INSTITUTO FEDERAL DE CIÊNCIA, EDUCAÇÃO E TECNOLOGIA
DE SÃO PAULO - IFSP - Campus São Paulo

Engenharia de Controle e Automação - 6º Semestre

N5LB5 - Laboratório Integrado III - Laboratório de Microcontroladores
N6MCL - Microcontroladores

Professores:

Me. Rodrigo Rech

Me. Alexandre de Jesus Aragão

Departamento de Elétrica – DEL

Projeto de Um Controle de Iluminação com Tiristores

Datas de entrega: 04/12/2019 (LB5), 06/12/2019 (MCL)

Integrantes:

Deivid De Campos Cazuza - SP1766309

Pedro Henrique Alves Rosendo - SP176652X

Weslley Lacerda da Silva - SP1761595

São Paulo

2019

SUMÁRIO:

1. INTRODUÇÃO

A realização desse projeto tem como objetivo o desenvolvimento de um sistema microcontrolado capaz de atuar como controlador de luminosidade de uma lâmpada em diferentes modos. Para tal foi projetado um conjunto de hardware, para detecção de pontos de interesse da rede de alimentação (zero-cross detector) e acionamento da lâmpada dimerizável, através de um TRIAC (Triode for Alternating Current), e software com base no microcontrolador PIC programado a partir da linguagem C.

O controle de nível de luminosidade da lâmpada é realizado através de um pulso de acionamento gerado pelo microcontrolador que, atuando sobre o TRIAC , regula o período de condução no qual a lâmpada estará inserida por semiciclo da rede de alimentação. Desta forma, valores de intervalo de tempo (**α** na figura abaixo) menores entre o ponto de início do semiciclo (indicado pelo circuito de zero-cross) e o pulso de acionamento gerado pelo PIC resultam em uma potência maior entregue à lâmpada e, portanto, um nível maior de luminosidade, assim como valores de intervalo de tempo maiores resultam em uma potência menor entregue.

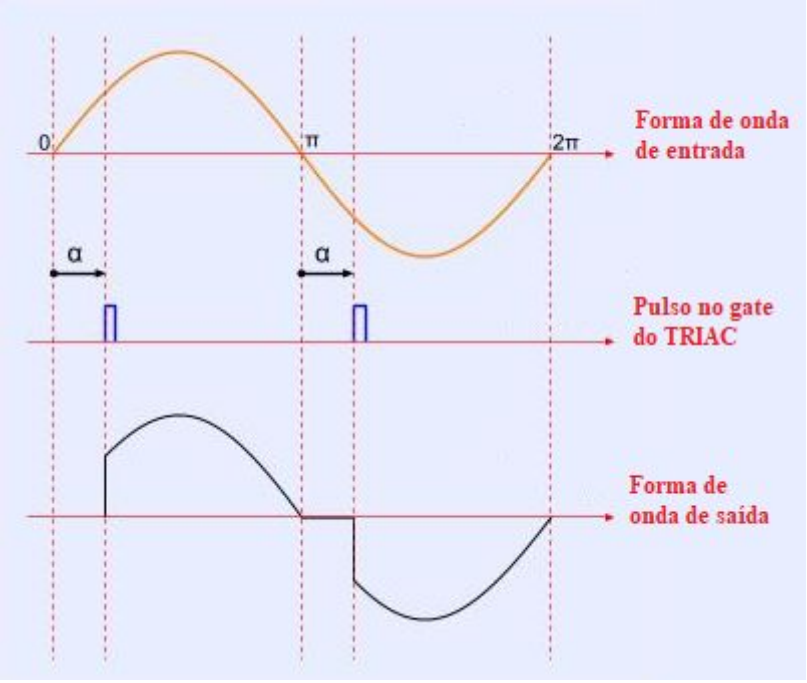


Figura 1 - Formas de onda de entrada, saída e de pulso de acionamento. (Fonte: Autores, 2019).

O projeto trouxe a proposta de funcionamento da lâmpada em 4 modos diferentes: modo de controle de luminosidade, fade, flash e timer. Além disto também foi requisitada a implementação de uma função adicional, a critério do grupo e sob autorização do professor,

que foi definida como um circuito de controle de luminosidade a partir das variações medidas em um sensor LDR (light dependent resistor ou fotoresistor).

Com a função de **controle de luminosidade** é possível regular o brilho da lâmpada em máximo, mínimo (desligada) e mais 100 intervalos intermediários.

Com a função **fade** é possível selecionar um intervalo de brilho até o qual a lâmpada acenderá ou apagará com variação gradual de nível de luminosidade.

Com a função **flash** é possível escolher entre 5 valores de tempo (1, 2, 3, 4 ou 5 segundos) para o período que a lâmpada ficará tanto ligada como desligada durante seu ciclo de liga-desliga.

Com a função **timer** é possível definir um valor de tempo (de 5 à 60 segundos) durante o qual a lâmpada deve ficar acesa e, depois disso, apagar.

A função adicional de **controle por LDR** consiste em utilizar a leitura pelo microcontrolador da variação de tensão sobre o LDR (um sinal analógico) para controlar a luminosidade da lâmpada, algo muito utilizado em sistemas de controle automático de iluminação pública para postes. Caso o nível do sinal medido com o LDR seja menor que um nivel pré estabelecido pelo usuário a lâmpada deve ligar, desligando caso seja maior. Por se tratar de um sinal de amplitude muito pequena utiliza-se um circuito de amplificação para ajustar o sinal de leitura à níveis aceitáveis pelo PIC.

2. DESENVOLVIMENTO

HARDWARE INTERNO MICROCONTROLADOR (PIC 16F876A)

Device	Program Memory		Data SRAM (Bytes)	EEPROM (Bytes)	I/O	10-bit A/D (ch)	CCP (PWM)	MSSP		USART	Timers 8/16-bit	Comparators
	Bytes	# Single Word Instructions						SPI	Master I^2C			
PIC16F876A	14.3K	8192	368	256	22	5	2	Yes	Yes	Yes	2/1	2

Figura 2: Tabela de especificações PIC 16f876a (Fonte: Microchip, 2003).

❖ TIMER 0

O timer0 é um contador de 08 bits responsável por uma das flags de interrupção do PIC 16f876a. Seu diagrama de blocos pode ser observado na figura 3.

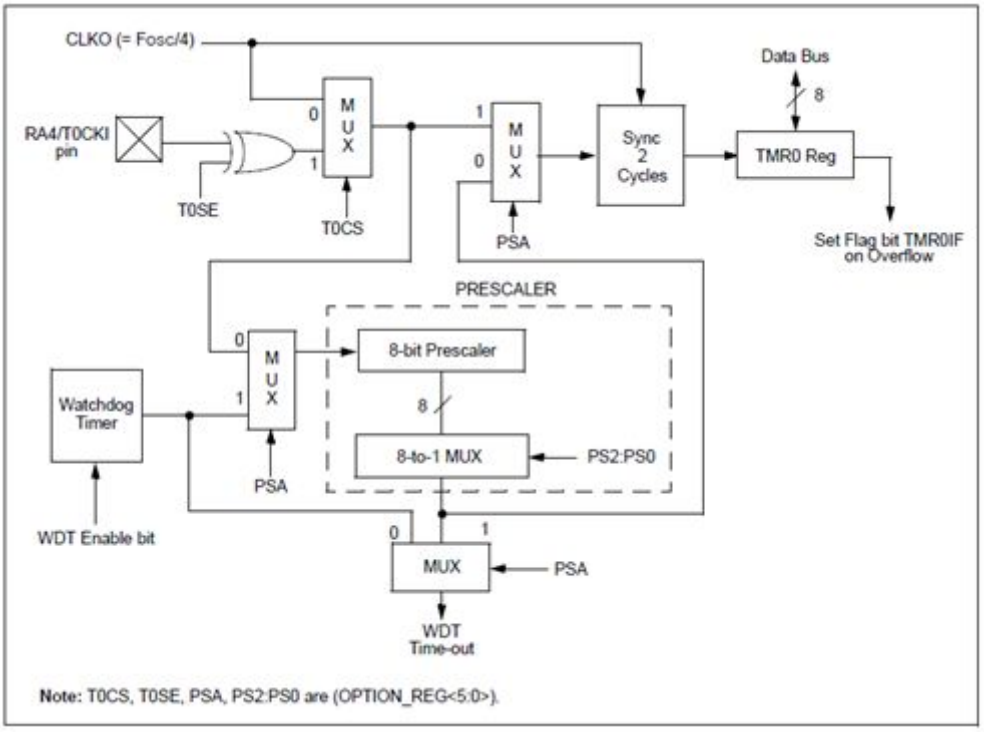


Figura 3: Timer0 PIC 16f876a (Fonte: Microchip, 2003).

Como pode ser observado no diagrama de blocos, este contador possui um prescaler de 8 bits, podendo seu incremento ser configurado para acontecer através da frequência de clock dividida por 4, na família de PIC midrange este é o conhecido ciclo de máquina, ou através do pino RA4. Independente do sinal de sincronia utilizado na contagem, o bit TMR0IF é setado no overflow do registrador TMR0, podendo o tempo de estouro ser ajustado através de reload neste registrador ou através da divisão da frequência a partir do prescaler.

A varredura dos botões ENTER, BACK, ON e OFF foi feita através deste timer com tempo de varredura de 100 ms. O valor pré-carregado no registrador de contagem será 0x3C, assim ocorrerão 196 contagens. No caso do timer0, o tempo de overflow é dado pela expressão 1. Note que o termo 2^8 resulta do número de bits do registrador.

$$T_{Ovf} = (2^8 - TMR0) \times CM \times Prescaler \qquad (1)$$

$$T_{Ovf} = (256 - 60) * 200 \times 10^{-9} \times 256$$

$$\therefore\ T_{Ovf} \cong 10\ ms$$

Utilizando variáveis auxiliares para contagem, pode-se obter facilmente tempos múltiplos do tempo de overflow conforme demandas do projeto. Para a varredura dos botões, optou-se por um tempo de 100ms pois é suficientemente pequeno para evitar que o não tratamento do pressionar do botão e grande suficiente para evitar ruídos mecânicos da chave.

- **TIMER 1**

O timer1 é utilizado neste projeto devido ao seu registrador de contagem ser de 16 bits – Na verdade, trata-se de dois registradores de 8 bits, um responsável pela contagem dos oito bits menos significativos e o outro pelos oito bits mais significativos - o que é conveniente uma vez que o ciclo de máquina é de apenas 200 ns. A equação para o seu tempo de estouro é similar a equação 1, porém, ao invés de considerarmos 2^8, consideramos 2^16, por motivo já explicado. O tempo de overflow máximo do timer 1 é:

$$T_{Ovf_{MAX}} = 2^{16} \times 200 \times 10^{-9} \times 8 \cong 104{,}86\ ms$$

É desejável que o tempo de estouro máximo seja igual ao tempo de um semiciclo pois assim o controle de ON/OFF será dado de forma mais simples. Logo, rearranjando os termos da equação de overflow e usando um prescaler de 1:1, podemos calcular o valor de contagem.

$$Cnt = \frac{8333}{200 \times 10^{-9}} \cong 41666$$

Logo, o registrador, para o valor máximo do período deve ser carregado com 5D3E_h. Devido ao fato do microcontrolador necessitar de um tempo de processamento, o valor de contagem (Cnt) serve apenas de parâmetro para a calibração que deve ser feita baseando-se em testes práticos. A figura 4 contempla o diagrama de blocos deste módulo.

