LED's Memory Game

Felipe Chermont
Engenharia de Eletrônica
Universidade de Brasília, FGA
Gama, Brasil
chernox27@gmail.com

Guilherme Simões Dias
Engenharia de Eletrônica
Universidade de Brasília, FGA
Gama, Brasil
g.simoesdias@gmail.com

I. Introdução

A. Revisão Bibliográfica

[1] O Transtorno de Déficit de Atenção com Hiperatividade (TDAH) é um transtorno neurobiológico que aparece geralmente na infância e se não acompanhado e trabalhado pode atrapalhar o âmbito pessoal e profissional durante a vida adulta. [2] Segundo os dados da Organização Mundial de Saúde, cerca de 4% da população adulta tem o TDAH, equivalente a aproximadamente 2 milhões de brasileiros. Já na população jovem, o TDAH afeta 6% das crianças e destas apenas 69% concluem os estudos.

Segundo o site NeuroSaber [3] o jogo da memória está entre as 10 melhores brincadeiras simples para acalmar crianças hiperativas. Sendo bastante usada pelos professores e pais de crianças que sofrem com este transtorno, por estimular habilidades como o pensamento, a memorização a identificação de cores e sons estabelecendo conceitos de igualdade e diferença, entre outros.

Pensando nisso, decidimos desenvolver um jogo da memória, utilizando a msp430, LEDs, botões e um buzzer, onde a pessoa deverá ver uma sequência pseudoaleatória de LEDs, com diferentes sons e após um sinal, repetir a sequência que foi mostrada anteriormente.

O projeto será baseado nos trabalhos [5] "*msp430launchpad-examples"* para inicializar os leds e como fazer o *debouncing* dos botões e no trabalho [6] "*Catch the LED"* onde veremos como criar uma sequência pseudoaleatória de LED´s

B. Justificativa

As capacidades cognitivas como memória e atenção são de extrema importância na vida pessoal de uma pessoa e o desenvolvimento incompleto dessas habilidades causa um impacto enorme na vida pessoal de um adulto. [4] Como destacado pela Organização Brasileira de Déficit de Atenção, os sintomas na vida adulta podem ocasionar prejuízos no trabalho, nas relações amorosas, problemas na condução de veículos dentre outros.

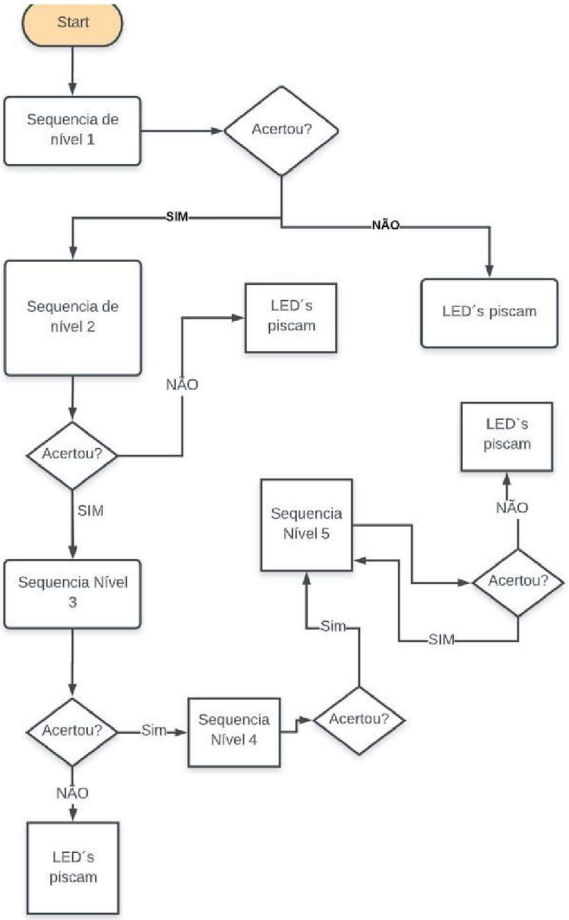
Este problema pode ser atenuado se algumas regiões do cérebro forem estimuladas desde a infância, para realizar este estímulo um método que vem se provando eficaz é a aplicação de jogos da memória. No entanto, para se manter a atenção de uma criança com TDAH é necessário fazer uso recursos chamativos como luzes e sons. Logo, decidimos fazer uso no jogo da memória de LEDs e buzzer.

