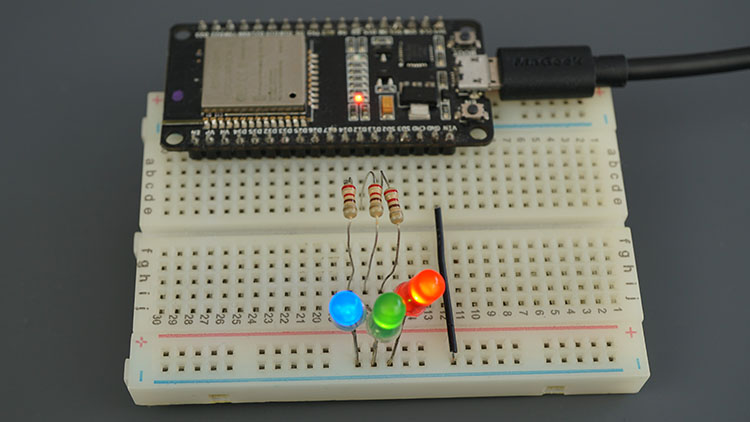


Figura 4 - Circuito eletrónico implementado pelo autor.

# Etapas do trabalho desenvolvido

Esta secção contém todas as etapas necessárias ao desenvolvimento desta fase, que se encontram descritas pelos mecanismos, ferramentas utilizadas e algoritmos implementados pelo grupo.

## 3.1. Protocolo de comunicação

## 3.2. Sistema Central

O sistema central implementado pelo grupo encontra-se mapeado, estruturado em 3 diretorias distintas:

* ***sample***: incorpora o código relativo ao gestor de serviço, que é responsável pelo processamento das amostras recolhidas e estabelece uma ponte de ligação entre a base de dados relacional e o serviço *web*.

Uma imagem com texto

Descrição gerada automaticamenteA figura seguinte ilustra o conjunto de ficheiros (interface e classes) presentes na diretoria descrita.

Figura 5 - Ficheiros presentes na diretoria sample.

* ***security*:** engloba o código responsável pela segurança do sistema central, ou seja, define um conjunto de funcionalidades de administração e monitorização, que por sua vez, são atribuídas a um conjunto de utilizadores, que possuem um determinado conjunto de funções e permissões.

A figura seguinte ilustra o conjunto de ficheiros (classes e enumerados) presentes na diretoria descrita.

Uma imagem com texto

Descrição gerada automaticamente

Figura 6 - Ficheiros presentes na diretoria security.

* ***template:*** composta pelo código intrínseco ao serviço *web*, isto é, possui todo o código inerente à interligação entre a componente gráfica do serviço referido e os endereços de ligação associados.

A figura seguinte ilustra o ficheiro (classe) presente na diretoria descrita.



Figura 7 - Ficheiro presente na diretoria template.

* ***resources***: constituída por duas diretorias e por um ficheiro responsável pela comunicação com a base de dados relacional. A diretoria *templates* possui o código inerente à componente gráfica do serviço *web* (menus, imagens, etc.) e o ficheiro *application.properties* contém um conjunto de instruções relativas à aplicação e à base de dados relacional.

A figura seguinte ilustra a constituição da diretoria referida.

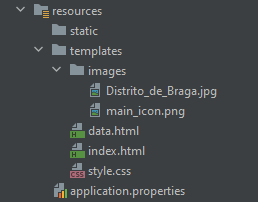


Figura 8 - Constituição da diretoria resources.

### 3.2.1 Sample

A diretoria *sample* contém um conjunto de classes e uma interface, que são necessários à transformação, tratamento e armazenamento das amostras.

A classe *WeatherSample* define a estrutura, representa o paradigma de qualquer amostra recolhida, isto é, contém todos os atributos e comportamentos inerentes à mesma.

A figura 9 ilustra os estados (variáveis de instância) e os comportamentos(métodos) relativos à classe *WeatherSample*.

Uma imagem com mesa

Descrição gerada automaticamente

Figura 9 - Estrutura da classe WeatherSample.

