

Dispositivo emissor de luz de LED para neoformação tecidual

Angélica Kathariny de Oliveira Alves
Faculdade UnB Gama - FGA
Universidade de Brasília - UnB
Brasília, Brasil
Angel.kyka@hotmail.com

Mayara Barbosa dos Santos
Faculdade UnB Gama - FGA
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. O resultado obtido foi um protótipo que seleciona o tempo de tratamento e faz o desligamento do LEDs ao final do tempo estabelecido. Ademais apresenta o nível de carga da bateria utilizada para alimentar o sistema.

Palavras-chave—MSP430; fototerapia; diabetes;

I. INTRODUÇÃO

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 Diabetes 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.

O custo elevado do tratamento dessas lesões é um dos fatores que motivam a busca por soluções alternativas para esse processo [2], além de soluções mais eficientes que auxiliem na efetiva cura da ferida.

A fototerapia, terapia que faz uso da luz em práticas clínicas, se insere nesse contexto como uma opção de baixo custo, quando comparada com os curativos tradicionais. Em geral são utilizados diodos emissores de luz (LED) com comprimentos de onda que variam 405nm (cor azul) a 940nm (infravermelho) [3].

Em 2010, foi realizado um teste com fototerapia utilizando LED vermelho (620-630nm). Nesse teste ratos foram submetidos ao tratamento e após avaliação das lesões ao final

do tratamento concluiu-se que houve deposição de colágeno, substância de extrema importância no processo cicatricial, e efeitos anti-inflamatórios [4].

Um estudo realizado em 2013 aponta a terapia luminosa com LEDs como forma eficaz de tratamento de feridas [4]. Nesse estudo foi comprovado que a irradiação luminosa atuou na aceleração do processo cicatricial e na reepitelização da área ferida.

As pesquisas supracitadas justificam o desenvolvimento do protótipo proposto neste trabalho.

II. OBJETIVO

Este projeto tem o objetivo de desenvolver um dispositivo eletrônico para fototerapia utilizando LEDs que serão controlados 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 a porcentagem de carga da bateria.

IV. BENEFÍCIOS

O projeto apresenta os seguintes benefícios:

- 1) *Seleção do tempo de tratamento.*
- 2) *Temporização do uso do dispositivo sem a utilização de dispositivos adicionais, como cronômetros e relógios.*
- 3) *aviso sonoro e desligamento do dispositivo ao final do tratamento.*
- 4) *Baixo custo.*

V. DESENVOLVIMENTO DO PROTÓTIPO

A. Materiais utilizados

TABELA I. MATERIAIS UTILIZADOS NA CONFEÇÃO DO PROTÓTIPO E PREÇOS.

Materiais	Preço
42 LEDs	R\$ 15,50
6 Resistores	R\$ 0,25
2 Botões	R\$ 0,95
1 MSP430	R\$ 50,00
1 Módulo Display 7 segmentos – 8 dígitos	R\$ 24,90
3 Protoboards	R\$ 36,00
3 Pilhas alcalinas	R\$ 5,25
1 Buzzer	R\$ 1,00
Total	R\$ 133,85

B. Descrição do sistema de neoformação tecidual proposto

O dispositivo proposto nesse projeto é composto uma matriz de LEDs e os resistores necessários para seu funcionamento. Além disso, a parte responsável pelo controle dos LEDs tem como componente principal o Microcontrolador MSP430.

Ao ligar o dispositivo o usuário seleciona através de botões o tempo em que os LEDs devem permanecer acessos e em seguida dá início ao tratamento.

Durante o tempo de tratamento o tempo decorrido será mostrado no display 7 segmentos. Ao lado desse timer, também no display, será exibida a porcentagem de carga da bateria.

Ao término do tempo estabelecido será emitido dois bips, aviso sonoro, os LEDs irão piscar e em seguida desligar.

C. Descrição de hardware

Os conversores analógico-digital do MSP430 usam como referência interna a tensão de 2.5V, pois a regulação da tensão interna da launchpad converte 2.5V em 1.8V para o MSP que é tensão mínima de operação.