Agora, para realizar a execução jogo foi necessário escolher um microcontrolador, dentre as opções estavam a MSP430 e a Armel AVR (Presente no Ardunio) por conta do preço e acessibilidade, a escolha pelo MSP, além de ser a placa utilizada na matéria, oferece vários modos de LPM (Low Power Mode), contém uma arquitetura de 16 bits e a MSP 430 permite a mudança do clock, algo que o Arduino não permite.

C. Objetivos

O objetivo do projeto é, através da msp430, desenvolver um jogo da memória que ajude pessoas com TDAH no desenvolvimento da memória e da concentração, por meio da memorização e repetição das sequencias geradas, de forma divertida fazendo uso de estímulos visuais e sonoros.

Conforme o Diagrama a seguir:



D. Requisitos

O projeto deve atender aos requisitos:

1. acender os LEDs numa sequência pseudoaleatória ao se iniciar o jogo;
2. gerar sons diferentes através do buzzer; correspondentes a cada LED.
3. enviar um sinal de cada botão que corresponde ao LED para verificar se a pessoa acertou a sequência.
4. acionar outro nível, aumentando a sequência sempre que a pessoa acertar a sequência, limitado até 5 vezes.
5. Piscar todos os LEDs 3 vezes caso a pessoa perca o jogo e entrar em modo de espera até ser inicializado novamente.

Componentes usados:

- 4 LEDs Difusos (5mm) nas cores azul, verde, amarelo e vermelho; - 5 Botões sem trava;
- 1 Protoboard;
- 1 Buzzer de 5V;
- 1 MSP430G

***Imagem de cada componente em anexo**

E. Benefícios

O projeto visa beneficiar pessoas com dificuldade de concentração devido ao transtorno. Proporcionando uma forma de treinar a concentração e à memória de forma divertida e intuitiva, além de desafiadora.

Link Para o GitHub e Trello:

GitHub Disponível em:

<https://github.com/chermont04/Eletronica-Embarcada>

Trello Disponível em:

<https://trello.com/b/VZx9T9tQ/eletronicaembarcada>

Figura 01: Led´s

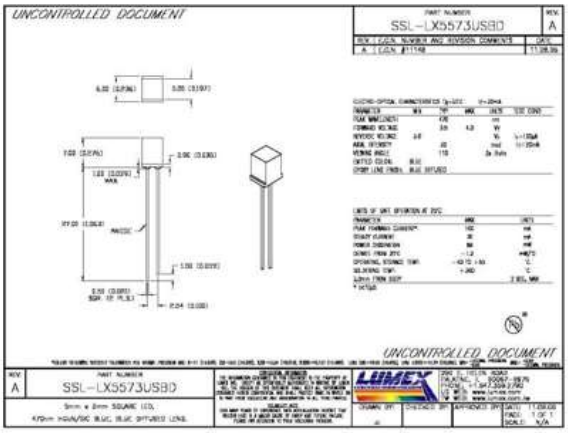


Figura 02: datasheet dos LEDs.

Figura 03: Botões sem trava

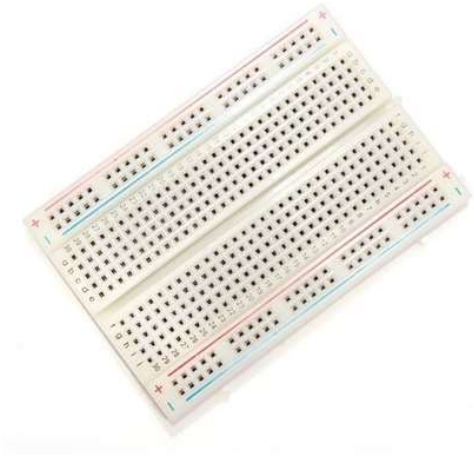


Figura 04: Protoboard.

Figura 05: Buzzer de 5V.

Figura 06: MSP340G .

V. Resultados

Na primeira parte do experimento, foram comprados e testado os materiais, sendo eles:

- Os quatro leds de 10mm aplicando uma voltagem de 3.3V, vista pelo datasheet deles;

- Os quatro botões de dois pinos, testados verificando a continuidade ao serem pressionados;

- O buzzer, similar ao teste dos Leds, foi aplicado uma voltagem de 5V;

Após o teste dos novos materiais, foi iniciada a implementação e desenvolvimento do Código, conforme o Apêndice, no microcontrolador MSP430G.

