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

Sumário

Página

1 Ementa da Disciplina

1. IDENTIFICAÇÃO DO CURSO
Bacharelado em Engenharia Elétrica e Computação

2. TIPO DE COMPONENTE		
Atividade ( )	Disciplina (✓)	Módulo ( )

3. NÍVEL		
Graduação (✓)	Mestrado ( )	Doutorado ( )

4. IDENTIFICAÇÃO DO COMPONENTE
Nome: Microprocessadores
Código: SBL0082
Carga Horária Prática: 32hrs
Carga Horária Teórica: 64hrs
Nº de Créditos: 6,0
Optativa: Sim ( ) Não (✓)
Obrigatória: Sim (✓) Não ( )

5. DOCENTE RESPONSÁVEL
Prof. Me. Alan Marques da Rocha

6. JUSTIFICATIVA
Microprocessadores e microcontroladores são a base de sistemas embarcados e de computação de propósito geral modernos. Para o engenheiro eletricista, compreender arquiteturas (Harvard e von Neumann), barramentos, memórias, periféricos e o ciclo de instruções é essencial para projetar soluções eficientes, confiáveis e com custo competitivo. A disciplina promove a integração entre teoria e prática por meio de atividades laboratoriais com ferramentas reais (MPLAB X, Compilador C/assembler) e uma plataforma didática (PIC18F452), desenvolvendo competências em especificação, programação de baixo nível e depuração de sistemas. Essa formação sustenta disciplinas avançadas e prepara o egresso para atuar em automação, instrumentação, IoT e produtos embarcados.

7. OBJETIVOS

Objetivo Geral: Capacitar o estudante a analisar, programar e integrar microprocessadores/microcontroladores a dispositivos de memória e periféricos, utilizando boas práticas de desenvolvimento em sistemas embarcados.

Objetivos Específicos:

- Entender a evolução e os conceitos fundamentais de sistemas a microprocessadores.
- Comparar arquiteturas básicas (Harvard vs von Neumann) e suas implicações de projeto.
- Diferenciar memórias voláteis e não voláteis e suas células básicas (ROM, EPROM, EEPROM, OTP, SRAM, DRAM, pilhas).
- Configurar e utilizar periféricos típicos (portas paralelas/seriais, temporizadores/contadores, conversores A/D e D/A).
- Programar, em C e assembler, a família PIC18 (com foco no PIC18F452), incluindo bits de configuração, registradores e interrupções.
- Utilizar o MPLAB X, compiladores e simuladores de hardware/software para desenvolvimento e depuração.
- Projetar e validar um sistema embarcado simples com leitura de sensores e atuação (PWM, DAC).

8. EMENTA

O microprocessador como elemento da arquitetura básica de um computador digital. Arquiteturas Harvard e von Neumann; Barramentos de dados, endereços e controle. Memórias (voláteis e não voláteis) e periféricos (E/S paralela e serial, temporizadores, A/D e D/A). Conjunto de instruções Assembly, montadores, programadores e simuladores. Programação em C e assembler. Microcontroladores PIC18 (PIC18F452): organização de memória, periféricos e sistema de interrupções. Ambiente de desenvolvimento MPLAB X e ferramentas de depuração. Exemplos de aplicações e atividades de laboratório (32h).

9. PROGRAMA DA DISCIPLINA (SUJEITO A ALTERAÇÕES)

Unidade 1 – Sistemas a microprocessadores

1.1 Evolução histórica

1.2 Conceitos básicos

1.3 Exemplos e aplicações

Unidade 2 – Arquiteturas básicas

2.1 Harvard e Von Neumann

2.2 Microprocessadores

- 2.2.1 8 bits, 16 bits e 32 bits
- 2.2.2 Arquiteturas RISC e CISC

2.3 Barramentos de dados, endereços e controle

2.4 Memórias voláteis e não voláteis

- 2.4.1 Células básicas
- 2.4.2 ROM, EPROM, EEPROM, OTP, SRAM, DRAM
- 2.4.3 Pilhas

2.5 Periféricos

- 2.5.1 Portas paralelas
- 2.5.2 Temporizadores e contadores
- 2.5.3 Conversores A/D e D/A
- 2.5.4 Portas seriais

Unidade 3 – Dispositivos microcontroladores

Unidade 4 – Microcontrolador Família PIC18F

4.1 Organização de memória

4.2 Conjunto de instruções

4.3 Periféricos (Portas de E/S, Temporizadores, Comunicação serial, osciladores externo e interno, watchdog timer)

4.4 Sistemas de Interrupção

Unidade 5 – Ferramentas de depuração e compilação

5.1 Ambiente de desenvolvimento integrado: MPLAB X

5.2 Compilador C e assembler

5.3 Simuladores de hardware e software

9. PROGRAMA DA DISCIPLINA (CONTINUAÇÃO)

Unidade 6 – Programação C e assembler

6.1 Estrutura de um programa

6.2 Bits de configuração (CONFIG)

6.3 Acesso aos registradores e configuração dos periféricos

6.4 Declaração de variáveis

6.5 Rotinas de interrupção em C e assembler

6.6 Acesso às portas de E/S para leitura de teclas e acionamento de LEDs

6.7 Acesso ao conversor A/D para leitura de sensores

6.8 Timers: geração de onda quadrada em porta de E/S

6.9 Timers: PWM

6.10 Interface com conversor D/A externo para geração de forma de onda analógica

10. FORMA DE AVALIAÇÃO (SUJEITO A ALTERAÇÕES)

Serão considerados dois componentes: REL (atividades práticas de laboratório) e PF (projeto final da disciplina). Após as práticas, será necessário apresentar relatório técnico. O projeto final será desenvolvido em equipe, com especificação, implementação, testes, demonstração e arguição.

Média Final = 0,70 · REL + 0,30 · PF

REL = nota das atividades práticas (peso 70%). PF = nota do projeto final (peso 30%).

11. BIBLIOGRAFIA

Básica:

[1] SOUSA, Daniel Rodrigues de; SOUZA, David José de. *Desbravando o microcontrolador PIC18: ensino didático*. São Paulo: Érica, 2012. 304 p.

[2] SOUZA, David José de; LAVINIA, Nicolas César; SOUZA, Daniel Rodrigues de. *Desbravando o microcontrolador PIC18: recursos avançados*. São Paulo: Érica, 2010. 336 p.

[3] PEREIRA, Fábio. *Microcontroladores PIC*. São Paulo: Érica, 2009. 360 p.

Complementar:

[4] MICROCHIP TECHNOLOGY INC. *PIC18FXX2 datasheet*. Chandler, AZ, EUA: Microchip Technology Inc., 2006. Disponível em: http://ww1.microchip.com/downloads/en/DeviceDoc/39564c.pdf. Acesso em: 6 out. 2025.

[5] MICROCHIP TECHNOLOGY INC. *MPLAB X IDE user's guide*. Chandler, AZ, EUA: Microchip Technology Inc., 2015. Disponível em: http://ww1.microchip.com/downloads/en/DeviceDoc/50002027D.pdf. Acesso em: 6 out. 2025.

[6] MICROCHIP TECHNOLOGY INC. *MPASM assembler, MPLINK object linker, MPLIB object librarian user's guide*. Chandler, AZ, EUA: Microchip Technology Inc., 2013. Disponível em: http://ww1.microchip.com/downloads/en/DeviceDoc/33014L.pdf. Acesso em: 6 out. 2025.

[7] MICROCHIP TECHNOLOGY INC. *MPLAB XC8 C compiler user's guide*. Chandler, AZ, EUA: Microchip Technology Inc., 2016. Disponível em: http://ww1.microchip.com/downloads/en/DeviceDoc/50002053G.pdf. Acesso em: 6 out. 2025.

2 Fundamentação Teórica

2.1 Sobre o Microcontrolador PIC18F4520

2.1.1 Visão Geral do Dispositivo

O PIC18F4520 é um microcontrolador de 8 bits da família PIC18 da Microchip, conhecido por sua arquitetura de alto desempenho (Arquitetura Harvard com *pipeline*) e um conjunto de instruções aprimorado. O modelo no encapsulamento PDIP de 40 pinos é popular para prototipagem. Ele oferece um equilíbrio robusto entre memória, velocidade e periféricos, tornando-o adequado para uma ampla gama de aplicações embarcadas. O diagrama do PIC18F4520 é ilustrado na Figura 11.1 a seguir:

- **Núcleo:** CPU de 8 bits (Conjunto de Instruções Aprimorado).
- **Velocidade Máxima:** Opera com uma frequência de clock de até 40 MHz, alcançando até 10 MIPS (Milhões de Instruções por Segundo).
- **Tensão de Alimentação:** Faixa típica de 4.2 V a 5.5 V (pode variar ligeiramente dependendo da variante e da temperatura).

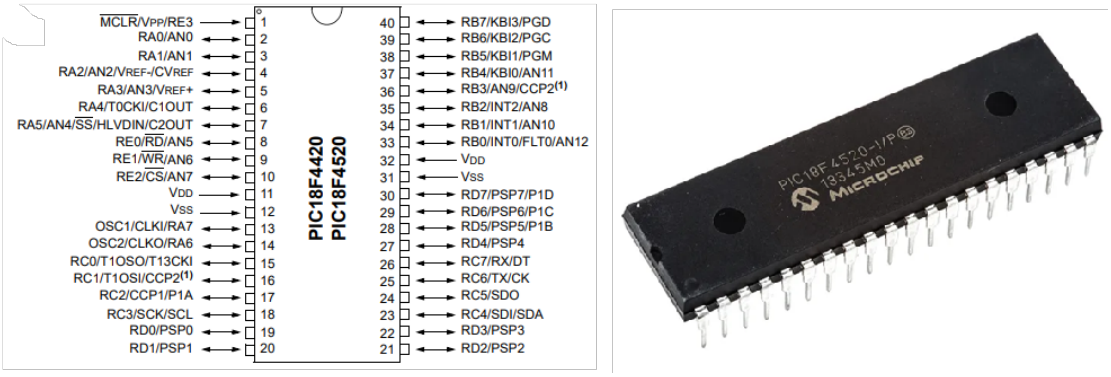

Figure 2.1: Diagrama e CI do PIC18F4520.

