Departamento de Engenharia Elétrica
Universidade Federal do Ceará *Campus* Sobral

(SBL0082) Microprocessadores

Prof. Me. Alan Marques da Rocha
Prof. Dr. Marcelo Marques Simões de Souza

1

Prática 01: Familiarização e E/S

4.1 Introdução

Nesta prática, o aluno aprenderá a configurar e criar um projeto no MPLAB X IDE, escolhendo o microcontrolador correto (PIC18F4520), a ferramenta de programação (gravador) e o montador (*assembler*) adequado, que neste caso é o `mpasm`. Em seguida, será criado o arquivo-fonte principal do projeto, no formato `.asm`, onde o código de controle será desenvolvido. O uso de *Assembly* nesta etapa permite observar com clareza a relação direta entre registradores do microcontrolador (por exemplo, `TRISD`, `PORTD`) e o comportamento físico nos pinos da placa, reforçando a importância da configuração de entradas e saídas digitais.

Após a escrita do código, o projeto é compilado (*build*). Nesse processo, o MPLAB X converte o arquivo `.asm` em um arquivo de saída no formato `.hex`. Esse arquivo `.hex` contém o programa já traduzido para instruções binárias, no formato que pode ser gravado diretamente na memória *flash* do PIC. Esse ponto é essencial: o microcontrolador não executa o arquivo `.asm`, mas sim o conteúdo convertido que está no `.hex`.

A etapa seguinte consiste na gravação do *firmware* no microcontrolador. Para isso, utiliza-se um gravador externo ligado ao computador. O gravador recebe o arquivo `.hex` e injeta esse conteúdo na memória de programa do PIC18F4520. Esse procedimento é denominado programação do microcontrolador. Após a gravação, o PIC pode ser removido do soquete do gravador, inserido novamente na placa UFC PICLAB-4520 e colocado em funcionamento para observação prática dos resultados.

Do ponto de vista pedagógico, esta prática consolida três conceitos fundamentais da disciplina de Microprocessadores: (i) a distinção entre software embarcado e hardware físico, (ii) o papel da IDE e do *toolchain* (cadeia de ferramentas) na geração do executável, e (iii) o processo de transferência segura do programa compilado para o microcontrolador. Ao final, espera-se que o aluno seja capaz de identificar as principais pastas do projeto no MPLAB X, compreender a função do arquivo `.hex`, operar o gravador de forma correta e relacionar o código desenvolvido com o comportamento observável na placa (por exemplo, o piscar de um LED controlado via porta digital).

4.2 Objetivos

Familiarizar-se com o kit da placa UFC PICLAB-4520 e seus componentes realizando a gravação de um programa em `Assembly` utilizando a IDE MPLABX.

4.3 Lista de Materiais

- Um kit da placa UFC PICLAB-4520;
- 1 gravador PICKit2 para programação do microcontrolador;
- Computador com a IDE de desenvolvimento MPLABX.

4.4 Procedimento Experimental

4.4.1 Parte I: Inicialização do MPLABX e Criação do Projeto

- **Passo 1:** Inicialize o MPLABX e crie um novo projeto (File → New Project). Em seguida, em **Categories**, escolha `Microchip Embedded` e em **Projects** → `Standalone Project`, conforme ilustrado na Figura 4.1.

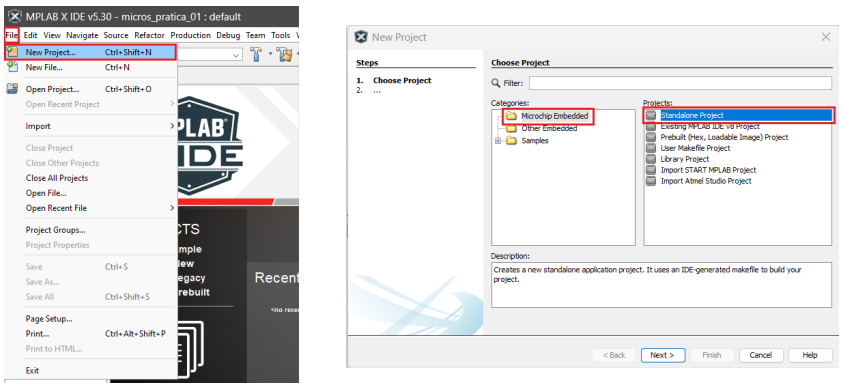

Figura 4.1: Criando um novo projeto no MPLAB X IDE.