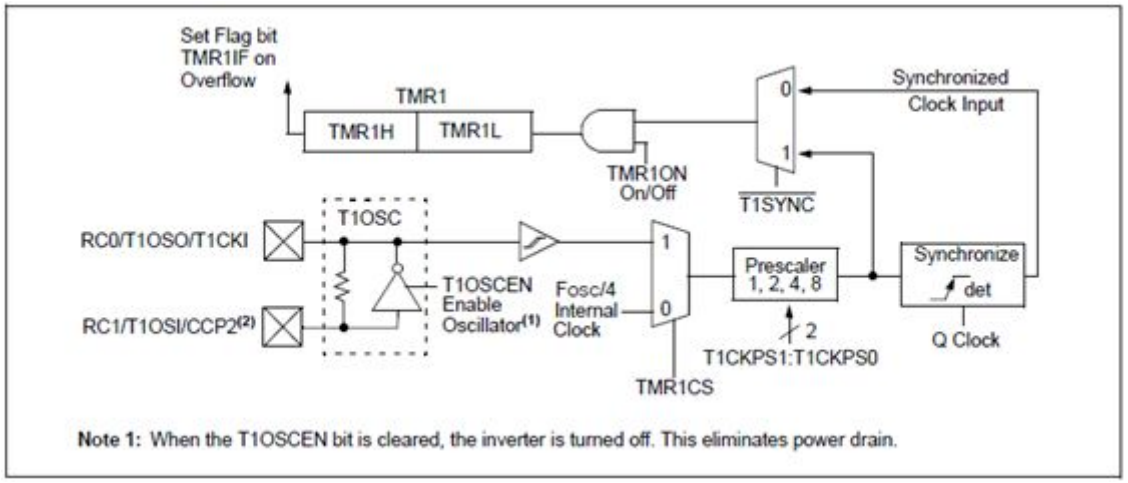


Figura 4: Diagrama de blocos Timer1 PIC 16f876a (Fonte: Microchip, 2003).

- **INTERRUPÇÃO EXTERNA**

Este modelo de microcontrolador possui apenas um pino de interrupção externa, sendo a mesma ocupada pelo circuito de zerocross. A interrupção externa é, basicamente, a responsável por fazer tratamentos de maior prioridade baseada em fenômenos externos dentro do microcontrolador. Ao detectar uma mudança no pino de interrupção, o processador efetua um desvio para a região de tratamento da interrupção externa, o que no caso dos microcontroladores PIC da família midrange é o endereço 0004h da memória para todas as interrupções. Neste endereço, cada flag é checada por pooling e consequentemente a mesma deve ser limpa em software, para garantir que já houve o tratamento desta.

HARDWARE EXTERNO MICROCONTROLADOR

Para tornar o microcontrolador capaz de identificar o início dos semiciclos da senóide se faz necessário a utilização de um circuito capaz de identificar o momento em que a senoide passa pelo zero. Este circuito é conhecido como zerocrossing. Existem diversas topologias para este tipo de circuito. Neste projeto, optou-se pela topologia da figura 5, utilizando um optoacoplador para detecção dos zeros.

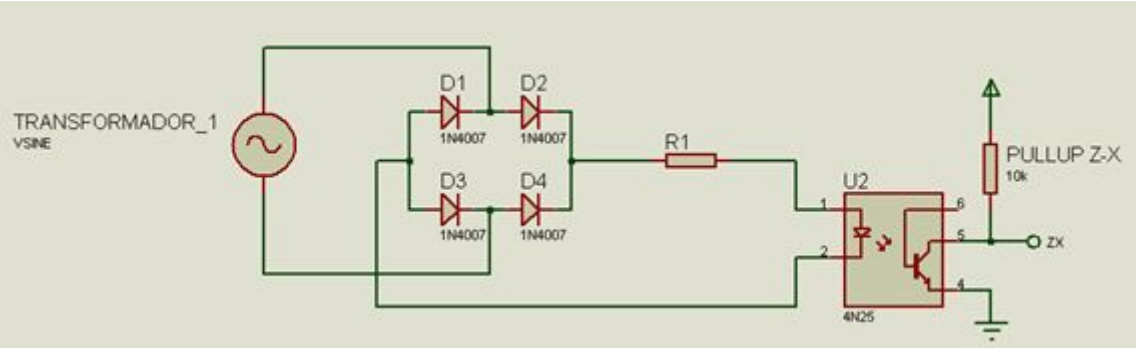


Figura 5: Circuito Zerocross (Fonte: Autores, 2019).

Para o cálculo do resistor R1, leva-se em consideração a corrente que o diodo do Optoacoplador suporta, segundo seu datasheet, algo em torno de 60 mA. Logo:

R_1 adotado: 470 Ω / 1/4 W

A saída do label ZX pode ser observada na figura 6, em verde, nela é possível notar que não há um sinal digital bem definido, o que pode gerar problemas nas componentes digitais do circuito. Para melhorar o sinal de circuitos analógicos tratados com circuitos digitais uma boa prática é o uso de buffers schmitt-trigger.

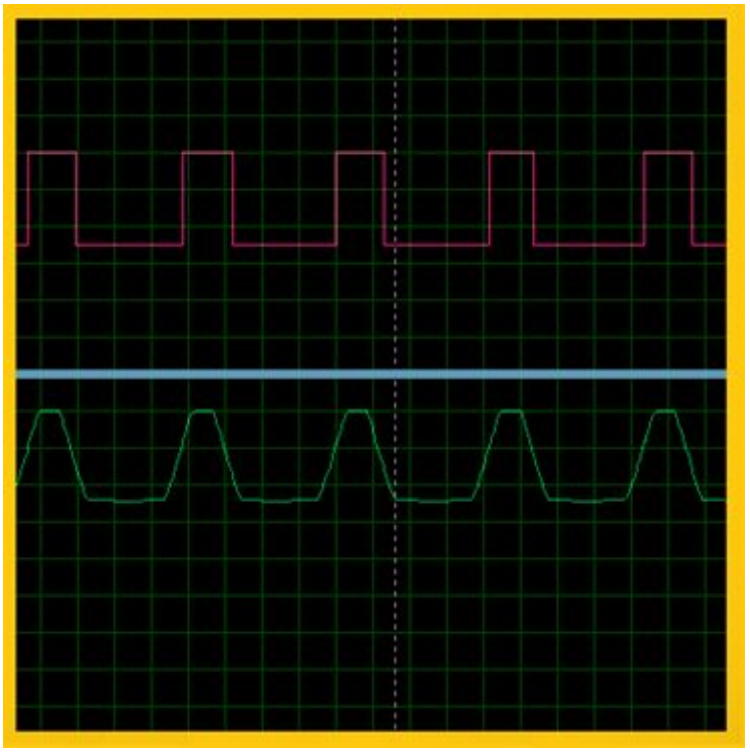


Figura 6: Sinal no label ZX antes e depois de ser recuperado (Fonte: Autores, 2019).

Existem diversas topologias para buffers schmitt-trigger, uma das mais comuns é utilizar portas lógicas da família schmitt-trigger. Por questão de componentes já disponíveis, optou-se pela utilização do CI 4093, conectando duas portas NAND schmitt-trigger na

configuração de inversor em série, como mostra a figura 7, assim gerando um buffer não inversor capaz de tratar o sinal como visto na figura 5.

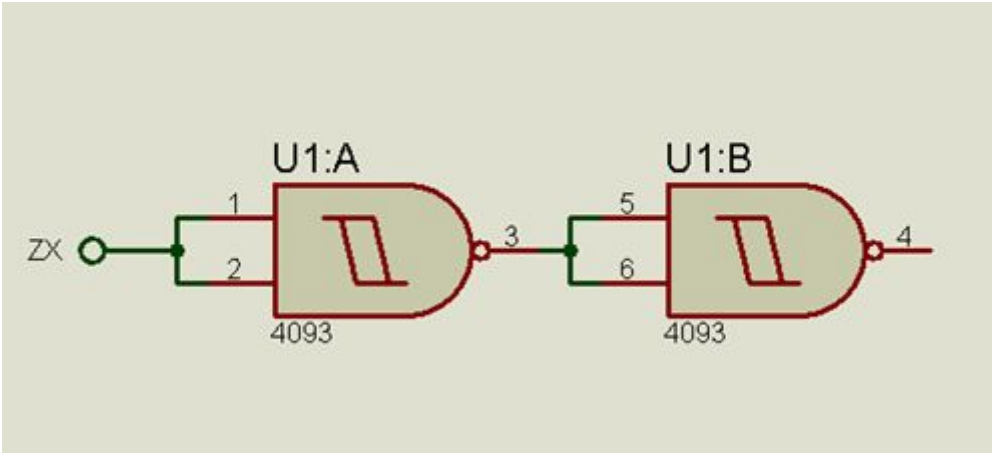


Figura 7: Buffer não inversor com 4093 (Fonte: Autores).

Para a fonte de alimentação, devido ao fato do circuito ter um consumo baixo, e sua facilidade de implementação, optou-se pelo 7805, regulador linear capaz de regular tensões de 7 a 25 volts para 5 Volts/1 Ampère. O diagrama da fonte pode ser visto na figura 8.

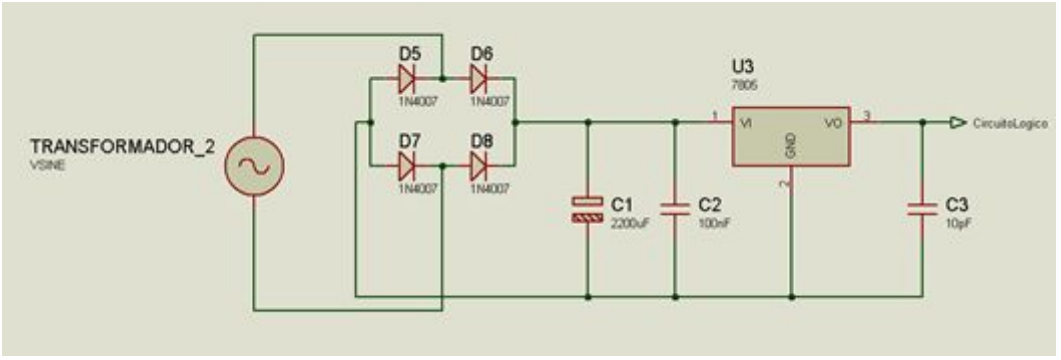


Figura 8: Circuito fonte de alimentação da parte lógica (Fonte: Autores).

Para a implementação extra, pensou-se fazer um modo onde a lâmpada pulsasse conforme o ritmo da música. Para isso, a topologia conhecida como transistor pump, (Thomas Keith Hemingway, 1970. Pg. 215) foi adotada, conforme pode ser visto na figura 9.

