

**CONTROLADOR DE PAINÉIS DE LED (RGB)**

**Paulo António da Costa Torres Neves**

**(Número 38714)**

LICENCIATURA DE ENGENHARIA ELETRÓNICA DE TELECOMUNICAÇÕES E DE COMPUTADORES

(LEETC)

PROJETO FINAL DE CURSO

ORIENTADORES: ANDRÉ RIBEIRO LOURENÇO

PEDRO MIGUEL FLORINDO MIGUENS MATUTINO

Setembro 2019

# Índice de conteúdos

[Índice de conteúdos 1](#_Toc18275510)

[Lista de Figuras i](#_Toc18275511)

[Lista de Tabelas ii](#_Toc18275512)

[Acrónimos iii](#_Toc18275513)

[1. Introdução 1](#_Toc18275514)

[1.1 Enquadramento 2](#_Toc18275515)

[1.2 Descrição do projeto 3](#_Toc18275516)

[1.3 Objetivos 4](#_Toc18275517)

[1.4 Abordagem 5](#_Toc18275518)

[2. Estado da Arte 6](#_Toc18275519)

[3. Análise de Requisitos 7](#_Toc18275520)

[3.1 Funcionamento de uma Matriz de LED 7](#_Toc18275521)

[3.1.1 Painel RGB 9](#_Toc18275522)

[3.2 Microcontroladores 11](#_Toc18275523)

[3.2.1 Microcontrolador 11](#_Toc18275524)

[3.3 Ambiente de Desenvolvimento Integrado 14](#_Toc18275525)

[3.3.1 Arduino IDE 14](#_Toc18275526)

[3.4 Requisitos Técnicos 14](#_Toc18275527)

[4. Implementação 15](#_Toc18275528)

[5. Resultados Experimentais 24](#_Toc18275529)

[6. Conclusões 25](#_Toc18275530)

[7. Bibliografia 26](#_Toc18275531)

[8. Anexos 27](#_Toc18275532)

[Anexo A – Esquema ESP32 Adafruit Feather 27](#_Toc18275533)

[Anexo B – Esquema Eléctrico 28](#_Toc18275534)

# Lista de Figuras

[Figura 1 - Diagrama de Blocos 3](file:////Users/pcostatorres/Documents/GitHub/PFC/410_v1.docx#_Toc18275535)

[Figura 2 - Diagrama de Blocos Completo 4](file:////Users/pcostatorres/Documents/GitHub/PFC/410_v1.docx#_Toc18275536)

[Figura 3 - Matriz de LED's 7](#_Toc18275537)

[Figura 4 – Seleção de LED da matriz 8](#_Toc18275538)

[Figura 5 - Esquema Eléctrico Shift Register 9](#_Toc18275539)

[Figura 6 - Distribuição de saídas Shift Register com correspondência de ativação 10](#_Toc18275540)

[Figura 7 - Adafruit Huzzah32 - ESP32 Feather 12](file:////Users/pcostatorres/Documents/GitHub/PFC/410_v1.docx#_Toc18275541)

[Figura 8 - Módulo ESP32-WROOM-32 12](file:////Users/pcostatorres/Documents/GitHub/PFC/410_v1.docx#_Toc18275542)

[Figura 9 - Sketch Arduino 14](#_Toc18275543)

[Figura 10 - Diagrama Lógico da aplicação e hardware 15](file:////Users/pcostatorres/Documents/GitHub/PFC/410_v1.docx#_Toc18275544)

[Figura 11 - Fluxograma Função Setup 16](#_Toc18275545)

[Figura 12 – Função Loop 16](#_Toc18275546)

[Figura 13 – Fluxograma da Função Connection Init(Alterar e acrescenta) 17](#_Toc18275547)

[Figura 14 - Fluxograma Função Connect Using BT 18](#_Toc18275548)

[Figura 15 - Fluxograma Wifi Process 19](#_Toc18275549)

[Figura 16 - Fluxograma da Função Process BT 20](#_Toc18275550)

[Figura 17 - Fluxograma Função Scroll Display 21](#_Toc18275551)

[Figura 18 - Fluxograma Matrix Drive 21](#_Toc18275552)

[Figura 19 - Bitmap Caracter 'A' 23](#_Toc18275553)

[Figura 20 - Esquema Eléctrico ESP32 Adafruit Feather 27](#_Toc18275554)

[Figura 21 - Pinout ESP32 Adafruit 27](#_Toc18275555)

[Figura 22 - Esquema Eléctrico 28](file:////Users/pcostatorres/Documents/GitHub/PFC/410_v1.docx#_Toc18275556)

# Lista de Tabelas

[Tabela 1 - Caracteristicas a ter em consideração na seleção de painel de LED's 6](#_Toc18275558)

[Tabela 2 - Carateristicas Técnicas Painel SDM P10RGB 9](#_Toc18275559)

[Tabela 3 - Diferentes Módulos ESP32 12](#_Toc18275560)

# Acrónimos

LED - Light Emissor Diode

LAN - Local Area Network

WLAN - Wireless Local Area Network

IDE - Integrated Development Environment

RGB - Red, Green, Blue

SMD - Surface Mounted Device

FIFO - First in First Out

RAM - Random Access Memory

HTTP - Hypertext Transfer Protocol

SMD - Surface Mounted Device

SMT - Surface Mounted Technology

GPIO - General Purpose Input/ Output

SOC - System on Chip

# Introdução

Hoje em dia vulgarizou-se a utilização de painéis de divulgação de informação não estáticos, habitualmente estes baseiam-se em painéis de LED’s, existem soluções no mercado, no entanto estas habitualmente têm interfaces de controlo proprietárias. A interoperabilidade entre estas dificulta a sua gestão e manutenção.

