Disciplina:	ME587 - Sistemas Digitais 2
Professor:	José Rodrigues de Oliveira Neto

Universidade Federal de Pernambuco
Centro de Tecnologia e Geociências
Departamento de Engenharia Mecânica

Lista de Exercícios #02 - 2024.1

O objetivo desta lista é configurar as entradas e saídas da placa BluePill[1] com os respectivos periféricos presentes no *Multifunction-Shield* Arduino[2] que iremos utilizar como expansão para nossa placa de desenvolvimento.

1 Instruções Gerais

- Sempre comente seu código e identifique o que ele faz e quem foi que fez (você, no caso);
- Coloque todas as pastas dos projetos criados em uma única pasta "`LE02_SEU_NOME`", em que `SEU_NOME` é o seu nome;
- Compacte a pasta e inclua como resposta da atividade.

2 Exercícios

Os exercícios dessa lista funcionarão como um passo-a-passo para conseguirmos nossos objetivos:

- **Conhecer como configurar entradas e saídas na CubeIDE;**
- **Conhecer mais sobre o *shield* multi-funções e configurar suas entradas e saídas;**
- **Introdução a programação em *bare-metal* utilizando a CubeIDE.**

2.1 Configurando o Projeto

Nesta etapa, vamos primeiramente refazer as configurações básicas que foram feitas na aula anterior:

1. Crie um novo projeto e o nomeie `LE02_Q1_SEU_NOME`, em que `SEU_NOME` é o seu nome;
2. Configure o placa para utilizar o cristal externo para gerar o *clock* do sistema;
3. Configure o sistema para trabalhar a 72 MHz;
4. Configure a forma de depuração para ser pela interface *Serial Wire*;
5. Configure o `PC13`, nosso *UserLed*, como saída;
6. "*Builde*" o projeto;
7. Grave na placa para garantir que todas as configurações estão corretas.

Se tiver dúvidas em qualquer dos passos, revisite o material da aula e/ou acesse a videoaula-VA03-CISE (PGEE918) - Primeiros Passos com a BluePill na STM32 CubeIDE: https://youtu.be/wBz28S6aEl8.

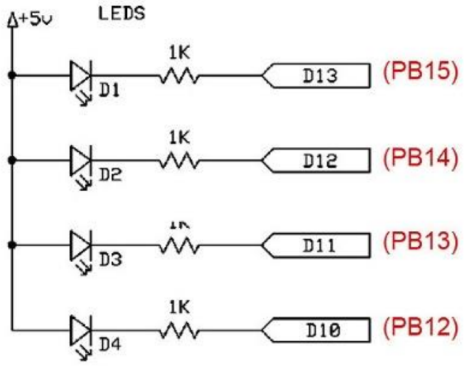
[1] Esquemático: https://stm32-base.org/assets/pdf/boards/original-schematic-STM32F103C8T6-Blue_Pill.pdf.

[2] Mais sobre: https://blog.eletrogate.com/guia-completo-do-shield-multi-funcoes-para-arduino/.

2.2 Configurando os Leds do *Shield* Arduino

A *Shiel*-Multifunções possui quatro Leds, cujo esquema é mostrado na Figura 1. Como é possível ver, os leds são ligados no VCC (como ocorre com o nosso *UserLed* da BluePill), logo, ligamos o led colocando 0 no pino indicado. A ligação entre os pinos com os correspondentes na placa BluePill é dado na Tabela 1

Figura 1: Esquemático dos Leds do *Shield*-Multifunções Arduino.



Fonte - Adaptado de (RANHEL, 2019, p. 18).

Tabela 1: Ligação entre os Pinos dos Leds no *Shield*-Multifunções Arduino e na Placa BluePill.

LED	*Shield*	BluePill
D1	D 13	PB15
D2	D 12	PB14
D3	D 11	PB13
D4	D 10	PB12

Fonte - O autor.

