Atividade_04 - Livro AVR e Arduino – Técnicas de Projeto
Capítulo: 5 (Display de 7 segmentos)

Título: Fazendo um placar eletrônico com displays de 7 segmentos

Objetivos: O objetivo é construir um placar eletrônico com pontuação e tempo restante de jogo. Devido a limitação no número de terminais da placa, vamos multiplexar os displays no tempo. Assim, cada novo display necessita apenas de um pino adicional.

Nesta prática utilizaremos o Tinkercad para simular um circuito simples usando o microcontrolador Atmega328, utilizado nas placas Arduino UNO.

1. Procedimentos:

1. Acesse sua conta no Tinkercad (tinkercad.com) e vá para a aba circuits (https://www.tinkercad.com/circuits).

2. Crie um novo circuito com 7 displays de 7 segmentos e dois botões. **Utilize displays de anodo comum.** Utilize placas de ensaio (protoboards) para ligar os displays. Os displays devem ser divididos da seguinte forma:

- 2 dígitos para pontuação do time A;
- 2 dígitos para pontuação do time B;
- 3 dígitos para o tempo restante, sendo um dígito para minutos e dois para segundos.

Sugestão de organização: 00 0:00 00

Dica: Utilize as linhas horizontais externas de dois protoboards como um barramento para alimentar os dígitos dos displays. Dica2: olhe o exemplo feito em sala de aula.

RESTRIÇÃO DE PROJETO: Os segmentos dos displays (abcdefg) devem ser ligados nas portas PD4 a PD7 e PC0 a PC2, respectivamente. Este item é obrigatório e zera a avaliação caso não seja atendido.

3. Cada botão avança a pontuação de um time. A cada clique, a pontuação é incrementada em 1 ponto.

4. Não é necessário botão para reset do placar. O tempo inicial (tempo de jogo) pode ser definido em código. O cronômetro é regressivo e exibe até 9 minutos e 59 segundos. Ao fim do tempo, os botões não podem incrementar a pontuação.

5. Cole o código fonte do microcontrolador ao final deste arquivo e inclua a imagem de seu design. Importante: Deixe seu circuito na opção compartilhar link no Tinkercad e cole o link para ele aqui:

ATENÇÃO: Documente seu código. Cada linha/bloco deve deixar explícito o seu papel.