É de salientar que dentro da base de dados, cada amostra é identificada pelo seu código identificador (ID), que corresponde à chave primária da tabela *weather.*

Uma imagem com texto, quadro de resultados

Descrição gerada automaticamenteA figura seguinte ilustra os atributos da tabela, entidade *weather*.

Figura 10 - Atributos da tabela weather.

As classes *WeatherSampleController*, *WeatherSampleService* e a interface *WeatherSampleRepo* atuam em diferentes camadas da aplicação.

A interface *WeatherSampleRepo* atua na camada de acesso à base de dados, ou seja, é responsável pela manipulação e gerenciamento da base de dados relacional. Esta interface estende a interface *JpaRepository*¸ que recebe duas classes como argumentos: a classe *WeatherSample*, que define os atributos da tabela *weather* e a classe *Long*, que representa o tipo da chave primária da tabela referida.

É de realçar que não foram definidos quaisquer métodos para a interface *WeatherSampleRepo*, uma vez que os métodos herdados, são suficientes para a complexidade deste projeto.

Relativamente à classe *WeatherSampleService*, esta estabelece a ponte de interligação entre a camada de serviço e a camada de acesso à base de dados, isto é, define um conjunto de métodos que operam sobre os métodos herdados pela interface *WeatherSampleRepo*.

Uma imagem com texto

Descrição gerada automaticamenteA figura 11 ilustra os estados (variáveis de instância) e os comportamentos (métodos) relativos à classe *WeatherSampleService*.

Figura 11 - Estrutura da classe WeatherSampleService.

No caso da classe *WeatherSampleController*, que equivale à camada de controlo, esta efetua um conjunto de *web requests* ao serviço web, do tipo: *GET*, *POST*, *PUT* e *DELETE*. Cada *web request* resulta numa *query* colocada à base de dados.

Em semelhança com o que acontece com a classe *WeatherSampleService*, a camada de controlo também define um conjunto de métodos, anotados com os tipos de *web requests* associados, que operam sobre os métodos definidos pela classe, que é responsável pela camada de serviço.

Uma imagem com texto

Descrição gerada automaticamenteA figura 12 ilustra os estados (variáveis de instância) e os comportamentos (métodos) relativos à classe *WeatherSampleController*.

Figura 12 - Estrutura da classe WeatherSampleService.

Uma imagem com texto

Descrição gerada automaticamenteA interação entre as diversas camadas do gestor de serviço, que foram descritas e analisadas ao longo desta subsecção, pode ser ilustrada de acordo com a figura seguinte.

Figura 13 - Interações entre as camadas do gestor de serviço.

## 3.3. ETAPA 3

## 3.4. ETAPA 4

## 3.5. ETAPA 5 (Servidor Web maybe?)

Nesta etapa, o servidor *Web* que desenvolvemos foi inspirado num *template* desenvolvido pela W3Schools [3]. Este template foi depois modificado por nós para obtermos o aspeto e definições que pretendíamos, como mostra a figura 10.