Para alimentar o circuito foi utilizado um conjunto de três pilhas AAA conectadas em série, promovendo uma tensão contínua de 4,5V. A placa regula essa tensão de para aproximadamente 3.6V, que é o valor necessário para a energização do chip microcontrolador que atua na faixa de operação de 1.8V a 3.6V.

Essa fonte externa também foi utilizada para garantir o funcionamento do display de 7 segmentos, pois o mesmo possui faixa de operação de 4.0V a 5.5V.

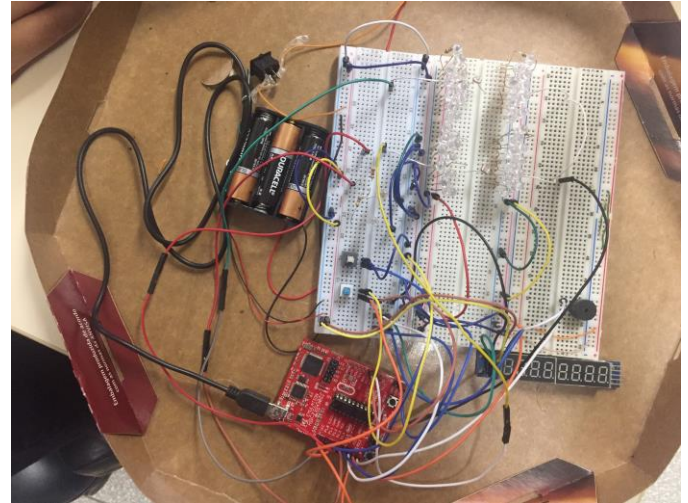


Fig. 1. Montagem do protótipo do dispositivo emissor de luz de LED para neoformação tecidual.

D. Descrição do software

O projeto foi desenvolvido em linguagem proprietária do fabricante do microcontrolador baseada na linguagem C. O compilador utilizado foi o software Energia versão 1.6.10E18.

As bibliotecas utilizadas foram: msp430G2553.h, proGetVcc, legacymsp430.h e stdio.h que habilita todas as funções de input e output dos pinos da placa.

Para tomar a medida da porcentagem de carga da bateria foi conectado o negativo da bateria junto ao negativo da placa launchpad, isto para garantir a mesma referência de tensão para todos os componentes. A equação 1 mostra o cálculo implementado em código para a obtenção da porcentagem de carga da bateria, onde o valor de 1800 que aparece dividindo na equação corresponde ao range em mV da operação.

$$\% \text{bateria} = (\text{valor lido} - 1800 * (100)) / 1800 \quad (1)$$

VI. RESULTADO

O dispositivo possui dois botões que servem para ajustar o tempo, iniciar e pausar o tratamento. O LED verde mostrado na figura 2 indica que os demais LEDs estão no modo *standby* aguardando os ajustes para o início do tratamento.

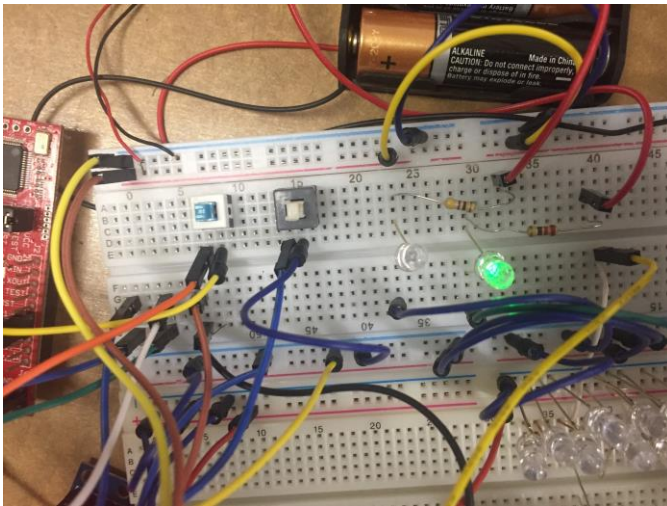


Fig. 2. Botões para ajuste do tempo de tratamento e LED verde indicativo do modo *standby*.