Inicialmente, focamos na criação da sequência pseudoaleatória fazendo uso de amostragens de dois clocks diferentes, que foram em seguida somados e em sequência multiplicados por uma constante.

Com a sequência gerada, foi criada uma rotina em que os LEDs piscam seguindo o vetor com a sequência pseudoaleatória, sendo que o número de LEDs que ascendem é de acordo com a variável round que é iniciada em 1 e é incrementada conforme o jogador vai acertando a sequência mostrada até alcançar o limite inicialmente estipulado de 9 rounds.

Para o ponto de controle 3 deu se continuidade ao código implementando a rotina da leitura dos botões para poder ser inserido os estímulos ao código e o adicional de que quando um botão é pressionado o respectivo LED associado ao botão é aceso indicando que a leitura foi feita.

Também fora adicionada uma rotina para verificação da sequência inserida, a qual verifica o valor de vetor em ‘i’ e compara com o valor atribuído a cada botão.

Ademais, foi adicionada uma rotina com a opção de selecionar a dificuldade dentre 3 possibilidades, em que a diferença entre elas é o número de rounds para alcançar a vitória no jogo, indicadas pelos LEDs verde (fácil), azul (médio) e amarelo (difícil).

Algumas outras implementações visuais foram adicionadas como rotinas de vitória e derrota que acendem os LEDs em um padrão ordenado.

Para o ponto de controle 4 buscamos inicialmente implementar uma ISR (interrupt service rotine) no reset/start e para isso alteramos o botão desta função para um dedicado totalmente a ela. Além disto foram adicionadas rotinas para implementar os sons do jogo, fazendo uso de diferentes frequências sendo enviadas ao buzzer para ajudar a identificar qual botão piscou pelas diferentes notas sonoras. Entretanto, a implementação de uma parte do código em assembly, que foi decido ser a parte de chamada de rotina para cada nota musical, ainda está pendente e sendo desenvolvida para a apresentação final.

REFERENCES

[1] VINOCUR – Evelyn – TDAH:o que é, sintomas e tratamento – Disponível em: < https://www.minhavida.com.br/saude/temas/tdah>. Acesso em: 13 set. 2019.

[2] JOSÉ - Fernando - Aumenta o número de pessoas com TDAH e o diagnóstico adequado é o maior desafio - 16 abril 2019 - Disponível em: <http://www.osul.com.br/aumenta-o-numero-de-pessoas-com-tdah-eodiagnostico-adequado-e-o-maior-desafio/>. Acesso em: 13 set. 2019.

[3]10 brincadeiras simples para desacelerar as crianças com TDAH - 05 ago. 2018 - Disponível em: <https://neurosaber.com.br/10brincadeirassimples-para-desacelerar-as-criancas-com-tdah/>. Acesso em: 13 set. 2019.

[4] GHIGIARELLI - Denise - O TDAH NO ADULTO E O PROCESSAMENTO DAS EMOÇÕES - 27 abr. 2016. Disponível em: <https://www.normaseregras.com/normas-abnt/referencias/>. Acesso em: 13 set. 2019.