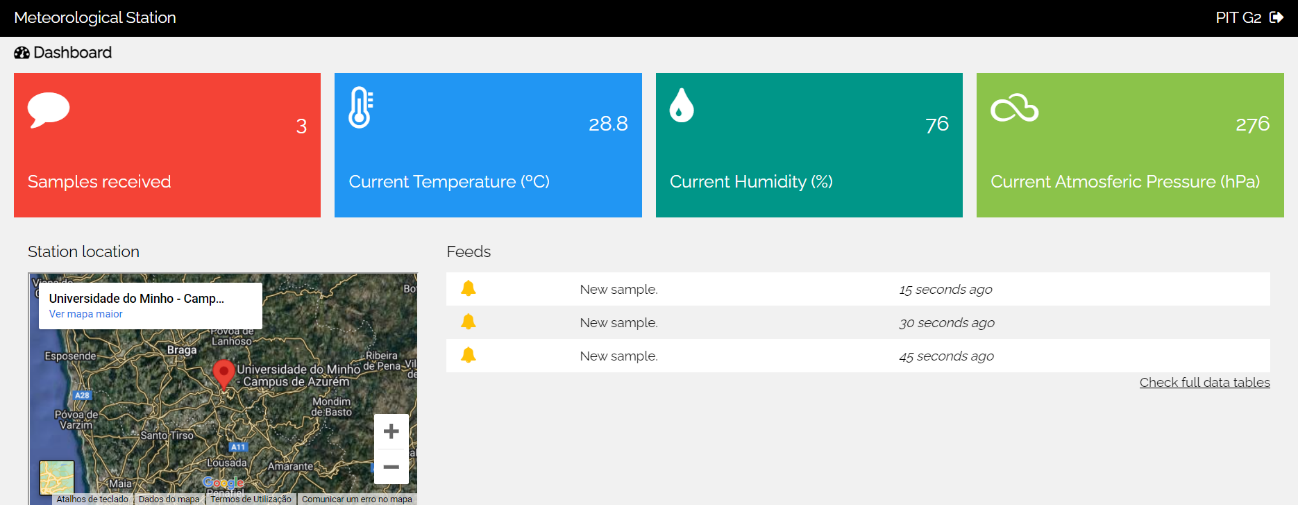


Figura 14 - Frontend da página inicial do servidor web

Nesta figura, podemos ver no topo o número de amostras que o servidor já obteve, tanto como a temperatura, a humidade e a pressão atmosférica da última amostra.

Por sua vez, se se premir o botão “*Check full data tables*”, vamos para a página que vemos na figura 12. Nesta página, temos acesso a todas as amostras que foram enviadas para o servidor *web*.



Figura 15 - Frontend da página com todas as amostras

# Análise de resultados e testes efetuados

# Conclusão

Concluída esta fase, na opinião do grupo, conseguimos atingir todos os objetivos propostos. Esta fase contribuiu para o conhecimento e estudo da implementação de *Sockets*, comunicação sobre TCP/IP e sobre o desenho de Servidores *Web*.

O grupo pretende que, com o trabalho desenvolvido ao longo deste fase intermédia, ter boas fundações para completar a etapa final deste projeto com sucesso.

## 5.1. Contribuição de cada aluno

Catarina Neves

* Conceção dos algoritmos;
* Tratamento de dados, tabelas e imagens a estes relacionados;
* MAIS QUALQUER COISA;
* MAIS QUALQUER COISA;
* Desenvolvimento do relatório;

Eduardo Cardoso

* Conceção dos algoritmos;
* MAIS QUALQUER COISA;
* MAIS QUALQUER COISA;
* MAIS QUALQUER COISA;
* MAIS QUALQUER COISA;
* Desenvolvimento do relatório;

José Gomes

* Conceção dos algoritmos;
* Tradução dos algoritmos implementados em fluxogramas;
* MAIS QUALQUER COISA;
* Desenvolvimento do relatório;

Luís Oliveira

* Conceção dos algoritmos;
* Desenvolvimento do *Frontend* e *Backend* do Servidor *Web*;
* Desenho de esquemas e gráficos;
* Desenvolvimento do relatório.

# Lista de referências

|  |  |
| --- | --- |
| [1] | R. Santos, “Random Nerd Tutorials,” 10 2020. [Online]. Available: https://randomnerdtutorials.com/esp32-websocket-server-arduino/. |
| [2] | R. Santos, “ESP32 Web Server (WebSocket) with Multiple Sliders: Control LEDs Brightness (PWM),” 5 2021. [Online]. Available: https://randomnerdtutorials.com/esp32-web-server-websocket-sliders/. |
| [3] | W3Schools, “W3Schools W3.CSS,” [Online]. Available: https://www.w3schools.com/w3css/. |
| [4] | ioBridge, “ThingSpeak for IoT Projects,” 2022. [Online]. Available: https://thingspeak.com/. |
| [5] | Bosch, “BME280 Combined humidity and pressure sensor,” setembro 2018. [Online]. Available: https://www.mouser.com/datasheet/2/783/BST-BME280-DS002-1509607.pdf. |
| [6] | NTP Pool Project, “Europe — europe.pool.ntp.org,” [Online]. Available: https://www.pool.ntp.org/zone/europe. |