A primeira ação a ser tomada pelo usuário após ligar o dispositivo é selecionar o tempo de tratamento. A figura 3 mostra o display que indica a seleção dos segundos. A figura 3 mostra o display na função de seleção dos minutos.

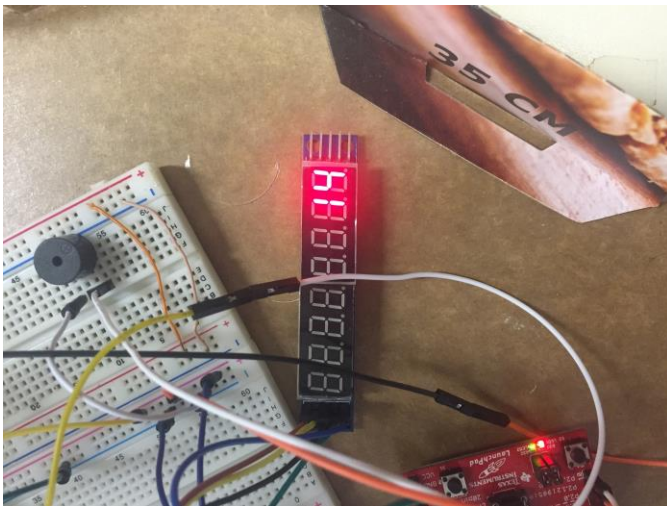


Fig. 3. Display no modo seleção dos segundos.

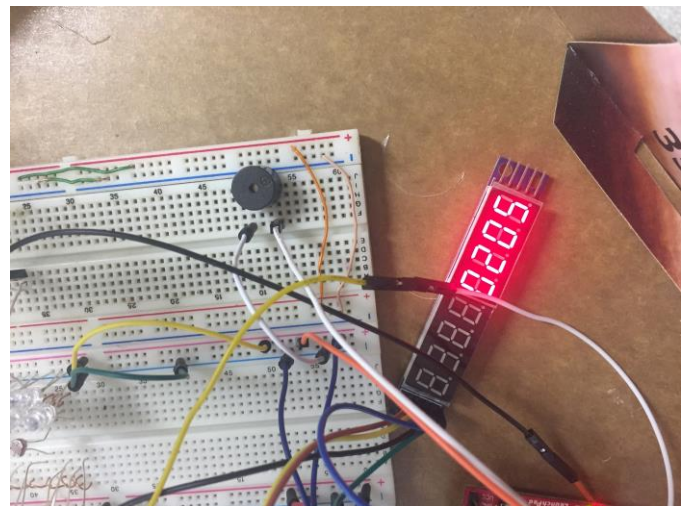


Fig. 4. Display no modo seleção dos minutos.

Além do *timer*, o display ainda apresenta a porcentagem de carga da bateria. Na figura 5, os tres digitos mais a esquerda representam a informação da bateria, que é atualizada a cada segundo.

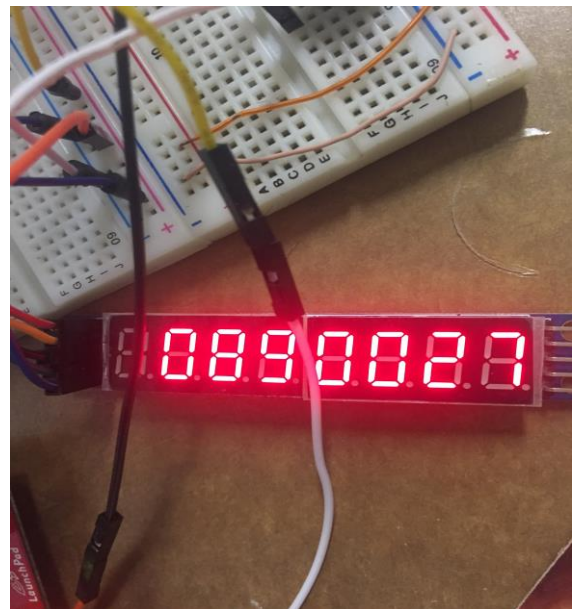


Fig. 5. Display: os três dígitos mais a esquerda representam a porcentagem de carga da bateria e os quatro dígitos mais a direita mostram o tempo decorrido do tratamento.