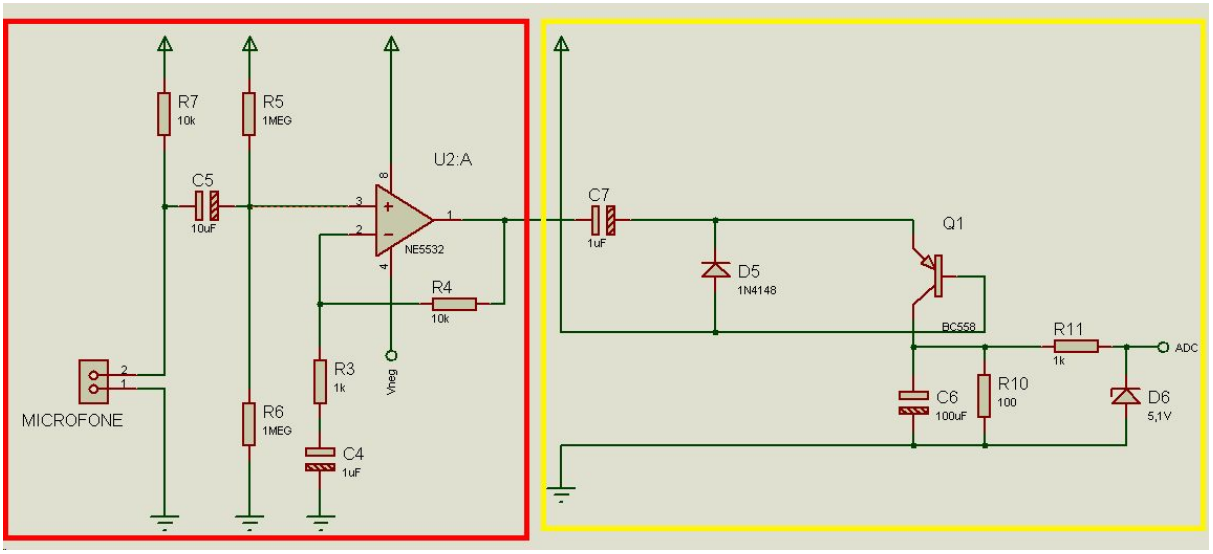


Figura 9: Amplificador de entrada e transistor pump (Fonte: Autores).

Na figura 8, destacada pelo retângulo vermelho, pode-se notar a clássica topologia de amplificador não-inversor, com ganho aproximado de 11 v/v. A única diferença para a topologia mais comum está no capacitor C4 contido no divisor resistivo. Este capacitor é o responsável por gerar um nível de de referência para a entrada inversora, desta forma, o nível DC aplicado devido ao divisor resistivo de R7 e Microfone, não é amplificado. No bloco amarelo, está a configuração de transistor pump. Esta topologia funciona gerando níveis de tensão em C7 menores que o da fonte, instantaneamente carregando o mesmo através do diodo D5. Quando a tensão em C7 for suficientemente maior que a tensão na base do transistor Q1, o mesmo entra em condução, carregando o capacitor C6. A descarga do capacitor C6 é feita através de R10. Note a aplicação do diodo zener para proteção da entrada AD do microcontrolador.

Para alimentação do amplificador operacional, utilizou-se a fonte simples de 5V e uma tensão negativa gerada através do circuito oscilador da figura 10. Este circuito oscilador funciona devido à não existência de uma região proibida em portas do tipo schmitt-trigger. Desta forma, o capacitor em processo de carga e descarga gera níveis lógicos High e Low nas entradas curto-circuitadas, fazendo a porta se comportar como uma inversora, gerando os ciclos de carga e descarga necessários para oscilação. A equação de oscilação deste circuito, embora simples, costuma não retornar boas aproximações, desta forma, os valores do resistor de feedback e do capacitor de oscilação foram escolhidos de forma prática de modo a obter uma frequência da casa de 20kHz.

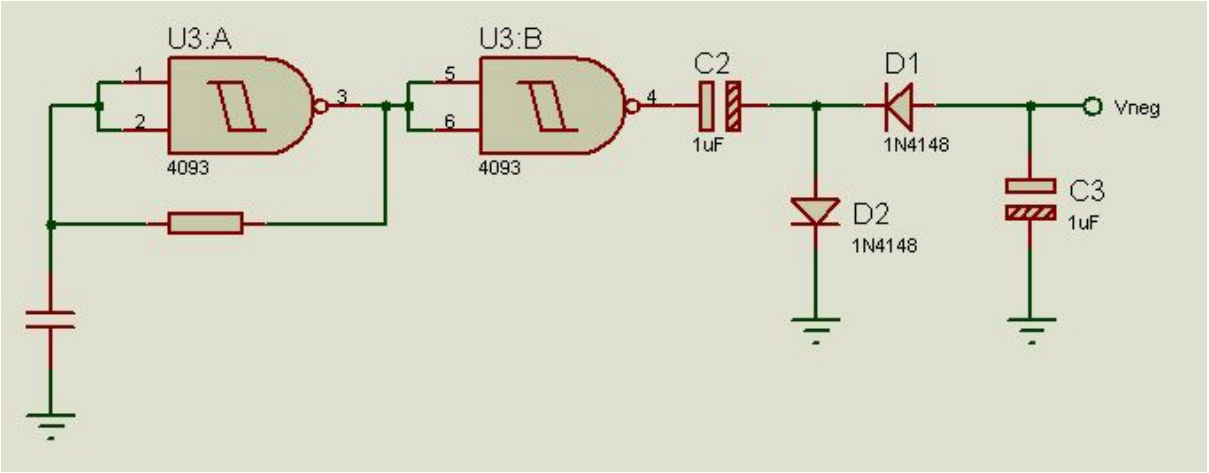


Figura10: Circuito oscilador e gerador de tensão negativa (Fonte: Autores).

Vale ressaltar que o microcontrolador poderia ser utilizado para gerar o circuito oscilador, todavia o chip 4093 possui quatro portas NAND, o que acaba sendo ideal para o projeto.

O circuito de acionamento da lâmpada foi construído baseando-se nas aplicações típicas apresentada no datasheet do optoacoplador MOC3023. O circuito pode ser visto na figura 11.

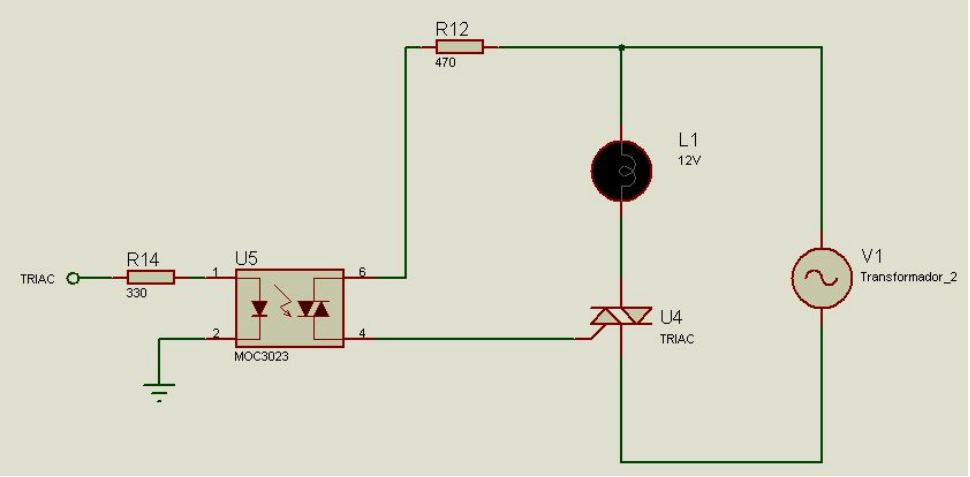


Figura 11: Circuito de acionamento (Fonte: Autores).

Finalmente, a última etapa do Hardware é o microcontrolador, responsável por adicionar maior autonomia ao projeto. Optou-se por utilizar um PIC da família midrange devido ao baixo custo e programação simples, uma vez que o programador possua certo grau de conhecimento em eletrônica digital, bem como a programação de qualquer microcontrolador.

No meio do processo de criação da função de sincronia, mapeou-se os valores de tensão de saída por frequência de entrada, obtendo a tabela 1, abaixo.

Frequência (Hz)	Tensão (mV)
500	59,7
1k	85,8
1,5k	93,9
2k	95,9
2,5k	102,4
3k	100
3,5k	107
4k	102,8
4,5k	104
5k	105,8
5,5k	106,7
6k	107,2
6,5k	107,7
7k	108,8
7,5k	108,8
8k	109,1
8,5k	108,2
9k	108,1
9,5k	108,4
10k	109
11k	108,2
12k	108,2
13k	117
14k	117,1
15k	117
16k	117
17k	117,2
18k	117,2
19k	116,9
20k	110

Tabela 1: Relação Frequência x Tensão

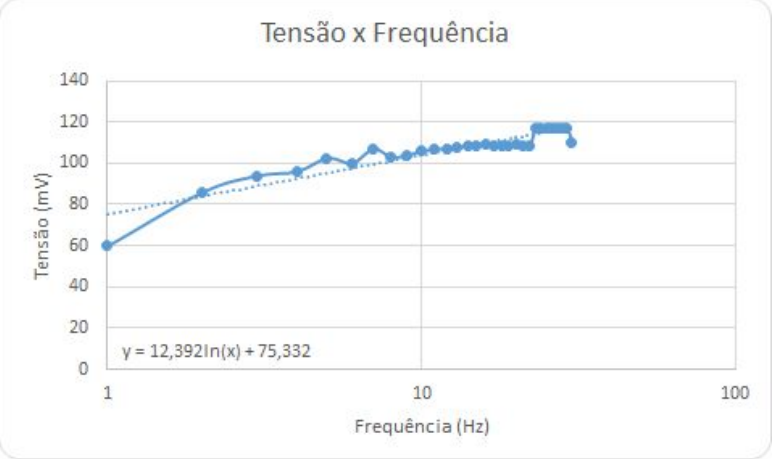


Grafico 1: Relação Tensão x Frequência em escala log (Fonte: Autores).

Ao plotar o gráfico da tabela, para verificação de sua curva de resposta, verificou-se que ela não é linear - isso foi comprovado pelo baixo índice de relação linear, abaixo de 0,6 - portanto, optou-se por trabalhar numa pequena faixa de valores de frequência, faixa entre 500Hz e 3kHz, aplicando um ganho antes da entrada do microcontrolador. O sinal medido, todavia, tinha uma amplitude tão baixa que, mal foi capaz de ser distinguido de ruídos. Com

isso, o projeto de implementação passou a ser a iluminação baseada nas medidas de um LDR, criando uma lâmpada crepuscular. A base para esta topologia pode ser observada na figura 12. Trata-se de um simples divisor de tensão.