- **Pinos de E/S:** Possui 36 pinos de Entrada/Saída (E/S) configuráveis.

2.1.2 Configurações do Oscilador

O PIC18F4520 oferece diversas opções de oscilador, configuradas por *Configuration Bits* (*Fuses*), permitindo ao projetista otimizar o consumo de energia e a precisão do *clock*. As principais opções incluem:

- **Oscilador Externo**:
  - **Cristal/Ressonador (LP, XT, HS):** Utiliza um cristal de quartzo ou ressonador cerâmico externo. LP (*Low-Power*) para baixas frequências, XT (Crystal/Resonator) para frequências médias, e HS (*High-Speed*) para frequências mais altas (até 40 MHz).
  - ***Clock* Externo (EC):** Utiliza uma fonte de clock externa, como um gerador de função.
- **Oscilador Interno:**
  - **Oscilador Interno de Alta Precisão (INTOSC):** Uma fonte de *clock* interna calibrada que pode ser selecionada para operar em várias frequências (31 kHz até 8 MHz). Permite que os pinos OSC1 e OSC2 sejam usados como pinos de E/S digital, economizando espaço e componentes externos
  - **PLL (*Phase-Locked Loop*):** O PLL pode ser habilitado para multiplicar a frequência do oscilador, alcançando o *clock* máximo de 40 MHz.

2.1.3 Modos de Gerenciamento de Energia

O microcontrolador suporta vários modos de gerenciamento de energia para otimizar o consumo, especialmente em aplicações alimentadas por bateria:

- **Modo de Operação Normal (Run):** A CPU e todos os periféricos habilitados estão operacionais. Maior consumo de energia.
- **Modo Idle (Ocioso):** O *clock* da CPU é desligado, mas os periféricos que utilizam o clock periférico continuam a operar.
- **Modo Sleep (Dormir):** A CPU e o oscilador são desligados para atingir o consumo de energia mais baixo possível. O dispositivo pode ser despertado por uma interrupção, como uma interrupção externa ou a interrupção do *Watchdog Timer* (WDT).

2.1.4 Reinicialização (Reset)

O PIC18F4520 pode ser reinicializado por diversas fontes, a saber:

- ***Power-on Reset* (POR)**: Ocorre automaticamente quando a tensão de alimentação (V_{DD}) atinge um nível operacional.
- ***Master Clear Reset* (MCLR)**: Reinicialização externa forçada através do pino MCLR.
- **Watchdog Timer Reset (WDT Reset):** Ocorre se o *Watchdog Timer* estourar, indicando que o programa pode ter entrado em loop ou travado.
- **Brown-out Reset (BOR):** Ocorre se a tensão de alimentação cair abaixo de um limite de segurança programável, garantindo que o dispositivo não opere com alimentação insuficiente.
- **Stack Full/Underflow Reset:** Reinicialização se a pilha de hardware (*Stack*) exceder seus limites (estouro ou esvaziamento).

2.2 Memória

O PIC18F4520 utiliza a **arquitetura Harvard**, que separa a memória de programa e a memória de dados, permitindo acesso simultâneo e, consequentemente, maior velocidade.

2.2.1 Organização de Memória

A memória de dados é organizada em Bancos de Registradores (Register Banks), onde o acesso a um registrador (ou variável) é determinado pela sua localização (endereço) e pelo Registrador de Seleção de Banco (Bank Select Register - BSR). O acesso rápido aos registradores mais comuns é simplificado através da área de Acesso Comum (Access RAM).

2.2.2 Memória de Programa Flash

- **Tamanho:** 32 Kbytes (ou 16 K de palavras de instrução de 16 bits).
- **Tipo:** Flash, o que significa que pode ser apagada e reprogramada eletronicamente.
- **Características:** Permite gravação/leitura de palavra única. Suporta aproximadamente 100.000 ciclos de escrita/apagamento e tem retenção de dados por cerca de 40 anos. Armazena o código do programa (*firmware*).

2.2.3 Memória EEPROM de Dados

O tamanho da mémória utilizada no PIC é de 256 bytes do tipo EEPROM (*Electrically Erasable Programmable Read-Only Memory*), possuindo as características para armazenar dados persistentes (que devem ser mantidos após o desligamento) que podem ser alterados durante a operação do programa (por exemplo, configurações, calibrações). Suporta aproximadamente 1.000.000 ciclos de escrita/apagamento.

2.3 Multiplicador por Hardware 8 × 8

O PIC18F4520 inclui um multiplicador de hardware que pode realizar a multiplicação de dois números de 8 bits com resultado de 16 bits ($8 \times 8 \rightarrow 16$) em um único ciclo de instrução. Isso acelera drasticamente as operações matemáticas complexas, que em arquiteturas sem multiplicador dedicado levariam muitas instruções.

2.4 Interrupções

O sistema de interrupção é crucial para o gerenciamento de eventos em tempo real. O PIC18F4520 possui um sistema de interrupção com prioridades (alta e baixa) para que eventos mais críticos possam interromper o tratamento de eventos menos críticos.

- **Fontes:** Múltiplas fontes de interrupção, incluindo temporizadores, módulos seriais, conversor A/D, pinos externos, EEPROM, etc.
- **Prioridades:** Permite configurar duas prioridades:
  - **Alta Prioridade:** A interrupção é atendida imediatamente. O vetor é fixo.
  - **Baixa Prioridade:** A interrupção só é atendida se nenhuma interrupção de alta prioridade estiver sendo atendida.

2.5 Portas e Módulos

2.5.1 Portas de E/S

O dispositivo possui até 5 Portas de E/S (Port A a Port E) no encapsulamento de 40 pinos, cada uma controlada por registradores, sendo:

- **TRISx:** Configura a direção do pino (Input/Output). 1 = Entrada, 0 = Saída.
- **LATx:** Armazena o valor de saída. Usado para escrever o valor na porta.
- **PORTx:** Usado para ler o estado dos pinos (ou para escrever na porta, embora LATx seja preferível para escrita para evitar o problema Read-Modify-Write).

Muitos pinos são multiplexados, ou seja, um único pino pode ser usado para E/S digital ou como entrada para um periférico (ex: A/D).

2.5.2 Módulo Timer 0

- **Tipo:** Temporizador/Contador de 8 bits ou 16 bits, selecionável.
- **Fonte de Clock:** Pode ser o clock interno (FOSC/4) ou uma fonte de clock externa (como contador de eventos).
- **Pré-escalador:** Possui um pré-escalador programável para estender o alcance de temporização.

2.5.3 Módulo Timer 1

- **Tipo:** Temporizador/Contador de 16 bits.
- **Características:** Pode operar de forma síncrona ou assíncrona com o clock do sistema. É frequentemente usado com um cristal de 32.768 kHz para funções de relógio em tempo real, mesmo no modo Sleep.

2.5.4 Módulo Timer 2

- **Tipo:** Temporizador de 8 bits.
- **Características:** Inclui um registrador de período de 8 bits (PR2) e um pós-escalador. É comumente usado como base de tempo para os módulos PWM (CCP/ECCP).

2.5.5 Módulo Timer 3

- **Tipo:** Temporizador/Contador de 16 bits, similar ao Timer 1.
- **Características:** Permite temporizações de alta resolução adicionais e pode ser usado como fonte de tempo alternativa para os módulos CCP.

2.5.6 Módulos CCP (Captura/Comparação/PWM)

O PIC18F4520 possui dois módulos CCP (Capture/Compare/PWM), sendo eles: Modo Captura: Captura o valor de um dos temporizadores (TMR1 ou TMR3) quando ocorre um evento no pino de entrada CCP, útil para medir a largura de pulsos e o modo comparação: Compara o valor do temporizador com um valor

registrado. Quando a correspondência ocorre, um evento (como mudar o estado de um pino ou gerar uma interrupção) é acionado.

Além disso, possui o modo PWM (*Pulse Width Modulation*) responsável por gerar um trem de pulsos com ciclo de trabalho e período controláveis, essencial para controle de motores, fontes chaveadas, e DACs digitais.

2.5.7 Módulo ECCP (Captura/Comparação/PWM Aprimorado)

O PIC18F4520 possui um módulo ECCP (Enhanced CCP). Ele inclui todas as funcionalidades do CCP padrão, mas adiciona recursos para:

PWM Half-Bridge e Full-Bridge: Gere pares de sinais PWM complementares com controle de dead-band (tempo morto), essencial para controle de motores DC e inversores de energia.

2.5.8 Módulo MSSP (Porta Serial Síncrona Mestre)

O MSSP permite comunicação serial síncrona e suporta dois protocolos, a saber: SPI (Serial Peripheral Interface): Protocolo de comunicação de alta velocidade full-duplex, frequentemente usado para comunicação com SD cards, displays, e outros PICs. Pode operar como Mestre ou Escravo.

I^2C (*Inter-Integrated Circuit*): Protocolo serial de 2 fios (two-wire interface) para comunicação com dispositivos como EEPROM externas, sensores, e RTCs (Real-Time Clocks). Pode operar como Mestre ou Escravo.

2.5.9 EUSART (Receptor/Transmissor Síncrono/Assíncrono Universal Aprimorado)

Também conhecido como Porta Serial. **Assíncrono (UART):** Usado para comunicação serial simples e padrão (ex: RS-232, RS-485). Ideal para depuração ou comunicação com PCs e outros microcontroladores. Suporta detecção de falha de transmissão e ruído. **Síncrono:** Usado para formatos de comunicação serial síncrona não-padrão.