[5] Alfy7 – MSP430 – Launchpad-Examples, Disponível em: <https://github.com/alfy7/MSP430-Launchpad-Examples> [6] *AdityaWadhwa - CatchTheLED, Disponível em:* < https://github.com/AdityaWadhwa/CatchTheLED>

```c
#include <msp430G2553.h>
#include <msp430.h>

// p1.3 , p2.0, p2.3, p1.0 --  p1.5


// Definindo os LEDs e botões usados para parar o timer e pegar o número aleatório.
#define RED_LED         BIT1                        // Red LED          (P1.1)
#define GREEN_LED       BIT2                        // Green LED        (P1.2)
#define BLUE_LED        BIT4                        // Blue LED         (P1.4)
#define YELLOW_LED      BIT7                        // Yellow LED       (P1.5)

#define RED_BTN         BIT1                        // Red button       (P2.1)
#define GREEN_BTN       BIT2                        // Green button     (P2.2)
#define BLUE_BTN        BIT4                        // Blue button      (P2.4)
#define YELLOW_BTN      BIT5                        // Yellow button    (P2.5)
#define RED_SND         BIT0                        // (P1.0)
#define GREEN_SND       BIT3                        // (P1.3)
#define BLUE_SND        BIT0                        // (P2.0)
#define YELLOW_SND      BIT3                        // (P2.3)

// Tamanhos de pausas diferentes para a função wait()
#define TEN_MS      1
#define CENTI_SEC   12
#define QUART_SEC   30
#define HALF_SEC    60
#define ONE_SEC     120
#define BLINK       80
#define PAUSE       30

// Timers
#define ENABLE_PINS 0xFFFE
#define ACLK        0x0100
#define SMCLK       0x0200
#define CONTINUOUS  0x0020
#define UP          0x0010

// Numero de rounds para acabar o jogo de acordo com a dificuldade
#define EASY            8
#define NORMAL          10
#define HARD            12
#define EXTREME         16

//Definition of the notes' frequecies in Hertz.
#define c 261
#define d 294
#define e 329
#define f 349
#define g 391
#define gS 415
#define a 440
#define aS 455
#define b 466
#define cH 523
#define cSH 554
#define dH 587
#define dSH 622
#define eH 659
#define fH 698
#define fSH 740
#define gH 784
#define gSH 830
```

```c
#define aH 880

// Funções para o funcionamento do game
void Reset(void);
int ChooseDifficulty(void);
void Wait(int t);

int GetFirstNumber(void);
int GetSecondNumber(void);
void MakeSequence(int sequence[16], int first_number, int second_number);

void BlinkLeds(int sequence[16], int round);
int GetAnswer(int sequence[16], int round);
void CorrectAnswer(void);

void Win(void);
void Loss(void);

void Tic();
void Toc();
void Tuc();
void Tac();

void delay_ms(unsigned int ms )
{
    unsigned int i;
    for (i = 0; i<= ms; i++)
       __delay_cycles(500); //Built-in function that suspends the execution for 500 cicles
}

void delay_us(unsigned int us )
{
    unsigned int i;
    for (i = 0; i<= us/2; i++)
       __delay_cycles(1);
}

//This function generates the square wave that makes the piezo speaker sound at a determinated frequency.
void beep(unsigned int note, unsigned int duration)
{
    int i;
    long delay = (long)(10000/note);  //This is the semiperiod of each note.
    long time = (long)((duration*100)/(delay*2));  //This is how much time we need to spend on the note.
    for (i=0;i<time;i++)
    {
        P1OUT |= BIT0;     //Set P1.2...
        delay_us(delay);   //...for a semiperiod...
        P1OUT &= ~BIT0;    //...then reset it...
        delay_us(delay);   //...for the other semiperiod.
    }
    delay_ms(20); //Add a little delay to separate the single notes
}

void main(void)
{
    // desativa o watch dog timer.
    WDTCTL  = WDTPW | WDTHOLD;

    // inicia o timerA0 com ACLK, contando até 10ms.
    TA0CTL |= ACLK | UP;
```

```
123     TA0CCR0 = 400;
124
125     // Inicia o Timer A1 com SMCLK.
126     TA0CTL |= ACLK | CONTINUOUS; // aqui
127     TA1CTL |= SMCLK | CONTINUOUS;
128
129     // habilita os leds como saída e os botões como entrada.
130     P1DIR |= BLUE_LED | YELLOW_LED | RED_LED | GREEN_LED;
131     P1DIR |= RED_SND | GREEN_SND;
132     P1DIR |= BLUE_SND | YELLOW_SND;
133     P1OUT &= (~RED_SND);
134     P1OUT &= (~BLUE_SND);
135     P1OUT &= (~GREEN_SND);
136     P1OUT &= (~YELLOW_SND);
137     P1OUT &= (~RED_LED);
138     P1OUT &= (~BLUE_LED);
139     P1OUT &= (~GREEN_LED);
140     P1OUT &= (~YELLOW_LED);
141     P2OUT |= YELLOW_BTN;
