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.

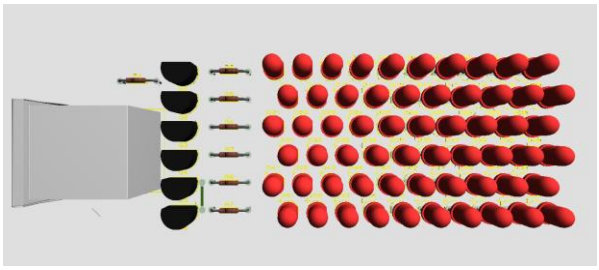

Fig. 1. Módulo 1: matriz de LEDs. *(Dos autores)*

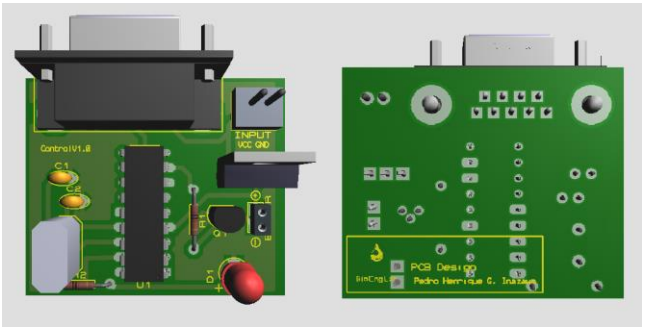

Fig. 2. Módulo 2: controlador. *(Dos autores)*

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

REFERÊNCIA 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

Anexo I – Código

```
//*****************************************************************************************
//Bibliotecas
#include <msp430g2553.h>
#include <legacymsp430.h>
#include <stdio.h>
//*****************************************************************************************

//*****************************************************************************************
//Defeinicao das variaveis
#define CS BIT0  //2.0 is CS do Display

#define MOSI BIT7  //1.7 is SPI MOSI Entrada de dados do display

#define SCLK BIT5  //1.5 is SPI clock do display

#define EXA BIT1  //1.1 is button ON/OFF exame do equipamento

#define LEDS BIT1  //1.6 Leds do produto
#define avaria BIT2  //1.6 Leds do produto

#define comandos (LEDS|avaria)

#define STDY BIT3  //1.3  Led modo standby (verde)

#define OP BIT4  //1.4 Led modo operacional (vermelho)

#define MODES (BIT3|BIT4)  //Agrupamento dos modos (LEDs)
```

```
//*****************************************************************************
//Declaração das funcoes
void Init_MAX7219(void);    //inicia o multiplexador do display

void SPI_Init(void); //Inicia o serial do display

void SPI_Write2(unsigned char, unsigned char);   //Escreve no display

int chamar(void);   //Chama as rotinas

unsigned int getVcc3();  //leitura da bateria

int analog(int sensorPinA0,int sensorPinA6);

void buzzer_inicial();
//*****************************************************************************

//*****************************************************************************
//Variaveis
int cont=3000; //contadores
int bateria();        //mostrar bateria
int segundos1=0;       //controla a quantidade de segundos (unidades)
int segundos2=0;       //controla a quantidade de segundos (dezenas)
int minutos1=0;        //controla a quantidade de minutos (unidades)
int minutos2=0;        //controla a quantidade de minutos (dezenas)
int atraso=0;         //atraso do delay, para acelerar o for
int sensorPin1 = A0;    // select the input pin for the potentiometer
```

```
int sensorPin2 = A6;    // select the input pin for the potentiometer
//int ledPin = 2;      // select the pin for the LED
int sensorValue1 = 0;  //memoria da leitura anterior do sensor A0
int sensorValue2 = 0;  //memoria da leitura anterior do sensor A6
//*******************************************************************************

int main(void)
{

  volatile unsigned int i;
  WDTCTL = WDTPW + WDTHOLD;
  //CONFIGURACAO DOS LEDS


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

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

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

  P2OUT |= 0;
  P2DIR = comandos;
```

```
88  //iniciando a configuração da comunicação serial
89  SPI_Init();
90
91  __delay_cycles(500);
92
93  //iniciando os parametros do display
94  Init_MAX7219();
95
96  __delay_cycles(500);
97
98  //iniciar com todos os displays mostrando zero
99  SPI_Write2(0x01,0x0);
100 SPI_Write2(0x02,0x0);
101 SPI_Write2(0x03,0x0);
102 SPI_Write2(0x04,0x0);
103 SPI_Write2(0x05,0x0);
104 SPI_Write2(0x06,0x0);
105 SPI_Write2(0x07,0x0);
106 SPI_Write2(0x08,0x0);
107
108 //Clock definido para 1MHz
109 BCSCTL1 = CALBC1_1MHZ;
110 DCOCTL = CALDCO_1MHZ;
111
112   P1OUT &= ~OP;   //Desliga LED MODE OPERATING
113   P1OUT |= (STDY);   //Liga LED STANDBY
```

```
114
115     bateria();   //mostrar nivel inicial da bateria
116
117     P2OUT |= LEDS;
118
119     __delay_cycles(10000);
120
121     while(cont>0){
122       analog(sensorPin1,sensorPin2);
123       cont = cont -1;
124     }
125