2.5.10 Módulo Conversor A/D de 10 bits

- **Resolução:** 10 bits, o que significa 1024 níveis de quantização (2^{10}).
- **Canais:** Possui até 13 canais de entrada analógica, multiplexados nos pinos de E/S.
- **Função:** Converte um sinal de tensão analógica em um valor digital que a CPU pode processar. Essencial para ler sensores, potenciômetros, etc.

2.5.11 Módulo Comparador

O PIC18F4520 possui dois módulos de comparador analógico. Função: Compara a tensão em duas entradas analógicas. Saída: A saída do comparador é digital (0 ou 1) e pode ser roteada para um pino de E/S ou para um temporizador/interrupção.

2.5.12 Módulo de Referência de Tensão do Comparador

O módulo CVREF (*Comparator Voltage Reference*) gera uma tensão de referência analógica estável e programável. Esta tensão é usada como entrada de comparação para os módulos comparadores, permitindo que o microcontrolador monitore se uma tensão externa está acima ou abaixo de um limite definido internamente.

2.6 Detecção de Alta/Baixa Tensão (HLVD)

O módulo HLVD (*High/Low-Voltage Detect*) permite ao microcontrolador detectar se a tensão de alimentação (V_{DD}) ou uma tensão de entrada está acima ou abaixo de um limite programável.

- **Aplicações:** Monitoramento de bateria (nível baixo) e detecção de sobretensão (nível alto).
- **Eventos:** Pode gerar uma interrupção para que o código de aplicação possa tomar uma atitude (ex: salvar dados na EEPROM e entrar em Sleep) antes de uma falha de energia.

2.7 Recursos Especiais da CPU

- ***Watchdog Timer* (WDT):** Temporizador independente com oscilador próprio. Deve ser reiniciado periodicamente pelo *software*. Se estourar, um WDT *Reset* é gerado, assumindo que o programa travou.
- **Instruções de Acesso de Tabela:** Instruções dedicadas para ler/escrever na *Flash* de programa (*Table Reads/Writes*), permitindo que o programa se configure ou armazene dados na *Flash*.
- **Extensão de Instrução *Access* RAM:** Permite que muitas instruções de 8 bits acessem diretamente o primeiro bloco de RAM (*Access* RAM), otimizando a velocidade.

2.8 Resumo do Conjunto de Instruções

O PIC18F4520 utiliza o conjunto de instruções aprimorado de 16 bits da família PIC18, possuindo as seguintes características: Todas as instruções são de palavra única (16 bits), exceto o `GOTO` e `CALL`, que são de palavra dupla (32 bits) onde a maioria das instruções é executada em um único ciclo de instrução (T_{CY}), que é 4 ciclos de clock do oscilador (`TOSC`).

2.9 Suporte ao Desenvolvimento

O desenvolvimento de *software* e *hardware* para o PIC18F4520 é amplamente suportado:

- Ferramentas de Software: MPLAB X IDE, MPLAB XC8 (Compilador C).
- Ferramentas de Hardware: Gravadores/Depuradores In-Circuit, como PICkit 3/4 ou ICD 3/4. Estes permitem a programação da memória Flash e a depuração em tempo real no circuito.

2.10 Características Elétricas

As características elétricas detalhadas são encontradas na folha de dados (*datasheet*), mas incluem:

- **Tensão de Operação (VDD):** Geralmente 4.2 V a 5.5 V.
- **Corrente de Saída/Entrada:** O PIC18F4520 pode fornecer e consumir uma corrente razoável por pino (tipicamente 25 mA), o que é suficiente para acionar diretamente alguns LEDs.
- **Consumo de Corrente:** Varia amplamente dependendo da frequência do oscilador e do modo de operação (`Run, Idle, Sleep`).

2.11 Gráficos e Tabelas de Características CC e CA

Estes dados são críticos para o projeto de *hardware* e são fornecidos no *datasheet* do fabricante:

- **Características CC:** Incluem limites de tensão e corrente de operação, corrente de consumo nos diferentes modos, limites de tensão de entrada/saída, e resistências dos *pull-ups* internos.
- **Características CA:** Referem-se ao tempo e à frequência, incluindo tempos de ciclo de instrução, tempos de subida/descida dos sinais de E/S, tempos mínimos de `pulse width` para periféricos seriais (SPI, I^2C), e a frequência máxima de operação (40 MHz).

3 Projeto e Utilização da Placa UFC PICLab-4520

3.1 Introdução

4

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 MPLASM™. Em seguida, será criado o arquivo-fonte do programa principal do projeto, no formato `.asm`, onde o código de controle será desenvolvido. Nesta etapa, a programação em linguagem *Assembly* permite observar com clareza a relação direta entre a manipulação dos 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 último contém a codificação do programa traduzida em instruções binárias, num formato transferível diretamente para a memória *flash* do microcontrolador. Assim, o microcontrolador não executa diretamente o arquivo de programa formato `.asm`, mas seu conteúdo convertido para binário e disponível para gravação no formato `.hex`.

Na gravação do *firmware* no microcontrolador, utiliza-se um gravador externo ligado ao computador. Esse gravador transfere o conteúdo do arquivo `.hex` à memória de programa do PIC18F4520 a partir do próprio MPLAB® X IDE. 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 MPLAB® X.

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 MPLAB® X.

4.4 Procedimento Experimental

Parte I: Inicialização do MPLAB® X e Criação do Projeto

- **Passo 1:** Inicialize o MPLAB® X 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.

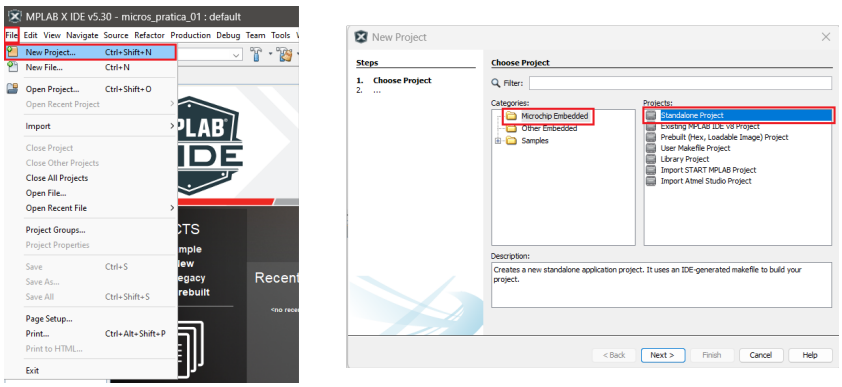

Figure 4.1: Criando um novo projeto no MPLAB® X IDE.

- **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.

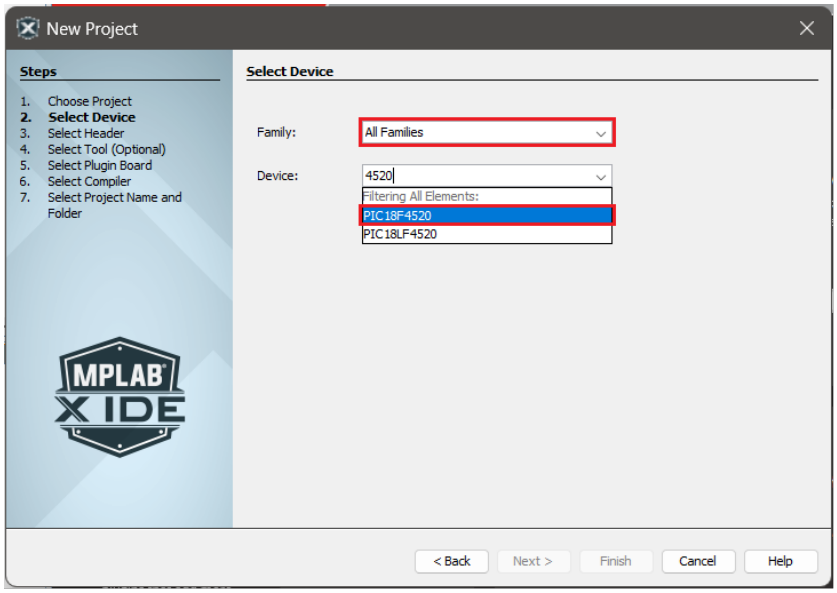

Figure 4.2: Escolhendo o PIC para a prática.

- **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**.

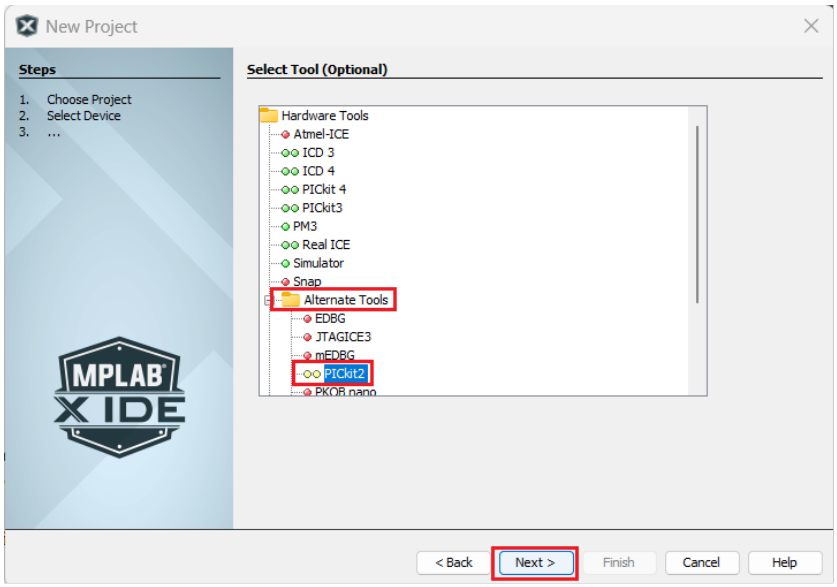

Figure 4.3: Escolha da ferramenta PICkit2 para gravação dos dados no PIC.