Na figura 6 é mostrado o LED vermelho, localizado ao lado do LED verde, aceso. Quando o tratamento inicia o LED verde apaga e o LED vermelho é aceso, permanecendo ligado até o fim do tempo estabelecido.

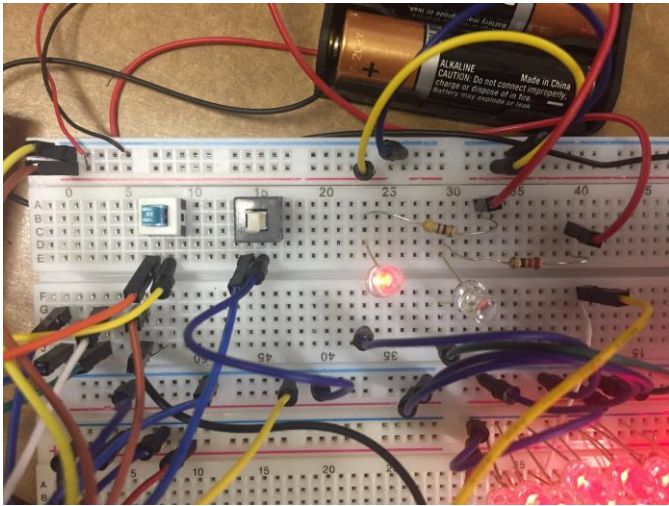


Fig. 6. LED vermelho aceso indicando o início do tratamento.

Na figura 7 é apresentado os LEDs em funcionamento. São eles os responsáveis por emitir a luz necessária para o tratamento das úlceras crônicas.

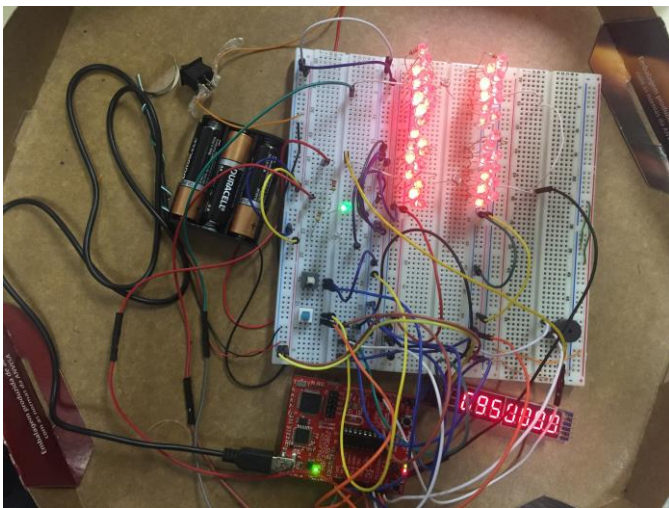


Fig. 7. Dispositivo em funcionemnto.

REFERÊNCIAS

- [1] MILECH, Adolfo. OLIVEIRA, José Egidio Paulo de. VENCIO, Sérgio. Diretrizes da Sociedade Brasileira de Diabetes (2015-2016) . São Paulo, 2016.
- [2] MOURA, Renata Oliveira, et al. "Efeitos da luz emitida por diodos (LED) e dos compostos de quitosana na cicatrização de feridas Revisão Sistemática." *Revista de Ciências Farmacêuticas Básica e Aplicada* 35.4 (2015).
- [3] MEYER, Patrícia Froes et al. Avaliação dos efeitos do LED na cicatrização de feridas cutâneas em ratos Wistar. *Fisioterapia Brasil*, v. 11, n. 6, p. 428-432, 2010.
- [4] 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.

