Proposta de Projeto

Dispositivo emissor de luz de LED para neoformação tecidual

Angélica Kathariny de Oliveira Alves
Graduanda em Engenharia Eletrônica
Universidade de Brasília - UnB
Brasília, Brasil
Angel.kyka@hotmail.com

Mayara Barbosa dos Santos
Graduanda em Engenharia Eletrônica
Universidade de Brasília - UnB
Brasília, Brasil
Mayara.b97@gmail.com

***Resumo*—Pacientes acometidos por Diabetes Mellitus apresentam dificuldades na cicatrização devido a deficiência de irrigação sanguínea que causa irregularidades na neoformação tecidual. Quando localizada nos pés as ulcerações podem evoluir e levar a amputação do membro. O uso de fototerapia com LED contribui para acelerar o processo de neoformação tecidual. Neste projeto é proposto um dispositivo emissor de luz de LED para o tratamento fototerápico controlado por intermédio do microcontrolador MSP430.**

Palavras-chave—MSP430; fototerapia; diabetes;

I. JUSTIFICATIVA

Uma pesquisa apresentada pela Organização Pan-Americana da Saúde/ Organização Mundial da Saúde (OPAS/OMS), publicada em 2016, mostrou o avanço mundial da diabetes nos últimos 24 anos. O estudo estima que até o ano de 2014 cerca de 8,5% da população é acometido por esta doença.

O Diabete Mellitus é uma síndrome metabólica caracterizada por taxas elevadas de açúcar no sangue [1]. Uma das complicações decorrentes dessa síndrome é chamada de pé diabético. Este termo refere-se a feridas que acometem os pés do indivíduo portador da síndrome decorrentes da deficiência de irrigação sanguínea do membro. A evolução dessas feridas pode acarretar na amputação do membro afetado.

Um estudo realizado em 2013 aponta a terapia luminosa com LEDs como forma eficaz de tratamento de feridas [2]. Nesse estudo foi comprovado que a irradiação luminosa atuou na aceleração do processo cicatricial e na reepitelização da área ferida.

II. OBJETIVO

Este projeto tem o objetivo de desenvolver um dispositivo eletrônico para fototerapia utilizando LEDs cujas cores serão controladas por intermédio do microcontrolador MSP430.

III. REQUISITOS

O dispositivo proposto neste projeto deve atender aos seguintes requisitos:

- O tempo que os LEDs ficarão ligados será definido de acordo com estudo da terapia luminosa.
- Após o término do tempo indicado o dispositivo deverá zerar o display, emitir um sinal sonoro e esperar para a reativação do sistema.
- Indicar um sinal luminoso se a bateria está com o nível de carregamento baixo ou alto.

IV. BENEFÍCIOS

O projeto apresenta como benefícios a temporização do uso do dispositivo sem a utilização de dispositivos adicionais, como cronômetros e relógios; seleção da cor do LED a ser utilizada bem como o uso do LED para cicatrizar feridas de diabéticos.

V. DESENVOLVIMENTO DO PROTÓTIPO

A. Módulos

O dispositivo proposto nesse projeto apresenta dois módulos. O primeiro módulo é composto por uma matriz de LEDs e os componentes necessários para seu funcionamento como resistores e diodos. Já o segundo módulo tem como componente principal o Microcontrolador MSP430 e é responsável pelo controle do primeiro módulo.

A fig. 1 mostra o layout que será usado como base da placa que abrigará a matriz de LEDs. Como mostrado na figura, o protótipo contará com fileiras de LEDs vermelhos. Com esta disposição é possível acionar as cores ao mesmo tempo ou quantas vezes forem necessárias. A fig 2 mostra o layout proposto para a placa de controle que contém o microcontrolador MSP430.