- **Passo 2:** Na próxima etapa, para escolher o dispositivo (micro) da prática vá em **Device**, selecione o `PIC18F4520` e clique em **Next**, conforme ilustrado na Figura 4.2.

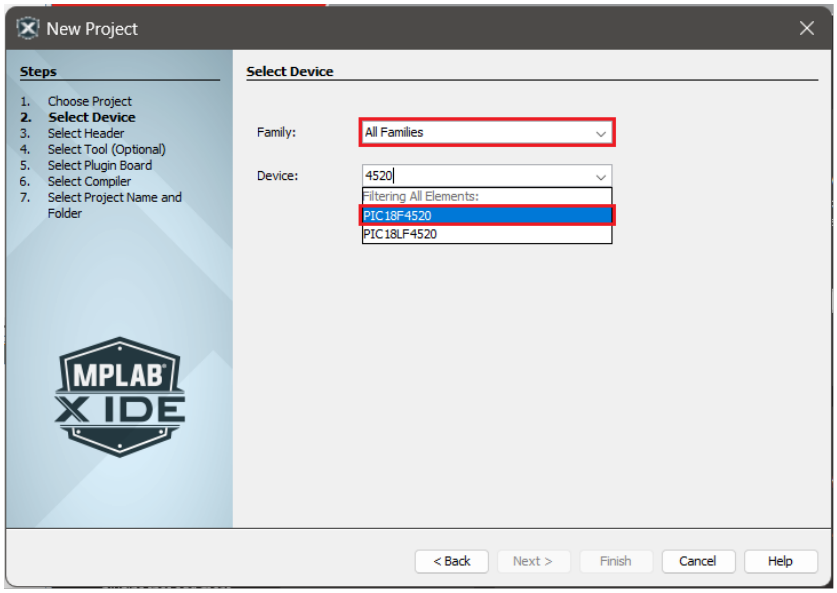

Figura 4.2: Escolhendo o PIC para a prática.

- **Passo 3:** Para escolher a ferramenta (**Select Tool**), vá na pasta `Alternate Tools` e escolha a opção `PICkit2`, conforme ilustrado na Figura 4.3. Em seguida, clique em **Next**.

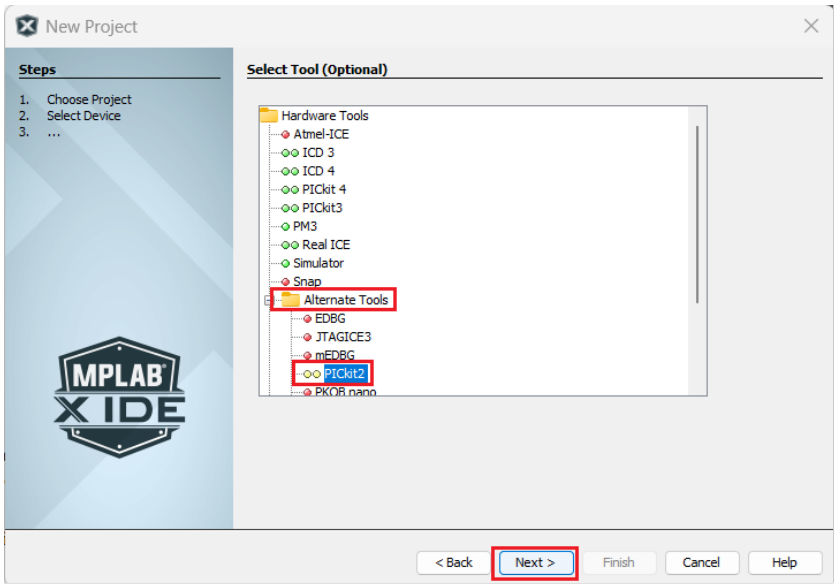

Figura 4.3: Escolha da ferramenta PICkit2 para gravação dos dados no PIC.

- **Passo 4:** Nesta prática, iremos trabalhar com o código em `ASSEMBLY (.asm)`,

então devemos escolher o compilador `mpasm` pré-instalado na IDE, conforme ilustrado na Figura 4.4.