Se for considerada a criação de uma *LED Wall,* conjunto de vários painéis de LED’s RGB, existem fatores importantes na escolha da melhor solução como:

* o número de interfaces necessárias para controlar a estrutura.
* o facto de ser uma opção modular.
* o seu baixo consumo.
* uma fácil instalação e manutenção.
* custos reduzidos.

Com a implementação de um sistema de controlo independente é possível integrar os vários tipos de painéis de LED’s com esta interface. Quando deparados com uma *LED Wall* de grandes dimensões poderá haver constrangimentos quanto ao seu sincronismo e à comunicação com os vários painéis de LED’s, quando se recorre a LAN ou WLAN.

## Enquadramento

Hoje em dia com a grande presença de painéis de LED’s em várias aplicações encontramos com frequência a necessidade de flexibilizar a sua utilização.

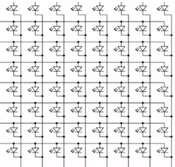
Assim surge a necessidade de criar uma interface que seja comum entre diferentes painéis, que seja dimensionável para tantos painéis quantos sejam necessários formando um agregado de painéis e que seja escalável.

Neste projeto propõe-se o desenvolvimento de uma interface para os painéis de LED’s, através de um microcontrolador, que faz a ligação entre o utilizador e os painéis de LED’s. O desenvolvimento desta interface será feito de forma modular e genérica, permitindo desta forma construir um sistema escalável.

A interface a desenvolver permitirá a centralização e uniformização da adaptação das várias aplicações dos painéis para os utilizadores, permitindo que a comunicação entre operador e interface seja realizada recorrendo a comunicação LAN ou WLAN.

## Descrição do projeto

Tal como descrito no diagrama de blocos representado na Figura 1, este projeto será dividido em 3 blocos principais. A aplicação desenvolvida a partir do IDE (1) selecionado, para assim enviar os dados para o microcontrolador (2). O microcontrolador que irá realizar a integração com o Painel de LED’s, controlando a informação visível nos painéis de LED’s. Por último temos o painel de LED’s que será uma matriz de LED que recebe a informação proveniente da aplicação e replicando-a nos seus LED’s.



MICROCONTROLADOR

APLICAÇÃO / IDE



USB

USB

WLAN

WLAN



PAINEIS DE LED’S RGB

Figura 1 - Diagrama de Blocos

## Objetivos

A existência de uma *LED Wall* é comum nos dias de hoje devido aos seus custos reduzidos de produção fazem com que sejam uma das principais opções quando se trata de reprodução de textos ou imagens.

O objetivo deste projeto é o desenvolvimento de uma estrutura modular com módulos básicos de 16x32 conjuntos de LED’s RGB, controlados através de um microcontrolador. Atualmente grande parte dos painéis de LED’s disponíveis são do tipo SMD e estes têm aplicações em ecrãs interiores e exteriores, diferentes densidades de LED’s por . Um pixel SMD é composto por um conjunto de LED’s com a côr vermelha, verde e azul todos juntos no mesmo encapsulamento, sendo montados estes na placa de circuito impresso, permitindo deste modo criar 8 níveis de cor.

O sistema a desenvolver irá utilizar *hardware* e *software* para permitir a transmissão de dados em blocos de 16x32 pixel, usando microcontroladores. A comunicação entre os vários módulos é feita através de um microcontrolador que possui interfaces de comunicação USB e sem fios para a receção de dados de um PC. Na Figura 2 está representado o diagrama de blocos de todo o sistema.



SHIFT REGISTER







SHIFT REGISTER

Figura 2 - Diagrama de Blocos Completo

## Abordagem

Este projeto tem como base o controlo de um painel de LED’s RGB através de um microcontrolador, onde foram analisados e selecionados, os componentes de hardware tais como a fonte de alimentação, o painel de LED’s RGB e o microcontrolador a utilizar no projeto, e relativamente ao software qual a linguagem mais adequada a utilizar na programação do microcontrolador, que neste caso foi selecionada a linguagem de programação em C devido ao seu comprovado uso nos mais variados sistemas embebidos.

A abordagem para o projeto está distribuída em 3 fases, onde na primeira fase é analisada a arquitetura do painel de LED’s e a sua integração com o microcontrolador, com a seleção do tipo de comunicação e bibliotecas indicadas.

Na fase seguinte, com os dados recolhidos e definidos nas fases anteriores, é feita uma prova de conceito na qual se demonstra o funcionamento do sistema.

Na última fase são corrigidos problemas detetados no desenvolvimento do projeto, melhorada a estabilidade e robustez do programa desenvolvido. Nesta fase será possível realizar uma demonstração funcional, onde o sistema deve apresentar-se com o comportamento definido nos requisitos.

# Estado da Arte

Para o projeto houve a necessidade de fazer a pesquisa sobre os produtos existentes no mercado que incluiu a pesquisa de painéis de LED’s RGB, microcontroladores e o produto acabado, que consiste num painel de LED’s RGB controlado por microcontrolador.

Com a pesquisa foi possível identificar tipos de painel que podem ter várias caraterísticas diferentes, as quais estão identificadas na Tabela 1, sendo que se optou por um painel com a dimensão de 16x32 conjuntos de LED’s RGB devido à sua dimensão possibilitar uma modularidade do painel, permite a visibilidade mais clara da informação projetada pelo painel, permitindo assim uma maior portabilidade durante o desenvolvimento do projeto.

Tabela 1 - Caracteristicas a ter em consideração na seleção de painel de LED's



Quanto ao estudo da escolha do microcontrolador efetuou-se uma pesquisa no qual existe uma grande diversidade no mercado, mas esta lista fica restringida devido ao projeto ter como requisitos a utilização de um microcontrolador de baixo custo, com ligação *Wifi* e *Bluetooth*. Perante estas restrições sobressaem duas famílias de produtos que são, a família Raspberry Pi (3) e a família Espressif (4).