2.2.1 Exercício Proposto

Com esse conhecimento, utilizando o mesmo projeto, `LE02_Q1_SEU_NOME`:

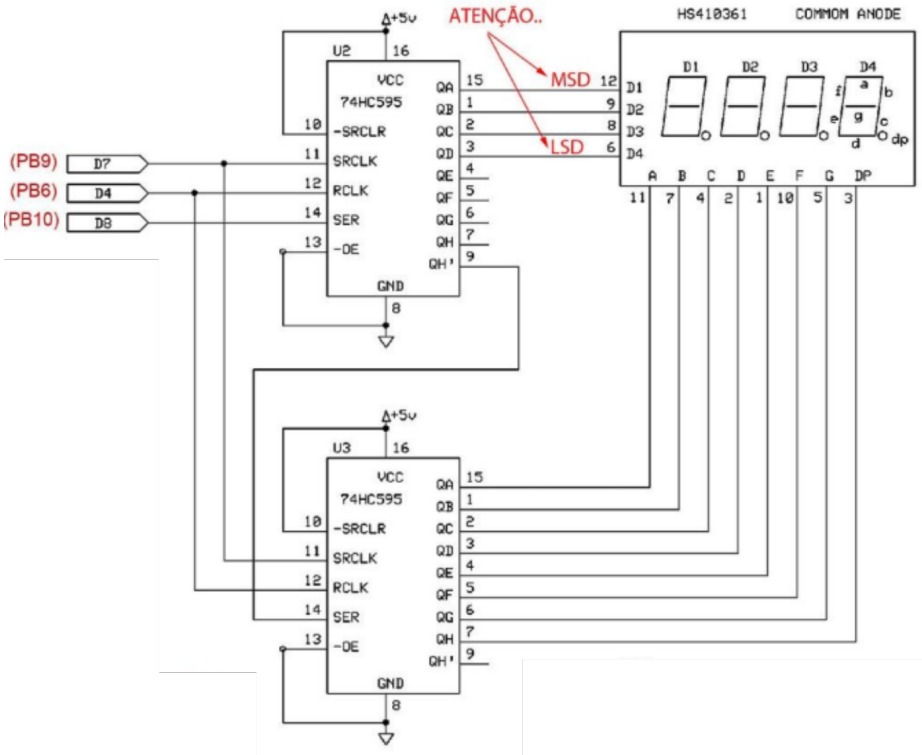
1. Configure os pinos relativos aos leds do *shield*-multifunções como saída;
2. Em **Pinout & Configurations → GPIO**, coloque "apelidos" neles (chamando-os de `D1`, `D2`, `D3` e `D4`), como feito para o `UserLed` na aula e coloque para eles começarem em nível lógico alto.
3. "Builde" o projeto;
4. Grave-o na placa;

É esperado que os quatro leds da placa, assim como o `UserLed`, fiquem desligados. Veja o corpo da função `MX_GPIO_Init`, será simples entender o porquê.

2.3 Configurando os Displays de Sete Segmentos

O circuito do Display usa um módulo com 4 dígitos (7 segmentos – anodo comum). Os segmentos são ativados através da multiplexação realizada pelos dois chips de Registradores de deslocamento (Shift registers) 74HC595[3] cujo esquemático é mostrado na Figura 2 e a pinagem é dada na Tabela 2.

Figura 2: Esquemático do Display do *Shield*-Multifunções Arduino.



Fonte - Adaptado de (RANHEL, 2019, p. 18).

Tabela 2: Ligação entre os Pinos do Display no *Shield*-Multifunções Arduino e na Placa BluePill.

74HC595	*Shield*	BluePill
SRCLK	D 7	PB9
RCLK	D 4	PB6
SER	D 8	PB10

Fonte - O autor.

2.3.1 Exercício Proposto

Com esse conhecimento, utilizando o mesmo projeto, `LE02_Q1_SEU_NOME`:

1. Configure os pinos relativos aos pinos do Display como saídas;
2. Configure-os com os nomes adequados (`SRCLK`, `RCLK` e `SER`);
3. "Builde" o projeto;

[3] *Datasheet*: https://www.ti.com/lit/ds/symlink/sn74hc595.pdf?ts=1618420630363&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FSN74HC595.

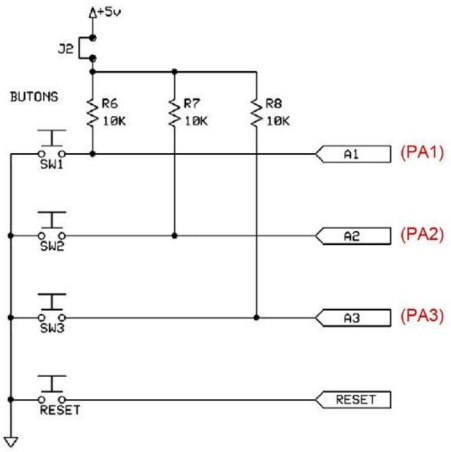
4. Grave-o na placa;

Trabalharemos futuramente para criar funções para escrever no Display. Agora o mais importante é garantir a configuração dos pinos.