126
127
128       buzzer_inicial();
129
130       P2OUT = 0;
131
132       while(1){
133
134         chamar();
135
136       }
137
138
139
140   return 0;
```

```
}

int chamar(void)
{


  if ((P1IN & EXA) == 0){
  for(;;){
    P2OUT |= LEDS;   //Liga LEDS para tratamento
    P1OUT |= OP;   //Liga LED MODE OPERATING
    P1OUT &= ~(STDY);   //Desliga LED STANDBY

    if ((P1IN & EXA) == 0){
      analog(sensorPin1,sensorPin2);
    }
    while((P1IN & EXA) != 0){
      P1OUT &= ~OP;   //Desliga LED MODE OPERATING
      P1OUT |= (STDY);   //Liga LED STANDBY
      P2OUT &= ~(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;
```

```
169       segundos1=0;
170       segundos2=0;
171       minutos1=0;
172       minutos2=0;
173     }
174
175     if(atraso==10){
176       segundos1=segundos1+1;        //incremeta o digito de unidade dos segundos
177       if (segundos1==10){
178         segundos1=0;
179         segundos2=segundos2+1;
180         SPI_Write2(0x01,segundos1);
181         if (segundos2<6){
182           SPI_Write2(0x02,segundos2);  //mostra a casa decimal dos segundos
183         }
184       }
185       SPI_Write2(0x01,segundos1);
186       bateria();
187       atraso=0;
188     }
189
190       if (segundos2==6){               //dezena dos segundos, ou seja, 60 segundos
191         segundos1=0;
192         segundos2=0;
193         minutos1=minutos1+1;          //incremeta o digito de unidade dos minutos
194         SPI_Write2(0x02,segundos2);  //mostra a casa decimal dos segundos
195         SPI_Write2(0x03,minutos1);    //mostra a casa unitaria dos minutos
196       }
```

```
197
198       if (minutos1==10){
199         segundos1=0;
200         segundos2=0;
201         minutos1=0;
202         minutos2=minutos2+1;           //incremeta o digito de dezena dos minutos
203         SPI_Write2(0x02,segundos2);  //mostra a casa decimal dos segundos
204         SPI_Write2(0x03,minutos1);    //mostra a casa unitaria dos minutos
205         SPI_Write2(0x04,minutos2);    //mostra a casa decimal dos minutos
206       }
207       if (minutos2==6){
208         segundos1=0;
209         segundos2=0;
210         minutos1=0;
211         minutos2=0;
212         SPI_Write2(0x02,0x00);  //mostra a casa unidade dos segundos
213         SPI_Write2(0x02,segundos2);  //mostra a casa decimal dos segundos
214         SPI_Write2(0x03,minutos1);    //mostra a casa unitaria dos minutos
215         SPI_Write2(0x04,minutos2);    //mostra a casa decimal dos minutos
216       }
217
218       if(minutos1==3){
219          buzzer_inicial();
220          while((P1IN & EXA) == 0){
221           P2OUT &= ~(LEDS);   //Desliga LEDS para tratamento
222           P2OUT &= ~(LEDS);   //Desliga LEDS para tratamento
223           SPI_Write2(0x01,0x0);
```

```
224          SPI_Write2(0x02,0x0);
225          SPI_Write2(0x03,0x0);
226          SPI_Write2(0x04,0x0);
227          SPI_Write2(0x05,0x0);
228          SPI_Write2(0x06,0x0);
229          SPI_Write2(0x07,0x0);
230          SPI_Write2(0x08,0x0);
231          cont=0;
232          segundos1=0;
233          segundos2=0;
234          minutos1=0;
235          minutos2=0;
236         }
237      }
238
239
240     __delay_cycles(98346);  //delay 1s
241     atraso++;
242
243
244  }
245  }
246
247  P1IFG = 0; //flag que chama a interrupcao
248
249  return 0;
```

```
}

int bateria(void)
{
      //getVcc pega tensão em mV, o msp430 travalha entre 1,8 V e 3,6 V.
      //Sendo assim o range de bateria é de 1,8 V, ou seja, 1,8V = 0% e 3,6 V = 100%

      float carga;
      int   carga2;

      carga=((getVcc3())-1800);        //bateria em mV

      carga=(carga/1800*100);          //carga em %
      carga2=carga;                    //conversao para int, display so aceita int

      //o %10 mostra o numero que esta imediatamente apos a virgula, 653,1 (neste caso mostra o 3)

      SPI_Write2(0x05,carga2%10);         //Escrever unidade da porcentadem no digito 5