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

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

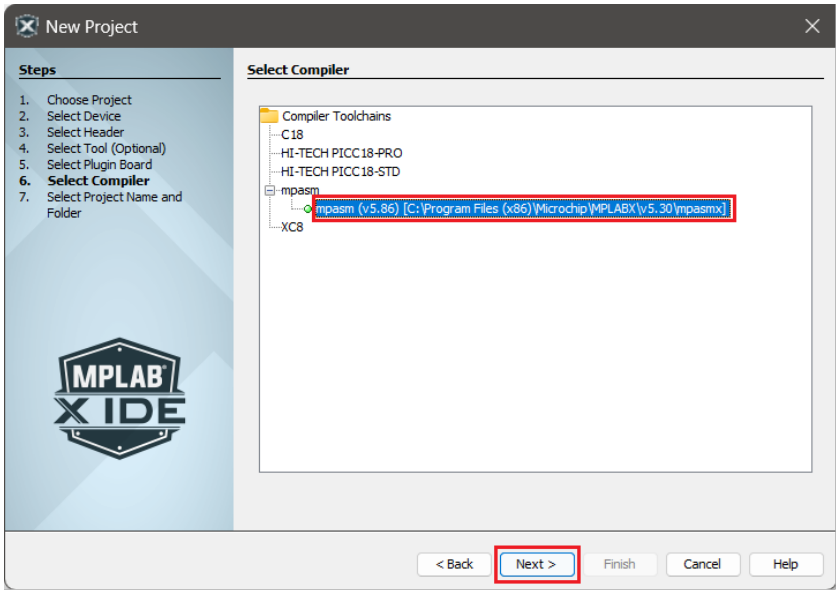

Figure 4.4: Escolhendo o compilador para o projeto.

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.

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.

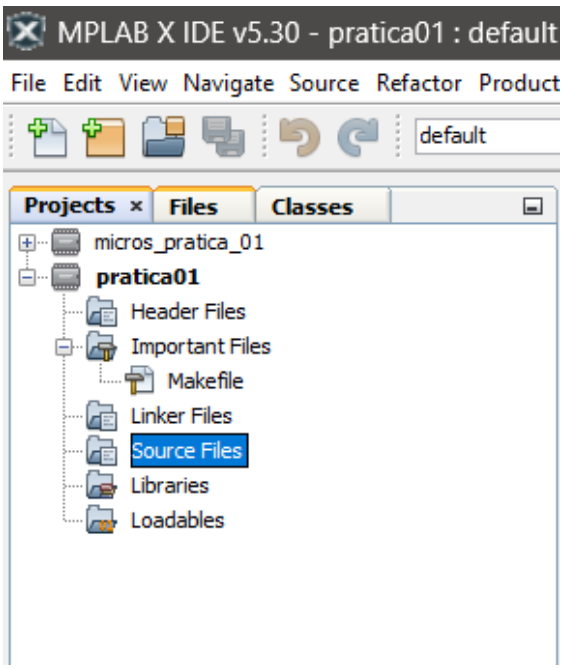

Figure 4.5: Pastas padrões após a criação do projeto.

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**.

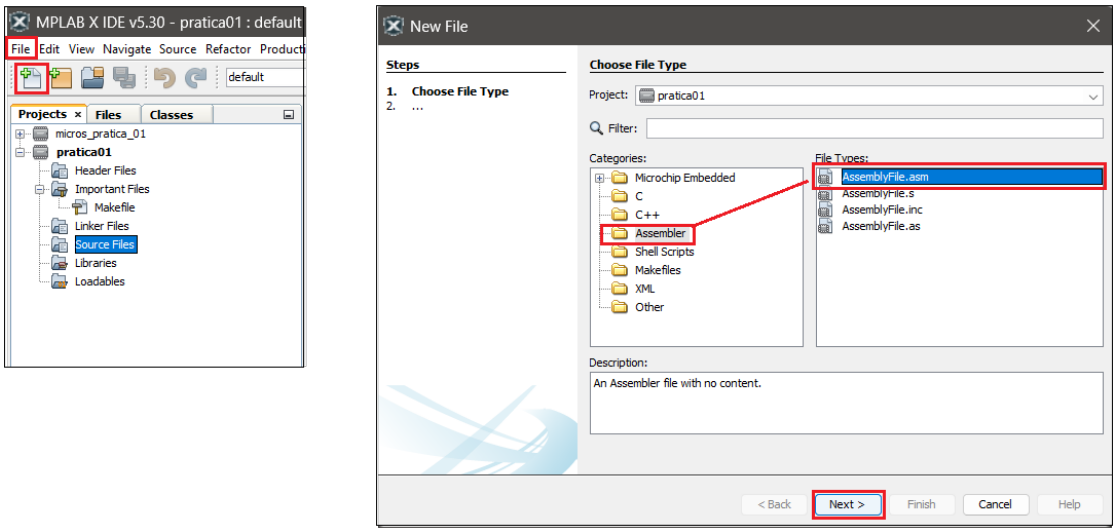

Figure 4.6: Criação do arquivo `main.asm` dentro de **Source Files**.

- **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.

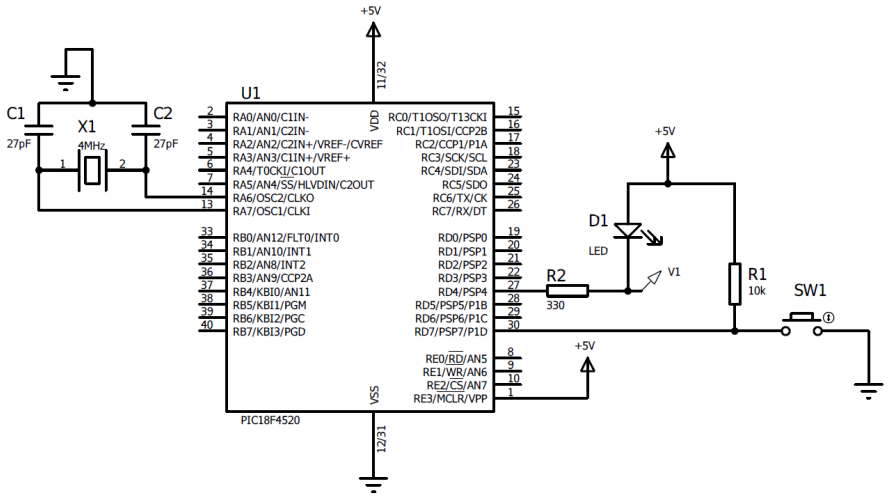

Figure 4.7: Circuito esquemático da prática 01.

- **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)`.

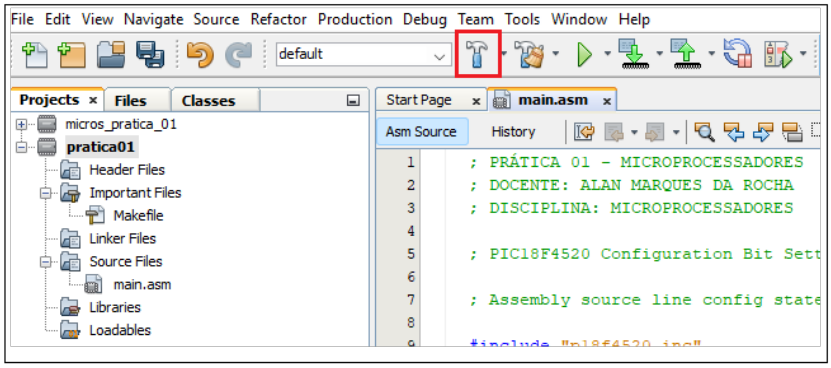

Figure 4.8: Compilação do projeto.

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.

Figure 4.9: Gravador para transferência dos dados.

- **Passo 01:** Realize a ligação do gravador a porta USB do PC/Notebook com

o cabo disponível. Observe o acionamento dos LEDs da parte superior do gravador.

- **Passo 02:** Insira o PIC18F4520 no soquete do gravador, respeitando o sentido correto (chanfro) para o lado da alavanca. Desça a alavanca e verifique se o microcontrolador encontra-se efetivamente travado ao soquete.
- **Passo 03:** 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.

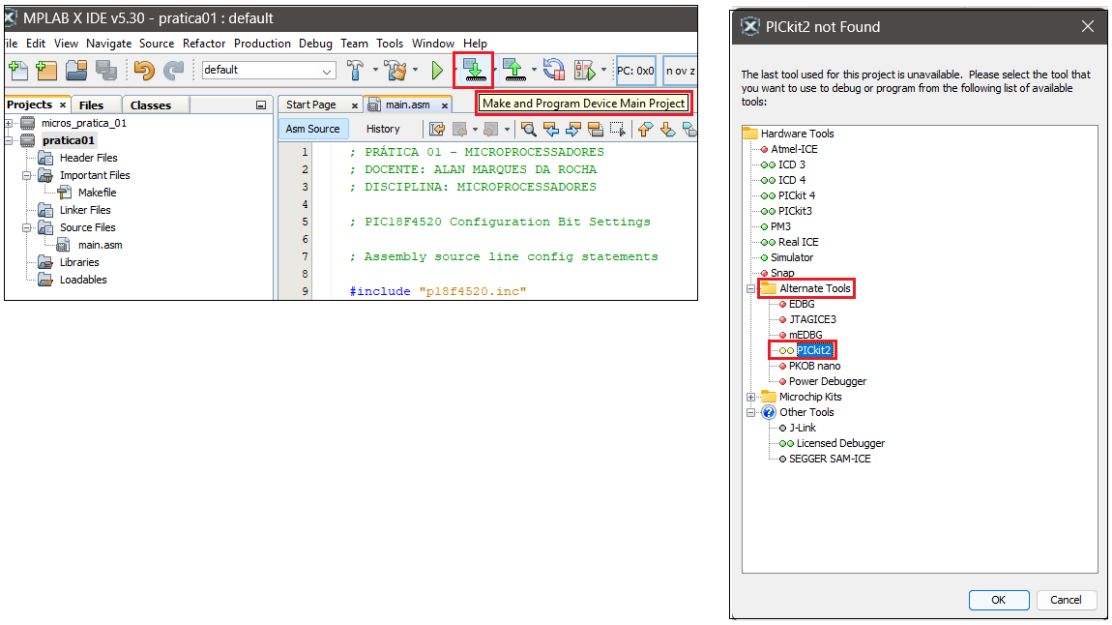

Figure 4.10: Processo de gravação do código no microcontrolador.