142     P2REN |= YELLOW_BTN;
143     P2OUT |= RED_BTN;
144     P2REN |= RED_BTN;
145     P2OUT |= GREEN_BTN;
146     P2REN |= GREEN_BTN;
147     P2OUT |= BLUE_BTN;
148     P2REN |= BLUE_BTN;
149
150     while(1)
151     {
152         // espera o jogador iniciar o jogo para então gerar o numero randomico.
153         int first_number = 0;
154         Reset();
155         first_number = GetFirstNumber();
156         Wait(QUART_SEC);
157         // espera o jogador selecionar a dificuldade.
158         int difficulty;
159         int second_number = 0;
160         difficulty = ChooseDifficulty();
161         second_number = GetSecondNumber();
162         // Preenche o array com uma combinação de dois números aleatórios.
163         int sequence[16] = {0.0};
164         MakeSequence(sequence, first_number, second_number);
165         int game_state = 1;
166         int round = 0;
167         while(game_state == 1)
168         {
169             Wait(ONE_SEC);
170             BlinkLeds(sequence, round);
171
172             // Espera o jogador prescionar o botão e verifica se é o correto.
173             Wait(TEN_MS);
174             game_state = GetAnswer(sequence, round);
175             Wait(TEN_MS);
176
177             // Se a resposta for correta, pisca o verde e segue para a proxima.
178             if (game_state == 1)
179             {
180                 CorrectAnswer();
181                 round++;
182             }
183             Wait(TEN_MS);
184
```

```
185             // Ao se terminar o maximo de rounds permitidos pela dificuldade, os leds
    piscaram indicando vitoria.
186             if (round == difficulty)
187             {
188                 game_state = 2;
189             }
190             Wait(TEN_MS);
191         }
192         if (game_state == 2)
193         {
194             Win();
195         }
196         // If not, then the loop quit because game_state == 0; show a loss
197         else
198         {
199             Loss();
200         }
201     }
202 }
203
204 void Reset(void)
205 {
206     P1OUT |= RED_LED;
207     int x = 0;
208     while (x < 1)
209     {
210         if ((P2IN & RED_BTN) == 0)
211         {
212             P1OUT &= (~RED_LED);
213             x = 1;
214         }
215     }
216 }
217
218 /*****************************************************************************
219 /   Mostra para o jogador 3 cores de LEDs (Verdem, azul e amarelo); Eles podem prescionar
220 /   o botão correspondente para selecionar a dificuldade, que decide o número de
221 /   rounds que devem ser cumpridos para ganhar o jogo, assim como a velocidade de
    amostragem.
222 *****************************************************************************/
223 int ChooseDifficulty(void)
224 {
225     P1OUT |= (BLUE_LED | YELLOW_LED | GREEN_LED);
226     int x = 0;
227     int i;
228     while (x < 1)
229     {
230         if ((P2IN & GREEN_BTN) == 0) //
    ------------------------------------------------------------------------------
231         {
232             P1OUT &= (~GREEN_LED);
233             P1OUT &= (~(BLUE_LED | YELLOW_LED));
234             x = EASY;
235             for (i = 0; i < 8; i++)
236             {
237                 P1OUT = (P1OUT ^ GREEN_LED);
238                 Wait(CENTI_SEC);
239             }
240         }
241         if ((P2IN & BLUE_BTN) == 0) //
    ------------------------------------------------------------------------------
242         {
```

```
243             P1OUT &= (~GREEN_LED);
244             P1OUT &= (~(BLUE_LED | YELLOW_LED));
245             x = NORMAL;
246             for (i = 0; i < 8; i++)
247             {
248                 P1OUT = (P1OUT ^ BLUE_LED);
249                 Wait(CENTI_SEC);
250             }
251         }
252         if ((P2IN & YELLOW_BTN) == 0) //
    ------------------------------------------------------------------------------
253         {
254             P1OUT &= (~GREEN_LED);
255             P1OUT &= (~(BLUE_LED | YELLOW_LED));
256             x = HARD;
257             for (i = 0; i < 8; i++)
258             {
259                 P1OUT = (P1OUT ^ YELLOW_LED);
260                 Wait(CENTI_SEC);
261             }
262         }
263         if ((P2IN & RED_BTN) == 0) //
    ------------------------------------------------------------------------------