Optou-se pela utilização da família Espressif, pois a família Raspberry Pi apresenta um custo superior e um maior consumo de energia. Dentro da família Espressif existem dois produtos (ESP 8266 (5) e o ESP32 (6)) que foram identificados como possíveis elementos para a realização do projeto, sendo que foi escolhido o ESP32 devido ao facto de ser o único com suporte de ligação por *Bluetooth*.

Foi realizada uma pesquisa de soluções de painéis de LED’s RGB com microcontrolador integrado, na qual se concluir que existe pouca oferta para este tipo de solução e das soluções encontradas não se encontra disponível informação sobre o seu funcionamento interno.

Para o desenvolvimento do software foram pesquisados quais os IDE disponíveis para o microcontrolador selecionado para a realização do projeto. Nesta matéria existem vários tipos de ambientes identificados como o ESP-IDF, MicroPython, Arduino e o NodeMCU. Das quais se conclui que a plataforma Arduino (7) é a que possui maior documentação, comunidade de suporte e maior número de bibliotecas.

# Análise de Requisitos

Neste capítulo é apresentada uma visão geral teórica acerca dos vários componentes e as técnicas de controlo que compõem o projeto.

Por forma a identificar as funcionalidades do sistema, definição dos requisitos, analisa-se e descreve-se detalhadamente cada uma das funcionalidades, de modo a que as mesmas possam ser validadas e desse modo se poder optar pelos módulos mais adequados ao seu funcionamento.

## Funcionamento de uma Matriz de LED

Os LED’s são frequentemente organizados em forma de matriz e desse modo ficam dispostos em linhas e colunas. Com esta organização é possível controlar um maior número de LED’s utilizando menos pinos de seleção, como por exemplo um painel com uma matriz de 4 linhas e 4 colunas, caso não estivessem dispostos de forma matricial seriam necessárias 16 saídas para ativar cada um dos LED, mas com a sua construção matricial apenas são necessárias ativar 8 saídas, como é ilustrado na Figura 3.



Figura 3 - Matriz de LED's 4x4 com 16 LED’s

Utilizando esta técnica todos os LED’s de cada linha e coluna estão interligados e quando se liga um LED ativa-se a respetiva linha e coluna.

Este tipo de montagem pode levantar um problema caso seja pretendido ativar um determinado LED, ou seja, no caso de se querer ativar o LED da linha dois e coluna dois, que iria ligar o LED conforme está apresentado na Figura 4.



Figura 4 – Seleção de LED da matriz

Agora, caso se opte também por ativar o LED da linha quatro e coluna três, iria ativar outros dois LED’s que não há interesse em ativar, conforme demonstrado na Figura 5.



Figura 5 – Seleção de 2 linhas e 2 colunas da matriz

Esta situação ocorre, pois, está a ser imposta uma tensão nas linhas dois e quatro, e nas colunas dois e três. Para resolver estes casos foi desenvolvida a multiplexagem.

A multiplexagem consiste em ativar uma linha de cada vez. Ao selecionar a coluna com o LED que se pretende ligar, os LED’s da linha ativada serão ligados. A linha é então desligada, ligando a linha seguinte repetindo o processo. Este processo torna-se num ciclo, repetido da primeira à última linha.

A frequência com que cada linha é ativada é chamada de *refresh rate* e caso esta tenha uma frequência suficientemente elevada, não permite que o olho humano se aperceba da transição, o que origina que quando se observa se verifiquem um conjunto de LED ligados.

### Painel RGB

Para o desenvolvimento do projeto optou-se por um painel com a correspondência de ativação dos LED’s na matriz não linear e com uma *refresh rate* de ¼. Na Tabela 2 estão apesentadas as características técnicas do painel escolhido.

Tabela 2 - Carateristicas Técnicas Painel SDM P10RGB



O painel utilizado é composto por 512 LEDs RGB distribuídos numa matriz de 16 linhas e 32 colunas.

Para ativar os LED’s da matriz o painel possui 8 *shift registers* de 64 bits que têm a responsabilidade sobre cada côr da matriz de LED’s e estão divididos entre a metade superior do painel e a metade inferior, cada uma é composta por 8 linhas e 32 colunas. Cada *shift register* tem um latch (LAT), *active high*, e um *output enable* (OE), *active low* assíncronos.

Uma imagem com mapa

Descrição gerada automaticamente

Figura 6 - Esquema Eléctrico Shift Register

Ao colocar o LAT com o valor ‘0’ e o OE com o valor ‘1’ irá permitir que os dados fiquem à saída do *shift register* e ao mesmo tempo desativar a saída de modo a que nenhum LED fique ativo, este estado permite que o painel seja atualizado sem que seja visualizado a deslocação dos bits e consequente ativação dos LEDs. Os *shift register* presentes no painel seguem a filosofia FIFO, o primeiro bit a entrar é o primeiro a sair.

As linhas são ativas com a utilização de 4 seletores, do tipo *active low*, identificados com A, B, C e D, onde cada um é responsável pela ativação de 4 linhas do painel. A distribuição das linhas é feita com 2 linhas na metade superior e 2 linhas na metade inferior do painel, que correspondem às 64 saídas do *shift register* de cada metade do painel, com a respetiva cor selecionada. São selecionadas 2 linhas de cada metade do painel devido à distribuição das 64 saídas do *shift register* no painel que têm de ser repartidas por 2 linhas de 32 colunas, estas saídas estão distribuídas no painel de uma forma não linear como se verifica na Figura 7.



Figura 7 - Distribuição de saídas Shift Register com correspondência de ativação

## Microcontroladores

Um microcontrolador é um computador dentro de um circuito integrado ou também conhecido como SOC. Com a capacidade de ser programado, permite ao projeto onde se utiliza uma grande adaptabilidade.