VII. CONCLUSÃO

O projeto final apresentado alcançou os objetivos propostos. Foi implementado o controle dos LEDs, a seleção do tempo que eles deveriam ficar ligados, a apresentação do tempo decorrido e da porcentagem da bateria no display.

O desenvolvimento do protótipo proporcionou a dupla ampliar o conhecimento do microcontrolador MSP-430 e aplicar os vários conhecimentos adquiridos na disciplina, como o uso do conversor analógico-digital, comunicação do microcontrolador com dispositivos externos a placa e divisor de clock para a temporização.

APÊNDICE

Código utilizado no projeto.

```
1 //*****
2 //Bibliotecas
3 #include <msp430g2553.h>
4 #include <legacymsp430.h>
5 #include <stdio.h>
6 //*****
7
8 //*****
9 //Defeinicao das variaveis
10 #define CS BIT0 //2.0 is CS do Display
11
12 #define MOSI BIT7 //1.7 is SPI MOSI Entrada de dados do display
13
14 #define SCLK BIT5 //1.5 is SPI clock do display
15
16 #define EXA BIT1 //1.1 is button ON/OFF exame do equipamento
17
18 #define LEDS BIT1 //1.6 Leds do produto
19 #define avaria BIT2 //1.6 Leds do produto
20
21 #define comandos (LEDS|avaria)
22
23 #define STDY BIT3 //1.3 Led modo standby (verde)
24
25 #define OP BIT4 //1.4 Led modo operacional (vermelho)
26
27 #define MODES (BIT3|BIT4) //Agrupamento dos modos (LEDs)
```

```

31 //*****
32 //Declaração das funcoes
33 void Init_MAX7219(void);    //inicia o multiplexador do display
34
35 void SPI_Init(void); //Inicia o serial do display
36
37 void SPI_Write2(unsigned char, unsigned char);    //Escreve no display
38
39 int chamar(void);    //Chama as rotinas
40
41 unsigned int getVcc3(); //leitura da bateria
42
43 int analog(int sensorPinA0,int sensorPinA6);
44
45 void buzzer_inicial();
46 //*****
47
48 //*****
49 //Variaveis
50 int cont=3000; //contadores
51 int bateria();    //mostrar bateria
52 int segundos1=0;    //controla a quantidade de segundos (unidades)
53 int segundos2=0;    //controla a quantidade de segundos (dezenas)
54 int minutos1=0;    //controla a quantidade de minutos (unidades)
55 int minutos2=0;    //controla a quantidade de minutos (dezenas)
56 int atraso=0;    //atraso do delay, para acelerar o for
57 int sensorPin1 = A0;    // select the input pin for the potentiometer