- **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?

5

Prática 02: Interface LCD 16x2 com PIC18F4520 em Assembly

5.1 Introdução

A presente prática tem como finalidade integrar conceitos fundamentais de microcontroladores, eletrônica digital e instrumentação computacional por meio da comunicação entre um microcontrolador PIC18F4520 e um *display* LCD alfanumérico de 16 colunas por 2 linhas (LCD 16×2). O experimento realiza a escrita de duas mensagens no *display* a partir de um programa desenvolvido integralmente em linguagem *Assembly*.

O LCD utilizado segue o padrão de controladores compatíveis com o HD44780, amplamente empregado em aplicações embarcadas devido à sua simplicidade de interface paralela. O barramento de dados é de 8 bits e é conectado diretamente ao PORTD do PIC18F4520, enquanto os sinais de controle *RS* (*Register Select*) e *E* (*Enable*) são conectados respectivamente aos pinos RB0 e RB1 do microcontrolador. O sinal *RW* do LCD é conectado permanentemente ao terra, de forma que a comunicação é feita apenas no sentido "microcontrolador → *display*", isto é, apenas escrita.

Além da programação, a prática aborda aspectos essenciais de *hardware*: alimentação do circuito em +5 V, uso adequado do pino de *reset* (MCLR) com *pull-up* de 10 kΩ, ajuste de contraste do LCD via potenciômetro e uso de um cristal externo de aproximadamente 4 MHz para definir o *clock* do PIC. Também é explorada a simulação completa do sistema no *software* Proteus, que permite validar tanto o *firmware* quanto o *hardware* sem necessidade imediata de montagem física em bancada.

Dessa forma, esta atividade funciona como um primeiro "Hello World" em sistemas embarcados: o aluno escreve código *Assembly*, gera o arquivo executável em formato `.hex`, associa esse código a um microcontrolador em um ambiente de simulação e observa um resultado visual direto no *display*. Esse fluxo corresponde exatamente ao ciclo típico de desenvolvimento embarcado profissional (desenvolver → compilar → gravar → testar).

5.2 Objetivos

- Compreender o mapeamento de pinos entre o microcontrolador PIC18F4520 e um *display* LCD 16x2 no modo de 8 bits.
- Desenvolver, em *Assembly*, uma rotina capaz de:
  - inicializar o LCD (configuração de interface, liga *display*, limpa tela);
  - posicionar o cursor no início da primeira e da segunda linha;
  - escrever sequências de caracteres armazenadas na memória de programa (*Flash*).
- Configurar corretamente os *Configuration Bits* (ou "bits de configuração") do PIC18F4520, definindo tipo de oscilador (`OSC = XT` para cristal externo de aproximadamente 4 MHz), desabilitando o *watchdog* (`WDT = OFF`) e habilitando o uso de MCLR externo.
- Gerar o arquivo `.hex` a partir do código *Assembly* utilizando o ambiente MPLAB X.
- Montar e simular o circuito completo no Proteus, associando o arquivo `.hex` ao componente PIC18F4520, fornecendo alimentação de +5 V, *clock* e *reset* adequados, e verificando a exibição de mensagens pré-definidas.
- Consolidar o fluxo de desenvolvimento típico: projeto → compilação → gravação → simulação funcional.

5.3 Lista de Materiais

- Um kit da placa UFC PICLAB-4520;
- 1 gravador PICKit2 para programação do microcontrolador;
- Computador com a IDE de desenvolvimento MPLABX e *software* Proteus;
- *display* LCD 16×2;
- Potenciômetro de 5KΩ ou 10kΩ.

5.4 Procedimento Experimental

Etapa 1: Criação do projeto no MPLAB X e configuração do dispositivo

1. Abrir o MPLAB X IDE.
2. Criar um novo projeto (*New Project*):

- Selecionar "*Microchip Embedded*" → "*Standalone Project*".
- Escolher o dispositivo **PIC18F4520**.
- Selecionar o *assembler* apropriado (MPASM / PIC18 *toolchain*).
- Criar o projeto em uma pasta conhecida (por exemplo, `PIC_LCD_MICROS`).

3. Dentro do projeto, criar um novo arquivo fonte Assembly (*New* → *Assembly File*).

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

Etapa 2: Compilação e geração do arquivo `.hex`

1. No MPLAB X, selecionar o projeto e executar *Build Project* (ou *Clean and Build*).

2. O assembler irá:

   - montar o código Assembly;
   - aplicar os *CONFIG bits*;
   - gerar um arquivo objeto e, ao final, gerar um arquivo `.hex`.

3. Ao término da compilação, localizar o arquivo `.hex` na pasta de saída do projeto. Em projetos MPLAB X típicos, esse arquivo aparece em um subdiretório parecido com:

   ```
   dist/default/production/<nome_do_projeto>.production.hex
   ```

4. Esse arquivo `.hex` contém o programa pronto para ser executado pelo PIC18F4520. É esse arquivo que será carregado no Proteus como "firmware" do microcontrolador.

Etapa 3: Montagem do circuito no Proteus

1. Abrir o Proteus e criar um novo projeto/schematic (`.pdsprj`).

2. Inserir os seguintes componentes:

- PIC18F4520;
- LCD 16x2 compatível com HD44780 (por exemplo LM016L);
- Cristal (aprox. 4 MHz);
- Dois capacitores cerâmicos ≈ 22 pF (um de cada lado do cristal para GND);
- Potenciômetro de 10 kΩ (ajuste de contraste do LCD);
- Resistor de 10 kΩ entre MCLR e VDD (pull-up de reset);
- Fonte DC de +5 V e referência GND;
- (Opcional) **Resistor** em série com o backlight do LCD (típico 220 Ω) se o módulo tiver pinos de LED.

O circuito simulado deverá seguir a montagem apresentada na Figura 5.1, conforme ilustrado a seguir:

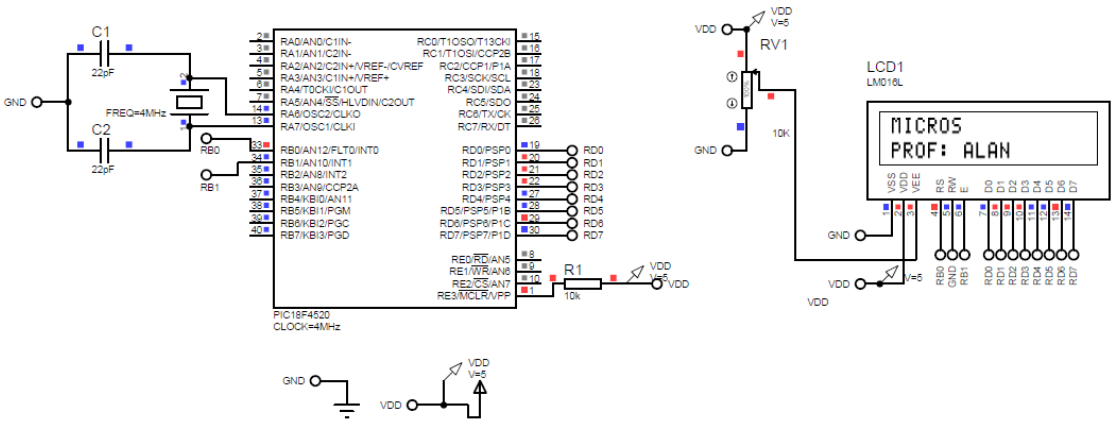

Figure 5.1: Circuito esquemático da prática 02.

Etapa 4: Carregar o firmware `.hex` no PIC dentro do Proteus

1. No *schematic* do Proteus, dar *duplo clique* no componente **PIC18F4520**.
2. Na janela de propriedades do PIC:
   - Em *Program File*, selecionar o arquivo `.hex` gerado na Etapa 3.
   - Ajustar o campo *Clock Frequency* para **4 MHz**, coerente com o cristal externo e com `CONFIG OSC = XT`.
3. Confirmar que o PIC está alimentado: VDD ligado a +5 V e VSS a GND. Se o nó VDD aparecer "cinza" durante a simulação, significa que a fonte de 5 V ainda não está realmente inserida no circuito ou conectada corretamente.

4. Ajustar o potenciômetro de contraste no LCD até que os caracteres fiquem visíveis. Se o contraste estiver muito alto ou muito baixo, a tela pode parecer completamente vazia mesmo que o código esteja rodando.