Na escolha de um microcontrolador há determinados aspetos a ter em consideração e que influenciam a sua escolha:

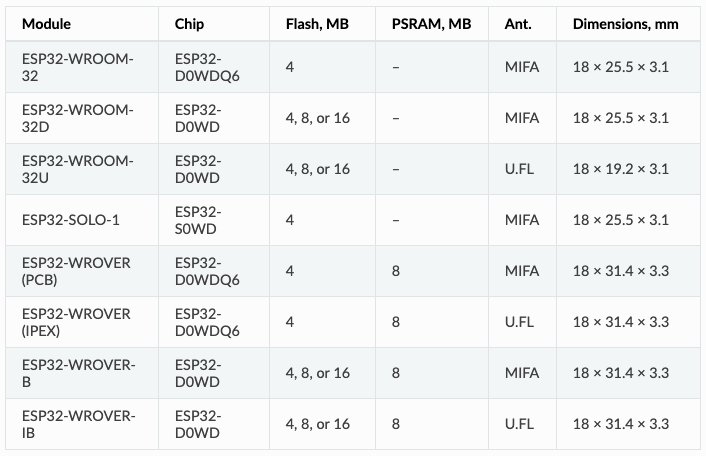
* Dimensão da memória RAM: é utilizada pelo programa principal para executar as suas rotinas. Quanto maior a memória RAM maior vai ser a sua capacidade de trabalho.
* Dimensão da memória FLASH: é utilizada para armazenar o programa e não necessita de fonte de alimentação para guardar o seu conteúdo.
* Número de Pinos de Entrada e Saída (GPIO): para se poder receber dados, processá-los e enviar para uma saída. O número de pinos pode ser importante quando se está a desenvolver um projeto.
* Conectividade: para se poder estabelecer comunicação com outros instrumentos é importante verificar os tipos de conexão que são possíveis, além da comunicação por cabo, também é importante para alguns projetos a conexão sem fios.

### Microcontrolador

O microcontrolador da Espressif ESP32 tem num circuito integrado o microcontrolador, Wifi e Bluetooth.

A família de microcontroladores ESP32 tem diferentes módulos como o WROOM, SOLO, WROVER e PICO. As principais características destes módulos são descritas na Tabela 3.

Tabela 3 - Diferentes Módulos ESP32



Para o desenvolvimento do projeto o módulo utilizado foi o ESP32-WROOM-32, integrado numa placa desenvolvida pela Adafruit Huzzah Feather. Esta placa tem incorporado um conversor USB-to-Serial, *bootloader reset* automático, conexão para bateria *Lithium Ion/Polymer* e todos os GPIO disponíveis no microcontrolador.

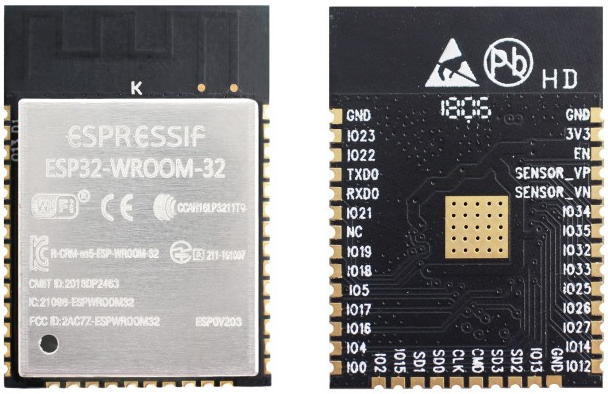
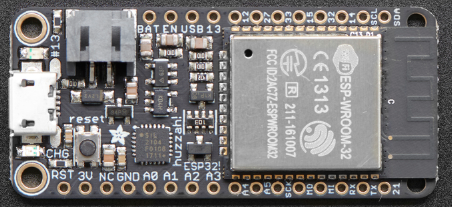


Figura 8 - Adafruit Huzzah32 - ESP32 Feather

Figura 9 - Módulo ESP32-WROOM-32

As características e as interfaces para periféricos deste módulo da Espressif são descritas:

* Processadores:
  + CPU: Xtensa dual-core (ou single-core) de 32 bits LX6 microprocessador, operando em 160 ou 240 MHz e realizar até 600 DMIPS
  + Ultrabaixa potência (ULP) co-processador
* Memória: 520 KB SRAM
* Conectividade sem fio:
  + Wi-Fi: [802.11](https://pt.wikipedia.org/wiki/IEEE_802.11) b/g/n
  + Bluetooth: v4.2 BR/EDR e BLE
* Interfaces para periféricos:
  + 12-bit SAR ADC até 18 canais
  + 2 × 8-bit [DACs](https://pt.wikipedia.org/wiki/Conversor_digital-anal%C3%B3gico" \o "Conversor digital-analógico)
  + 10 × Sensores de toque (GPIOs de [deteção capacitiva](https://pt.wikipedia.org/wiki/Sensores_capacitivos))
  + Sensor de temperatura
  + 4 × [SPI](https://pt.wikipedia.org/wiki/Serial_Peripheral_Interface)
  + 2 × Interfaces I²S
  + 2 × Interfaces [I²C](https://pt.wikipedia.org/wiki/I%C2%B2C)
  + 3 × [UART](https://pt.wikipedia.org/wiki/USART)
  + Controladores de host [SD](https://pt.wikipedia.org/wiki/Secure_Digital_Card)/[SDIO](https://pt.wikipedia.org/wiki/Secure_Digital_Card)/CE-ATA/[MMC](https://pt.wikipedia.org/wiki/MultiMediaCard)/eMMC
  + Controlador escravo SDIO/SPI
  + Interface [Ethernet](https://pt.wikipedia.org/wiki/Ethernet) MAC da com DMA dedicado e Suporte [Protocolo IEEE 1588 com Tempo Preciso](https://pt.wikipedia.org/wiki/IEEE_1588)
  + [Barramento CAN](https://pt.wikipedia.org/wiki/Controller_Area_Network) 2.0
  + Controlador remoto infravermelho (TX/RX, até 8 canais)
  + Motor [PWM](https://pt.wikipedia.org/wiki/Modula%C3%A7%C3%A3o_por_largura_de_pulso)
  + LED [PWM](https://pt.wikipedia.org/wiki/Modula%C3%A7%C3%A3o_por_largura_de_pulso) (até 16 canais)
  + [Sensor de efeito Hall](https://pt.wikipedia.org/wiki/Sensor_de_efeito_Hall)
  + Pré-amplificador analógico de ultrabaixa potência

