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.

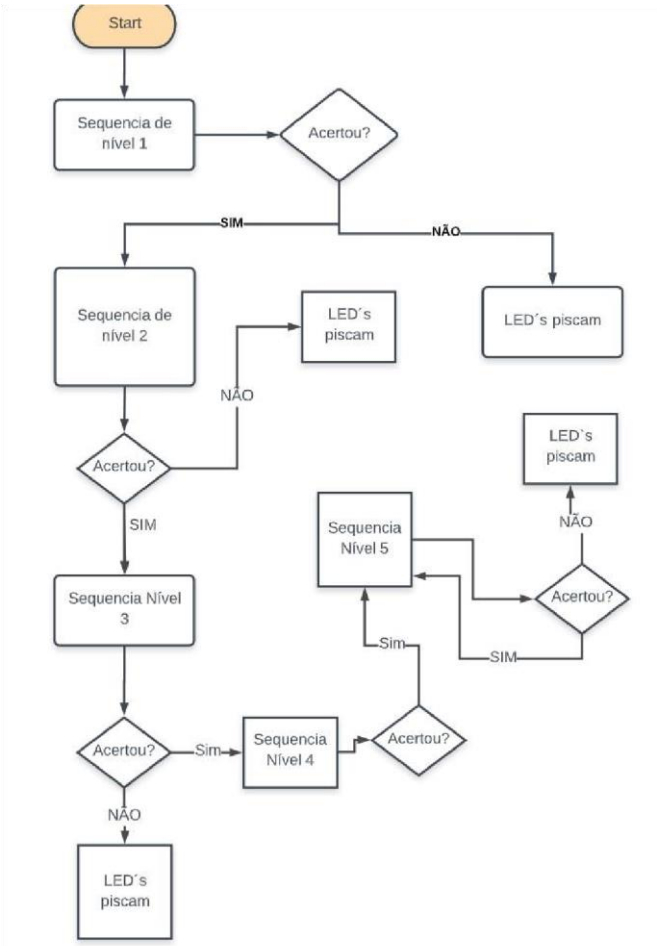
Este problema pode ser atenuado se algumas regiões do cérebro forem estimuladas desde a infância, para realizar este estímulo um método que vem se provando eficaz é a aplicação de jogos da memória. No entanto, para se manter a atenção de uma criança com TDAH é necessário fazer uso recursos chamativos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

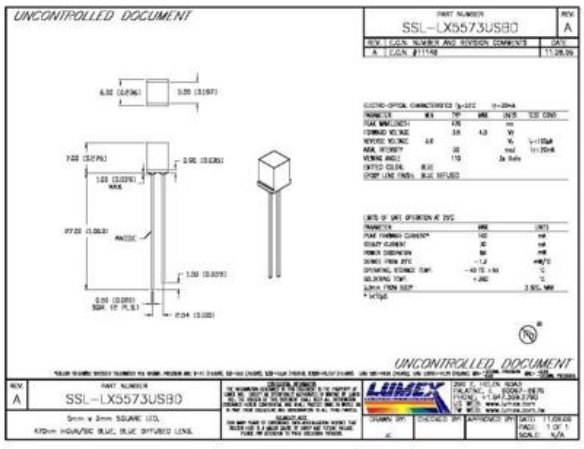


Figura 02: datasheet dos LEDs.

Figura 03: Botões sem trava

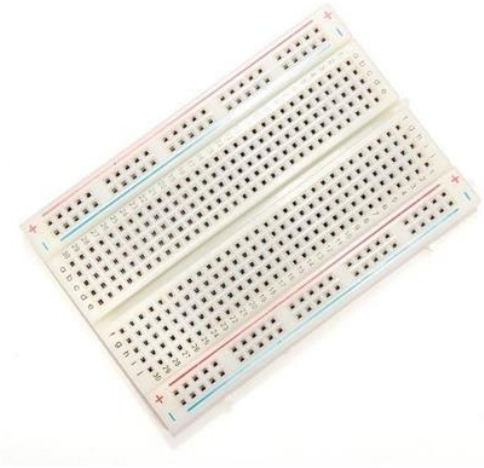


Figura 04: Protoboard.

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 no reset/start e adicionar uma parte do código em assembly que seria as rotinas de vitória e derrota, no entanto os locais podem estar sujeitos a mudança caso necessário. Como também adicionar o buzzer para ajudar o jogador ao orientá-lo com sons e tentaremos adicionar um placar usando um display LCD para indicar quantas respostas corretas foram dadas.

