Camila Motta Renó – 2019003833 – engenharia de computação – ECOP14 – Universidade Federal de Itajubá (UNIFEI)

Impressora 3D

As funcionalidades de uma impressora 3D são ilimitadas. Diversos objetos são feitos com essa tecnologia. Dessa forma, escolheu-se um projeto para a disciplina ECOP14 para simular uma impressora 3D utilizando o microcontrolador PIC18F4520, através de um simulador da placa PICGenios que fornece suporte para a IDE MPLAB.

A proposta estabelece objetos a serem impressos, que são selecionados pelo teclado. O buzzer (responsável por avisar o término da impressão e quando falta filamento), o LCD (informa as opções do usuário e o estado da impressora), os LEDs (indica inicialização ou desligamento), o display de 7 segmentos (timer da impressão) e o rele (representa a impressora funcionando) são usados para avisar o usuário sobre os estágios do funcionamento do projeto. Além disso, a ventoinha é usada para resfriar a placa.

Durante o projeto todo foi utilizado:

- Simulador PICSimLab (link para download: https://sourceforge.net/projects/picsim/)
- Ambiente de programação para sistemas embarcados: MPLAB (link para download: https://www.microchip.com/mplab/mplab-x-ide)
- Compilador (link para download: https://www.microchip.com/en-us/development-tools-tools-and-software/mplab-xc-compilers)

Os componentes da placa PICGenios utilizada dentro do simulador foram:

- LCD 16x4
- 4 displays de 7 segmentos
- 1 rele
- 1 ventoinha/ cooler
- 1 buzzer
- 1 teclado matricial 4x3

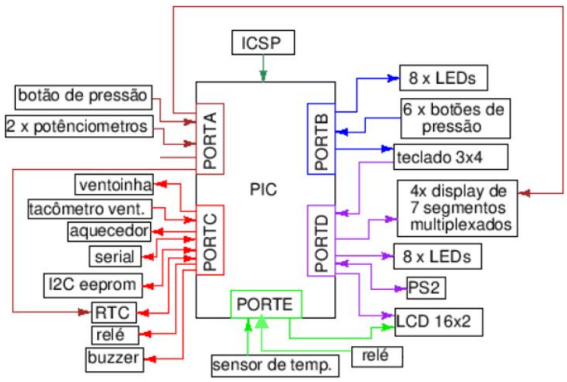
PASSO 1: O HARDWARE

Figura 1 - placa do simulador PICSimLab

Após a instalação de todos os programas necessários, selecionou-se a placa PICGenios, o microcontrolador PIC18F4520, o LCD com 16x4 e modificou a porta do buzzer para ser possível ligá-la. Na figura 1, mostra-se onde se modifica cada uma dessas configurações e como elas deviam estar. Caso apareça algum problema no simulador, como aconteceu durante o desenvolvimento dessa aplicação, instalar uma versão antiga pode resolve-lo.

As funcionalidades das portas são organizadas conforme a figura 2. Por meio da análise da figura ao lado, percebe-se que, para o projeto proposto, será necessário a multiplexação somente para os displays de 7 segmentos. Sendo assim, a porta utilizada é a mesma para cada display e, para mostrar o valor em cada um deve-se alternar entre cada display rapidamente, dando a impressão de que todos estão ligados simultaneamente.

Figura 2 - características da placa PICGenios com o PIC18F4520

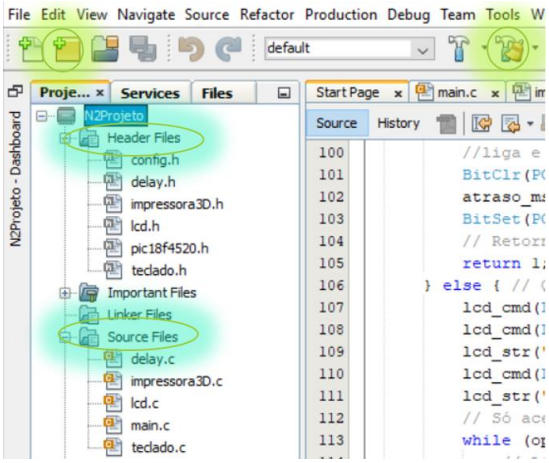


Fonte: disponível na aba "help" do simulador, com alguma alterações

PASSO 2: O SOFTWARE

Depois de instalar a IDE no computador, cria-se um novo projeto no MPLAB, no botão com uma pasta. As configurações usadas para o desenvolvimento do código dessa proposta foram "New Project -> Microship Embbeded -> Standolone Project -> PIC18F4520 -> PICSimLab -> XC8 -> Nome do novo projeto".