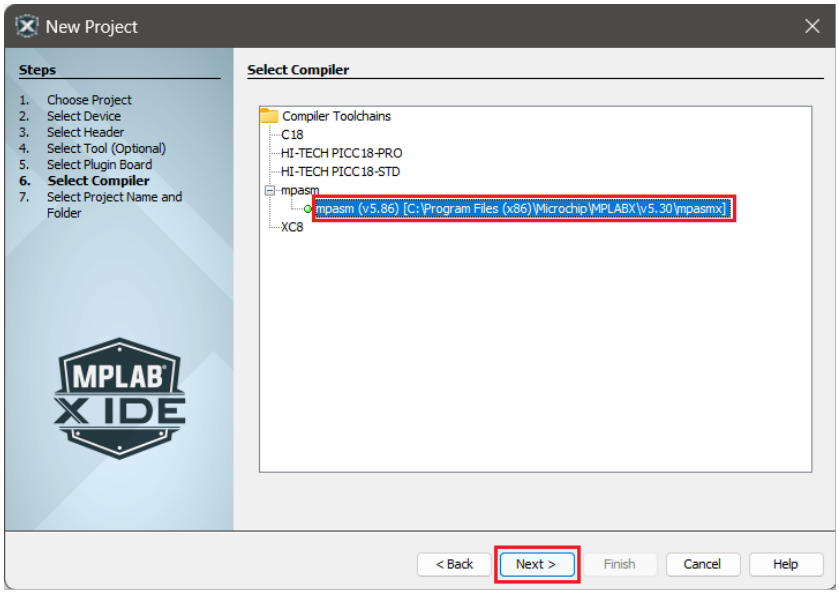

Figura 4.4: Escolhendo o compilador para o projeto.

Observe que o compilador `XC8` **não encontra-se instalado**. Este é o compilador para códigos escritos na linguagem C. Será necessário instalá-lo para as próximas práticas.

- **Passo 5:** Após escolher o compilador, escolha um novo para o projeto e inclua-o em `Project Name`. Marque também a opção: `Set as main project`. Se tudo tiver corrido bem, então o ambiente já deverá estar pronto.

4.4.2 Parte II: Criação e Envio do Código para o Microcontrolador

Após a criação do projeto você observará que o mesmo possui algumas pastas padrões, conforme ilustrado na Figura 4.5.

- **Header Files** → Contém arquivos `.h`. Guardam definições, constantes, macros e protótipos de funções que serão incluídos no código-fonte com `#include`.
- **Important Files** → Lista arquivos considerados críticos pelo projeto (por exemplo, configuração de bits do microcontrolador). Funciona como uma área de "arquivos importantes" para acesso rápido.

- **Makefile** → Arquivo de automação da compilação. Define como o projeto deve ser compilado, quais arquivos entram na *build* e quais opções de compilador serão usadas.
- **Linker Files** → Arquivos de linkedição (`.lkr`, `.ld`) que definem o mapeamento de memória do microcontrolador (endereços de programa, vetores de interrupção etc.).
- **Source Files** → Contém o código-fonte principal do projeto (`.c` para XC8 ou `.asm` para *mpasm*). É onde o programa do microcontrolador é realmente escrito.
- **Libraries** → Armazena bibliotecas externas reutilizáveis utilizadas no projeto (código adicional já pronto).
- **Loadables** → Armazena arquivos carregáveis no simulador/depurador (por exemplo, um arquivo `.hex` já compilado) que podem ser executados sem recompilar o projeto.

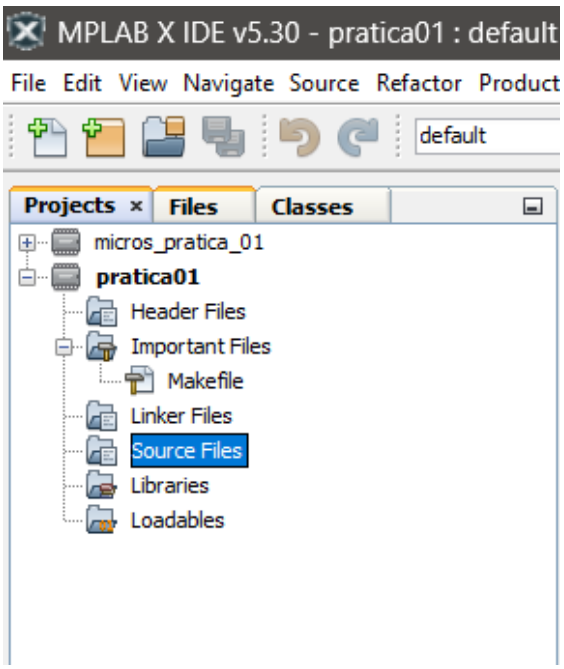

Figura 4.5: Pastas padrões após a criação do projeto.