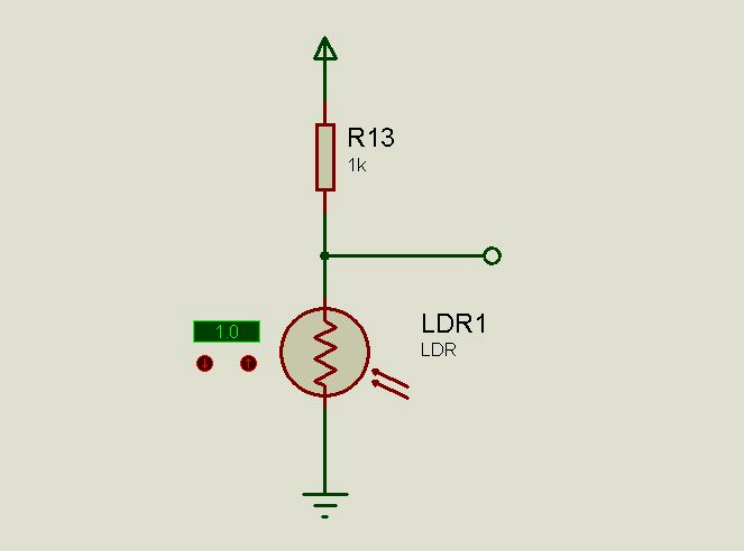


Figura 12: Circuito com LDR (Fonte: Autores).

Todo o código fonte foi escrito no software MikroC Pro for PIC v.4.15.0.0. Este software, embora pago, possui uma versão gratuita, sem limite de caracteres, porém com limite de tamanho de código compilado.

3. FLUXOGRAMA

Abaixo, são apresentados os fluxogramas das funções: Controle de luminosidade, efeito fade, efeito flash e modo timer.