Figura 3 - imagem da IDE MPLAB



Agora, pode-se criar novos arquivos, que servem como a main ou como headers ou como um arquivo C para as funções. É padrão que os arquivos .c novos ou adicionados ao projeto fiquem dentro da pasta "Source Files", enquanto os arquivos .h no "Header Files". Para compilar o código adicionado seleciona-se o botão de martelo. Após a compilação do código escrito, um arquivo .hex é gerado.

PASSO 3: ENTENDENDO O CÓDIGO

Antes de fazer o código, escreveu-se um planejamento sobre as funções necessárias para os objetivos propostos. Chegou-se a quantidade de 8 funções, com algumas que surgiram durante o projeto. Alguns métodos se relacionam entre si, enquanto outros realizam um objetivo específico sem depender de informações de outros lugares, nem fornecer dados para outras funções.

Primeiro, na main são configuradas as portas para cada funcionalidade, determinando se são entradas ou saídas, a partir do TRIS e se são analógicos ou digitais, pelo ADCON. Após esse passo, disponibiliza-se no LCD a mensagem "Digite 0 para ligar".O código chama a função opcaoEscolhida, que fica esperando a entrada do usuário.

Em vários lugares do arquivo é usado as funções deslocarCursor() e atraso_ms(int x), que possui vários laços internos e consegue atrasar o programa em x ms.

Depois disso, chama-se a InicializarImpressora, a qual liga sequencialmente os LEDs. No primeiro método, muda-se as portas para ler a entrada no teclado matricial e chama-se uma função disponibilizada para a realização da disciplina, que retorna um char. Esse valor é passado para a main. Enquanto no segundo método, determina-se um vetor com os valores

hexadecimais responsáveis por ligar cada LED do PORTB. Com o uso da função auxiliar atraso_ms, espera-se, aproximadamente 1000 ms de intervalo entre cada LED acionado.

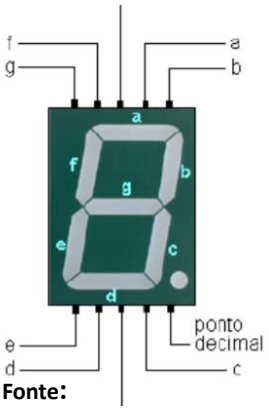
Escreve-se no LCD as opções disponíveis. Percebe-se que, mesmo ocupando as 4 linhas do display, só existem dois endereços para os 4, pois o L_L1 se refere a primeira e a terceira linha e o L_L2, a segunda e quarta. Com isso, chama-se a opcaoEscolhida para esperar a resposta do usuário.

Sabendo qual a opção do usuário, escreve-se uma estrutura de escolha com o "switch". Dentro de cada caso, mostra uma mensagem no LCD, através do lcdMensagem que usa códigos disponibilizados para a disciplina. Além disso, chama-se a função ativarRele com um tempo determinado pelo define. Primeiro, esse método verifica se existe filamento o suficiente e, se necessária a troca, ele é chamado novamente para retomar o pedido feito. O filamento é medido através da variável global quantFilamento, crescendo proporcionalmente ao tempo que cada impressão demora. Se for menor que 25, a impressora funciona sem precisar de nenhuma troca.

Dentro do ativarRele, existe um condicional ("if") que verifica o estado da variável acima. Se for possível ele liga o rele através do PORTE, chama o tempoCooler (passando o tempo de cada operação) e liga o buzzer ao final da operação. Caso a condição não seja satisfeita, ele expõe a mensagem "Não há filamento suficiente. Quando preencher aperte o 9", liga e desliga o buzzer e zera a variável, pois nenhum filamento foi usado.

Para a função tempoCooler, cria-se um outro vetor com valores hexadecimais que ligam cada LED dos 7 para formar um número. Conforme observado na figura 4, pode-se encontrar esse valores fazendo-os no formato 0b0gfedcba, colocando cada letra como 0 (para desligado) ou 1 (para ligado) e, depois, calculando o hexadecimal.

