Universidade Federal de Itajubá

Engenharia de Computação

Programação Embarcada e Laboratório de Programação Embarcada

Documento Técnico – ECOP04 e ECOP14

Simulação de um MP3 Player

Aluno:

João Pedro Rabelo Melo Ferreira – Matrícula: 2018004282

01 de agosto de 2021

Introdução

Um sistema embarcado (ou sistema embutido) é um sistema microprocessado no qual o computador é completamente encapsulado ou dedicado ao dispositivo ou sistema que ele controla. Diferentemente de computadores de propósito geral, como o computador pessoal, um sistema embarcado realiza um conjunto de tarefas predefinidas, geralmente com requisitos específicos. Já que o sistema é dedicado a tarefas específicas, através de engenharia pode-se otimizar o projeto reduzindo tamanho, recursos computacionais e custo do produto.

Por conseguinte, este relatório visa apresentar o desenvolvimento de um sistema embarcado, o qual fará com que um microcontrolador funcione como um aparelho de MP3.

O Projeto foi desenvolvido durante o primeiro semestre do ano de 2021 nas disciplinas de Programação Embarcada (ECOP04) e Laboratório de Programação Embarcada (ECOP14), da Universidade Federal de Itajubá (UNIFEI).
O aparelho de MP3, terá um menu de comunicação com o usuário, informações de duração e nome da música que está sendo tocada, nível de volume da música entre outras funcionalidades que serão melhores relatadas no decorrer do relatório.
Foram utilizados simuladores para o desenvolvimento da proposta sendo eles o PicGenios, com o microcontrolador PIC18F4520, parte da popular família de 8 bits e núcleo de 14 bits, produzido pela MICROSHIP, sendo usado para simular a placa, o software MPLAB para desenvolvimento dos códigos e o compilador XC8.
Para a criação do programa, foi utilizada a linguagem C e várias bibliotecas, que nos foram dadas ao longo da disciplina, para o microcontrolador esse, que possui as funcionalidade de Display LCD, display de 7-segmentos, LEDs, teclado numérico, cooler, relés, buzzer, entre outras.

Objetivo

1. Simular um aparelho MP3

O objetivo do projeto é simular o funcionamento de uma aparelho MP3, onde o usuário terá acesso a um Menu com a opção de escolher as músicas que quer executar, sendo possível, ao executar uma música, controlar o volume do som, aumentando ou abaixando, mudar para as próximas músicas ou voltar em outra playlist.

Especificações

A placa PicGenios, com o microcontrolador PIC18F4520 possui diversos periféricos para utilização, e alguns foram usados no projeto, sendo eles:

• Display LCD: É o menu por onde o usuário irá se comunicar;

• Display 7 Segmentos: Mostra ao usuário as informações da música, por exemplo, o tempo discorrido ao apertar o play;

• Teclado: É a interação com o menu, acessar próxima música, música anterior e pausar;

• LEDs: Nível de volume das músicas;

• Cooler: Simula a caixa de som.

• Buzzer: Aviso sonoro indicando o fim da música.

Foram usados os seguintes softwares para o desenvolvimento do projeto:

MPLAB X IDE: https://www.microchip.com/mplab/mplab-x-ide

Compilador XC8: https://www.microchip.com/mplab/compilers

PICSimLab: PICSimLab - Prog. IC Simulator Lab. - Browse Files at SourceForge.net

Desenvolvimento

O programa utiliza diversas funcionalidades disponíveis na placa, sendo elas o display LDC, display de 7- segmentos, teclado numérico, cooler e buzzer.
O display LCD é utilizado para toda interação com o jogador, mostrando segundos de músicas já tocadas. O teclado numérico é utilizado para inserir informação no programa, sendo elas os números da sequência, conta com números 1 e 2, além do '*' e '#'.
O cooler é utilizado como um indicativo visual, para demonstrar que o Mp3 está tocando perfeitamente.

Código

1. **Bibliotecas**

   O código foi desenvolvido na linguagem C utilizando o programa MPLAB e simulado utilizando o PicSimLab, diversas bibliotecas próprias para o PIC18F4520 foram utilizadas, como:

pic18f4520.h – biblioteca padrão para utilização do PIC18F4520.

config.h – Biblioteca fornecida pelo professor.

bits.h - Biblioteca fornecida pelo professor.

lcd.h - Biblioteca fornecida pelo professor.

ssd.h - Biblioteca fornecida pelo professor.

keypad.h - Biblioteca fornecida pelo professor, visa simplificar o uso do teclado

delay.h - Biblioteca desenvolvida para criar um atraso no programa.

delay o delay(char): Pausar o programa pelo tempo em segundos recebido no parâmetro.

o delay_ms(unsigned int): Pausar o programa pelo tempo em milisegundos recebido no parâmetro.

o cronômetro(unsigned long int): Apresentar uma contagem regressiva no display de 7 segmentos do tempo recebido pelo parâmetro.

pwm.h - Biblioteca fornecida pelo professor.