Etapa 5: Execução da simulação

1. Clicar em *Run* (Play) no Proteus.

2. Se todas as conexões e o arquivo `.hex` estiverem corretos, o PIC18F4520 executará:

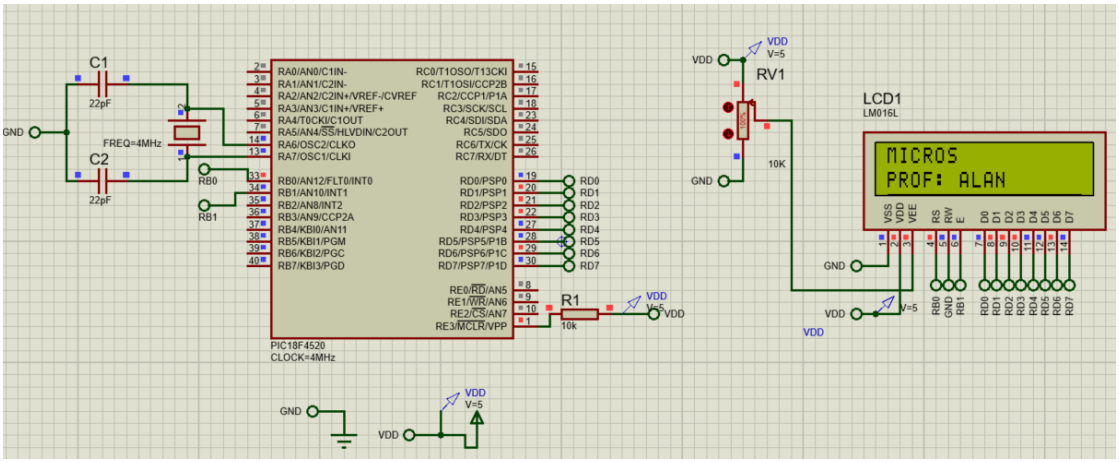

Figure 5.2: Circuito simulado no proteus da prática 02.

Etapa 6: Montagem do circuito físico

1. Monte o circuito em *protoboard* replicando o esquemático anteriormente validado em simulação no Proteus.

2. Ao utilizar a placa UFC PICLAB-4520 não se faz necessário realizar a montagem do MCLR ligado ao VDD por meio de um resistor de 10kΩ.

3. Também não se faz necessário a montagem do cristal externo com os capacitores de desacoplamento ao terra.

4. Ligue o *display* LCD 16×2 conforme definido (barramento D0..D7 no PORTD, RS no RB0, E no RB1, RW em GND), ajuste o contraste do LCD com o potenciômetro entre 5 V e GND e alimente tanto o PIC quanto o LCD com a mesma fonte regulada de 5 V. Certifique-se de reproduzir fielmente a mesma fiação usada na simulação.

5. Verifique a pinagem do *display* LCD, realizando a montagem seguindo as informações apresentadas na Figura 5.3. Para mais informações sobre o *display* LCD, visite: (https://abre.ai/nWxX)

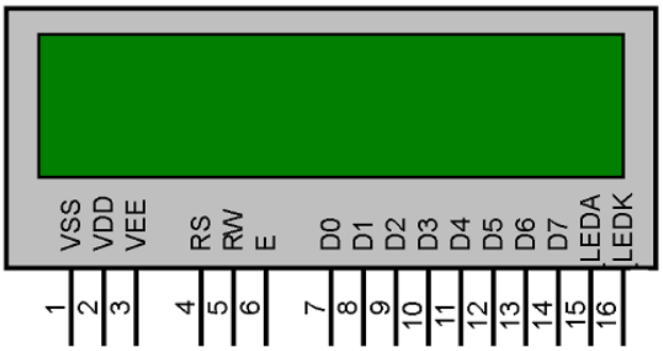

Figure 5.3: Pinagem do LCD 16×2.

6. Verifique as conexões e observe a mensagem que aparece no *display*. Registre o resultado para a inclusão no relatório.

5.5 Questionário

1. Explique qual é o papel do sinal RS, do sinal *Enable* (E) e do sinal *Read/Write* (RW) no LCD 16×2. Indique quais desses sinais são conectados ao PIC18F4520 e quais são forçados a um nível fixo no circuito desta prática.

2. No código em *Assembly*, o barramento de dados do LCD (D0–D7) é conectado em qual porta do PIC18F4520? Justifique por que essa porta foi configurada como saída usando o registrador `TRIS` correspondente.

3. Qual é a função da sub-rotina `LCD_Init`? Descreva os comandos enviados ao LCD (`0x38`, `0x0C`, `0x01`, `0x06`) e o efeito que cada um deles provoca no *display*.

4. Por que o potenciômetro (entre 5 V e GND) precisa estar ligado ao pino de contraste (VEE/V0) do LCD? O que acontece visualmente no *display* se o contraste estiver muito alto ou muito baixo?

5. A prática utiliza um cristal externo de aproximadamente 4 MHz ligado aos pinos OSC1 e OSC2 do PIC18F4520, junto com capacitores de aproximadamente 22 pF para GND. Explique a importância desse cristal para o funcionamento do programa e justifique por que o bit de configuração `CONFIG OSC` foi ajustado para `XT`.

6. Na simulação no Proteus, por que é necessário carregar manualmente o arquivo `.hex` no componente PIC18F4520 e ajustar a frequência de *clock* no campo *Clock Frequency*? Descreva o que aconteceria se o PIC estivesse sem programa ou com *clock* incorreto.

7. Mostre, no código em *Assembly*, onde as strings `"MICROS"` e `"PROF: ALAN"` estão declaradas. A seguir, explique como a rotina `LCD_PrintString` percorre essas strings e envia caractere por caractere ao LCD. Use o trecho abaixo como referência:

```
        ORG     0x300
MsgLinha1:
        DB      "MICROS",0x00
MsgLinha2:
        DB      "PROF: ALAN",0x00
```

e

```
LCD_PrintString:
NextChar:
        TBLRD*+                         ; TABLAT <- [TBLPTR], TBLPTR++
        MOVF    TABLAT, W, ACCESS       ; W = caractere lido
        BNZ     SendChar                ; se W != 0x00, imprime
        RETURN                          ; se W == 0x00, acabou a string

SendChar:
        CALL    LCD_Data                ; manda caractere em W
        BRA     NextChar
```

8. Quais são as diferenças práticas entre simular o circuito no Proteus e montar fisicamente o circuito em bancada (por exemplo, utilizando a placa UFC PICLAB-4520)? Comente sobre: alimentação de +5 V, *reset* (MCLR), oscilador, ligação do LCD e necessidade de ajustes físicos (potenciômetro de contraste).

9. Reproduza em esquema o diagrama mínimo de ligação entre o PIC18F4520 e o LCD 16×2 utilizado na prática, indicando:

   - VDD e VSS (alimentação +5 V / GND);
   - MCLR com *pull-up* de 10 kΩ;

- Cristal externo e capacitores de 22 pF;
- RS → RB0, E → RB1, RW → GND;
- D0–D7 → RD0–RD7;
- Potenciômetro no pino de contraste do LCD.

10. Refaça a prática utilizando linguagem C em vez de *Assembly*. O comportamento final deve ser o mesmo: escrever `"MICROS"` na primeira linha e `"PROF: ALAN"` na segunda linha do LCD. Para isso:

    (a) Crie um novo projeto em C no MPLAB X para o PIC18F4520.

    (b) Implemente em C funções equivalentes a `LCD_Init()`, `LCD_Command()`, `LCD_Data()` e `LCD_PrintString()`.

    (c) Armazene as duas mensagens (linha 1 e linha 2) e envie-as ao LCD após a inicialização, posicionando o cursor corretamente.

    (d) Gere o arquivo `.hex` a partir do código C, carregue esse arquivo no PIC no Proteus e valide o funcionamento.

    Inclua no relatório final:

    - o código em C (em ambiente LaTeX, use `\begin{verbatim}` ... `\end{verbatim}` para apresentá-lo);
    - o arquivo `.hex` gerado;
    - uma captura de tela do Proteus exibindo as mensagens no LCD.

6

Prática 03: Cronômetro com Interrupção Externa

6.1 Introdução

Uma característica dos sistemas embarcados é serem reativos a eventos. Além disso, esses sistemas *contam tempo*, seja para execução de tarefas periódicas, medida de tempos ou geração de temporizações. O microcontrolador PIC18F4520 utiliza para esse fim três elementos: (i) bases de clock configuráveis (oscilador interno, cristal externo HS, PLL), (ii) temporizadores de propósito geral com *prescalers* e **interrupções**, e (iii) periféricos que se beneficiam diretamente de uma base de tempo estável (PWM, comunicação serial). Ao articular esses recursos, os microcontroladores PIC permite temporizações de "pulsos imprecisos" a bases de tempo de 1 Hz confiáveis (fundamento de cronômetros, relógios, agendadores e protocolos).

Nesta prática você implementará um cronômetro de segundos (`00-59`), cuja contagem é apresentada em dois dígitos, em código BCD (decimal codificado em binário), no `PORTD` (*nibble* mais significativo = dezenas; *nibble* menos significativo = unidades). A codificação BCD permite a apresentação dos dígitos em displays de sete segmentos com decodificadores BCD→7 segmentos. O controle do usuário é assíncrono: dois botões conectados às **interrupções externas** `INT0`/`INT1` alternam *iniciar/pausar* e *reset*. A precisão temporal não depende de `delays` de software, mas do **Timer0** operando como contador de 16 bits, com **preload**, a partir do clock do sistema. Assim, a temporização dos segundos ocorre por **interrupção** periódica, liberando a CPU para realização de outras tarefas.

6.2 Objetivos

- Compreender o uso de interrupções externas (`INT0/INT1`) para interação com botões.
- Projetar a base de tempo de 1 Hz com `Timer0` (16 bits) para um cristal externo (por exemplo, 20 MHz).
- Implementar, em linguagem C (XC8) um cronômetro 00–59, com codificação BCD para a exibição do tempo em displays de sete segmentos.
- Simular e validar o projeto no *software* Proteus, observando o comportamento do contador e das interrupções.