Após a criação do projeto a pasta **Source Files** fica vazia. Nela, criaremos o arquivo `.asm` para a escrita do código. Para isso, seguiremos as seguintes etapas:

- **Passo 01:** Selecione a pasta **Source Files**, depois vá em **File** ou clique no ícone de novo arquivo (marcado com o retângulo vermelho na Figura 4.6 ou utilize o atalho (`Ctrl+N`), escolha a categoria `Assembler` e o tipo de arquivo `AssemblyFile.asm`, conforme ilustrado na Figura 4.6, depois clique em **Next**. Dê um nome ao arquivo em **File Name**.

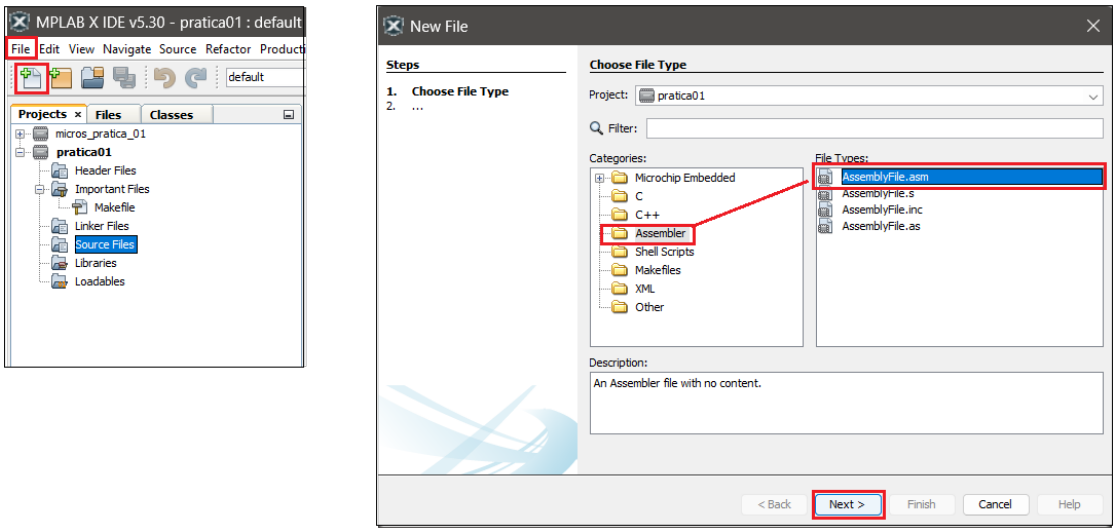

Figura 4.6: Criação do arquivo `main.asm` dentro de **Source Files**.

- **Passo 02:** Analise o esquemático apresentado na Figura 4.7 e faça uma comparação com a pinagem da placa UFC PICLAB-4520. Monte o circuito (se necessário) ou faça as modificações necessárias.

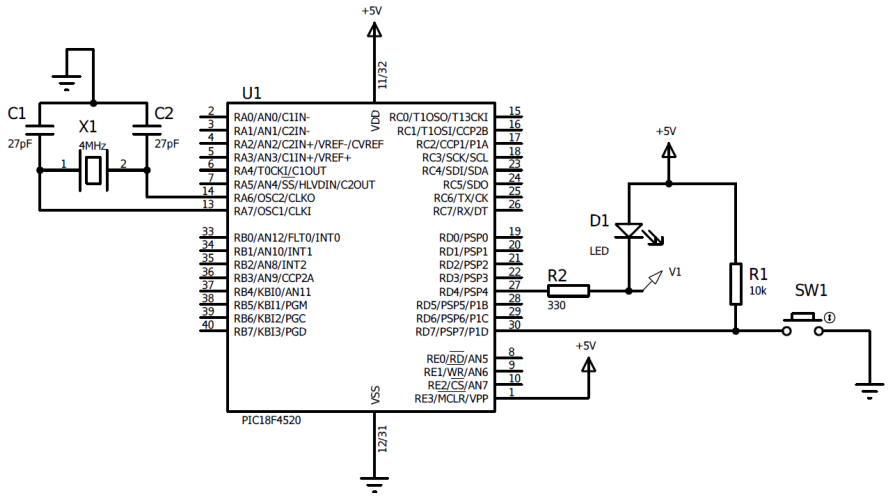

Figura 4.7: Circuito esquemático da prática 01.

