Projeto Final - Simulador MP3

Gabriel B. Vargas - 2019004680

Universidade Federal de Itajubá

gabriel.vargas@unifei.edu.br

8 de dezembro de 2020

I. Introdução

O PIC18F4520 é um microcontrolador extremamente difundido, parte da popular família de 8 bits e núcleo de 14 bits, produzido pela MICROSHIP. De modo a se programar esse microcontrolador, utilizou-se a IDE MPLAB-X, desenvolvida também pela empresa MICROSHIP.

Para que os testes das aplicações fossem realizados, utilizou-se o software PICsimLab, desenvolvido pelo professor Luis Cláudio Gambôa. Tal placa possui suporte total ao PIC18F4520, e a placa de desenvolvimento simulada é a PICGenios.

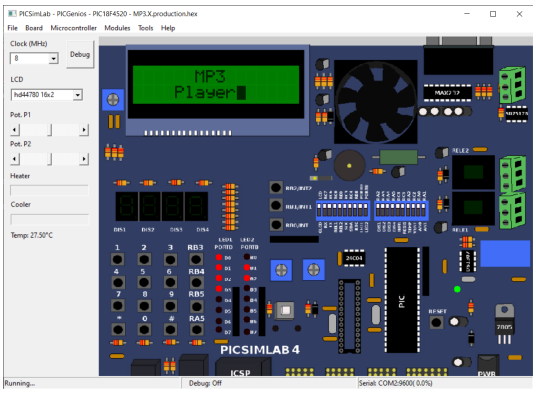

Figura 1: *Placa PICGenios, simulada no sofwtare PICsimLab.*

O projeto desenvolvido com essas ferramentas trata-se de um simulador de MP3 Player, e utiliza das seguintes funções da placa PICGenius:

- **Display LCD:** É utilizado para realizar a comunicação primária com o usuário, por meio de menus;
- **Display 7 Segmentos:** Apresenta informações numéricas ao usuário, como duração restante da música;
- **Teclado:** Interface de comunicação do usuário com o sistema, utilizado para interagir com os menus;
- **Barramento de LEDs:** Demonstra o nível de volume das músicas;
- **Cooler:** É executado de modo a simular a caixa de som do aparelho de MP3.
- **Buzzer:** Aviso sonoro de que o processo finalizou, no caso, a música.

II. Desenvolvimento

Inicialmente, foi realizado um planejamento do código, que seria separado em duas principais funções:

- Escolha da música;
- Execução da música.

Na primeira etapa, o menu inicial foi implementado, enquanto a segunda etapa seria responsável pelas funcionalidades propostas pelo projeto, que são executadas enquanto a música é tocada. Essas funções foram implementadas na biblioteca "songs.h", que foi desenvolvida especificamente para o projeto. Outras bibliotecas também foram utilizadas durante o desenvolvimento do projeto, que foram fornecidas pelo professor, e possuem funções para controlar, por exemplo, o teclado e o display LCD.

Além disso, de maneira a facilitar o controle de versionamento do código, foi utilizada a ferramenta git, sendo o projeto hospedado no GitHub.

i. Inicialização da biblioteca

Durante o processo de criação do algoritmo, notou-se a necessidade de uma função exclusiva para a correta inicialização da biblioteca, assim como carregar as listas de músicas em uma struct com seus nomes e durações. Desse modo, a função "songsInit()"foi criada, assim como a struct "musica", que armazena uma string contendo o nome da música atual, e um inteiro, que armazena sua duração em segundos.

```
void songsInit(void) {
    TRISC = 0x00;
    for (unsigned int i = 0; i < 10; i++) {
        musicas[i].duracao = duracoes[i];
        strcpy(musicas[i].nome, (char*) nomes[i
    ]);
    }
    return;
}

```

Acima está a implementação dessa função, que configura o TRISC, para futura ativação do buzzer, e carrega as listas com os nomes das músicas e durações, em uma lista do tipo musica. Com isso, torna-se possível associar o nome de uma música com sua duração de maneira facilitada.

ii. Escolher a Música

Sendo a biblioteca iniciada, e a lista de música prontas para ser utilizada, prosseguiu-se para a próxima etapa, que trata-se do menu principal de esoclha da música. Essa etapa foi implementada na função "chooseSong()", que deve exibir no display LCD o nome da música atual, no display de 7 segmentos o índice da música, e realizar uma interação com o usuário pelo teclado.

```
void chooseSong(void) {
    kpDebounce();
    tecla = kpRead();
    ssdUpdate();
    if (bitTst(tecla, 3) || bitTst(tecla, 7)) {
        flag = 1;
        for (;;) {
            ssdUpdate();
            kpDebounce();
            atraso_ms(10);
            if ((kpRead() != tecla) || flag ==
    1) {
                tecla = kpRead();
                if (bitTst(tecla, 3)) { //1
                    if (indice == 0) {
                        indice = 9;
                    } else {
                        indice -= 1;
                    }
                } else if (bitTst(tecla, 7)) {
    //2
                    if (indice == 9) {
                        indice = 0;
                    } else {
                        indice += 1;
                    }
                } else if (bitTst(tecla, 0)) {
    //*
                    flag = 0;
                    break;
                }
                lcdCommand(CLR);
                lcdPosition(1, 0);
                lcdStr("<-(1) (*)  (2)->");
                lcdPosition(0, 0);
                lcdStr(musicas[indice].nome);
                ssdDigit(indice, 3);
                flag = 0;
            }
        }
        playSong();
    }
}

```

Acima encontra-se o código da função chooseSong(), que precisa que uma das teclas de seleção (1 ou 2) sejam pressionadas para que se prossiga para a próxima etapa. Com isso, entra-se em um for infinito, que realiza a interação com o usuário pelo teclado. A cada execução desse for, o display LCD é atualizado com as novas informações. Quando o usuário pressionar a teclada de seleção de música (*), o for é interrompido, e a próxima função é executada.