## Ambiente de Desenvolvimento Integrado

Um ambiente de desenvolvimento integrado é um software que integra várias ferramentas numa só aplicação.

* Editor de Texto
* Compilador
* Programador

### Arduino IDE

Esta plataforma é caracterizada pelo seu conceito de *sketch*, que é o *upload* da unidade de código para ser processado no microcontrolador.

Ao contrário de um programa tipo na linguagem C, não existe a função de entrada *Main*, esta é substituída por uma função de inicialização de periféricos e variáveis com o nome *Setup* que é executada no arranque do *sketch* e uma função *Loop*, que como o seu nome sugere repete-se consecutivamente enquanto a placa estiver ligada, permitindo ao programa mudar e responder a essas mudanças. É usada para controlar ativamente um microcontrolador.

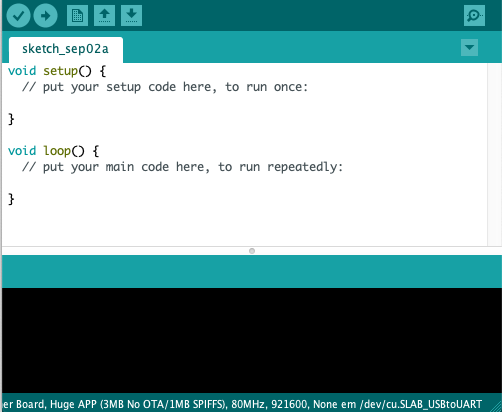


Figura 10 - Sketch Arduino

## Requisitos Técnicos

RT01: A alimentação dos componentes do painel de LED RGB deve ser garantido com uma fonte de alimentação externa adequada ao painel.

O consumo de corrente caso não venha explicitado no manual de utilização do painel poderá ser dado pela expressão em baixo:

RT02: A alimentação do microcontrolador deverá ser feita por USB 2.0 no mínimo.

RT03: Painel de RGB modular com montagem de LED SMD.

RT04: O microcontrolador deverá ter disponível os GPIO necessários \*\*\*, deverá ter conexão WIFI a um router e Bluetooth local.

# Implementação

Neste capítulo descreve-se a implementação no microcontrolador os algoritmos para o processamento de dados, o controlo do painel RGB, analisando assim as várias camadas que compõem a aplicação, estruturas de dados desenvolvidas e a interação entre as várias peças desenvolvidas, usando o estilo *top-down*.



Figura 11 - Diagrama Lógico da aplicação e hardware

Para o desenvolvimento do *software* foi dividido o código em camadas lógicas, sendo esta uma forma de dividir responsabilidades, integrar bibliotecas nativas e tornar a aplicação portável para outras plataformas. O código desenvolvido neste projeto segue a orientação representada na Figura 11.

* Camada Aplicacional – Onde é aplicada toda a lógica da aplicação e o processamento de eventos.
* Camada Intermédia – Nesta camada estão presentes as bibliotecas que permitem a comunicação da aplicação com o painel LED, comunicação com uma rede *Wifi* usando protocolo 802.11 b/g/n, comunicação *Bluetooth* usando protocolo v4.2 BR/EDR e acesso à Eeprom.
* Camada Física – Esta camada é constituída pelo microcontrolador ESP32 e por um painel de LED’s RGB.

## Camada Aplicacional

### Inicialização do sistema

A aplicação começa por inicializar o hardware dentro da função *Setup*, na qual se efetua uma leitura da Eeprom com o propósito de se obter o ssid e a password da última ligação estabelecida. De seguida é efetuada a inicialização do painel, onde são definidos os pinos de output do microcontrolador que irão fazer a conexão ao painel e inicializa os *shift registers* do painel com valor ‘0’. Por fim procede à inicialização de uma tarefa (*task*), que é responsável pelo refrescamento continuo da matriz do painel, como demonstrado no fluxograma presente na Figura 12.



Figura 12 - Fluxograma Função Setup

Após o *Setup* entra em execução a função *Loop,* a qual como é referido anteriormente é chamada continuamente, onde são executadas as ações do fluxograma presente na figura \* a cada iteração.

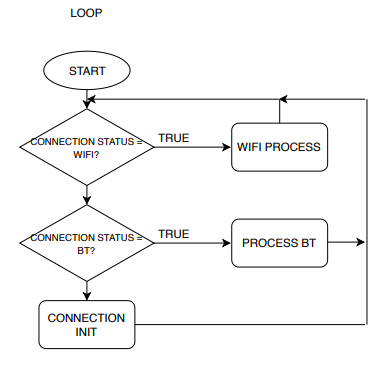


Figura 13 – Função Loop

### Gestor de Ligação

Em cada iteração é analisado o estado de uma variável e executada a ação correspondente. Quando não encontra nenhuma ligação válida, por defeito executa a função *Connection Init*. Esta função é responsável por estabelecer uma tentativa de ligação por *Wifi* ou por *Bluetooth*, como representado na figura \*.