- **Passo 03:** Escreva o código disponível em (`https://abre.ai/nUWH`) no arquivo criado na etapa anterior. Faça as considerações e observações necessárias. Após a criação, compile o projeto clicando no ícone do martelo, conforme ilustrado na Figura 4.7 ou utilize o atalho `F11`. Caso o código seja compilado sem erro, irá aparecer a seguinte mensagem no Output: `BUILD SUCCESSFUL (total time: 1s)`.

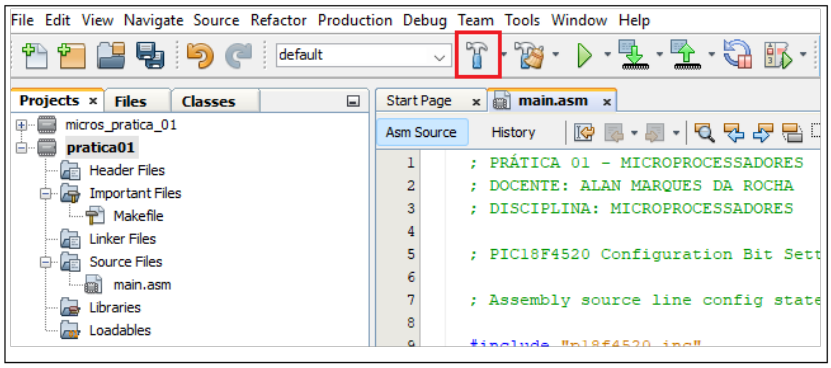

Figura 4.8: Compilação do projeto.

4.4.3 Parte III: Transferindo o código para o PIC via Gravador de Dados

Após a compilação do projeto é necessário realizar a transferência dos dados para o microcontrolador. Para isso, utilizaremos o gravador apresentado na Figura 4.9.

Figura 4.9: Gravador para transferência dos dados.

- **Passo 01:** Realize a ligação do gravador utilizando o cabo disponível. Ligue-o na porta USB do PC/Notebook e a outra extremidade do cabo na entrada do gravador. Verique a ligação dos leds na parte superior do gravador.
- **Passo 02:** Insira o PIC no soquete do gravador, respeitando o sentido correto (chanfro) para o lado da alavanca. Verifique se o microcontrolador encontra-se efetivamente preso.
- **Passo 03:** Após a realização das Etapas 01 e 02, clique no ícone de transferência de dados (`Make and Program Device Main Project`, conforme ilustrado na Figura 4.10. Vá até a pasta **Alternate Tools** e em `PICkit2` deverá aparecer o nome do gravador semelhante a `MultiPROG`. Um led laranja deverá piscar o que indicará que o código foi efetivamente gravado no micro.