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

A implementação da função iniciaMusica, configura o TRISC, para uma futura ativação do buzzer, e carrega a lista com o nome de todas as músicas e suas durações.

2. Main

A função main foi implementada para apenas iniciar as bibliotecas importadas e preparar a apresentação do projeto.

Na Main foi implementado o visual do nosso Mp3 Player, como por exemplo o desenho da Unifei, o seu logo, o meu nome(João Pedro Rabelo Melo Ferreira) e minha matrícula(2018004282) como forma de apresentação.

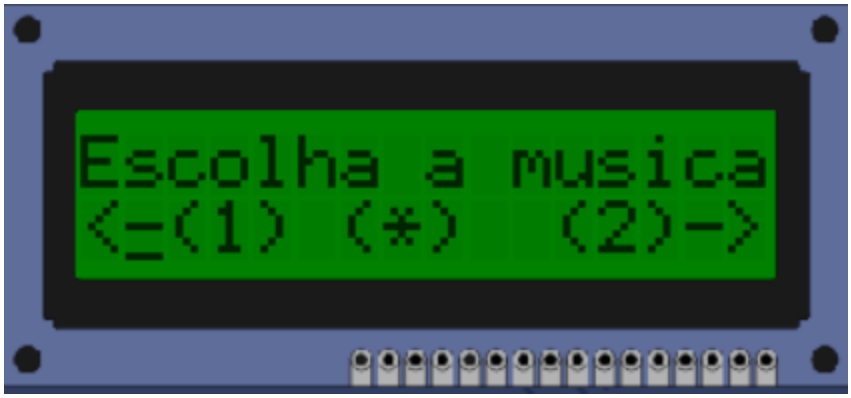
```
char logo[48] = {
    0x01, 0x03, 0x03, 0x0E, 0x1C, 0x18, 0x08, 0x08,
    0x11, 0x1F, 0x00, 0x01, 0x1F, 0x12, 0x14, 0x1F,
    0x10, 0x18, 0x18, 0x0E, 0x07, 0x03, 0x02, 0x02,
    0x08, 0x18, 0x1C, 0x0E, 0x03, 0x03, 0x01, 0x00,
    0x12, 0x14, 0x1F, 0x08, 0x00, 0x1F, 0x11, 0x00,
    0x02, 0x03, 0x07, 0x0E, 0x18, 0x18, 0x10, 0x00,
};
```

Código para o desenho da Logo Unifei.

```
char text6[8]="  UNIFEI";
char text7[10]="Joao Pedro   ";
char text8[16]="      2018004282";
```

Apresentação inicial do programa.

3. Escolha da Música



Dando início na biblioteca, vamos para a próxima etapa, que é o menu principal e a escolha da música a ser tocada pelo usuário.

A função escolheMusica() foi criada para atender nessa questão,

que exibe no Display de 7 segmentos o índice da música e, por fim, realiza uma interação com o usuário pelo teclado.

```
void escolheMusica(void) {
    kpDebounce();
    tecla = kpRead();
    ssdUpdate();
    if (bitTst(tecla, 3) || bitTst(tecla, 7)) { //Tecla
        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;
```

```
                flag = 0;
            }
        }
        tocaMusica();
    }
}
```

Pelo código da função escolhaMusica entendemos o que foi feito e como ela funciona, é necessário que o usuário pressione uma das teclas, ou 1 ou 2, para prosseguir na escolha da música. Após essa etapa, entramos em um for infinito para a interação com o usuário pelo teclado, sendo que a cada execução do for, o display de LCD é atualizado, ou para a próxima música ou para a anterior, conforme pressionado pelo usuário.

Para interromper o For, o usuário precisa pressionar a tecla (*), que indica que ele quer iniciar a música escolhida, sendo assim é executada a próxima função do código.

4. Tocar música

Após o processo de escolher a música, chega o momento em que o usuário pode ouvir sua música desejada, portanto, é agora que implementamos a função tocarMusica, onde o usuário pode dar Play ou Pause na música e alterar o seu volume, como preferir.

```
void tocaMusica() {
    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 as tarefas do cooler, buzzer e o retorno ao menu principal 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 mantém 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 10 ms de atraso, a fim 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.

5. Alterar Volume

Por último, foi implementado uma função volume, uma função auxiliar alterarVolume(). Essa função é chamada pela função tocarMusica() quando o usuário pressiona ou a tecla 1 ou a tecla 2, sendo essas função dedicada a somar ou subtrair do volume atual, e por fim exibe na barra de led’s.

```
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;
}
```

Link vídeo YOUTUBE:

https://youtu.be/bToDZhGGams

Link GitHub:

https://github.com/Jotape01/Simulador-MP3-Embarcada-FInal