```

```

58 int sensorPin2 = A6;    // select the input pin for the potentiometer
59 //int ledPin = 2;      // select the pin for the LED
60 int sensorValue1 = 0; //memoria da leitura anterior do sensor A0
61 int sensorValue2 = 0; //memoria da leitura anterior do sensor A6
62 //*****
63
64 int main(void)
65 {
66
67     volatile unsigned int i;
68     WDCTL = WDTPW + WDTHOLD;
69     //CONFIGURACAO DOS LEDS
70
71
72     //configura botao com resistor de pullup(P1REN abilitado e P1OUT em 1) **para pulldown(P1REN abilitado e P1OUT em 0)
73
74     P1DIR &= ~EXA;
75     P1REN |= EXA;
76     P1OUT |= EXA;
77
78     P1DIR &= ~MODES;
79     P1REN |= MODES;
80     P1OUT |= MODES;
81
82     P2OUT |= 0;
83     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;
```

```
141 }
142
143 int chamar(void)
144 {
145
146
147     if ((P1IN & EXA) == 0){
148         for(;;){
149             P2OUT |= LEDS;    //Liga LEDS para tratamento
150             P1OUT |= OP;      //Liga LED MODE OPERATING
151             P1OUT &= ~(STDY);  //Desliga LED STANDBY
152
153             if ((P1IN & EXA) == 0){
154                 analog(sensorPin1,sensorPin2);
155             }
156             while((P1IN & EXA) != 0){
157                 P1OUT &= ~OP;    //Desliga LED MODE OPERATING
158                 P1OUT |= (STDY); //Liga LED STANDBY
159                 P2OUT &= ~(LEDS); //Desliga LEDS para tratamento
160                 SPI_Write2(0x01,0x0);
161                 SPI_Write2(0x02,0x0);
162                 SPI_Write2(0x03,0x0);
163                 SPI_Write2(0x04,0x0);
164                 SPI_Write2(0x05,0x0);
165                 SPI_Write2(0x06,0x0);
166                 SPI_Write2(0x07,0x0);
167                 SPI_Write2(0x08,0x0);
168                 cont=0;
```

```

169     segundos1=0;
170     segundos2=0;
171     minutos1=0;
172     minutos2=0;
173 }
174
175 if(atraso==10){
176     segundos1=segundos1+1;           //incrementa o dígito 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;           //incrementa o dígito de unidade dos minutos
194     SPI_Write2(0x02,segundos2); //mostra a casa decimal dos segundos
195     SPI_Write2(0x03,minutos1);     //mostra a casa unitária dos minutos
196 }

```

```

197
198     if (minutos1==10){
199         segundos1=0;
200         segundos2=0;
201         minutos1=0;
202         minutos2=minutos2+1;           //incrementa o dígito de dezena dos minutos
203         SPI_Write2(0x02,segundos2); //mostra a casa decimal dos segundos
204         SPI_Write2(0x03,minutos1);   //mostra a casa unitária 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 unitária 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;
```



```

251 }
252
253 int bateria(void)
254 {
255     //getVcc pega tensão em mV, o msp430 trabalha entre 1,8 V e 3,6 V.
256     //Sendo assim o range de bateria é de 1,8 V, ou seja, 1,8V = 0% e 3,6 V = 100%
257
258     float carga;
259     int carga2;
260
261     carga=((getVcc3())-1800);          //bateria em mV
262
263     carga=(carga/1800*100);           //carga em %
264     carga2=carga;                     //conversao para int, display so aceita int
265
266     //o %10 mostra o numero que esta imediatamente apos a virgula, 653,1 (neste caso mostra o 3)
267
268     SPI_Write2(0x05,carga2%10);       //Escrever unidade da porcentagem no digito 5
269     SPI_Write2(0x06,(carga2/10)%10); //Escrever dezena da porcentagem no digito 5
270     SPI_Write2(0x07,(carga2/100)%10); //Escrever centena da porcentagem no digito 5
271
272
273
274     return (0);
275 }

```

```

277 unsigned int getVcc3()
278 {
279     ADC10CTL0 = SREF_1 + REFON + REF2_5V + ADC10ON + ADC10SHT_3; // use internal ref, turn on 2.5V ref,
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 ADC100SC.
284     while (!(ADC10CTL0 & ADC10IFG)); // Conversion done?
285     unsigned long mv = (ADC10MEM * 50001);
286     return ((unsigned int) (mv / 10241));
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     SPI_Write2(0x09, 0xFF);      //decode mode - converte decimal (0x09, 0x00)
345
346     SPI_Write2(0x0A, 0x0F);      //intensity control - intensidade maxima
347
348     SPI_Write2(0x0B, 0x06);      //scan limit original (0x0B, 0x07) digitos do display de 8 digitos
349
350     SPI_Write2(0x0C, 0x01);      //shutdown - operacao normal
351
352     SPI_Write2(0x0F, 0x01);      //display teste
353
354     SPI_Write2(0x0F, 0x00);      //display teste
355
356 }
357
358
359
360
361
362
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
414         P2OUT = ~avaria;
415         __delay_cycles(400000); //delay 1s
416         i--;
417     }
418 }

```