2.4 Configurando os Botões de Entrada

Nosso *shield*-multifunções possui 4 botões, 3 para uso geral (`SW1`, `SW2`, `SW3`) e um para resetar a placa. O esquemático é mostrado na Figura 3 e a pinagem é dada na Tabela 3.

Figura 3: Esquemático dos Botões do *Shield*-Multifunções Arduino.



Fonte - Adaptado de (RANHEL, 2019, p. 18).

Tabela 3: Ligação entre os Pinos dos Botões no *Shield*-Multifunções Arduino e na Placa BluePill.

CHAVES	*Shield*	BluePill
SW1	A1	PA1
SW2	A2	PA2
SW3	A3	PA3

Fonte - O autor.

2.4.1 Exercício Proposto

Com esse conhecimento, utilizando o mesmo projeto, `LE02_Q1_SEU_NOME`:

1. Configure os pinos relativos aos botões como entradas (`GPIO_Input`);
2. Configure-os com os nomes adequados (`SW1`, `SW2` e `SW3`);
3. "Builde" o projeto;
4. Grave-o na placa;

No futuro voltaremos a esses botões para configurá-los como interrupções de pinos externos. Assim como configurar os demais componentes da *shield*-multifunções. Se quiser ver a configuração de todos os pinos, o esquemático está disponível na página 18 de (RANHEL, 2019). Aproveite, veja qual o pino da buzina e o configure como uma saída. Lembre-se que o CubeMX sempre inicializa resetando os pinos de saída, logo, para não ficar com a buzina apitando, você tem que colocar a saída para iniciar com nível lógico alto.

2.5 Copiando as Configurações de um Projeto para um Novo Projeto

Para não termos que configurar sempre todos os pinos de entrada e saída, clock do sistema, etc. É possível criar um projeto copiando o arquivo de configurações (arquivo com a extensão .ioc). Para isso:

I. No menu superior e vá em:

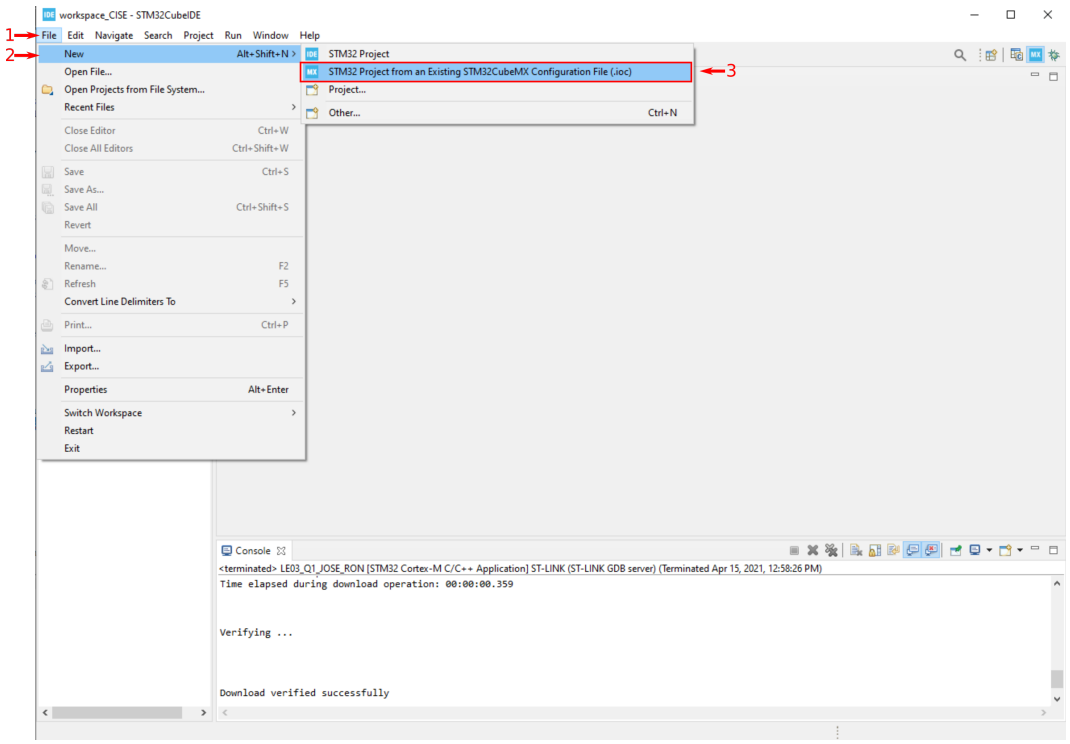
1 **File**;

2 **New**;

3 **STM32 Project from a Existing STM32CubeMx Configuration file (.ioc)**.