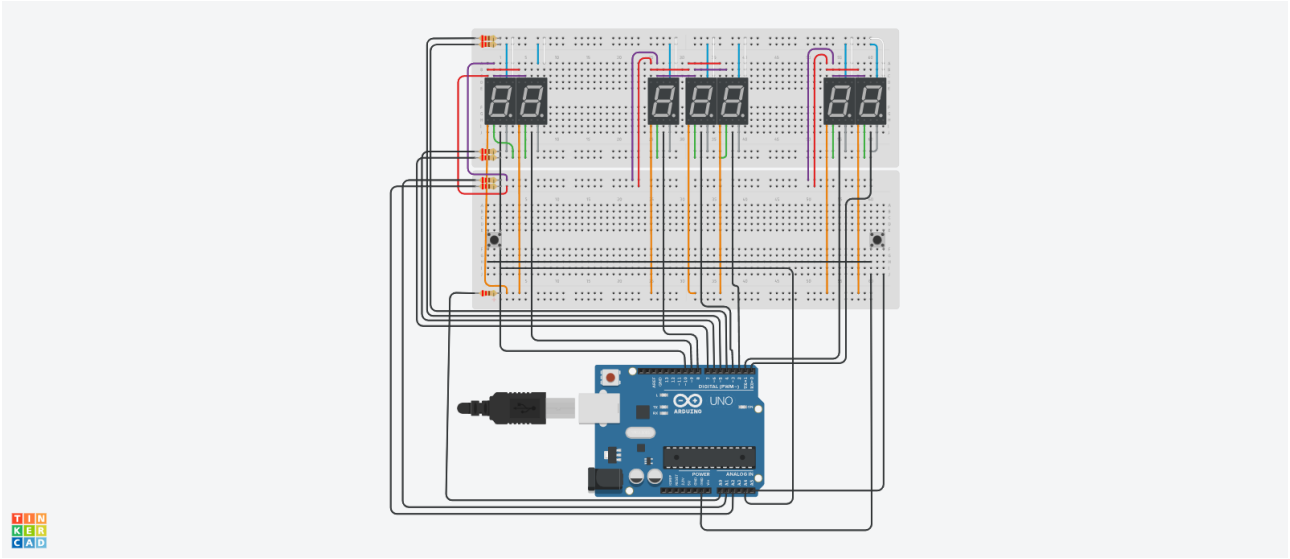
```
#define F_CPU 16000000UL        //define a frequência do microcontrolador em 16MHz

#include <avr/io.h>             //definições do componente especificado
#include <util/delay.h>         //biblioteca para o uso das rotinas de _delay_
#include <avr/pgmspace.h>       //biblioteca para poder gravar dados na memória flash

#define D_DISPLAY PORTD
#define C_DISPLAY PORTC

unsigned char disp;
```

```
const unsigned char Tabela[] PROGMEM = {0x40, 0x79, 0x24, 0x30, 0x19, 0x12, 0x02, 0x78,
0x00, 0x18, 0x08, 0x03, 0x46, 0x21, 0x06, 0x0E};

char digA0=0, digA1 = 0;
char digB0=0, digB1 = 0;

unsigned char valorA = 0;
unsigned char valorB = 0;

#define debounceInterval 1 // ms
unsigned int debounceTimeA = 0;
unsigned int debounceTimeB = 0;

int lastStateA = 1; // Ultima leitura do ruido
int lastStateB = 1; // Ultima leitura do ruido

int btnA = 1;   // Estado do botao
int btnB = 1;   // Estado do botao

long timer, tempo = 90000;
int min, seg1, seg0, btt, teamA = 0, teamB = 0;

//--------------------------------------------------------------------------------

void setup(){
        UCSR0B = 0x00;

        DDRD = 0b11111111;       // Definindo PD0, PD1, PD2, PD3 como saidas
        PORTD = 0b11110000;

        DDRC = 0b00000111;       // Definindo PC0, PC1, PC2 como saidas
        PORTC = 0b00000111;

        DDRB = 0b00011100;       // Definindo PB2, PB3, PB4 como saidas
        PORTB = 0b00000011;

        if(tempo > 599000){
                tempo = 599000;
        }

}

void loop(){

        // Calculo do tempo
        timer = tempo - millis();
        if(timer > 0){

                // ---------- TRATANDO DOS DISPLAYS DO MEIO ----------
                min = timer/60000;
        seg1 = (timer%60000)/10000;
        seg0 = (timer%10000)/1000;

                // ---------- TRATANDO DOS DISPLAYS DA ESQUERDA ----------
                // Se o botão A for pressionado, incrementa o valor da variável valorA
                int leituraA = PINB & (1<<PB1);

                if(leituraA != lastStateA){
                        debounceTimeA = millis();
```

```
        }

        if((millis() - debounceTimeA) > debounceInterval){
                if(btnA != leituraA){
                        btnA = leituraA;
                        if(btnA == 0){
                                if(valorA == 99){
                                        valorA = 0;
                                } else{
                                        valorA++;
                                }
                                digA0 = valorA % 10;
                                digA1 = valorA / 10;
                        }
                }
        }
        lastStateA = leituraA;


        // ---------- TRATANDO DOS DISPLAYS DA DIREITA ----------
        // Se o botão B for pressionado, incrementa o valor da variável valorB
        int leituraB = PINB & (1<<PB0);

        if(leituraB != lastStateB){
                debounceTimeB = millis();