Figura 14 – Fluxograma da Função Connection Init

Esta função começa por verificar se existem condições para realizar uma ligação *Wifi*, a verificação é feita durante vinte iterações. Durante as tentativas se for estabelecida uma ligação com sucesso, apresenta os dados da ligação estabelecida e guarda na *Eeprom* os valor do *ssid* e da *password* correspondentes à ligação.

Caso sejam realizadas as vinte iterações e não tiver sido estabelecida uma ligação por *Wifi* é chamada a função *Connection Using BT* cujo retorno afeta a variável de estado com o valor correspondente ao tipo de ligação estabelecida.

A função *Connetion Using BT* é responsável pela ativação do *Bluetooth* e o seu emparelhamento com um dispositivo que irá permitir uma ligação série e onde se configura o tipo de ligação, *Bluetooth* ou *Wifi*.

Como ilustrado na figura \*, começa-se por inicializar o *Bluetooth* e aguarda uma ligação. Quando o emparelhamento é estabelecido, é criada uma ligação série que permite a comunicação bidirecional entre o dispositivo e o microcontrolador.

A aplicação aguarda que seja inserida uma linha de texto no dispositivo com o comando a executar. Ao inserir um comando, verifica se o comando inserido é “BLUE” e neste caso, mantem a ligação *Bluetooth* ativa e retorna o valor a indicar que está estabelecida uma ligação *Bluetooth*, caso seja inserido outro comando, verifica se os primeiros quatro caracteres são “SSID” e no caso de se verificar esta condição os caracteres inseridos após o quarto caracter são copiados para a variável global denominada “ssid”, após este passo pode ser inserida a *password*, escrevendo “PASS” seguido da *password* onde os caracteres inseridos após o quarto caracter são copiados para a variável global denominada “password”.

Validada a inserção do “SSID” e “PASS” é retornado o valor a indicar que se irá realizar a tentativa de ligação por *Wifi.*



Figura 15 - Fluxograma Função Connect Using BT

### Processamento de pedidos HTTP

Após o retorno da função *Connection Init*, é executado o processamento correspondente com o valor da variável de estado.

No caso de a variável de estado ter o valor *Wifi*, é efetuado o processamento da função *Wifi Process*, cujo fluxograma se apresenta na figura 13.

A função *Wifi Process* tem como responsabilidade o processamento da *query String* recebida através dos pedidos realizados pelo utilizador através do protocolo HTTP.



Figura 16 - Fluxograma Wifi Process

Ao entrar na função é verificada se a ligação *Wifi* está ativa, no caso da ligação se ter perdido é retornado o valor a indicar que não existe ligação. Se a ligação estiver ativa é criado um cliente, após a criação do mesmo é realizada uma tentativa de ligação a um determinado endereço de IP e a um porto definido. Com a ligação estabelecida é feita uma verificação se existem carateres disponíveis para serem recebidos, caso hajam carateres disponíveis é feita uma leitura dos mesmos, onde se realiza um *parse* sobre a *string* lida, para ser convertida de modo a poderem ser processados os dados para a matriz. Após a leitura de todos os *bytes* disponíveis, ou se não existirem *bytes* disponíveis para receber entra em execução a função *Scroll Display,* como representado no fluxograma da figura \*, que tem como função tratar o deslocamento dos caracteres horizontalmente e verticalmente. Quando a ligação com o cliente deixar de estar disponível, o cliente é desligado e a função retorna o valor a indicar o valor *Wifi*.

### Processamento de pedidos *Bluetooth*

No caso de a variável ter o valor BT é efetuado o processamento da função *Process BT* cujo fluxograma é apresentado na figura \*.



Figura 17 - Fluxograma da Função Process BT

A função *Process BT* tem como responsabilidade gerir a informação enviada pelo utilizador para ser transmitida no painel através da ligação por *Bluetooth*.

Após entrar na função é verificado se existem carateres disponíveis no buffer de receção, caso existam é realizada a leitura dos mesmos e é feita uma verificação se a *String* lida é igual à *String* “WIFISTART”. Se a verificação for falsa a função retorna o valor de NCON a indicar que a ligação por *Bluetooth* será interrompida, caso contrário a *String* lida é processada de modo a poder ser transmitida no painel. Após o processamento de todos os dados entra em execução a função *Scroll Display*.

## Camada Intermédia

### Deslocamento de texto

A função *Scroll Display* como referido anteriormente é responsável pela gestão do deslocamento horizontal, quando existem mais caracteres que colunas disponíveis e vertical, quando existem menos linhas disponíveis que as que são pretendidas pelo utilizador.

Colocar figura ilustrativa scroll



Figura 18 - Fluxograma Função Scroll Display

Quando a função é executada, existe uma variável *TimeStamp* que é usada para verificar se decorreu o tempo definido após a última iteração. Com este processo é possível manter o ritmo de deslocação do texto no painel de LED’s, deste modo é atualizada a posição da matriz horizontalmente e verticalmente. Após a atualização das variáveis de posição, entra em execução a função que faz a conversão de matrizes.

### Atualização de matrizes

Como existem diversos tipos de painéis, com diferentes tipos de matrizes e diferentes distribuições de LED’s no painel, tal como demonstrado na figura 5, onde a distribuição dos LED’s na matriz não segue uma distribuição linear, foi necessário criar um módulo *Matrix Driver*, cujo fluxograma está representado na figura\*, que tem como função a conversão de matrizes e o envio da matriz final para o painel de LED’s.



Figura 19 - Fluxograma Matrix Driver

Foi necessário criar uma matriz (*displayRGB*) que guarde o bitmap correspondentes a cada caracter visível no painel. A dimensão desta matriz define o número de caracteres visíveis no painel com base na dimensão do *bitmap* de cada caracter. Caso a sequência de *bitmaps* for superior à dimensão do painel de LED’s, o driver efetua um deslocamento sobre esta matriz de forma a mostrar os restantes *bitmaps* presentes nesta matriz.