264         {
265             P1OUT &= (~GREEN_LED);
266             P1OUT &= (~(BLUE_LED | YELLOW_LED));
267             x = EXTREME;
268             for (i = 0; i < 8; i++)
269             {
270                 P1OUT = (P1OUT ^ RED_LED);
271                 Wait(CENTI_SEC);
272             }
273         }
274     }
275     return x;
276 }
277
278 void Wait(int t)
279 {
280     int i = 0;
281     // enquanto a contagem não atingir o limite:
282     while (i <= t)
283     {   // quando outros 10 ms tiverem passado.
284         if (TA0CTL & TAIFG)
285         {
286             // aumenta o contador e conta novamente 10 ms.
287             i++;
288             TA0CTL &= (~TAIFG);
289         }
290     }
291 }
292
293 void Tic ()
294 {
295     beep(a, 500);
296 }
297 void Toc ()
298 {
299     beep(cH, 150);
300 }
301 void Tac ()
302 {
```

```
303     beep(b, 125);
304 }
305 void Tuc ()
306 {
307     beep(aH, 880);
308 }
309
310 int GetFirstNumber(void)
311 {
312     int first_num = 0;
313     first_num = TA0R;
314     return first_num;
315 }
316
317 int GetSecondNumber(void)
318 {
319     int second_num = 0;
320     second_num = TA1R;
321     return second_num;
322 }
323
324 // cira uma sequencia semi aleatória usando os dois arrays criados anteriormente.
325 void MakeSequence(int sequence[8], int first_number, int second_number)
326 {
327     int i;
328     int first_array[16] = {0.0};
329     int second_array[16] = {0.0};
330     for (i = 0; i < 16; i++)
331     {
332         first_array[(15 - i)] = ((first_number >> i) & 0x01);
333         second_array[(15 - i)] = ((second_number >> i) & 0x01);
334     }
335     for (i = 0; i < 2; i++)
336     {
337         sequence[i] = (first_array[i]) + (second_array[i] * 1);
338     }
339     for (i = 2; i < 3; i++)
340     {
341         sequence[i] = (first_array[i]) + (second_array[i] * 3);
342     }
343     for (i = 3; i < 4; i++)
344     {
345         sequence[i] = (first_array[i]) + (second_array[i] * 4);
346     }
347     for (i = 4; i < 5; i++)
348     {
349         sequence[i] = (first_array[i]) + (second_array[i] * 1);
350     }
351     for (i = 5; i < 6; i++)
352     {
353         sequence[i] = (first_array[i]) + (second_array[i] * 3);
354     }
355     for (i = 6; i < 7; i++)
356     {
357         sequence[i] = (first_array[i]) + (second_array[i] * 4);
358     }
359     for (i = 7; i < 8; i++)
360     {
361         sequence[i] = (first_array[i]) + (second_array[i] * 4);
362     }
363     for (i = 8; i < 9; i++)
364     {
```

```
365         sequence[i] = (first_array[i]) + (second_array[i] * 1);
366     }
367     for (i = 9; i < 10; i++)
368     {
369         sequence[i] = (first_array[i]) + (second_array[i] * 3);
370     }
371     for (i = 10; i < 11; i++)
372     {
373         sequence[i] = (first_array[i]) + (second_array[i] * 4);
374     }
375     for (i = 11; i < 12; i++)
376     {
377         sequence[i] = (first_array[i]) + (second_array[i] * 3);
378     }
379     for (i = 12; i < 13; i++)
380     {
381         sequence[i] = (first_array[i]) + (second_array[i] * 1);
382     }
383     for (i = 13; i < 14; i++)
384     {
385         sequence[i] = (first_array[i]) + (second_array[i] * 3);
386     }
387     for (i = 14; i < 15; i++)
388     {
389         sequence[i] = (first_array[i]) + (second_array[i] * 1);
390     }
391     for (i = 15; i < 16; i++)
392     {
393         sequence[i] = (first_array[i]) + (second_array[i] * 4);
394     }
395 }
396
397 void BlinkLeds(int sequence[16], int round)
398 {
399     int i = 0;
400     do
401     {
402         switch (sequence[i])
403         {
404         case (0):   P1OUT |= RED_LED;
405                     Wait(BLINK);
406                     P1OUT &= (~RED_LED);
407                     Tic();
408                     Wait(PAUSE);
409                     break;
410
411         case (1):   P1OUT |= GREEN_LED;
412                     Wait(BLINK);
413                     P1OUT &= (~GREEN_LED);
414                     Tac();
415                     Wait(PAUSE);
416                     break;
417
418         case (3):   P1OUT |= BLUE_LED;
419                     Wait(BLINK);
420                     P1OUT &= (~BLUE_LED);
421                     Tuc();
422                     Wait(PAUSE);
423                     break;
424
425         case (4):   P1OUT |= YELLOW_LED;