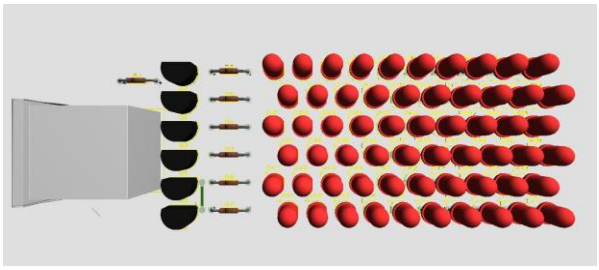

Fig. 1. Módulo 1: matriz de LEDs. *(Dos autores)*

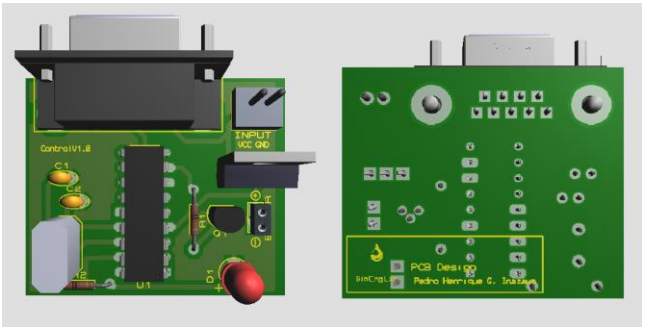

Fig. 2. Módulo 2: controlador. *(Dos autores)*

B. Componentes

- LEDs
- Microcontrolador MSP430
- Resistores
- Reguladores de tensão
- Display 8 dígitos de 7 segmentos
- Circuito impresso
- Pilha AA 1,5V
- Case para pilha
- Botão para ligar e desligar
- Botão para realizar o exame