Como referido no ponto 3.1.1 devido à estrutura do painel de LED’s, não é possível fazer um varrimento direto da matriz *displayRGB* para o painel, para esse feito é necessário realizar uma conversão para uma matriz intermédia (*arrayTX*) com base numa tabela de máscaras (*convTable*). A leitura da matriz *displayRGB* é feita a partir do índice linha / coluna dada pelos valores das variáveis *scroll* vertical e *scroll* horizontal.

Com a matriz *convTable* é possível mapear a matriz *displayRGB* na matriz *arrayTX*, como por exemplo o mapeamento da posição, na matriz *displayRGB*, linha 7 coluna 10, corresponde na matriz *ArrayTX*, à linha 3 coluna 38, como apresentado na figura \*. Antes de se desenvolver a matriz intermédia de conversão, começou por se desenvolver um algoritmo que fizesse diretamente essa função, e verificou-se que o algoritmo encontrado obrigava o programa a fazer vários ciclos fazendo com o programa ficasse mais lento e pesado. Cada elemento da matriz *convTable* é do tipo inteiro (*Int*) no qual permite guardar 2 *bytes* em que cada um dos *bytes* corresponde a um par linha/ coluna da matriz *arrayTX*, onde na parte alta é guardado o valor da linha correspondente e na parte baixa a posição da coluna que está atribuída.





Após a matriz *arrayTX* estar completa estão reunidas as condições necessárias para o envio dos dados para o painel. Devido às características dos *shift registers* do painel, é enviado o conteúdo da matriz *arrayTX* em vectores de 64 bytes correspondente a oito *shift registers*.

A conversão dos caracteres em *bitmaps* é da responsabilidade do módulo *Text Interface*. Neste módulo está presente a fonte dos caracteres que é convertida para um *bitmap* com base na tabela ASCII do caracter, como representado na figura \*. Sendo que existem várias bibliotecas de fonte disponíveis na internet, optou-se por utilizar uma já desenvolvida, a qual possui os caracteres necessários para a representação de texto no painel. A escrita de um caracter na matriz *displayRGB* consiste na cópia do *bitmap* do caracter ponto a ponto.



A

Figura 20 - Bitmap Caracter 'A'

# Resultados Experimentais

Neste capítulo é apresentado o resultado final do trabalho desenvolvido. São abordadas duas formas de se poder apresentar texto no painel, usando uma ligação *Bluetooth* ou utilizando uma ligação *Wifi*. A ligação por Wifi pode ser configurada utilizando uma ligação por Bluetooth.

O programa desenvolvido ocupa 1 356 807 bytes, divididos entre o rascunho do programa e as variáveis globais. O rascunho do programa está inserido na memória de armazenamento do programa e tem uma ocupação de 41% o que corresponde a 1 291 235 bytes. As variáveis globais estão inseridas na memória dinâmica e usam 20% do espaço disponível, o que corresponde a uma ocupação de 65 572 bytes, este espaço foi otimizado pois a matriz *convTable* ...

## Ligação *Bluetooth*

Ao não existir nenhuma ligação *Wifi* ativa, o sistema inicializa a antena *Bluetooth* e permite o emparelhamento com um dispositivo externo permitindo uma comunicação série com o qual é possível trocar mensagens. Para a realização deste projeto e para o emparelhamento e consequente ligação série, optou-se pela utilização da aplicação *BluTerm*, este *software* foi obtido gratuitamente na plataforma *Apple Store*.

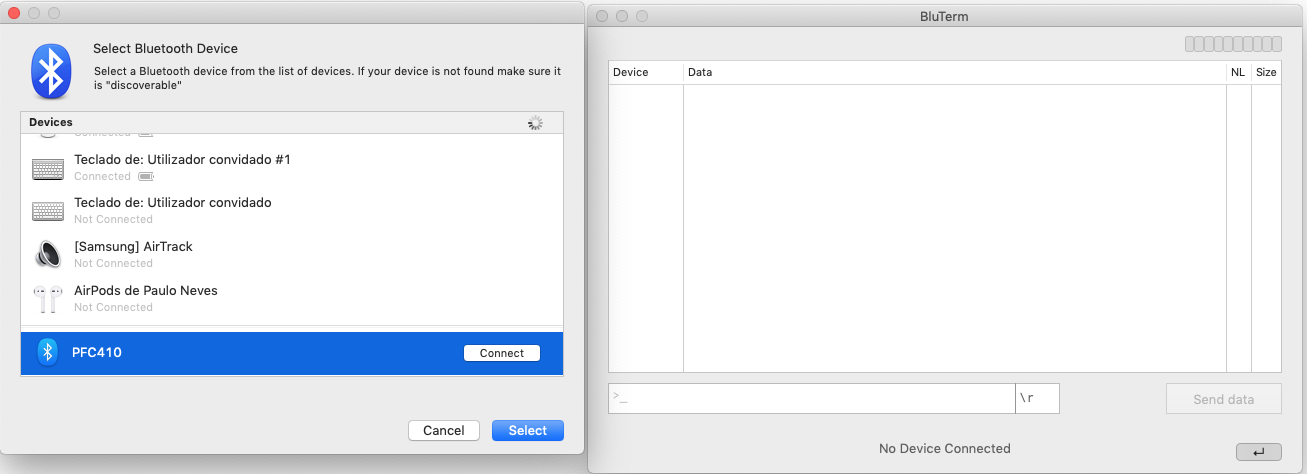


Figura 21 - Emparelhamento ESP32 com Dispositivo Externo utilizando a Aplicação BluTerm

O painel tem dois LED’s ativos que durante o período de estabelecimento de ligações que dão o *status* da ligação, onde a posição linha 1 coluna 1 dá a indicação se a ligação por Wifi está estabelecida e a posição linha 1 coluna 2 dá dois tipos de indicação, se aguarda o tipo de ligação ou se a ligação por Bluetooth está ativa.