426                     Wait(BLINK);
```

```
427                     P1OUT &= (~YELLOW_LED);
428                     Toc();
429                     Wait(PAUSE);
430                     break;
431         }
432
433         i = i + 1;
434
435     }
436     while (i <= round);
437 }
438
439 /*************************************************************************
440 / Essa função espera o jogador aplicar uma resposta para depois julga-la.
441 /
442 / Isto é feito esperando o jogador aplicar uma resposta e depois verificando se
443 / se é a entrada certa para o elemento i da sequencia do array, em que i é o
444 / n-nesimo LED mostrado.
445 *************************************************************************/
446 int GetAnswer(int sequence[16], int round)
447 {
448     int i = 0;
449     int game_over = 0;
450     // Permanece no loop até que uma resposta errada seja emitida.
451     while (i <= round && game_over == 0)
452     {
453     //    Wait(HALF_SEC);
454         if ((P2IN & RED_BTN) == 0)
455         {
456             P1OUT |= RED_LED;
457             if (sequence[i] == 0)
458                 i++;
459             else
460                 game_over = 1;
461             Wait(QUART_SEC);
462             P1OUT &= (~RED_LED);
463         }
464         if ((P2IN & GREEN_BTN) == 0)
465         {
466             P1OUT |= GREEN_LED;
467             if (sequence[i] == 1)
468                 i++;
469             else
470                 game_over = 1;
471             Wait(QUART_SEC);
472             P1OUT &= (~GREEN_LED);
473         }
474         if ((P2IN & BLUE_BTN) == 0)
475         {
476             P1OUT |= BLUE_LED;
477             if (sequence[i] == 3)
478                 i++;
479             else
480                 game_over = 1;
481             Wait(QUART_SEC);
482             P1OUT &= (~BLUE_LED);
483         }
484         if ((P2IN & YELLOW_BTN) == 0)
485         {
486             P1OUT |= YELLOW_LED;
487             if (sequence[i] == 4)
488                 i++;
```

```
489             else
490                 game_over = 1;
491             Wait(QUART_SEC);
492             P1OUT &= (~YELLOW_LED);
493         }
494     }
495     // Se a resposta for errada, envia um sinal para a função main() para mostrar
496     // Loss().
497     if (game_over == 0)
498         return 1;
499     // Caso contrário, envia um sinal para a função main() para indicar que a
500     // resposta foi correta e continuar.
501     else
502         return 0;
503 }
504
505
506 /*****************************************************************************
507 /   Essa função pisca o LED verde 8 vezes bem rapido indicando que a sequencia
508 /   inserida é correta.
509 *****************************************************************************/
510 void CorrectAnswer(void)
511 {
512     int i;
513     for (i = 0; i < 8; i++)
514     {
515         P2OUT = (P2OUT ^ GREEN_LED);
516         Wait(CENTI_SEC);
517     }
518 }
519
520 /*****************************************************************************
521 /   Essa função faz com que os LEDs pisquem rapido indicando a vitória
522 *****************************************************************************/
523 void Win(void)
524 {
525     int i;
526     for (i = 0; i < 3; i++)
527     {
528         P1OUT |= RED_LED;
529         Wait(CENTI_SEC);
530         P1OUT &= (~RED_LED);
531         Wait(CENTI_SEC);
532         P1OUT |= GREEN_LED;
533         Wait(CENTI_SEC);
534         P1OUT &= (~GREEN_LED);
535         Wait(CENTI_SEC);
536         P1OUT |= BLUE_LED;
537         Wait(CENTI_SEC);
538         P1OUT &= (~BLUE_LED);
539         Wait(CENTI_SEC);
540         P1OUT |= YELLOW_LED;
541         Wait(CENTI_SEC);
542         P1OUT &= (~YELLOW_LED);
543         Wait(CENTI_SEC);
544     }
545 }
546
547 /*****************************************************************************
548 / Essa função faz com que o LED vermelho pisque 3 vezes devagar indicando a derrota
549 *****************************************************************************/
550 void Loss(void)
```

```
551 {
552     int i;
553     for (i = 0; i < 3; i++)
554     {
555         P1OUT |= RED_LED;
556         Wait(QUART_SEC);
557         P1OUT &= (~RED_LED);
558         Wait(QUART_SEC);
559     }
560     Wait(ONE_SEC);
561 }
562
```