Figura 4 - display de 7 segmentos com cada LED em destaque



Fonte: https://docente.ifrn.edu.br/aryalves/disciplinas/semestre-letivo-2015.2/eletronica-digital/manipulando-display-de-7-seguimentos-com-ci-decodificados-bcd-7s

Dessa forma, realiza-se a multiplexação dos displays de 7 segmentos, escrevendo o algarismo equivalente ao tempo, várias vezes rapidamente para parecer que todos fiquem ligados. Após a passagem de aproximadamente um segundo, a variável cont que representa o tempo é subtraída em uma unidade. Caso queira mudar a passagem tempo, é essencial observar o efeito flicker gerado quando a atualização dos displays é baixa e eles parecem piscando. Por fim, utiliza-se a variável j, que mostra quanto tempo já se passou, para ligar e desligar o cooler.

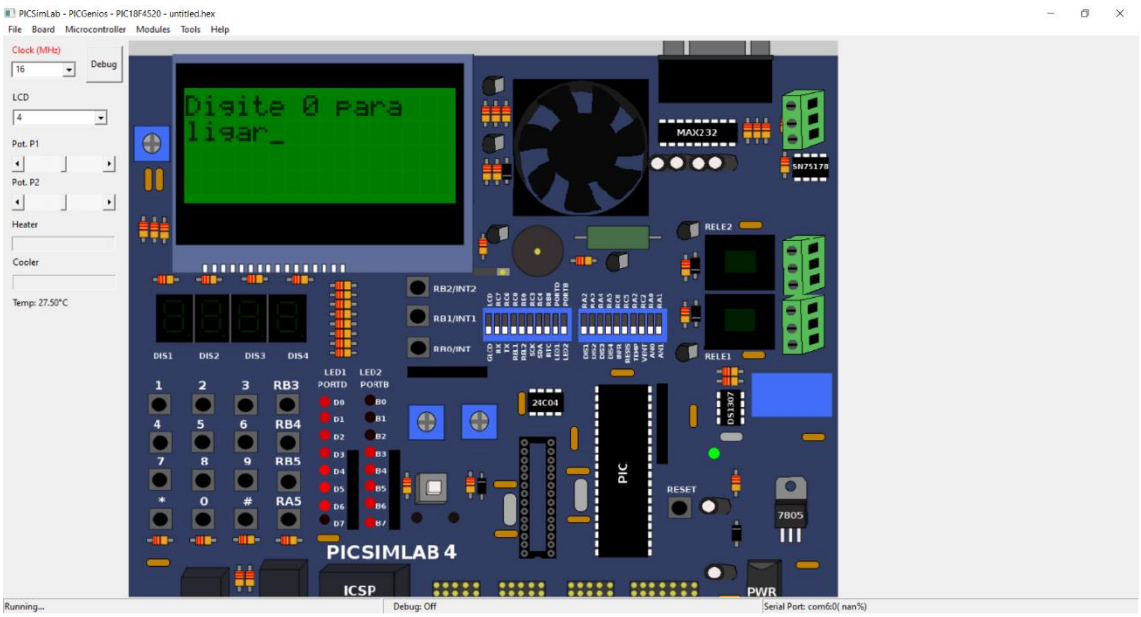
Por fim, existe a opção de desligar a impressora. Ao selecioná-la, aparece a mensagem "Desligando" no LCD 16x4. Chama-se a função desligarImpressora semelhante com a InicializarImpressora, mas que percorre o laço ("for") do maior para o menor índice nas posições do vetor.

PASSO 4: SIMULAR NA PLACA DO PICSIMLAB

Com a compilação dos arquivos, é possível ler o .hex pelo simulador. Assim, para rodar o código através dele deve selecionar "File -> Load Hex". Para encontrar o arquivo segue-se essa ordem "Nome_do_projeto -> dis -> default -> production". Algumas vezes, ao selecionar o botão do teclado matricial pela interface gráfica ou pelo teclado do próprio computador pode não receber o valor, mas isso é um problema do simulador.

Camila Motta Renó – 2019003833 – engenharia de computação – ECOP14 – Universidade Federal de Itajubá (UNIFEI)

Foto da simulação do projeto:



PASSO 5: DISPONIBILIZAÇÃO DO CÓDIGO

Os códigos estão disponibilizados no GitHub através do link: https://github.com/kcami/Projeto-ECOP14.git.