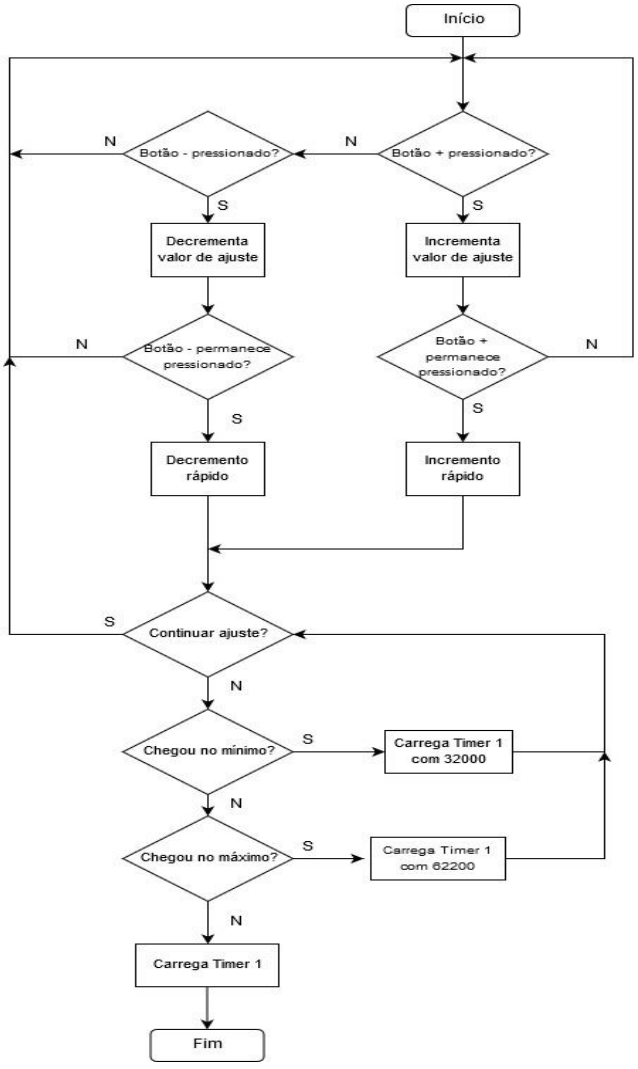


Figura 13 - Fluxograma referente à função Controle de Luminosidade

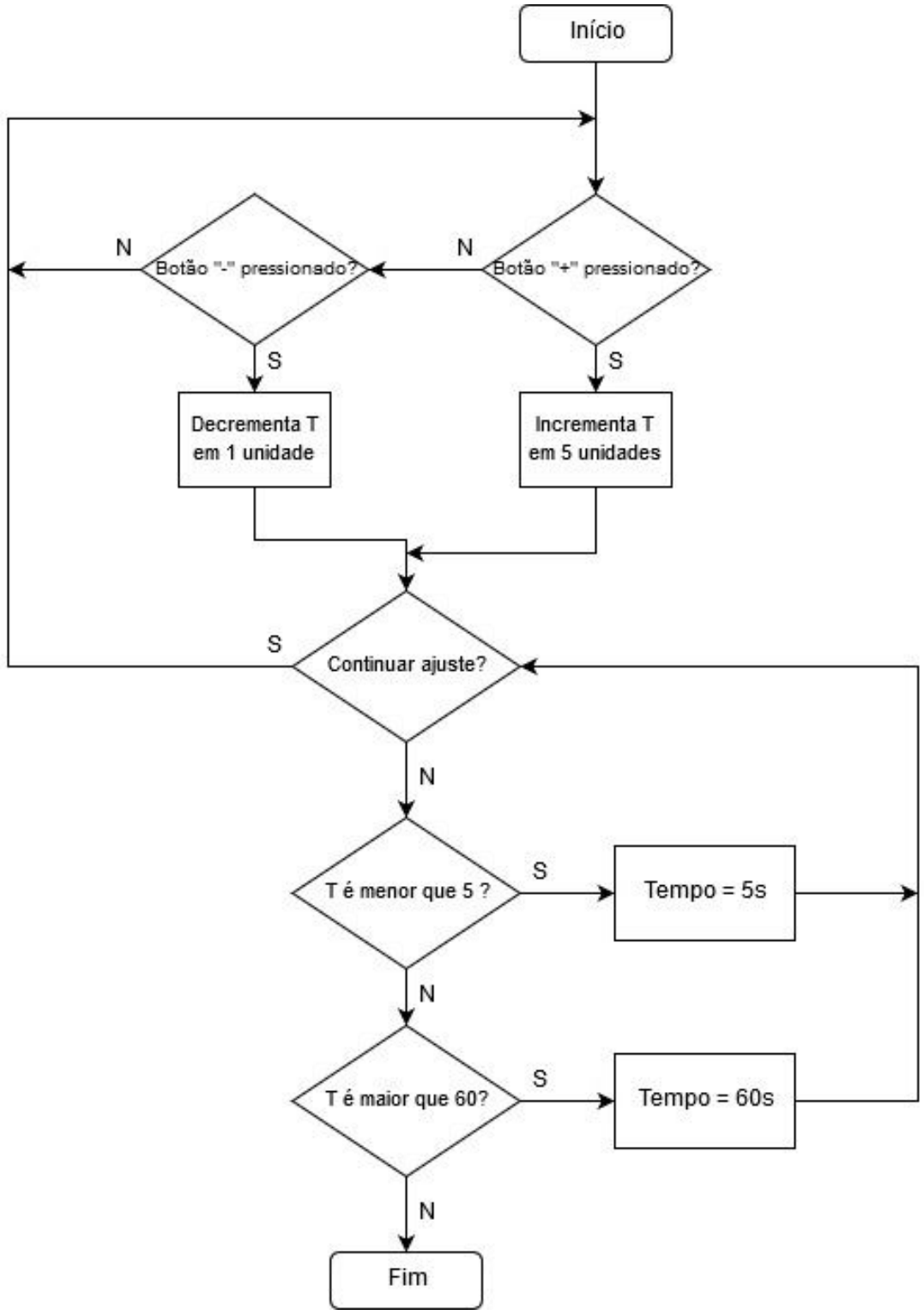


Figura 14 - Fluxograma referente ao Modo Timer

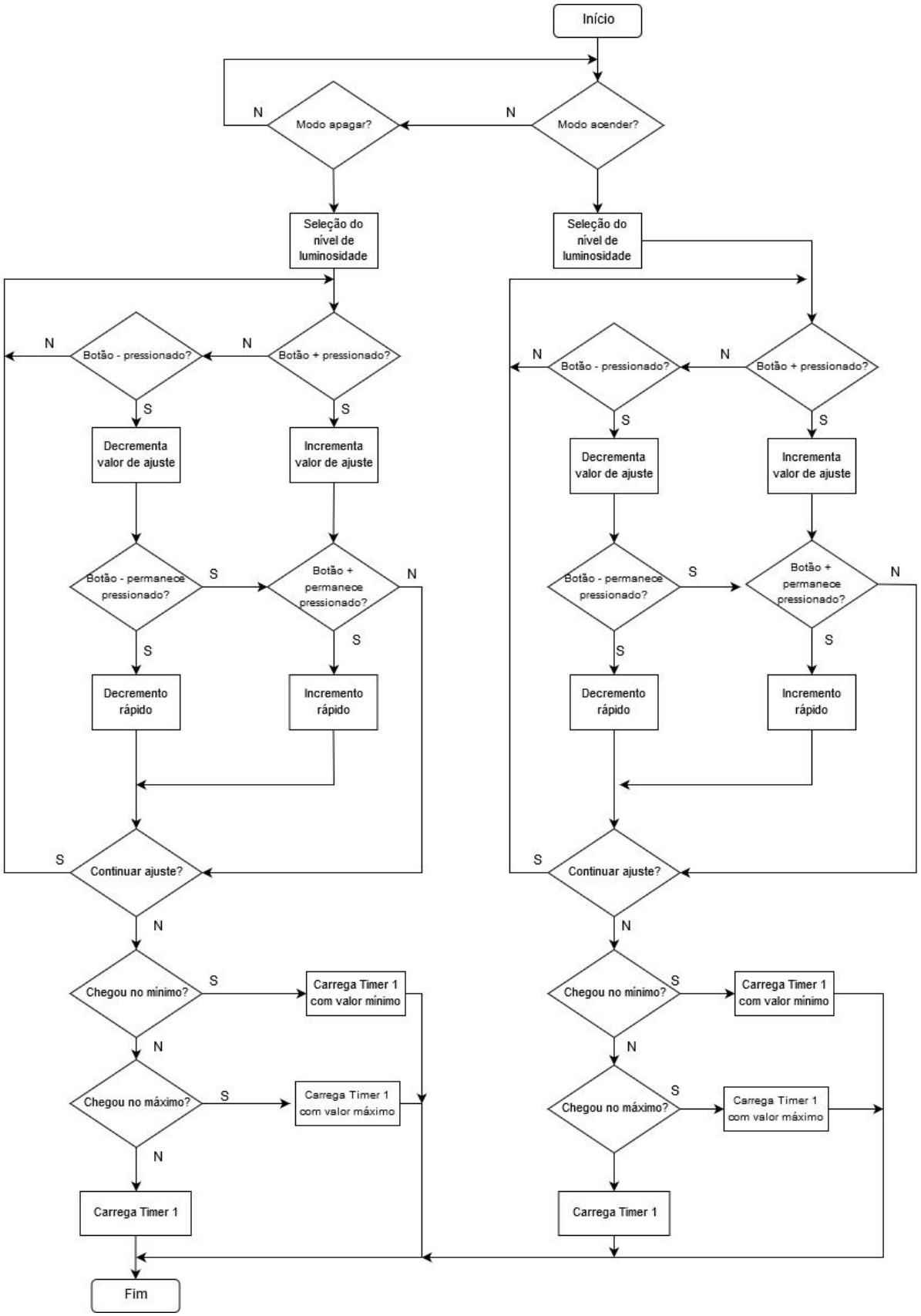


Figura 15 - Fluxograma referente ao efeito fade

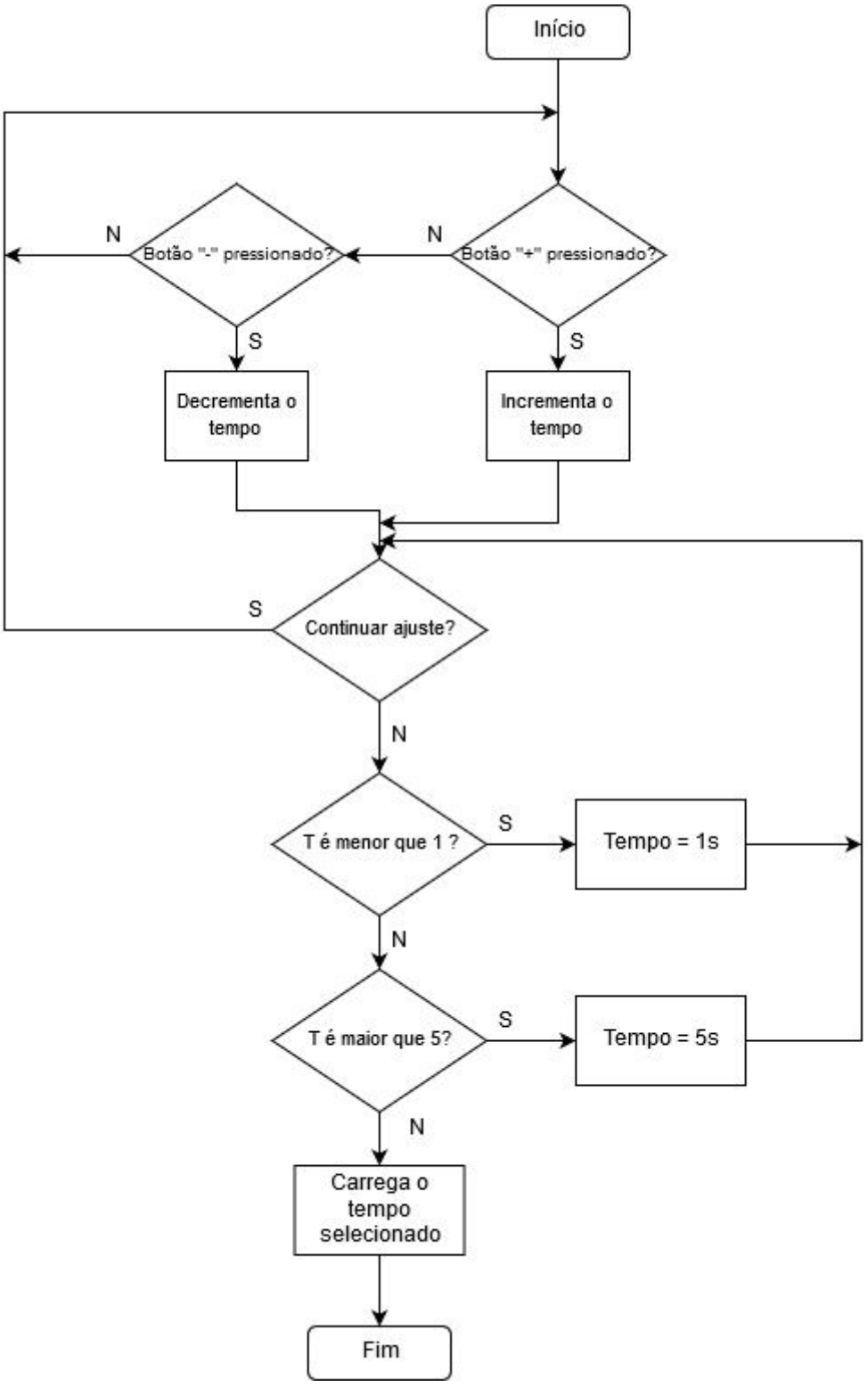


Figura 16 - Fluxograma referente ao efeito flash

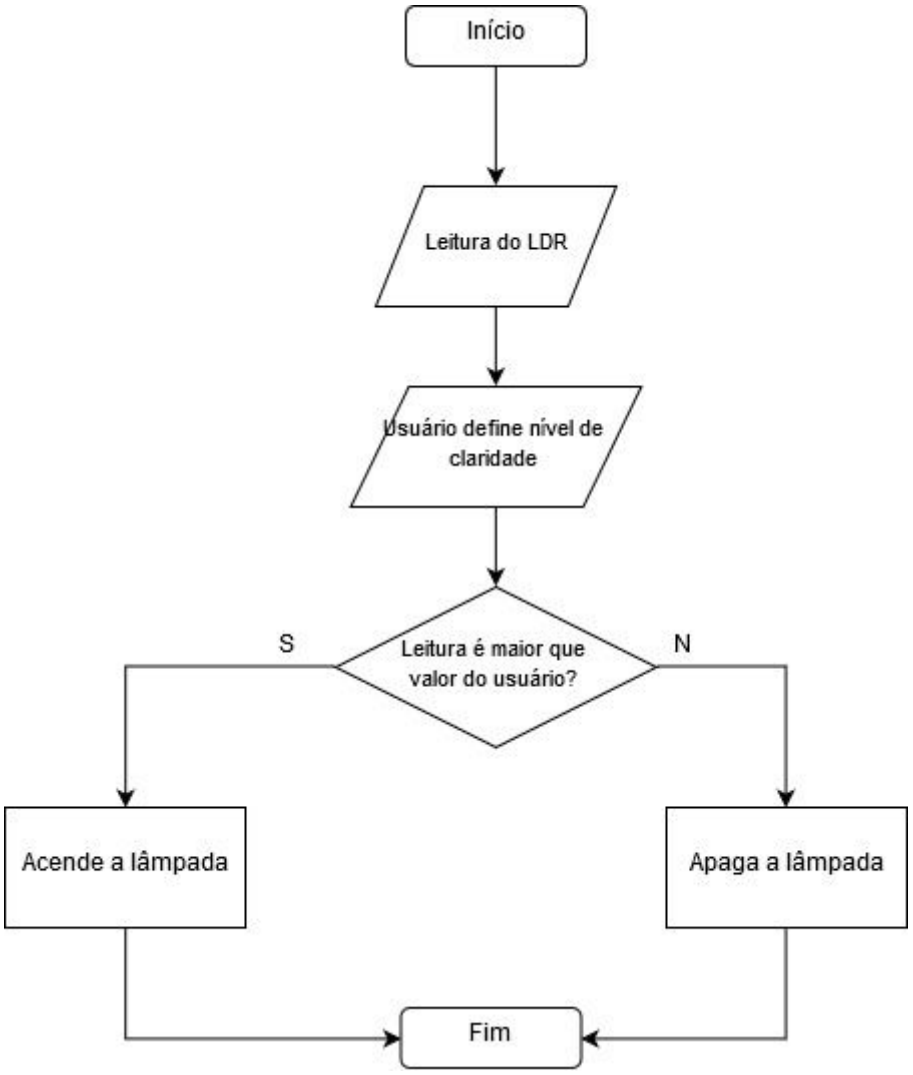


Figura 17 - Fluxograma referente à função personalizada

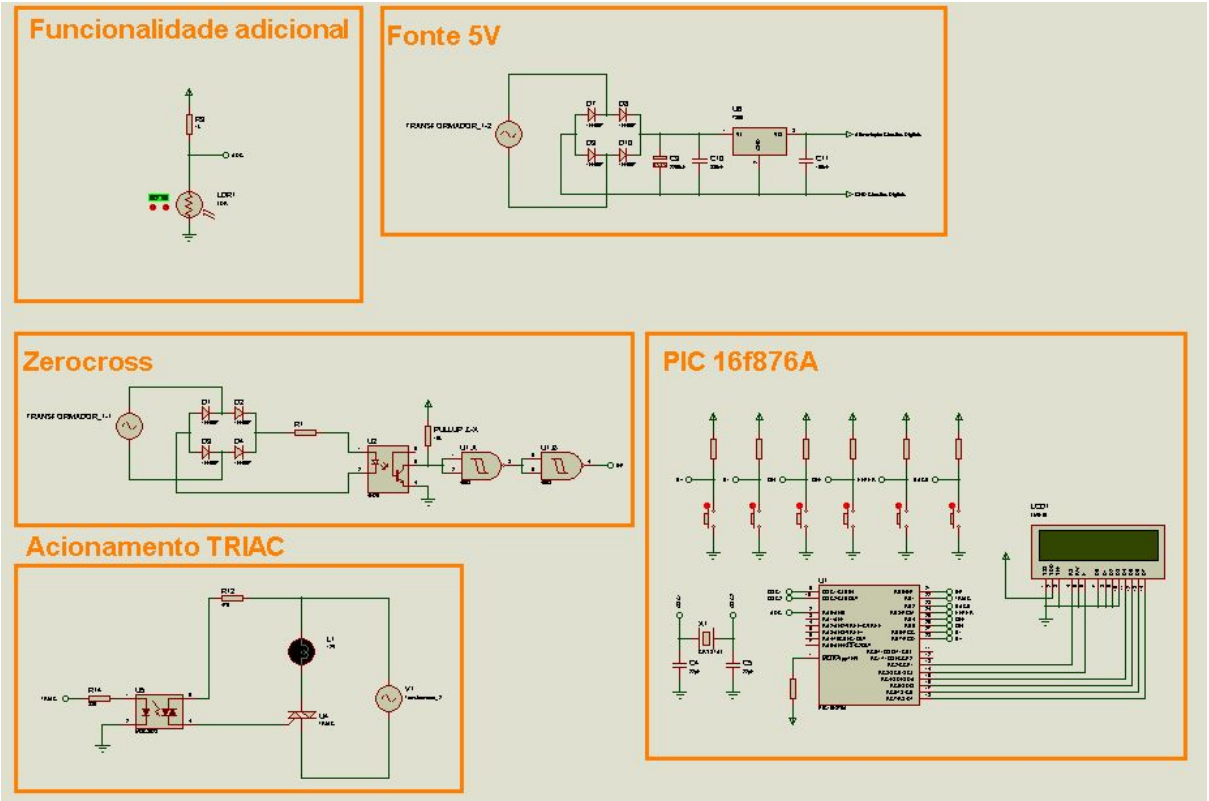
4. CONCLUSÃO

No decorrer do projeto, foram enfrentados alguns problemas na montagem do hardware, que não foram observados na simulação do circuito. Um deles foi o comportamento irregular da lâmpada, que piscava de forma aparentemente aleatória, mesmo não havendo o acionamento do TRIAC. Tal problema foi solucionado com a adição de um diodo na entrada do pino do microcontrolador, pois possivelmente o acionamento estava sendo feito por uma tensão muito baixa, e o diodo, que conduz a partir de uma tensão de próxima de 0,7 V, impediu a passagem dessa tensão indesejada.

No início do projeto, decidiu-se fazer como funcionalidade adicional um amplificador que fazia a lâmpada variar seu brilho a partir das variações de um áudio, no entanto, com a montagem do circuito, não foi observado nenhum sinal amplificado. Optou-se, portanto, por utilizar um LDR para ligar a lâmpada no escuro e desliga-la na presença de luz.