        }

        if((millis() - debounceTimeB) > debounceInterval){
                if(btnB != leituraB){
                        btnB = leituraB;
                        if(btnB == 0){
                                if(valorB == 99){
                                        valorB = 0;
                                } else{
                                        valorB++;
                                }
                                digB0 = valorB % 10;
                                digB1 = valorB / 10;
                        }
                }
        }
        lastStateB = leituraB;

}

// DISPLAYS DO TIMER
// Ativa o seg0
PORTD |= (1 << PD2);
disp = pgm_read_byte(&Tabela[seg0]);
D_DISPLAY &= 0b00001111;
D_DISPLAY = (0b11110000 & (disp << 4));
C_DISPLAY &= 0b11111000;
C_DISPLAY = (0b00000111 & (disp >> 4));
_delay_ms(1);
PORTD &= ~(1 << PD2);

// Ativa o seg1
PORTD |= (1 << PD3);
disp = pgm_read_byte(&Tabela[seg1]);
```

```
D_DISPLAY &= 0b00001111;
D_DISPLAY = (0b11110000 & (disp << 4));
C_DISPLAY &= 0b11111000;
C_DISPLAY = (0b00000111 & (disp >> 4));
_delay_ms(1);
PORTD &= ~(1 << PD3);

// Ativa o min
PORTB |= (1 << PB2);
disp = pgm_read_byte(&Tabela[min]);
D_DISPLAY &= 0b00001111;
D_DISPLAY = (0b11110000 & (disp << 4));
C_DISPLAY &= 0b11111000;
C_DISPLAY = (0b00000111 & (disp >> 4));
_delay_ms(1);
PORTB &= ~(1 << PB2);


// DISPLAYS DO TIME A
// Ativa a unidade
PORTB |= (1<<PB3);
disp = pgm_read_byte(&Tabela[digA0]);
D_DISPLAY &= 0b00001111;        // Limpa o display
D_DISPLAY |= (0b11110000 & (disp << 4));      // Escreve o valor no display
C_DISPLAY &= 0b11111000;        // Limpa o display
C_DISPLAY |= (0b00000111 & (disp >> 4));      // Escreve o valor no display
_delay_ms(1);
PORTB &= ~(1<<PB3);

// Ativa a dezena
PORTB |= (1<<PB4);
disp = pgm_read_byte(&Tabela[digA1]);
D_DISPLAY &= 0b00001111;        // Limpa o display
D_DISPLAY |= (0b11110000 & (disp << 4));      // Escreve o valor no display
C_DISPLAY &= 0b11111000;        // Limpa o display
C_DISPLAY |= (0b00000111 & (disp >> 4));      // Escreve o valor no display
_delay_ms(1);
PORTB &= ~(1<<PB4);


// DISPLAYS DO TIME B
// Ativa a unidade
PORTD |= (1<<PD0);
disp = pgm_read_byte(&Tabela[digB0]);
D_DISPLAY &= 0b00001111;        // Limpa o display
D_DISPLAY |= (0b11110000 & (disp << 4));      // Escreve o valor no display
C_DISPLAY &= 0b11111000;        // Limpa o display
C_DISPLAY |= (0b00000111 & (disp >> 4));      // Escreve o valor no display
_delay_ms(1);
PORTD &= ~(1<<PD0);

// Ativa a dezena
PORTD |= (1<<PD1);
disp = pgm_read_byte(&Tabela[digB1]);
D_DISPLAY &= 0b00001111;        // Limpa o display
D_DISPLAY |= (0b11110000 & (disp << 4));      // Escreve o valor no display
C_DISPLAY &= 0b11111000;        // Limpa o display
C_DISPLAY |= (0b00000111 & (disp >> 4));      // Escreve o valor no display
_delay_ms(1);
```

```
        PORTD &= ~(1<<PD1);
}
```



RÚBRICA:
Circuito: 25%
Lógica da programação e funcionamento: 75%

Valor desta atividade na média: 1.0