6.3 Lista de Material

- 1 kit da placa UFC PICLAB-4520.
- 1 gravador PICKit2 para programação do microcontrolador.
- Protolab com *displays* BCD de sete segmentos já decodificados.
- *Jumpers* para ligação dos circuitos.
- *Software* Proteus para simulação e averiguação antes da montagem.

6.4 Procedimento Experimental

Etapa 1: Criação do projeto no MPLAB-X e configuração do dispositivo

1. Abra o MPLAB X e crie um *Standalone Project* para o dispositivo **PIC18F4520**.

2. Selecione o compilador linguagem C XC8 e crie um arquivo `main.c`.

3. Escreva o código disponível em (https://abre.ai/nYlX) no arquivo criado na etapa anterior. Compile o projeto clicando no ícone do martelo ou através da tecla de atalho F11. Caso o código seja compilado sem erros, deverá aparecer a seguinte mensagem no Output: BUILD SUCCESSFUL (total time: 1s).

Etapa 2: Compilação e geração do arquivo `.hex`

1. No MPLAB X, selecione o projeto e execute em *Build Project* (ou *Clean and Build*).

2. O PROJETO NÃO É EM C ??? O assembler irá:

   - montar o código Assembly;
   - aplicar os `CONFIG bits`;
   - gerar um arquivo objeto e, ao final, gerar o arquivo `.hex`.

3. Ao término da compilação, localize o arquivo `.hex` na pasta de saída do projeto. Em projetos MPLAB X típicos, esse arquivo aparece em um subdiretório parecido com:

   ```
   dist/default/production/<nome_do_projeto>.production.hex
   ```

4. Esse arquivo `.hex` contém o programa pronto para ser executado pelo PIC18F4520. É esse arquivo que será carregado no Proteus como "firmware" do microcontrolador.

Etapa 3: Simulação do circuito no Proteus

1. Abra o Proteus e crie um novo projeto/**schematic (`.pdsprj`)**

2. Insira os seguintes componentes:

   - PIC18F4520;
   - Um *Crystal* com frequência de 20 MHz;
   - Dois capacitores cerâmios ≈ 22 pF;
   - Resistor de 10kΩ entre o MCLR e VDD (*pull-up* de *reset*);
   - Fonte DC de +5V e referência GND;
   - *Switch* (`SW-SPDT`).

O circuito simulado deverá seguir a montagem apresentada na Figura 6.1, conforme ilustrado a seguir:

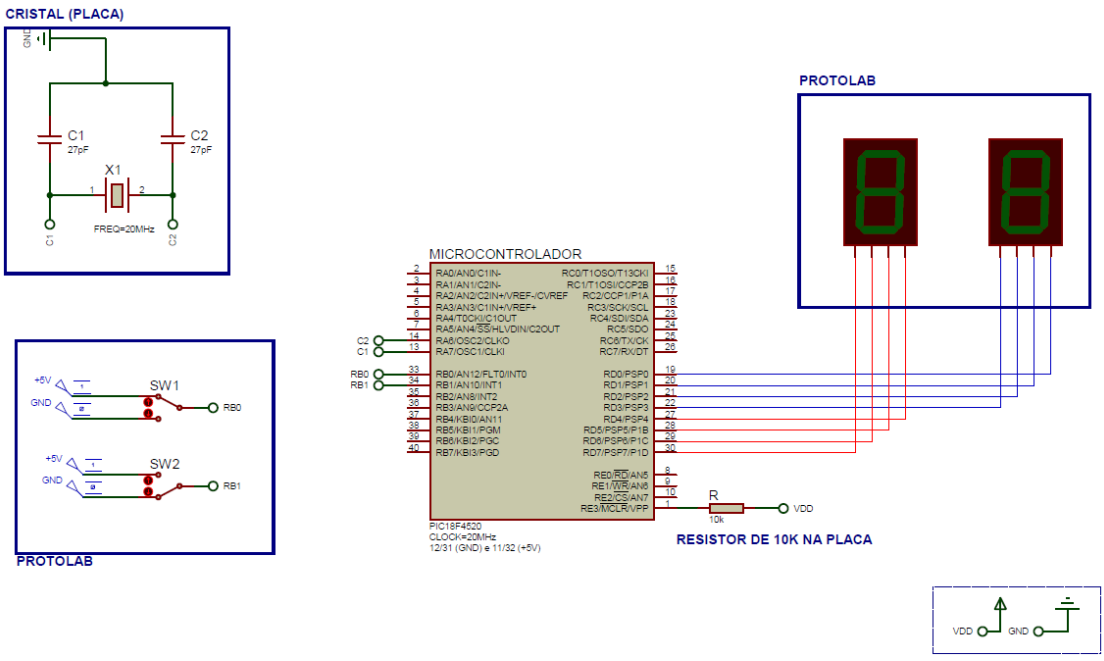

Figure 6.1: Circuito esquemático da prática 03.

Etapa 4: Carregar o firmware `.hex` no PIC dentro do Proteus

1. Após a montagem do circuito, dê um duplo clique no componente PIC18F4520.

2. Na janela de propriedades do PIC, siga os seguintes passos:

   - Em *Program File*, selecione o arquivo `.hex`, conforme ilustrado na Figura 6.2.

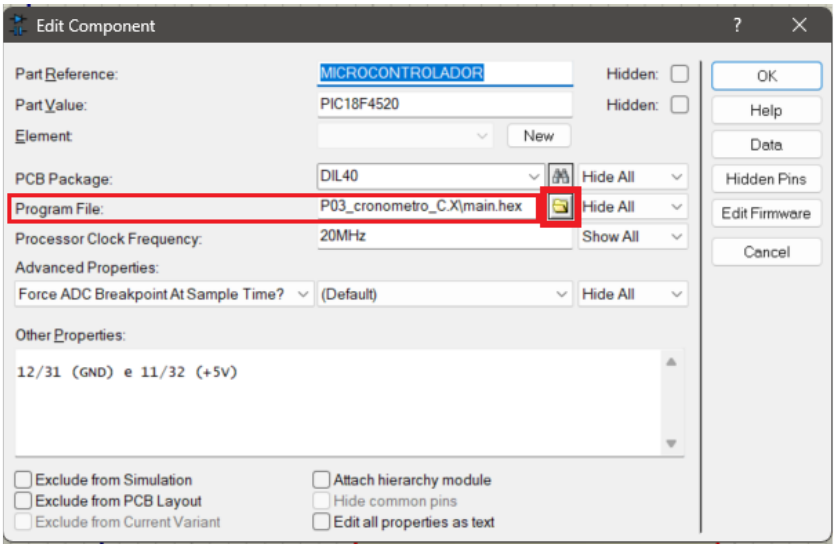

Figure 6.2: Exportação do arquivo `.hex` para o PIC no Proteus.

3. Confirme se o PIC está realmente alimentado: VDD ligado a +5 V e VSS ao GND. Se o nó VDD aparecer "cinza" durante a simulação, significa que a fonte de +5 V ainda não está realmente inserida no circuito ou conectada corretamente.

Etapa 5: Execução da simulação

1. Para realizar a simulação, clique em *Run* no Proteus (canto inferior esquerdo).

2. Se todas as conexões e o arquivo `.hex` estiverem corretos, o programa será executado, conforme ilustrado na Figura 6.3.

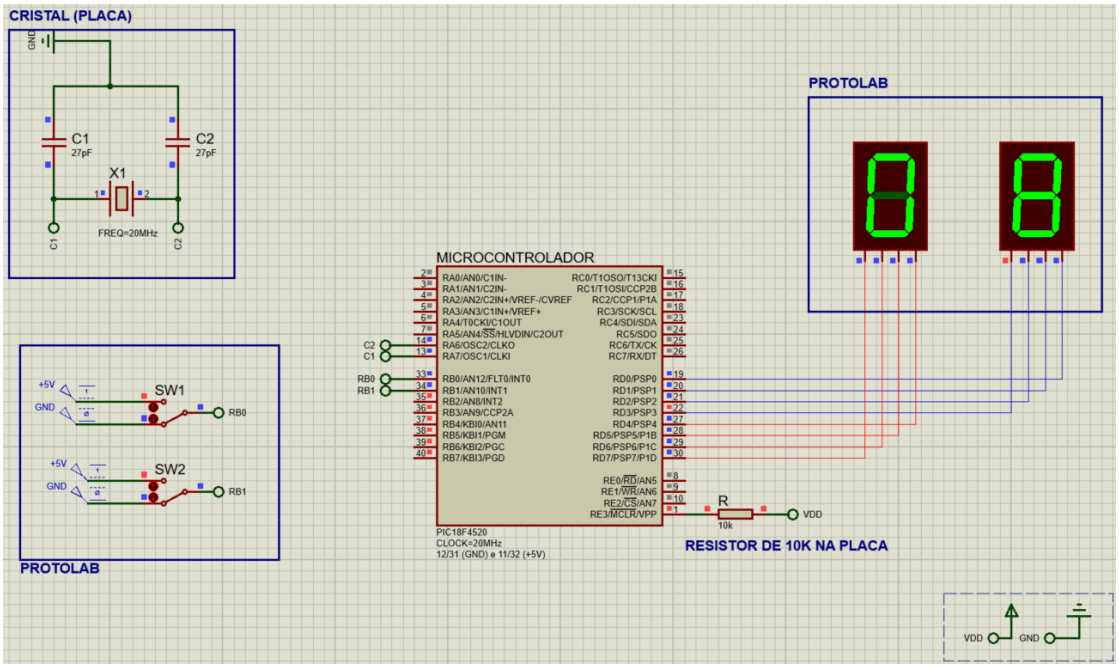

Figure 6.3: Circuito da prática 03 simulado no Proteus.