Como mostrado na Figura 4;

Figura 4: Criando um novo projeto copiando a configuração de outro.



II. Irá abrir a janela mostrada na Figura 5, nela vá em:

1 **Browse...**, e caminhe nas pastas até o arquivo `LE2_Q1_SEU_NOME.ioc`, dentro da pasta `LE2_Q1_SEU_NOME`;

2 Escreva o nome do novo projeto no campo **Project Name:**, e coloque o nome `LE2_Q2_SEU_NOME`, em que textttSEU_NOME é o seu nome;

3 Clique em **Finish** para gerar o projeto.

Como mostrado na Figura 5.

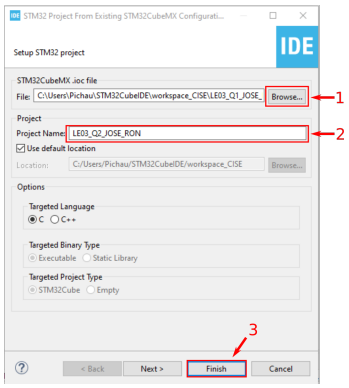
III. Será gerado o projeto com mas mesmas configurações do projeto copiado, dado isso:

1 Compile o projeto;

2 Grave na placa;

OBS: Criar um projeto a partir de outro apenas copia as configurações de placa e periféricos. Não copia o código feito pelo usuário!!!

Figura 5: Criando um novo projeto copiando a configuração de outro.



2.6 Fazendo algum código

Para testas algumas das entradas e saídas implementadas, faça um projeto cujos LEDs seguem 3 comportamentos diferentes:

1. Se o botão `SW1` estiver pressionado, os leds `D1`, `D2`, `D3` e `D4` ficarão piscando (invertendo o valor) todos com o mesmo valor a cada 500 ms;
2. Se o botão `SW2` estiver pressionado, os leds `D1` e `D3` devem piscar 100 ms acessos, 500 ms apagados; e `D2` e **D4** devem piscar 300 ms apagados, 300 ms acesos (nessa ordem);
3. Se o botão `SW3` estiver pressionado, apenas um led deve ficar acesso a cada 500 milissegundos seguindo a ordem: `D1` → `D2` → `D3` → `D4` → `D1` ... ;
4. Deixe o `UserLed` da BluePill sempre piscando com período: 50 ms aceso, 950 ms apagado.

2.6.1 Observações

- Para facilitar, utilize que `SW1` tem precedência sobre `SW2` que tem precedência sobre `SW3`;
- Caso nenhum dos botões esteja pressionado, os leds `D1` a `D4` devem permanecer com os valores anteriores (**ou seja, eles só mudam se um dos botões está pressionado**);
- Para ler o estado de um pino, utilize a função

  `HAL_GPIO_ReadPin (GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin)`,

  se tiver dúvida de seu funcionamento, veja o documento **Description of STM32F1 HAL and low-layer drivers**;

2.7 Delay não Blocante

Para todo o código anterior é possível fazer o código apenas utilizando a função `HAL_Delay(TEMPO)`, no entanto essa função é blocante. Ou seja, o microcontrolador fica parado sem fazer nada pelo tempo em ms estipulado em `TEMPO`. Assim com em outros *hardware abstraction layer* (HAL), o do ARM já possui uma função que devolve o tempo em milissegundos desde que a placa foi inicializada pela última vez, a função `HAL_GetTick()`.

2.7.1 Exercício Proposto

1. Refaça o projeto anterior substituindo a função `HAL_Delay(TEMPO)` por um modo de contar o tempo de forma não blocante utilizando a função `HAL_GetTick()`.

2. Para isso crie um novo projeto: `LE02_Q3_SEU_NOME` em que `SEU_NOME` é o seu nome.

3 Links Importantes

- MURTA, José Gustavo Abreu, "**Guia completo do shield multi-funções para Arduino**", Blog Eletrogate, 13 de junho de 2018, Disponível em: https://blog.eletrogate.com/guia-completo-do-shield-multi-funcoes-para-arduino/, acessado em 15 de abril de 2021.

- STM32-base, "**Blue Pill STM32F103C8T6**", STM32-base project website , Disponível em: https://stm32-base.org/boards/STM32F103C8T6-Blue-Pill.html, acessado em 15 de abril de 2021.

- RANHEL, J. , "**Apostila: Sistemas Microprocessados – ARM CORTEX**", UFABC, ver 5, 2019.