      SPI_Write2(0x06,(carga2/10)%10);    //Escrever dezena da porcentadem no digito 5
      SPI_Write2(0x07,(carga2/100)%10);   //Escrever centena da porcentadem no digito 5



    return (0);
}
```

```
277 unsigned int getVcc3()
278 {
279   ADC10CTL0 = SREF_1 + REFON + REF2_5V + ADC10ON + ADC10SHT_3;  // use internal ref, turn on 2.5V ref, set samp time = 64 cycles
280   ADC10CTL1 = INCH_11;
281     //delayMs(1);           //allow internal reference to stabilize
282     ADC10CTL0 |= ENC + ADC10SC;             // Enable conversions
283     //also verify that sample and hold time is long enough; maybe prescale ADC10OSC.
284   while (!(ADC10CTL0 & ADC10IFG));               // Conversion done?
285   unsigned long mv = (ADC10MEM * 5000l);
286   return ((unsigned int) (mv / 1024l));
287 }
288
289 void SPI_Init(void) //SPI initialization
290
291 {
292
293     P2DIR |= CS;  //cs is output
294
295     P1SEL |= MOSI + SCLK; //spi init (binario)
296
297     P1SEL2 |= MOSI + SCLK; //spi init
298
299     UCB0CTL1 = UCSWRST;
300
301      UCB0CTL0 |= UCMSB + UCMST + UCSYNC + UCCKPH; // 3-pin, 8-bit SPI master
302
303      UCB0CTL1 |= UCSSEL_2;                     // SMCLK
```

```
305     UCB0BR0 = 10;                           // spi speed is smclk/10
306
307     UCB0BR1 = 0;                               //
308
309     UCB0CTL1 &= ~UCSWRST;                      // **Initialize USCI state machine**
310
311
312
313     __enable_interrupt(); // enable all interrupts
314
315 }
316
317
318
319 void SPI_Write2(unsigned char MSB, unsigned char LSB) //SPI write one byte
320
321 {
322
323
324
325       P2OUT &= ~CS;
326
327       __delay_cycles(50);
328
329       UCB0TXBUF = MSB ;
330
331       while (UCB0STAT & UCBUSY);
```

```
332
333         UCB0TXBUF = LSB ;
334
335         while (UCB0STAT & UCBUSY);
336
337         P2OUT |= CS;
338
339 }
340
341 void Init_MAX7219(void)
342
343 {
344
345         SPI_Write2(0x09, 0xFF);         //decode mode - converte decimal (0x09, 0x00)
346
347         SPI_Write2(0x0A, 0x0F);         //intensity control - intensidade maxima
348
349         SPI_Write2(0x0B, 0x06);         //scan limit original (0x0B, 0x07) digitos do display de 8 digitos
350
351         SPI_Write2(0x0C, 0x01);         //shutdown - operacao normal
352
353         SPI_Write2(0x0F, 0x01);         //display teste
354
355         SPI_Write2(0x0F, 0x00);         //display teste
356
357 }
```

```
363 int analog(int sensorPinA0,int sensorPinA6) {
364
365   int defeito1;
366   int defeito2;
367   int valorA0;
368   int valorA6;
369
370   // read the value from the sensor:
371   __delay_cycles(100);
372   valorA0 = analogRead(sensorPinA0);
373
374   __delay_cycles(100);
375
376   valorA6 = analogRead(sensorPinA6);
377
378   __delay_cycles(100);
379
380   if(sensorValue1==0){                          //testa primeira leitura do sistema
381     sensorValue1 = valorA0;                     //guarda o valor de referencia de sesnor 1
382   }else if( (sensorValue1 - valorA0) > 6 ){     //testa se leitura atual diverge muito da antiga
383     defeito1 = 1;
384   }else{
385     defeito1 = 0;
386   }
```

```
386   }
387
388   if(sensorValue2==0){                          //testa primeira leitura
389     sensorValue2 = valorA6;                     //guarda o valor de referencia de sesnor 2
390   }else if( (sensorValue2 - valorA6) > 6 ){     //testa se leitura atual diverge muito da antiga
391     defeito2 = 1;
392   }else{
393     defeito2 = 0;
394   }
395
396   if( (defeito1==1) or (defeito2==1) ){
397       P2OUT |= avaria;
398   }else{
399       P2OUT = ~avaria;
400   }
401
402
403
404   return (0);
405 }
406
407 void buzzer_inicial(){
408
409   int i=3;
410   while(i>0){
411     P2OUT |= avaria;
412     __delay_cycles(400000);  //delay 1s
413     P2OUT = ~avaria;
414     __delay_cycles(400000);  //delay 1s
415     i--;
416   }
417
418 }
```