3. Utilize `SW1` para iniciar a contagem e `SW2` para encerrar. O cronômetro deverá iniciar em 0 e seguir até 59, reiniciando a contagem.

Etapa 6: Montagem em laboratório

1. Monte o circuito com o auxílio da protolab, replicando o esquemático anteriormente validado em simulação no Proteus.

2. Na placa UFC PICLAB-4520 não é necessário ligar o pino MCLR ao VDD por meio de um resistor de 10kΩ (a ligação já está na placa), assim como ligar ao microcontrolador o cristal externo com os capacitores de desacoplamento ao terra.

3. Verifique as conexões e a continuidade dos *jumpers* e observe o que acontece com os *displays* BCD. Registre o resultado para a inclusão no relatório.

6.5 Questionário

1. Explique por que *delays* por *software* não garantem uma base de tempo estável e como o uso de `Timer0` com `preload` corrige esse problema.

2. Com Fosc = 20 MHz, calcule o *preload* do `Timer0` (16 bits, prescaler 1:256) para obter 1 Hz. Mostre as etapas do cálculo e compare com o valor usado no código.

3. Se o cristal for alterado para 4 MHz, qual deve ser o *novo preload* para manter 1 Hz? Apresente a dedução.

4. Modifique o projeto para `MM:SS` (quatro dígitos). Especifique a lógica de *overflow* (00:00 → 59:59 → 00:00) e como distribuir a saída (BCD ou segmentos) para os quatro dígitos. Realize essa etapa na linguagem C. (O código em C e a simulação do Proteus devem ser encaminhadas junto com o relatório).

7

Prática 04: Controle de Ventoinha

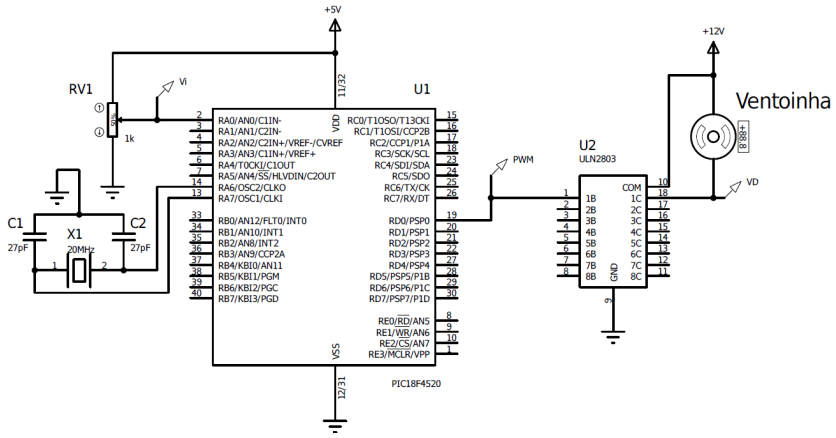

Figure 7.1: Circuito esquemático da prática 04.

8 Prática 05: Incremento e Decremento

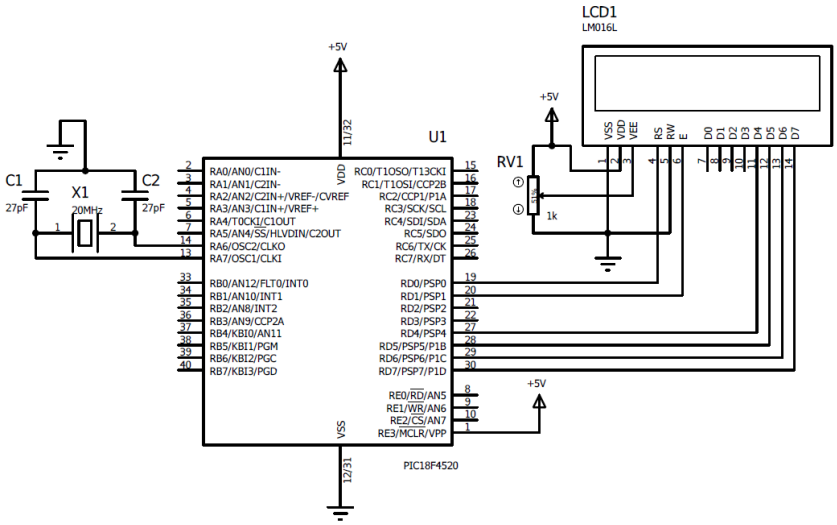

Figure 8.1: Circuito esquemático da prática 05.

9 Prática 06: Voltímetro (ADC+LCD)

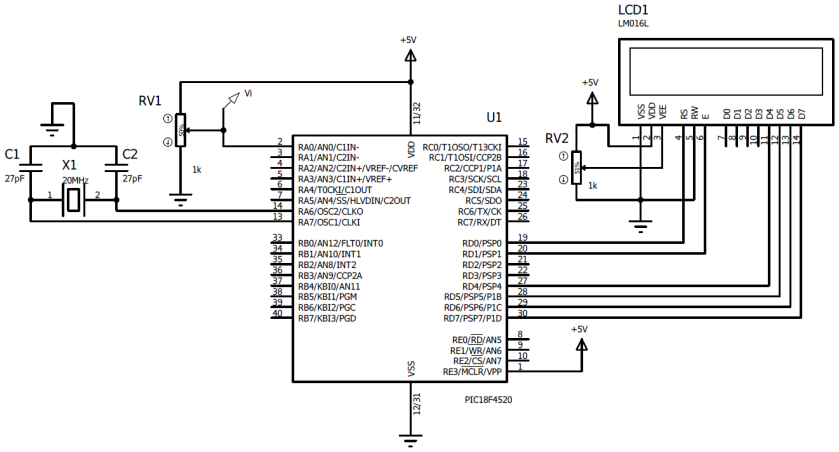

Figure 9.1: Circuito esquemático da prática 06.

10 Prática 07: DAC ou rede R–2R (8 bits)

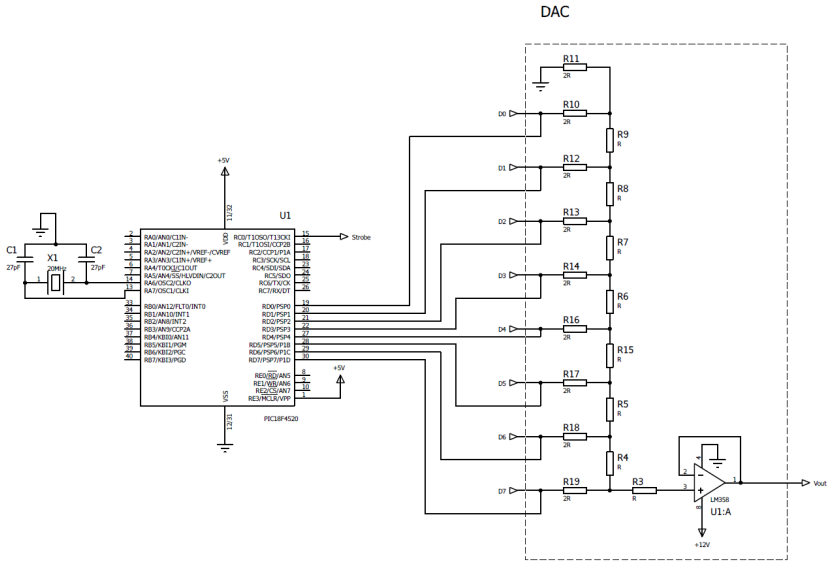

Figure 10.1: Circuito esquemático da prática 07.

11 Prática 08: Comunicação Serial Síncrona

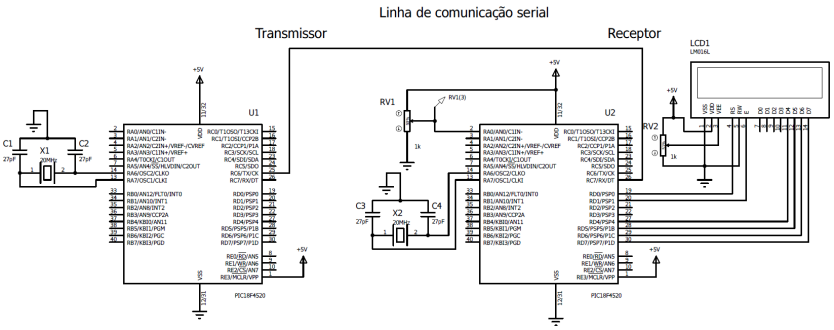

Figure 11.1: Circuito esquemático da prática 08.

12

Prática 09: Nome da Prática

12.1 Introdução
12.2 Objetivos
12.3 Lista de Material
12.4 Procedimento Experimental
Etapa 1: Criação do projeto no MPLAB-X e configuração do dispositivo
Etapa 2: Compilação e geração do arquivo `.hex`
Etapa 3: Montagem do circuito no Proteus
Etapa 4: Carregar o firmware `.hex` no PIC dentro do Proteus
Etapa 5: Execução da simulação
Etapa 6: Montagem do circuito físico
12.5 Questionário

13 Prática 10: Nome da Prática

13.1 Introdução
13.2 Objetivos
13.3 Lista de Material
13.4 Procedimento Experimental
Etapa 1: Criação do projeto no MPLAB-X e configuração do dispositivo
Etapa 2: Compilação e geração do arquivo `.hex`
Etapa 3: Montagem do circuito no Proteus
Etapa 4: Carregar o firmware `.hex` no PIC dentro do Proteus
Etapa 5: Execução da simulação
Etapa 6: Montagem do circuito físico
13.5 Questionário

14
Conclusões

Bibliografia

[1] Sergio Martínez-Losa Del Rincón. Unofficial LaTeX template for reports/books/thesis with corporate logos of Universidad de Zaragoza with a beautiful look and feel. https://github.com/sergiomtzlosa/latex-template-report-unizar, 2021.