Com o emparelhamento concluído é enviada uma mensagem para o dispositivo a informar que aguarda a inserção dos comandos, que neste caso será o comando “BLUE” para a reprodução de texto no painel utilizando a ligação Bluetooth. Verifica-se na figura \* que o painel muda a cor do LED da linha 1 coluna 2 para rosa, o que significa que aguarda a inserção do tipo de ligação pretendida.

|  |  |
| --- | --- |
|  |  |

Figura 22 - Aguarda inserção do tipo de ligação

Ao selecionar a ligação por Bluetooth utilizando o comando “BLUE” verifica-se na figura \* a mudança de cor do LED da linha 1 coluna 2 para azul.

|  |  |
| --- | --- |
|  |  |

Figura 23 - Ligação por Bluetooth ativa

Após estabelecida a ligação por Bluetooth é possível enviar a mensagem que se pretende mostrar no display. A *String* com a mensagem deverá respeitar alguns critérios, como apresentado na *String* seguinte.

“Linha1=MENSAGEMLINHA1&Linha2=MENSAGEMLINHA2&Cor1=NumeroCor&Cor2=NumeroCor”

Mudança de texto usando o browser

## Ligação Wifi via Bluetooth

# Conclusões

Ponto de melhoria

# Bibliografia

1. **IDE. [Online] https://pt.wikipedia.org/wiki/Ambiente\_de\_desenvolvimento\_integrado.**

**2. Microcontrolador. [Online] https://pt.wikipedia.org/wiki/Microcontrolador.**

**3. Rapberry Pi. [Online] https://www.raspberrypi.org/.**

**4. ESP32. [Online] https://pt.wikipedia.org/wiki/ESP32.**

**5. Programming ESP32 Board with Arduino IDE. [Online] https://circuitdigest.com/microcontroller-projects/programming-esp32-with-arduino-ide.**

1. Texas Instruments: 16-CHANNEL CONSTANT-CURRENT LED SINK DRIVERS
2. Sanders Full Color LED Module: P10 RGB Module

# Anexos

## Anexo A – Esquema ESP32 Adafruit Feather

Uma imagem com texto

Descrição gerada automaticamente

Figura 24 - Esquema Eléctrico ESP32 Adafruit Feather

Uma imagem com captura de ecrã, mapa

Descrição gerada automaticamente

Figura 25 - Pinout ESP32 Adafruit

## Anexo B – Esquema Eléctrico

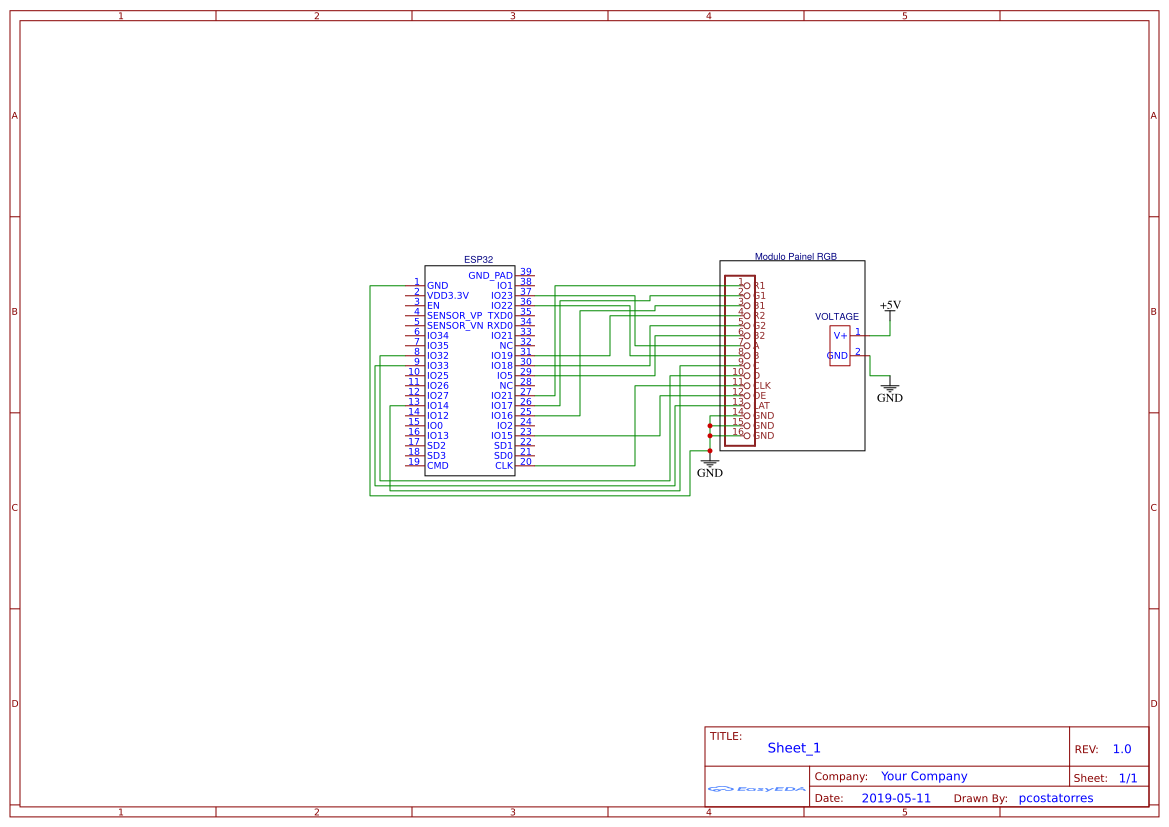


Figura 26 - Esquema Eléctrico