Por fim, pode-se dizer que o presente projeto foi fundamental para a assimilação do conteúdo visto nas disciplinas de teoria e laboratório de microcontroladores. Apesar de ser estudado o funcionamento do microcontrolador 8051 nas aulas, o conhecimento pôde ser aplicado para a programação do PIC sem maiores problemas, visto que o princípio de funcionamento das interrupções e dos timers é o mesmo para ambos microcontroladores.

5. ESQUEMÁTICO



6. CÓDIGO

```
/*### Dimmer microcontrolado com PIC ###

 uC: PIC 16f876A
 Ciclo de máquina: 200ns
 Plataforma de desenvolvimento do software: MikroC PRO for PIC v.4.15.0.0

 Data de inicio: 12 Outubro de 2019
 Última atualização: 28 Outubro de 2019

 OBS: Não é recomendado fazer a chamada de uma função dentro da pilha de
 interrupção.

*/

// ###########################################################################
// --- Mapeamento do Display LCD ---
// LCD module connections
sbit LCD_RS at RC2_bit;
sbit LCD_EN at RC3_bit;
sbit LCD_D7 at RC7_bit;
sbit LCD_D6 at RC6_bit;
sbit LCD_D5 at RC5_bit;
sbit LCD_D4 at RC4_bit;

// Pin direction
sbit LCD_RS_Direction at TRISC2_bit;
sbit LCD_EN_Direction at TRISC3_bit;
sbit LCD_D7_Direction at TRISC7_bit;
sbit LCD_D6_Direction at TRISC6_bit;
sbit LCD_D5_Direction at TRISC5_bit;
sbit LCD_D4_Direction at TRISC4_bit;

// ###########################################################################
// --- Flags ---
#define inc         flagsA.B0       // Flag do botão de incremento
#define dec         flagsA.B1       // Flag do botão de decremento
#define GO          flagsA.B2       // Flag de entrada nos menus
#define Func_up     flagsA.B3       // Flag de incremento
#define ClearLCD    flagsA.B4       // Flag de limpeza do LCD
#define Fade_M      flagsA.B5       // Flag do modo Fade
#define Flash_M     flagsA.B6       // Flag do modo Flash
#define Timer_M     flagsA.B7       // Flag do modo Timer
#define Func_dw     flagsB.B0       // Flag para decremento
#define _fFS        flagsB.B1
#define _fTM        flagsB.B2
#define _fTm        flagsB.B3
#define _fIF        flagsB.B4       //
#define _fFO        flagsB.B5       // Flag usada p/ indicar que está no modo flash
#define entTim      flagsB.B6       // Flag de sinalização de enter dentro de timer (Usada para acionar o
timer)
#define changeSt    flagsB.B7       // Flag p/ mudança do estado da Lampada no Flash
#define flinc       flagsC.B0       // Flag de incremento tempo flash
#define fldec       flagsC.B1       // Flag de decremento tempo flash
```

```
#define statusF  flagsC.B2          // Flag de status do Flash
#define ModoFade flagsC.B3          // Flag de status do Fade
#define atualiza flagsC.B3          // Flag de atualização
#define hab_f    flagsC.B4          // Flag que habilita o flash
#define regula   flagsC.B5

// ###############################################################################
// --- Mapeamento de Hardware ---
#define nmenus   5
#define butinc   RB7_bit   // Pino 28
#define butdec   RB6_bit   // Pino 27
#define ON       RB5_bit // Pino 26
#define OFF      RB4_bit // Pino 25
#define ENTER    RB3_bit // Pino 24
#define BACK     RB2_bit // Pino 23
#define triac    RB1_bit // Pino 22
#define Desl     32000    // ### Valores para ligar e desligar a lâmpada ###
#define Lig      62200    // valor display = (T_ajust - Desl)/((Lig - Desl)/100)
#define Vel      50       // Parametro que altera a velocidade do fade
                          // Tempo = (T_incremento/100)*10^3
                          // T_incremento é o tempo desejado de duração do Fade
                          // dado em segundos

// ###############################################################################
// --- Variáveis Globais ---
char auxcont0 = 0x00,     // Variável auxiliar de contagem Timer0
     sel = 0x01,          // Variável para seleção dos menus
     estado = 0x00,       // Variável de estado Fade
     _butcontp = 0x00,          // Variável usada na rotina de incremento rápido
     _butcontm = 0x00,          // Variável usada na rotina de decremento rápido
     _1seg,               // Variável usada para contar 01 segundos (TMR0)
     contseg,             // Variável usada para contar o tempo no modo Timer
     ContFlash,           // Variável usada para contar o tempo no modo Flash
     auxflash, comparadora, step,
     flagsA, flagsB,      // Registradores de Flags (Mapeados acima)
     flagsC;

unsigned long int  ajust_tmr1 = Desl,          // Valor da lampada apagada
                   ajust_fade = Desl,
                   aux;

unsigned int       LDR_Read, antLe;
// ###############################################################################
// --- Vetor de Interrupção ---
void interrupt ()
{
     // --- Interrupção externa ---
     if (INTF_bit)                    // Testa se houve interrupção externa
     {
     T1CON.F0 = 0x01;                 // Habilita a contagem do TMR1
     triac = 0x00;                    // Descomentar essa linha mais tarde
     INTF_bit = 0x00;                 // Limpa a flag de interrupção externa
     } // end INTF if

     // --- Timer 1 ---
     if (TMR1IF_bit)                  // Testa se houve estouro do TMR1
     {
```

```
        triac = 0x01;                   // Gera o pulso em High no pino triac
        //delay_us (200);               // Comentar essa linha mais tarde
        //triac = 0x00;                 // Comentar essa linha mais tarde

        TMR1L = (ajust_tmr1 & 0x00FF);          // Recarrega para contagem
        TMR1H = (char)(ajust_tmr1 >> 8);

        T1CON.F0 = 0x00;                // Desliga o Timer 1 até a prox. interrupção externa
        TMR1IF_bit = 0x00;              // Limpa a flag de overflow TMR1
        } // end TMR1IF if

        // --- Timer 0 ---
        if (TMR0IF_bit)                 // Testa se houve estouro do timer0
        {
        // === Base de tempo de 10ms ===
        auxcont0++;                     // Incrementa a variável auxiliar de contagem do TMR0




        // === Base de tempo 100 ms ===

        if (auxcont0 == 10)
        {
        _1seg++;                // Incrementa a variavel de contagem de segundo

        auxcont0 = 0x00;        // Zera a variável de contagem

        // === Rotina para implementação do botão com incremento inteligente ===
        // Para a parte de controle...
        if (!_fIF)      // Teste para aproveitar a variável de
        {               // contagem rápida para modo controle e fade

        if (Func_up)    _butcontp++; // Incrementa a variável se flag setada (botão inc pressionado)
        if (Func_dw)    _butcontm++; // Incrementa a variável se flag setada (botão dec pressionado)

        if (_butcontp > 15)     // Verifica se a variável de contagem chegou a quinze sem o botão ser
solto por 1,5s
        {
                _butcontp = 15;         // Garante que não haja overflow da variável
                ajust_tmr1 += 960;      // Incrementa a variavel de ajuste rapidamente
        } // ent _butcontp if

        if (_butcontm > 15)     // Verifica se a variável de contagem chegou a quinze sem o botão ser
solto por 1,5s
        {
                _butcontm = 15;         // Garante que não haja overflow da variável
                ajust_tmr1 -= 960;      // Decrementa a variavel de ajuste rapidamente
        } // ent _butcontm if
        } // end !Fade_M if
        // === Fim da rotina do botão com incremento inteligente ===

        // === Rotina para implementação do botão com incremento inteligente ===
        // Para a parte do fade...
        if (_fIF)               // Se esta flag está setada,
        {               // o programa principal está em fade
```

```
        if (Func_up)        _butcontp++; // Exatamente a mesma explicação
        if (Func_dw)        _butcontm++; // da parte logo acima desta

        if (_butcontp > 15)
        {
                _butcontp = 15;
                ajust_fade += 960;
        } // ent _butcontp if

        if (_butcontm > 15)
        {
                _butcontm = 15;
                ajust_fade -= 960;
        } // ent _butcontm if
        } // end !Fade_M if
        // === Fim da rotina do botão com incremento Fade ===

        // === Teste do pressionar de botão de entrada e saída dos menus ===
        if (!ENTER)                 // Botão enter pressionado? Sim...
        {
        GO = 0x01;                  // Seta a flag de entrada do menu

        if (entTim)         Timer_M = 0x01;         // Verifica se o programa está na função timer
        else                Timer_M = 0x00;         // caso esteja e enter seja pressionado seta a flag
Timer_M

        if (_fFO)
        {
                Flash_M = 0x01; // Verifica se o programa está na função flash
                statusF = 0x01;
        }
        else                Flash_M = 0x00; // caso esteja e enter seja pressionado seta a flag Flash_M

        if (_fFS && estado == 0)            estado = 1;

        if (Flash_M && _fFO && comparadora > 0)
        {
        auxflash = comparadora;
        hab_f = 1;
        }

        } // end if ENTER

        else if (!BACK)                     // Botão se saida pressionado? Sim...
        {
        GO = 0x00;                  // Limpa a flag de entrada do menu
        _fFS = 0x00;
        ajust_tmr1 = 32000;                 // Garante a lampada apagada no menu
        } // end if BACK

        // === Teste do pressionar de botões de ON e OFF ===
        else if (!ON)
        {
        ajust_tmr1 = Lig;
        GO = 0x00;
        }
```

```
        else if (!OFF)
        {
                ajust_tmr1 = Desl;
                GO = 0x00;
        }

        // === Do efeito fade 100ms ===
        if (estado == 2)
        {
                atualiza = 0x01;
                contseg++;
        }


        // === Base de tempo 1 segundo ===
        if (_1seg == 10)            // Testa se já passaram 1 segundo
        {
        _1seg = 0x00;               // Zera o valor da contagem

        if (Timer_M)
        {
                ClearLCD = 0x01;
                contseg--;
                if (contseg == 0x00)
                {
                Timer_M = 0x00;
                ClearLCD = 0x00;
                }
        }
        if (hab_f )
        {
                auxflash--;
                if (auxflash == 0)
                {
                auxflash = comparadora;
                changeSt = !changeSt;
                }
        }
        } // end _1seg if
        } // end base 100 ms

        TMR0 = 0x3C;                    // Carrega o timer0 com o valor 60 novamente
        TMR0IF_bit = 0x00;              // Limpa a flag do timer0 após o fim do processamento da