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>

// 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      BIT5                    // 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)

// 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       50
#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

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

    // inicia o timerA0 com ACLK, contando até 10ms.
    TA0CTL |= ACLK | UP;
    TA0CCR0 = 400;

    // Inicia o Timer A1 com SMCLK.
```

```
63      TA0CTL |= ACLK | CONTINUOUS; // aqui
64      TA1CTL |= SMCLK | CONTINUOUS;
65
66      // habilita os leds como saída e os botões como entrada.
67      P1DIR |= BLUE_LED | YELLOW_LED | RED_LED | GREEN_LED;
68      P1OUT &= (~RED_LED);
69      P1OUT &= (~BLUE_LED);
70      P1OUT &= (~GREEN_LED);
71      P1OUT &= (~YELLOW_LED);
72      P2OUT |= YELLOW_BTN;
73      P2REN |= YELLOW_BTN;
74      P2OUT |= RED_BTN;
75      P2REN |= RED_BTN;
76      P2OUT |= GREEN_BTN;
77      P2REN |= GREEN_BTN;
78      P2OUT |= BLUE_BTN;
79      P2REN |= BLUE_BTN;
80
81      while(1)
82      {
83          // espera o jogador iniciar o jogo para então gerar o numero randomico.
84          int first_number = 0;
85          Reset();
86          first_number = GetFirstNumber();
87          Wait(QUART_SEC);
88          // espera o jogador selecionar a dificuldade.
89          int difficulty;
90          int second_number = 0;
91          difficulty = ChooseDifficulty();
92          second_number = GetSecondNumber();
93          // Preenche o array com uma combinação de dois números aleatórios.
94          int sequence[16] = {0.0};
95          MakeSequence(sequence, first_number, second_number);
96          int game_state = 1;
97          int round = 0;
98          while(game_state == 1)
99          {
100             Wait(ONE_SEC);
101             BlinkLeds(sequence, round);
102
103             // Espera o jogador prescionar o botão e verifica se é o correto.
104             Wait(TEN_MS);
105             game_state = GetAnswer(sequence, round);
106             Wait(TEN_MS);
107
108             // Se a resposta for correta, pisca o verde e segue para a proxima.
109             if (game_state == 1)
110             {
111                 CorrectAnswer();
112                 round++;
113             }
114             Wait(TEN_MS);
115
116             // Ao se terminar o maximo de rounds permitidos pela dificuldade, os leds
    piscaram indicando vitoria.
117             if (round == difficulty)
118             {
119                 game_state = 2;
120             }
121             Wait(TEN_MS);
122         }
123         if (game_state == 2)
```

```
124         {
125             Win();
126         }
127         // If not, then the loop quit because game_state == 0; show a loss
128         else
129         {
130             Loss();
131         }
132     }
133 }
134
135 void Reset(void)
136 {
137     P1OUT |= RED_LED;
138     int x = 0;
139     while (x < 1)
140     {
141         if ((P2IN & RED_BTN) == 0)
142         {
143             P1OUT &= (~RED_LED);
144             x = 1;
145         }
146     }
147 }
148
149 /*************************************************************************
150 /   Mostra para o jogador 3 cores de LEDs (Verdem, azul e amarelo); Eles podem prescionar
151 /   o botão correspondente para selecionar a dificuldade, que decide o número de
152 /   rounds que devem ser cumpridos para ganhar o jogo, assim como a velocidade de
    amostragem.
153 *************************************************************************/
154 int ChooseDifficulty(void)
155 {
156     P1OUT |= (BLUE_LED | YELLOW_LED | GREEN_LED);
157     int x = 0;
158     int i;
159     while (x < 1)
160     {
161         if ((P2IN & GREEN_BTN) == 0) //
    ------------------------------------------------------------------------------
162         {
163             P1OUT &= (~GREEN_LED);
164             P1OUT &= (~(BLUE_LED | YELLOW_LED));
165             x = 3;
166             for (i = 0; i < 8; i++)
167             {
168                 P1OUT = (P1OUT ^ GREEN_LED);
169                 Wait(CENTI_SEC);
170             }
171         }
172         if ((P2IN & BLUE_BTN) == 0) //
    ------------------------------------------------------------------------------
173         {
174             P1OUT &= (~GREEN_LED);
175             P1OUT &= (~(BLUE_LED | YELLOW_LED));
176             x = 4;
177             for (i = 0; i < 8; i++)
178             {
179                 P1OUT = (P1OUT ^ BLUE_LED);
180                 Wait(CENTI_SEC);
181             }
182         }
```