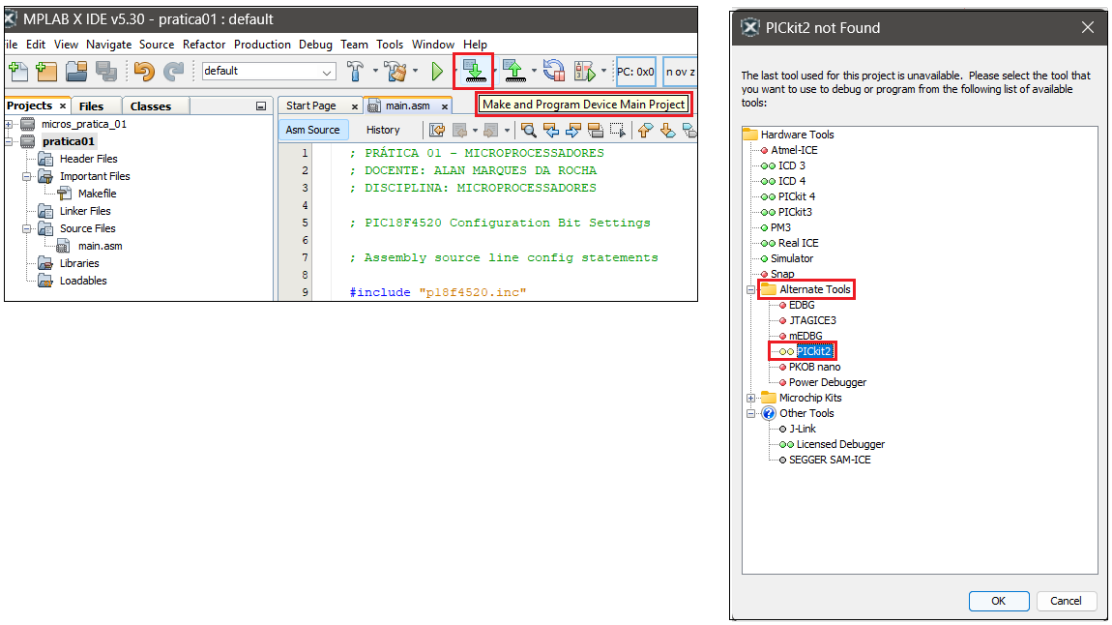

Figura 4.10: Processo de gravação do código no microcontrolador.

- **Passo 04:** Inclua o microcontrolador novamente na placa, ligue-a e observe o que acontece com o LED. Conte quantas vezes o mesmo pisca por segundo e realize suas observações.

4.5 Questionário

1. Durante a etapa de compilação no MPLAB X (clicando no ícone do martelo, *Build Main Project*), quais arquivos são gerados a partir do código-fonte

`.asm`? Explique a função do arquivo `.hex` no processo de gravação do microcontrolador PIC18F4520.

2. Descreva o caminho completo para que um programa escrito em Assembly chegue a ser executado fisicamente no PIC18F4520 presente na placa UFC PICLAB-4520. Cite as etapas: edição do código, compilação/montagem, geração do arquivo final e gravação via gravador externo.

3. Qual é a função do gravador (por exemplo, `PICkit2`/`MultiPROG`) na Prática 01? Explique por que não basta apenas escrever o código no MPLAB X para que o PIC passe a executar o programa.

4. No projeto criado no MPLAB X, há várias pastas (`Header Files`, `Source Files`, `Linker Files`, `Loadables` etc.). Em qual pasta o aluno deve criar o arquivo `main.asm`? Justifique por que esse arquivo deve ficar nessa pasta específica.

5. Na etapa de programação do microcontrolador, é mencionado o comando/função `Make and Program Device Main Project`. O que esse comando faz de diferente em relação apenas ao *Build Main Project* (martelo)?

6. Quais as diferenças entre a montagem do circuito representado na Figura 4.7 e o circuito utilizado na própria placa? Quais alterações foram necessárias realizar no código? Porquê?

7. No início do código Assembly, são feitas configurações de bits, por exemplo:

   ```
   CONFIG WDT = OFF
   ```

   Explique o que significa desabilitar o *Watchdog Timer* (WDT) e por que isso é importante em um primeiro teste de laboratório.

8. Ainda nas configurações iniciais, temos:

   ```
   CONFIG OSC = HS
   ```

   O que essa configuração informa ao microcontrolador sobre o tipo de oscilador? Por que o clock/oscilação do sistema é algo essencial para o funcionamento correto dos atrasos de tempo e da temporização do programa?

9. No trecho:

   ```
   MOVLW b'11101101'MOVWF TRISD
   ```

   Explique o objetivo dessas instruções. O que significa escrever em `TRISD`? Com base nisso, quais pinos da porta `PORTD` serão configurados como entrada e quais serão saída?

10. O programa faz leitura de uma chave (`SW1`) usando:

```
BTFSS PORTD,7
```

    e decide acender ou apagar o LED em `PORTD,1`. Explique com suas palavras a lógica implementada: em que situação o LED acende e em que situação ele apaga?

11. As subrotinas `ATRASO_2ms` e `ATRASO_500ms` criam atrasos de tempo usando laços (`loops`) e instruções `NOP`. Por que precisamos implementar manualmente essas rotinas de atraso em Assembly no PIC18F4520? O que poderia acontecer visualmente com o LED se não houvesse nenhum atraso entre ligar e desligar?