VI. CÓDIGO

```
#include <msp430g2553.h>
#include <legacymsp430.h>
#include <stdio.h>

#define CS BIT0  //2.0 is CS

#define MOSI BIT7  //1.7 is SPI MOSI

#define SCLK BIT5  //1.5 is SPI clock

#define EXA BIT1  //1.3 is butao que habilita exame do equipamento

#define BTN (EXA)  //define exa como btn

#define LEDS BIT6  //Leds medicinais

#define STDY BIT3  //Led de standby aguardando exame

#define OP BIT4  //Leds realizando exame

#define MODES (BIT3|BIT4)  //Grupo dos leds de modos de operacao STDY e OP (para inicilaizar os dois de uma vez)

//*****************************************
*****************************************
**
//Declaração das funcoes
void Init_MAX7219(void);

void SPI_Init(void); //SPI initialization

void SPI_Write(unsigned char);

void SPI_Write2(unsigned char, unsigned char);

void SPI_Write3(unsigned char, unsigned char, unsigned char);

int chamar(void);

unsigned int getVcc3();
//*****************************************
*****************************************
**

//*****************************************
*****************************************
**
//Variaveis
long int cont, cont1; //contadores
int bateria();       //mostrar bateria
int segundos=0;          //controla a quantidade de segundos dezena
int minutos1=0;          //controla a quantidade de minutos unidade
int minutos2=0;          //controla a quantidade de minutos dezena
//*****************************************
*****************************************
**

int main(void)
{
  //Desabilita o watchdog
```

```
  WDTCTL = WDTPW + WDTHOLD;
  //CONFIGURACAO DOS LEDS
  P1OUT = 0;                //desligas as portas de P1
  P1DIR = LEDS;             //Define Leds como saidas

  //configura botao com resistor de pullup(P1REN abilitado e P1OUT em 1) **para pulldown(P1REN abilitado e P1OUT em 0)

  P1DIR &= ~EXA;  //Define o botao EXA como entrada
  P1REN |= EXA;   //Define como binario
  P1OUT |= EXA;   //

  P1DIR &= ~MODES;
  P1REN |= MODES;
  P1OUT |= MODES;

  SPI_Init(); //Configuracao da comunicacao serial


  __delay_cycles(500);      //Pausa de 500 ciclos de clock

  Init_MAX7219();           //cofigura o chip max7219 do display

  __delay_cycles(500);

  //SPI_Write2("Digito","valor") como escrever na funcao SPI_Write2

  SPI_Write2(0x01,0x0);             //Escreve o valor "zero", no digito 1 do display
  SPI_Write2(0x02,0x0);             //Escreve o valor "zero", no digito 2 do display
  SPI_Write2(0x03,0x0);             //Escreve o valor "zero", no digito 3 do display
  SPI_Write2(0x04,0x0);             //Escreve o valor "zero", no digito 4 do display
  SPI_Write2(0x05,0x0);             //Escreve o valor "zero", no digito 5 do display
  SPI_Write2(0x06,0x0);             //Escreve o valor "zero", no digito 6 do display
  SPI_Write2(0x07,0x0);             //Escreve o valor "zero", no digito 7 do display
  SPI_Write2(0x08,0x0);             //Escreve o valor "zero", no digito 8 do display


  //habilita interrupcao no botao
  //P1IES |= BTN;
  //P1IE  |= BTN;
 // _BIS_SR(LPM3_bits+GIE);
  //Clock definido para 16MHz
  BCSCTL1 = CALBC1_16MHZ;
  DCOCTL = CALDCO_16MHZ;

   P1OUT &= ~OP;      //Desliga LED MODE OPERATING
   P1OUT |= (STDY);   //Liga LED STANDBY

   while(1){
   chamar();
   //break;
   }


 return 0;
}

int chamar(void)
{


  if ((P1IN & EXA) == 0){
          //testa se o botao de exame esta apertado
  for(cont=0;cont<=920010;cont++){
                  //92000->1s
   P1OUT |= LEDS;
                  //Liga LEDS para tratamento
   P1OUT |= OP;
                  //Liga LED MODE OPERATING
   P1OUT &= ~(STDY);
          //Desliga LED STANDBY

   while((P1IN & EXA) != 0){
                  //se o botao de exame for liberado volta-se para as condicoes iniciais
    P1OUT &= ~OP;
                  //Desliga LED operadno
    P1OUT |= (STDY);
          //Liga LED standby
    P1OUT &= ~(LEDS);
                  //Desliga LEDS para tratamento
    SPI_Write2(0x01,0x0);
    SPI_Write2(0x02,0x0);
    SPI_Write2(0x03,0x0);
    SPI_Write2(0x04,0x0);
    SPI_Write2(0x05,0x0);
    SPI_Write2(0x06,0x0);
    SPI_Write2(0x07,0x0);
    SPI_Write2(0x08,0x0);
    cont=0;
                  //Zera o contador
    segundos=0;
          //Zerar o tempo
    minutos1=0;
    minutos2=0;
   }
```

```
   if(cont>=0 & cont<18000){
                    //intervalor de 0s -> 1s
//     P1OUT |= LED0;
       SPI_Write2(0x01,0x0);
                    //Mostra o valor 0 nas unidades
dos segundos
       SPI_Write2(0x02,segundos);
            //mostra o valro das dezenas de segundos
   }

   if(cont>=18000 & cont<36000){
            //intervalor de 1s -> 2s
//     P1OUT |= LED1;
       SPI_Write2(0x01,0x1);
//       SPI_Write2(0x02,segundos);
   }

   if(cont>=36000 & cont<54000){
            //intervalor de 2s -> 3s
//     P1OUT |= LED2;
       SPI_Write2(0x01,0x2);
//       SPI_Write2(0x02,segundos);
   }

   if(cont>=54000 & cont<72000){
            //intervalor de 3s -> 4s
//     P1OUT |= LED3;
       SPI_Write2(0x01,0x3);
//       SPI_Write2(0x02,segundos);
   }

   if(cont>=72000 & cont<90000){
            //intervalor de 4s -> 5s
//     P1OUT |= LED4;
       SPI_Write2(0x01,0x4);
//       SPI_Write2(0x02,segundos);
   }

   if(cont>=90000 & cont<108000){
                    //intervalor de 5s -> 6s
//     P1OUT |= LED5;
       SPI_Write2(0x01,0x5);
//       SPI_Write2(0x02,segundos);
   }

   if(cont>=108000 & cont<126000){
                    //intervalor de 6s -> 7s
//     P1OUT |= LED6;
       SPI_Write2(0x01,0x6);
//       SPI_Write2(0x02,segundos);
   }

   if(cont>=126000 & cont<144000){
                    //intervalor de 7s -> 8s
//     P1OUT |= LED7;
       SPI_Write2(0x01,0x7);