interrupção
} // end TMR0IF if


} // end interrupt

// ##############################################################################
// --- Declaração de Funções Auxiliares ---
void registradores ();              // Função que configurará os regitradores
void testabotoes ();                // Função que testa os botoes
void controlight ();                // Função que controla a luminosidade
void fade ();                       // Função responsável pelo fade
void flash ();                      // Função responsável pelo flash
void timer ();                      // Função responsável pelo timer
```

```
void calcDisplay ();
void ldr ();                    // Função responsável pelo
                                // tratamento do ldr (Extra!!)

// ###########################################################################
// --- Função Principal ---
void main()
{
        GO      = 0x00;                 // Inicializa todas as flags limpas...
        inc     = 0x00;
        dec     = 0x00;
        ClearLCD = 0x00;
        Func_up  = 0x00;
        Func_dw  = 0x00;
        Timer_M  = 0x00;
        entTim   = 0x00;
        Fade_M   = 0x00;
        Flash_M  = 0x00;

        registradores ();               // Faz chamada da função que configura os registradores

        Lcd_Init();                     // Inicia o display LCD
        Lcd_Cmd (_LCD_CLEAR);                   // Limpa o display LCD
        Lcd_Cmd(_LCD_CURSOR_OFF);               // Desliga o cursor do display LCD
        Lcd_Out (1, 6,"DIMMER");                // Imprime mensagem de inicialização no display LCD
        Lcd_Out (2, 2,"MICROCONTROLADO");               // (Mensagem generica)

        delay_ms (1000);                // Aguarda 1 segundo com a mensagem na tela
        Lcd_Cmd (_LCD_CLEAR);

        while (1)
        {
        testabotoes();          // Chama a função que testa os botões de incremento e decremento

        switch (sel)            // Entra no menu de funcionalidades do dimmer
        {
        case 0x01:              // Menu Controle

        if (ClearLCD)
        {
        LCD_Cmd (_LCD_CLEAR);
        ClearLCD = 0x00;
        } // end if Clear LCD

        Lcd_Chr (1,1, '<');
        Lcd_Chr (1,16, '>');
        Lcd_Chr (1,3, 'C');
        Lcd_Chr_Cp ('O');
        Lcd_Chr_Cp ('N');
        Lcd_Chr_Cp ('T');
        Lcd_Chr_Cp ('R');
        Lcd_Chr_Cp ('O');
        Lcd_Chr_Cp ('L');
        Lcd_Chr_Cp ('E');
        Lcd_Chr (1,11, ' ');
        Lcd_Out (2,1, "         ");
```

```
_fIF = 0x00;
controlight ();
break;

case 0x02:                      // Menu Fade

if (ClearLCD)
{
LCD_Cmd (_LCD_CLEAR);
ClearLCD = 0x00;
} // end if Clear LCD

Lcd_Chr (1,1, '<');
Lcd_Chr (1,16, '>');
Lcd_Chr (1,3, 'F');
Lcd_Chr_Cp ('A');
Lcd_Chr_Cp ('D');
Lcd_Chr_Cp ('E');
Lcd_Chr (1,7, ' ');
Lcd_Out (2,1, "                 ");


estado = 0x00;                  // Zera a variável de estado para o modo fade
Fade_M          = 0x00;                 // Talvez eu adicione uma flag
_fIF      = 0x01;               // para atualizar os flags
_fFS      = 0x00;
ajust_fade          = 0x00;
atualiza            = 0x00;

contseg             = 0x00;

fade ();
break;

case 0x03:                      // Menu Flash

if (ClearLCD)
{
LCD_Cmd (_LCD_CLEAR);
ClearLCD = 0x00;
} // end if Clear LCD

Lcd_Chr (1,1, '<');
Lcd_Chr (1,16, '>');
Lcd_Chr (1,3, 'F');
Lcd_Chr_Cp ('L');
Lcd_Chr_Cp ('A');
Lcd_Chr_Cp ('S');
Lcd_Chr_Cp ('H');
Lcd_Out (2,1, "                 ");


hab_f   = 0;
Flash_M = 0;                    // Flag de indicação de modo
_fFO      = 0;                  // Flag que indica que o botão foi pressionado ao menos uma vez
flash ();                       // Faz a chamada da função flash
break;
```

```
        case 0x04:                  // Menu Timer

        if (ClearLCD)
        {
        LCD_Cmd (_LCD_CLEAR);
        ClearLCD = 0x00;
        } // end if Clear LCD

        Lcd_Chr (1,1, '<');
        Lcd_Chr (1,16, '>');
        Lcd_Chr (1,3, 'T');
        Lcd_Chr_Cp ('I');
        Lcd_Chr_Cp ('M');
        Lcd_Chr_Cp ('E');
        Lcd_Chr_Cp ('R');
        Lcd_Chr_Cp (' ');
        Lcd_Out (2,1, "             ");

        contseg = 0x00;             // Garante que a contagem sempre inicia em zero
        entTim  = 0x00;             // Garante que a flag entTim esteja limpa ao entrar na função timer
        timer ();
        break;

        case 0x05:                  // Menu Sinc

        if (ClearLCD)
        {
        LCD_Cmd (_LCD_CLEAR);
        ClearLCD = 0x00;
        } // end if Clear LCD

        Lcd_Chr (1,1, '<');
        Lcd_Chr (1,16, '>');
        Lcd_Chr (1,3, 'L');
        Lcd_Chr_Cp ('D');
        Lcd_Chr_Cp ('R');
        Lcd_Chr_Cp (' ');
        Lcd_Chr_Cp (' ');
        Lcd_Out (2,1, "             ");

        ldr ();
        break;

        } // end switch case sel

        } // end while
  } // end main

// ###############################################################################
// --- Desenvolvimento de Funções Auxiliares ---
void registradores ()
{

// --- Configuração TMR0 e External Interrupt ---
        INTCON = 0xF0;              // 1111 0000
        // Habilita as interrupções globais para configuração (INTCON.F7)
```

```
        // Habilita as interrupções por periféricos (INTCON.F6)
        // Habilita interrupção do timer 0 (INTCON.F5)
        // Habilita interrupção externa no pino RB0 (INTCON.F4)
        // Desabilita interrução por mudança do PORTB (INTCON.F3)
        // Limpa a flag de interrupção do Timer0 (INTCON.F2)
        // Limpa a flag de interrupção externa (INTCON.F1)
        // Limpa a flag de mudança do PORTB (INTCON.F0)

        OPTION_REG = 0x87; // 1100 0111 | 1000 0111 (0x87 - Testar se é melhor com borda de subida ou
descida)
        // Desabilita os PULLUPS do PORTB (OPTION_REG.F7)
        // Habilita interrupção Externa por borda de descida (OPTION_REG.F6)
        // Associa o clock do timer0 ao ciclo de máquina (OPTION_REG.F5)
        // Associa o preescaler ao timer0 (OPTION_REG.F3)
        // Configura o preescaler em 1:256 (OPTION_REG <F2:F0>)

// --- Configuração TMR1 ---
        T1CON = 0x00;  // 0000 0000
        // Não são implementados (T1CON <F7:F6>)
        // Habilita o prescale em 1:1 (T1CON <F5:F4>)
        // Desabilita o oscilador do TMR1 (T1CON.F3)
        // Bit de controle de sincronia do TMR1, don't care pois T1CON.F1 é setado (T1CON.F2)
        // Config. o incremento do TMR pelo ciclo de máquina (T1CON.F1)
        // Inicia o TMR1 desligado, deve ser ligado após interrupção externa (T1CON.F0)

        PIE1 = PIE1 | 0x01;// 0000 0001
        // Habilita a interrupção do TMR1 por overflow

        PIR1 = PIR1 & 0xFE;// 1111 1110
        // Limpa a flag de overflow do TMR1 (garantia!)

        TMR1H = 0x7D;           // Configura o TMR1 para contagem do periodo desligado
        TMR1L = 0x00;

// --- Configuração dos demais periféricos ----
        CMCON = 0x07;           // Desabilita os comparadores
        CVREN_bit = 0;
        CVROE_bit = 0;

        TMR0 = 0x3C;   // Carrega o timer0 com o valor 60 inicialmente

        ADON_bit = 0x00;  // Desabilita o modulo de conversão AD
        ADCON1 = 0x0E;          // Configura os pinos do PORTA como digitais

        TRISA = 0b11111101;
        TRISB = 0b11111101;
        TRISC = 0xF0;   // 1111 0000


        PORTA = 0x00;  // Inicia o PORTB em LOW
        PORTB = 0xFF;  // Inicia o PORTB em HIGH
        PORTC = 0x00;  // Inicia o PORTC em LOW

} // end registradores

void testabotoes ()
{
```

```
        if (!butinc)        inc = 0x01;         // Se botão de mais pressionado, seta a flag inc
        if (!butdec)        dec = 0x01;         // Se botão de menos pressionado, seta a flag dec

        if (butinc && inc)          // Botão+ solto e flag inc setada? Sim...
        {
        inc = 0x00;                 // Limpa a flag
        sel++;              // Incrementa a seleção de menus
        ClearLCD = 0x01;
        } // end but+ && inc

        if (butdec && dec)                  // Botão- solto e flag dec setada? Sim...
        {
        dec = 0x00;                 // Limpa a flag
        sel--;              // Decrementa a seleção de menus
        ClearLCD = 0x01;
        } // end if but- && dec

        if (sel > nmenus)           sel = 0x01;         // Cria o efeito de menu ciclico...
        if (sel < 0x01)     sel = nmenus;

} // end testabotoes

void controlight ()
/*
  1. Controle de luminosidade: A luminosidade deverá aumentar ou diminuir por
  meio do acionamento de dois botões (+ e -). Deve ser possível escolher no
  mínimo entre 5 níveis diferentes de luminosidade, que deverão ser indicados
  no display e visíveis pela luminosidade da lâmpada. Deve existir, também,
  um botão ON e um botão OFF, que ligam (100% de luminosidade) ou
  desligam (0% de luminosidade) instantaneamente a lâmpada.
*/
{

        while (GO)
        {

        Lcd_Chr (1,1, ' ');
        Lcd_Chr (1,16, ' ');
        Lcd_Chr (1,11,':');

        if (!butinc)        Func_up = 0x01;  // Se botão de mais pressionado, seta a flag Func_up
        if (!butdec)        Func_dw = 0x01;         // Se botão de menos pressionado, seta a flag Func_dw

        if (butinc && Func_up)              // Botão+ solto e flag inc setada? Sim...
        {
        Func_up = 0x00;             // Limpa a flag
        _butcontp = 0x00;           // Limpa a variavel de contagem do incremento inteligente
        _butcontm = 0x00;           // Redundância para segurança do código
        ajust_tmr1 += 302;          // Incrementa o valor de ajuste
        }

        if (butdec && Func_dw)              // Botão- solto e flag Func_dw setada? Sim...
        {
        Func_dw = 0x00;             // Limpa a flag
        _butcontm = 0x00;           // Limpa a variavel de contagem do incremento inteligente
        _butcontp = 0x00;           // Redundância para segurança do código
        ajust_tmr1 -= 302;          // Decrementa a variável de ajuste
```

```
        }

        if (ajust_tmr1 > Lig)       ajust_tmr1 = 62200;
        if (ajust_tmr1 < Desl)      ajust_tmr1 = 32000;

        calcDisplay ();
        } // end while
} // end controlight

void fade ()
/*
  2. Efeito fade: Este modo, que deve ser escolhido pelo usuário por meio de
  um botão e indicado que está ativo através de um led (ou via display), deve
  apagar ou acender (até o nível de luminosidade já programado) de forma
  gradual e ininterrupta.
*/
{
  while (GO)
  {
        //contseg = 20;

        switch (estado)
        {
        case 0x00:


                Lcd_Chr (1,3, 'F');
                Lcd_Chr_Cp ('a');
                Lcd_Chr_Cp ('d');