```
183         if ((P2IN & YELLOW_BTN) == 0) //
    ------------------------------------------------------------------------------------
184         {
185             P1OUT &= (~GREEN_LED);
186             P1OUT &= (~(BLUE_LED | YELLOW_LED));
187             x = 5;
188             for (i = 0; i < 8; i++)
189             {
190                 P1OUT = (P1OUT ^ YELLOW_LED);
191                 Wait(CENTI_SEC);
192             }
193         }
194         if ((P2IN & RED_BTN) == 0) //
    ------------------------------------------------------------------------------------
195         {
196             P1OUT &= (~GREEN_LED);
197             P1OUT &= (~(BLUE_LED | YELLOW_LED));
198             x = 9;
199             for (i = 0; i < 8; i++)
200             {
201                 P1OUT = (P1OUT ^ RED_LED);
202                 Wait(CENTI_SEC);
203             }
204         }
205     }
206     return x;
207 }
208
209 void Wait(int t)
210 {
211     int i = 0;
212     // enquanto a contagem não atingir o limite:
213     while (i <= t)
214     {   // quando outros 10 ms tiverem passado.
215         if (TA0CTL & TAIFG)
216         {
217             // aumenta o contador e conta novamente 10 ms.
218             i++;
219             TA0CTL &= (~TAIFG);
220         }
221     }
222 }
223
224 int GetFirstNumber(void)
225 {
226     int first_num = 0;
227     first_num = TA0R;
228     return first_num;
229 }
230
231 int GetSecondNumber(void)
232 {
233     int second_num = 0;
234     second_num = TA1R;
235     return second_num;
236 }
237
238 // cira uma sequencia semi aleatória usando os dois arrays criados anteriormente.
239 void MakeSequence(int sequence[8], int first_number, int second_number)
240 {
241     int i;
242     int first_array[16] = {0.0};
```

```
243     int second_array[16] = {0.0};
244     for (i = 0; i < 16; i++)
245     {
246         first_array[(15 - i)] = ((first_number >> i) & 0x01);
247         second_array[(15 - i)] = ((second_number >> i) & 0x01);
248     }
249     for (i = 0; i < 2; i++)
250     {
251         sequence[i] = (first_array[i]) + (second_array[i] * 1);
252     }
253     for (i = 2; i < 3; i++)
254     {
255         sequence[i] = (first_array[i]) + (second_array[i] * 3);
256     }
257     for (i = 3; i < 4; i++)
258     {
259         sequence[i] = (first_array[i]) + (second_array[i] * 4);
260     }
261     for (i = 4; i < 5; i++)
262     {
263         sequence[i] = (first_array[i]) + (second_array[i] * 1);
264     }
265     for (i = 5; i < 6; i++)
266     {
267         sequence[i] = (first_array[i]) + (second_array[i] * 3);
268     }
269     for (i = 6; i < 7; i++)
270     {
271         sequence[i] = (first_array[i]) + (second_array[i] * 4);
272     }
273     for (i = 7; i < 8; i++)
274     {
275         sequence[i] = (first_array[i]) + (second_array[i] * 4);
276     }
277     for (i = 8; i < 9; i++)
278     {
279         sequence[i] = (first_array[i]) + (second_array[i] * 1);
280     }
281     for (i = 9; i < 10; i++)
282     {
283         sequence[i] = (first_array[i]) + (second_array[i] * 3);
284     }
285     for (i = 10; i < 11; i++)
286     {
287         sequence[i] = (first_array[i]) + (second_array[i] * 4);
288     }
289     for (i = 11; i < 12; i++)
290     {
291         sequence[i] = (first_array[i]) + (second_array[i] * 3);
292     }
293     for (i = 12; i < 13; i++)
294     {
295         sequence[i] = (first_array[i]) + (second_array[i] * 1);
296     }
297     for (i = 13; i < 14; i++)
298     {
299         sequence[i] = (first_array[i]) + (second_array[i] * 3);
300     }
301     for (i = 14; i < 15; i++)
302     {
303         sequence[i] = (first_array[i]) + (second_array[i] * 1);
304     }
```