iii. Tocar a Música

Tendo a música sido selecionada, a próxima etapa é responsável por implementar as funcionalidades do sistema que dependem da execução da música, como Play/Pause e escolha do volume. Portanto, foi criada a função playSong(), que é responsável pelos métodos

Figura 2: *Menu inicial de interação com o usuário, para escolha da música.*

já citados, e também pelo buzzer, cooler, e retorno ao menu principal.

```
void playSong() {
    pwmInit();
    lcdCommand(CLR);
    lcdPosition(0, 0);
    lcdStr(musicas[indice].nome);
    lcdPosition(1, 0);
    lcdStr("-(1)  (*)  (2)+");

    tempo = musicas[indice].duracao;
    pwmSet(100);
    while (tempo != 0) {

        minuto1 = (tempo / 60) % 10;
        minuto2 = (tempo / 60) / 10;
        segundo1 = (tempo % 60) % 10;
        segundo2 = (tempo % 60) / 10;

        ssdDigit(minuto2, 0);
        ssdDigit(minuto1, 1);
        ssdDigit(segundo2, 2);
        ssdDigit(segundo1, 3);

        for (unsigned char j = 0; j < 100; j++)
    {
            ssdUpdate();
            atraso_ms(10);
            kpDebounce();
            tecla = kpRead();
            if (bitTst(tecla, 3)) {
                while(bitTst(tecla, 3)) {
                    ssdUpdate();
                    kpDebounce();
                    tecla = kpRead();
                }
                alterarVolume(0);
            }
            else if (bitTst(tecla, 7)) {
                while(bitTst(tecla, 7)) {
                    ssdUpdate();
                    kpDebounce();
                    tecla = kpRead();
                }
                alterarVolume(1);
            }
            else if (bitTst(tecla, 0)) {
                while(bitTst(tecla, 0)) {
                    ssdUpdate();
                    kpDebounce();
                    tecla = kpRead();
                }
                if (pause == 0) {pause = 1;}
    else {pause = 0;}
            }
            else if (bitTst(tecla, 4)) {
                while(bitTst(tecla, 4)) {
                    ssdUpdate();
                    kpDebounce();
                    tecla = kpRead();
                }
                ssdDigit(0, 0);
                ssdDigit(0, 1);
                ssdDigit(0, 2);
                ssdDigit(0, 3);
                return;
            }
        }
        if (pause == 0) {
            tempo -= 1;
            pwmSet(100);
        } else {
            pwmSet(0);
        }
    }
    TRISA=0x00;
    pwmSet(0);
    bitSet(TRISC, 1);
    atraso_ms(500);
    bitClr(TRISC, 1);
    return;
}
```

Essa função realiza tais tarefas enquanto a música não tiver terminado, e, assim, é implementada em sua maior parte dentro de um loop while, que depende do tempo de duração da música. Nesse loop, o tempo é convertido para o display de 7 segmentos, e o LCD se mantem ativo com o nome da música e os comandos. A interação com o usuário é realizada pelo teclado, mas, nesse caso, foi utilizada uma estrutura if/while para cada uma das teclas utilizadas, de modo a evitar problemas no debouncing.

Essa estrutura está disposta dentro de um for que é executado 100 vezes, cada um com 10ms de atraso,

afim de atualizar o tempo ao fim desse ciclo, enquanto ainda mantém a interatividade com o usuário. Ao fim da música, os displays de 7 segmentos e o cooler são desligados, e o buzzer ativado por meio segundo.

Figura 3: *Menu de interação equanto a música é executada.*

iv. Alterar Volume

Por fim, houve a necessidade de outra função auxiliar, a alterarVolume(). Essa função é chamada pela playSong() quando a tecla 1 ou 2 é pressionada. Tal função soma, ou subtrai o volume atual, e o exibe no barramento de LEDs brevemente.

```
void alterarVolume(char opt) {
    if (opt == 1) {
        if(volume!=8){volume += 1;}
    } else {
        if(volume!=0){volume -= 1;}
    }
    unsigned char old_D, old_A;
    old_D = TRISD;

    PORTA=0x00;
    TRISD = 0x00;

    if (volume == 0) {
        PORTD = 0b00000000;
    } else if (volume == 1) {
        PORTD = 0b10000000;
    } else if (volume == 2) {
        PORTD = 0b11000000;
    } else if (volume == 3) {
        PORTD = 0b11100000;
    } else if (volume == 4) {
        PORTD = 0b11110000;
    } else if (volume == 5) {
        PORTD = 0b11111000;
    } else if (volume == 6) {
        PORTD = 0b11111100;
    } else if (volume == 7) {
        PORTD = 0b11111110;
    } else if (volume == 8) {
        PORTD = 0b11111111;
    }
    atraso_ms(500);
    TRISD=old_D;
}
```

v. Main

Por fim, a função main foi implementada de maneira simples, apenas iniciando as bibliotecas importadas e preparando a apresentação do projeto. Além disso, a função chooseSong() é chamada dentro do for infinito.

```
void main(void) {

    songsInit();
    lcdInit();
    ssdInit();
    kpInit();
    lcdCommand(ON);

    lcdPosition(0, 6);
    lcdStr("MP3");
    lcdPosition(1, 4);
    lcdStr("Player");
    atraso_ms(5000);

    lcdCommand(CLR);
    lcdPosition(0, 0);
    lcdStr("Escolha a musica");
    for (;;) {
        lcdPosition(1, 0);
        lcdStr("<-(1) (*)  (2)->");
        chooseSong();
    }
}
```