//       SPI_Write2(0x02,segundos);
   }

   if(cont>=144000 & cont<162000){
                    //intervalor de 8s -> 9s
//     P1OUT |= LED8;
       SPI_Write2(0x01,0x8);
//       SPI_Write2(0x02,segundos);
   }

   if(cont>=162000 & cont<180000){
                    //intervalor de 9s -> 10s
//     P1OUT |= LED9;
       SPI_Write2(0x01,0x9);
//       SPI_Write2(0x02,segundos);
   }

   if(cont>=180000){
                    //zerar o contador para voltar a
contar as unidades dos segundos
     cont=0;

     segundos=segundos+1;
          //incremeta o digito das dezenas de
segundos
     SPI_Write2(0x02,segundos);
          //excreve no display o valor das dezenas
de segundos
     bateria();
          //Mostra o valor atual da bateria em %

     if (segundos==6){
                    //atualiza o relogio para o
primeiro minuto 60s
      segundos=0;
          //zera as dezenas de segundos
      minutos1=minutos1+1;
                    //incremente a unidade dos
minutos
      SPI_Write2(0x03,minutos1);
          //mostra a unidade dos minutos
      //bateria();
     }

     if (minutos1==10){
          //testa se temos 10 minutos (incrementa a
dezena dos minutos)
      minutos1=0;
                    //zera as unidades do minuto
      minutos2=minutos2+1;
                    //incremeta a dezena dos minutos
      SPI_Write2(0x04,minutos2);
          //mostra as dezenas dos minutos no digito
4
     }
     if (minutos2==6){
                    //para zerar a hora
      minutos2=0;
```

```
    }

    if(minutos1==3){
                    //Testa se tem 3 min de exame
      while((P1IN & EXA) == 0){
          //Testa se o botao ainda esta apertado (evita que o exame volte a iniciar sozinho)
        P1OUT &= ~(LEDS);
         //Desliga LEDS para tratamento
        SPI_Write2(0x01,0x0);
         //vamos zerar todo o display
        SPI_Write2(0x02,0x0);
        SPI_Write2(0x03,0x0);
        SPI_Write2(0x04,0x0);
        SPI_Write2(0x05,0x0);
        SPI_Write2(0x06,0x0);
        SPI_Write2(0x07,0x0);
        SPI_Write2(0x08,0x0);
        cont=0;
         //Zerar as condicoes iniciais do tempo
        segundos=0;
        minutos1=0;
        minutos2=0;
      }
    }

   }


  }
  }

  P1IFG = 0;                                    //flag que chama a interrupcao

  return 0;

}

int bateria(void)
{
    float carga;
    int   carga2;

    carga=((getVcc3()/10)-180);
//bateria (está em centivolts para facilitar e mostrar)

    carga=(carga/180*100); //                    //carga em %
    carga2=carga;                               //conversao para int, display so aceita int

    SPI_Write2(0x05,carga2%10);
    SPI_Write2(0x06,(carga2/10)%10);
    SPI_Write2(0x07,(carga2/100)%10);

   return (0);
}

unsigned int getVcc3()
{
  ADC10CTL0 = SREF_1 + REFON + REF2_5V + ADC10ON + ADC10SHT_3;  // use internal ref, turn on 2.5V ref, set samp time = 64 cycles
  ADC10CTL1 = INCH_11;
    //delayMs(1);
//allow internal reference to stabilize
    ADC10CTL0 |= ENC + ADC10SC;
// Enable conversions
    //also verify that sample and hold time is long enough; maybe prescale ADC10OSC.
  while (!(ADC10CTL0 & ADC10IFG));
                        // Conversion done
  unsigned long mv = (ADC10MEM * 5000l);
  return ((unsigned int) (mv / 1024l));
}

void SPI_Init(void)
                                    //SPI initialization (entre a placa e o computador)

{

    P2DIR |= CS;                                //cs is output

    P1SEL |= MOSI + SCLK;
//spi init (estabelece os pinos de comunicação serial)

    P1SEL2 |= MOSI + SCLK;
//spi init (estabelece os pinos de comunicação serial)

    UCB0CTL1 = UCSWRST;

    UCB0CTL0 |= UCMSB + UCMST + UCSYNC + UCCKPH;          // 3-pin, 8-bit SPI master

    UCB0CTL1 |= UCSSEL_2;
// SMCLK

    UCB0BR0 = 10;                               // spi speed is smclk/10 (mostra que o clock da comunicação serial é dividido por 10

    UCB0BR1 = 0;

    UCB0CTL1 &= ~UCSWRST;
// **Initialize USCI state machine**
```

```
    __enable_interrupt();                   //
enable all interrupts

}


void SPI_Write2(unsigned char MSB, unsigned
char LSB)     //SPI write one byte (escreve no
display)

{

     P2OUT                &=                ~CS;
//Habilita a comunicação serial

     __delay_cycles(50);

     UCB0TXBUF = MSB ;                       //
envia na comunicação serial o dígito que será
usado no display

     while     (UCB0STAT     &     UCBUSY);
// Testa se o canal CS ainda está ocupado

     UCB0TXBUF = LSB ;                       //
na comunicação serial o valor que será mostrado
no dígito

     while     (UCB0STAT     &     UCBUSY);
// Testa se o canal CS ainda está ocupado

     P2OUT |= CS;                            //Libera
a comunicação serial

}


void Init_MAX7219(void)

{

     SPI_Write2(0x09, 0xFF);       //decode mode -
converte decimal (0x09, 0x00)

     SPI_Write2(0x0A,  0x0F);            //intensity
control - intensidade maxima

     SPI_Write2(0x0B,  0x06);            //scan limit
original (0x0B, 0x07) display de 8 digitos

     SPI_Write2(0x0C,  0x01);            //shutdown -
operacao normal

     SPI_Write2(0x0F,  0x01);            //display teste
(Liga todos os displays)

     SPI_Write2(0x0F,  0x00);            //display teste
(libera todos os displays para controle)

}
```

REVISÃO BIBLIOGRÁFICA

[1] MILECH, Adolfo. OLIVEIRA, José Egidio Paulo de. VENCIO, Sérgio. Diretrizes da Sociedade Brasileira de Diabetes (2015-2016) . São Paulo, 2016.

[2] REIS. Maria do Carmo dos. Sistema indutor de neoformação tecidual para pé diabético com circuito emissor de luz de LEDs e utilização do látex natural[thesis], Brasília: Universidade de Brasília, 2013.