```
305     for (i = 15; i < 16; i++)
306     {
307         sequence[i] = (first_array[i]) + (second_array[i] * 4);
308     }
309 }
310
311 void BlinkLeds(int sequence[16], int round)
312 {
313     int i = 0;
314     do
315     {
316         switch (sequence[i])
317         {
318         case (0):   P1OUT |= RED_LED;
319                     Wait(BLINK);
320                     P1OUT &= (~RED_LED);
321                     Wait(PAUSE);
322                     break;
323
324         case (1):   P1OUT |= GREEN_LED;
325                     Wait(BLINK);
326                     P1OUT &= (~GREEN_LED);
327                     Wait(PAUSE);
328                     break;
329
330         case (3):   P1OUT |= BLUE_LED;
331                     Wait(BLINK);
332                     P1OUT &= (~BLUE_LED);
333                     Wait(PAUSE);
334                     break;
335
336         case (4):   P1OUT |= YELLOW_LED;
337                     Wait(BLINK);
338                     P1OUT &= (~YELLOW_LED);
339                     Wait(PAUSE);
340                     break;
341         }
342
343         i = i + 1;
344
345     }
346     while (i <= round);
347 }
348
349 /*****************************************************************************
350 / Essa função espera o jogador aplicar uma resposta para depois julga-la.
351 /
352 / Isto é feito esperando o jogador aplicar uma resposta e depois verificando se
353 / se é a entrada certa para o elemento i da sequencia do array, em que i é o
354 / n-nesimo LED mostrado.
355 *****************************************************************************/
356 int GetAnswer(int sequence[16], int round)
357 {
358     int i = 0;
359     int game_over = 0;
360     // Permanece no loop até que uma resposta errada seja emitida.
361     while (i <= round && game_over == 0)
362     {
363     //    Wait(HALF_SEC);
364         if ((P2IN & RED_BTN) == 0)
365         {
366             P1OUT |= RED_LED;
```

```
367             if (sequence[i] == 0)
368                 i++;
369             else
370                 game_over = 1;
371             Wait(QUART_SEC);
372             P1OUT &= (~RED_LED);
373         }
374         if ((P2IN & GREEN_BTN) == 0)
375         {
376             P1OUT |= GREEN_LED;
377             if (sequence[i] == 1)
378                 i++;
379             else
380                 game_over = 1;
381             Wait(QUART_SEC);
382             P1OUT &= (~GREEN_LED);
383         }
384         if ((P2IN & BLUE_BTN) == 0)
385         {
386             P1OUT |= BLUE_LED;
387             if (sequence[i] == 3)
388                 i++;
389             else
390                 game_over = 1;
391             Wait(QUART_SEC);
392             P1OUT &= (~BLUE_LED);
393         }
394         if ((P2IN & YELLOW_BTN) == 0)
395         {
396             P1OUT |= YELLOW_LED;
397             if (sequence[i] == 4)
398                 i++;
399             else
400                 game_over = 1;
401             Wait(QUART_SEC);
402             P1OUT &= (~YELLOW_LED);
403         }
404     }
405     // Se a resposta for errada, envia um sinal para a função main() para mostrar
406     // Loss().
407     if (game_over == 0)
408         return 1;
409     // Caso contrário, envia um sinal para a função main() para indicar que a
410     // resposta foi correta e continuar.
411     else
412         return 0;
413 }
414
415
416 /*****************************************************************************
417 /   Essa função pisca o LED verde 8 vezes bem rapido indicando que a sequencia
418 /   inserida é correta.
419 *****************************************************************************/
420 void CorrectAnswer(void)
421 {
422     int i;
423     for (i = 0; i < 8; i++)
424     {
425         P2OUT = (P2OUT ^ GREEN_LED);
426         Wait(CENTI_SEC);
427     }
428 }
```

```
429
430 /*************************************************************************
431 /   Essa função faz com que os LEDs pisquem rapido indicando a vitória
432 *************************************************************************/
433 void Win(void)
434 {
435     int i;
436     for (i = 0; i < 3; i++)
437     {
438         P1OUT |= RED_LED;
439         Wait(CENTI_SEC);
440         P1OUT &= (~RED_LED);
441         Wait(CENTI_SEC);
442         P1OUT |= GREEN_LED;
443         Wait(CENTI_SEC);
444         P1OUT &= (~GREEN_LED);
445         Wait(CENTI_SEC);
446         P1OUT |= BLUE_LED;
447         Wait(CENTI_SEC);
448         P1OUT &= (~BLUE_LED);
449         Wait(CENTI_SEC);
450         P1OUT |= YELLOW_LED;
451         Wait(CENTI_SEC);
452         P1OUT &= (~YELLOW_LED);
453         Wait(CENTI_SEC);
454     }
455 }
456
457 /*************************************************************************
458 / Essa função faz com que o LED vermelho pisque 3 vezes devagar indicando a derrota
459 *************************************************************************/
460 void Loss(void)
461 {
462     int i;
463     for (i = 0; i < 3; i++)
464     {
465         P1OUT |= RED_LED;
466         Wait(QUART_SEC);
467         P1OUT &= (~RED_LED);
468         Wait(QUART_SEC);
469     }
470     Wait(ONE_SEC);
471 }
472
```