                Lcd_Chr_Cp ('e');
                Lcd_Chr (1,1, ' ');
                Lcd_Chr (1,16, ' ');
                Lcd_Chr (1, 7,':');
                //Lcd_Out (2,1, "               ");
                Lcd_Chr (2,1, ' ');
                Lcd_Chr (2,2, ' ');
                Lcd_Chr (2,3, ' ');
                Lcd_Chr (2,8, ' ');
                Lcd_Chr (2,9, ' ');

                if (!butinc)        Func_up = 0x01; // Se botão de mais pressionado, seta a flag Func_up
                if (!butdec)        Func_dw = 0x01;         // Se botão de menos pressionado, seta a flag
Func_dw

                if (butinc && Func_up)              // Botão+ solto e flag inc setada? Sim...
                {
                Func_up = 0x00;             // Limpa a flag
                _butcontp = 0x00;           // Limpa a variavel de contagem do incremento inteligente
                _butcontm = 0x00;           // Redundancia para segurança do código
                _fFS      = 0x01;           // Garante o inicio com um pressionar de botão ao menos
                ajust_fade -= 3020;         // Incrementa o valor de ajuste do fade
                }

                if (butdec && Func_dw)              // Botão- solto e flag Func_dw setada? Sim...
                {
                Func_dw = 0x00;             // Limpa a flag
                _butcontm = 0x00;           // Limpa a variavel de contagem do incremento inteligente
```

```
            _butcontp = 0x00;           // Redundancia para segurança do código
            _fFS      = 0x01;
            ajust_fade += 3020;         // Decrementa a variável de ajuste do fade
            }

            if (ajust_fade > Lig)       ajust_fade = Lig;
            if (ajust_fade < Desl)      ajust_fade = Desl;
            break;

case 0x01:                  // Neste caso o botão de enter foi pressionado

            Lcd_Chr (1, 1, 'A');
            Lcd_Chr_Cp ('C');
            Lcd_Chr_Cp ('E');
            Lcd_Chr_Cp ('N');
            Lcd_Chr_Cp ('D');
            Lcd_Chr_Cp ('E');
            Lcd_Chr_Cp ('R');
            Lcd_Chr_Cp (':');
            Lcd_Chr_Cp ('B');
            Lcd_Chr_Cp ('+');

            Lcd_Chr (2, 1, 'A');
            Lcd_Chr_Cp ('P');
            Lcd_Chr_Cp ('A');
            Lcd_Chr_Cp ('G');
            Lcd_Chr_Cp ('A');
            Lcd_Chr_Cp ('R');
            Lcd_Chr_Cp (':');
            Lcd_Chr_Cp ('B');
            Lcd_Chr_Cp ('-');

            if (!butinc)        inc = 0x01;
            if (!butdec)        dec = 0x01;

            if (butinc && inc)
            {
            inc = 0x00;
            ModoFade = 0x01;
            estado = 0x02;
            step = (Desl - ajust_fade)/Vel;
            //Lcd_Cmd (_LCD_CLEAR);
            Lcd_Chr (1,3, 'F');
            Lcd_Chr_Cp ('a');
            Lcd_Chr_Cp ('d');
            Lcd_Chr_Cp ('e');
            Lcd_Chr (1,7, ':');
            }

            if (butdec && dec)
            {
            dec = 0x00;
            ModoFade = 0x00;
            estado = 0x02;
            step = (Desl + ajust_fade)/Vel;
            Lcd_Chr (1,3, 'F');
            Lcd_Chr_Cp ('a');
```

```
                    Lcd_Chr_Cp ('d');
                    Lcd_Chr_Cp ('e');
                    Lcd_Chr (1,7, ':');
                    }
                    break;

          case 0x02:

                    if (ModoFade == 1)
                    {
                    if (atualiza)
                    {
                    ajust_tmr1 = Desl + (step*contseg);          //Qual a coondição que limita ajust_tmr1? Ajust_fade deveria ser usado aqui?
                    atualiza = 0;
                    if (contseg == 1)
                    {
                    Lcd_Cmd (_LCD_CLEAR);
                    Lcd_Chr (1,3, 'F');
                    Lcd_Chr_Cp ('a');
                    Lcd_Chr_Cp ('d');
                    Lcd_Chr_Cp ('e');
                    Lcd_Chr_Cp (':');
                    }
                    if (contseg == Vel)
                    {
                    estado = 0;
                    contseg = 0;
                    Lcd_Cmd (_LCD_CLEAR);
                    }
                    } // end atualiza
                    } // end if ModoFade

                    if (ModoFade == 0)
                    {
                    if (atualiza)
                    {
                    ajust_tmr1 = Desl - (step*contseg);          //Qual a coondição que limita ajust_tmr1? Ajust_fade deveria ser usado aqui?
                    atualiza = 0;
                    if (contseg == 1)
                    {
                    Lcd_Cmd (_LCD_CLEAR);
                    Lcd_Chr (1,3, 'F');
                    Lcd_Chr_Cp ('a');
                    Lcd_Chr_Cp ('d');
                    Lcd_Chr_Cp ('e');
                    Lcd_Chr_Cp (':');
                    }

                    if (contseg == Vel)
                    {
                    estado = 0;
                    contseg = 0;
                    Lcd_Cmd (_LCD_CLEAR);
                    }
                    } // end atualiza
```

```
                } // end else



                calcDisplay();
                break;

        } // end switch estado

        if (estado != 1)   calcDisplay ();


   }// end while
} // end fade

void flash ()
/*
  3. Efeito flash: No efeito flash, quando acionado, a lâmpada deve piscar
  ininterruptamente com um tempo selecionado. Deve ser considerada uma faixa
  selecionável entre 1 e 5 segundos.
*/
{
        while (GO)
        {
        Flash_M = 0x01;

        Lcd_Chr (1,1, ' ');
        Lcd_Chr (1,16, ' ');
        Lcd_Chr (1, 8, ':');

        if (!butinc)   flinc = 0x01;
        if (!butdec)   fldec = 0x01;

        if (butinc && flinc)                    // Botão+ solto e flag flinc setada? Sim...
        {
        flinc = 0x00;
        _fFO = 0x01;
        comparadora++;
        if (comparadora >= 5)                   comparadora = 5;
        }

        if (butdec && fldec)                    // Botão- solto e flag fldec setada? Sim...
        {
        fldec = 0x00;
        _fFO = 0x01;
        comparadora--;
        if (comparadora <= 0)                   comparadora = 0;
        }


        if (changeSt)       ajust_tmr1 = Lig;
        else if (!changeSt) ajust_tmr1 = Desl;

        calcDisplay();

        } // end while
} // end flash
```

```
void timer ()
/*
  4. Modo timer: O usuário deve inserir o tempo, em segundos, via botões em que
        a lâmpada deverá ficar ligada. Deve ser considerada uma faixa selecionável
        entre 5 e 60 segundos, após esse tempo a mesma se apaga.
*/
{
        while (GO)
        {


        Lcd_Chr (1,1, ' ');
        Lcd_Chr (1,16, ' ');
        Lcd_Chr (1, 8, ':');
        Lcd_Chr (2, 7, 'S');
        Lcd_Chr_Cp ('e');
        Lcd_Chr_Cp ('g');

        if (!butinc)        _fTM = 0x01;   // Botão pressionado? Seta a flag

        if (!butdec)        _fTm = 0x01;   // Botão pressionado? Seta a flag


        if (butinc && _fTM && !Timer_M)          // Botão solto e flag setada?
        {
        _fTM = 0x00;                // Limpa a flag
        contseg+= 5;                // Incrementa o tempo de contagem
        entTim  = 0x01;
        //_StartTimer = 0x00;
        }

        if (butdec && _fTm)         // Botão solto e flag setada?
        {
        _fTm = 0x00;                // Limpa a flag
        contseg--;                  // Decrementa o tempo de contagem
        entTim = 0x01;
        //_StartTimer = 0x00;
        }

        if (contseg < 0x00)         contseg = 0x05; // Prende o valor
        else if (contseg > 0x3C)    contseg = 0x3C; // de contagem
                                              // no intervalo
                                              // [5,60]

        calcDisplay();

        if (ClearLCD)               ajust_tmr1 = 59000;         //65530; // Inverti os valores
        else                        ajust_tmr1 = 0x7D00;

        //if (debug) Lcd_Chr (2, 10, 'E');
        //else Lcd_Chr (2, 10, ' ');

        } // end while
        Timer_M = 0x00;
} // end timer
```

```
void ldr ()
{
 regula = 0;
 char cont = 0, tin = 0;
 unsigned int valor;

        while (GO)
        {

        Lcd_Chr (1,1, ' ');
        Lcd_Chr (1,16, ' ');
        Lcd_Chr (1, 7, ':');

        if (atualiza)
        {
        tin++;
        LDR_Read = Adc_Read(0);


        cont++
        if (cont = 2)
        {
        cont = 0;
        valor = (LDR_Read + antLe)/2;

        }

        antLe = LDR_Read;

        if (tin == 50)
        {
        regula = 1;
        }

        if (regula && )
        calcDisplay();
        }

        } // end while
} // end ldr

void calcDisplay()
{
        char _unidade,
        _dezena,
        _centena,
        _milhar;
        unsigned Cont0_100;

        switch (sel)
        {
        case 0x01:
                Cont0_100 = (ajust_tmr1 - Desl)/302;

                _unidade = ((Cont0_100/1)%10);
                _dezena  = ((Cont0_100/10)%10);
                _centena = ((Cont0_100/100)%10);
```

```
                Lcd_Chr (2,4, (_centena + 0x30));
                Lcd_Chr_Cp ((_dezena + 0x30));
                Lcd_Chr_Cp ((_unidade + 0x30));
                break;


        case 0x02:
                Cont0_100 = 100 - ((ajust_fade - Desl)/302);

                _unidade = ((Cont0_100/1)%10);
                _dezena  = ((Cont0_100/10)%10);
                _centena = ((Cont0_100/100)%10);

                Lcd_Chr (2,4, (_centena + 0x30));
                Lcd_Chr_Cp ((_dezena + 0x30));
                Lcd_Chr_Cp ((_unidade + 0x30));
                break;

        case 0x03:
                _unidade = ((comparadora/1)%10);
                Lcd_Chr (2, 6,(_unidade+0x30));
                break;

        case 0x04:

        _unidade = ((contseg/1)%10);
        _dezena = ((contseg/10)%10);

        Lcd_Chr (2,4, (_dezena + 0x30));
        Lcd_Chr_Cp ((_unidade + 0x30));
        break;

        case 0x05:
                _unidade = ((sinc_/1)%10);
                _dezena  = ((sinc_/10)%10);
                _centena = ((sinc_/100)%10);
                _milhar  = ((sinc_/1000)%10);

                Lcd_Chr (2,4, (_milhar + 0x30));
                Lcd_Chr_Cp (_centena + 30);
                Lcd_Chr_Cp (_dezena + 30);
                Lcd_Chr_Cp (_unidade + 30);

        } // end switch sel
} // end calcDisplay
```

7. REFERÊNCIAS

Thomas Keith Hemingway. Electronic Designers Handbook. Bussiness Books (1970). Disponível em <https://archive.org/details/ElectronicDesignersHandbook/page/n1>. Último acesso em 04/12/2019.

Microchip. Datasheet PIC 16f876A. Disponível em <https://ww1.microchip.com/downloads/en/devicedoc/39582b.pdf>.