Atividade_05 - Livro AVR e Arduino – Técnicas de Projeto
Capítulo: 5 (Acionando LCDs 16x2)

Título: Fazendo um placar eletrônico com display LCD 16x2

Objetivos: O objetivo é construir um placar eletrônico com pontuação e tempo restante de jogo usando um display LCD.

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 um display LCD e dois botões. Utilize a seguinte configuração para exibir as informações:
- 2 dígitos para pontuação do time A;
- 2 dígitos para pontuação do time B;
- 5 dígitos para o tempo restante, sendo um dígito para minutos, dois para segundos e dois para centésimos.

Sugestão de organização: 00 0:00.00 00

Utilize os projetos disponíveis na aula sobre LCDs como exemplo.

RESTRIÇÃO DE PROJETO: A interface de dados do display LCD deve ser ligado todo na porta C, incluindo os pinos de Enable e Register Select. 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.

DICA: utilize o millis para implementar seu contador de centésimos. Não tente atualizar o display muito rápido, isto provavelmente vai deixar a simulação lenta.

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

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

LINK TINKERCAD: https://www.tinkercad.com/things/jsikOFjlLc2-copy-of-lcd4bitsidentnum2021/editel?sharecode=FvUy927Mkl-rt_JpVpWJcuKhMR0cVoPerRLbWPl4ecM

CÓDIGO:

```
#ifndef _DEF_PRINCIPAIS_H
#define _DEF_PRINCIPAIS_H

#define F_CPU 16000000UL        //define a frequencia do microcontrolador - 16MHz

#include <avr/io.h>             //definições do componente especificado
#include <util/delay.h>         //biblioteca para o uso das rotinas de _delay_ms e _delay_us()
#include <avr/pgmspace.h>       //para o uso do PROGMEM, gravação de dados na memória flash
```

```
//Definições de macros para o trabalho com bits
#define set_bit(y,bit)  (y|=(1<<bit))        //coloca em 1 o bit x da variável Y
#define clr_bit(y,bit)  (y&=~(1<<bit))       //coloca em 0 o bit x da variável Y
#define cpl_bit(y,bit)  (y^=(1<<bit))        //troca o estado lógico do bit x da variável Y
#define tst_bit(y,bit)  (y&(1<<bit))         //retorna 0 ou 1 conforme leitura do bit

#endif


#ifndef _LCD_H
#define _LCD_H


//Definições para facilitar a troca dos pinos do hardware e facilitar a re-programação
#define DADOS_LCD       PORTC               //4 bits de dados do LCD no PORTD

#define nibble_dados    0                   //0 para via de dados do LCD nos 4 LSBs do PORT empregado (Px0-D4,
                                                    Px1-D5, Px2-D6, Px3-D7)
                                            //1 para via de dados do LCD nos 4 MSBs do PORT empregado (Px4-D4,
                                                    Px5-D5, Px6-D6, Px7-D7)

#define CONTR_LCD       PORTC               //PORT com os pinos de controle do LCD (pino R/W em 0).
#define E               PC4                 //pino de habilitação do LCD (enable)
#define RS              PC5                 //pino para informar se o dado é uma instrução ou caractere


//sinal de habilitação para o LCD
#define pulso_enable()  _delay_us(1); set_bit(CONTR_LCD,E); _delay_us(1); clr_bit(CONTR_LCD,E); _delay_us(45)

//protótipo das funções
void cmd_LCD(unsigned char c, char cd);
void inic_LCD_4bits();

#endif

//---------------------------------------------------------------------------------------
// Sub-rotina para enviar caracteres e comandos ao LCD com via de dados de 4 bits
//---------------------------------------------------------------------------------------
void cmd_LCD(unsigned char c, char cd){   //c = caractere ou comando, cd = 0 para comando, 1 para caractere
        if(cd==0)
        clr_bit(CONTR_LCD,RS);
        else
        set_bit(CONTR_LCD,RS);

        //primeiro nibble de dados - 4 MSB
        #if (nibble_dados)                  //compila código para os pinos de dados do LCD nos 4 MSB do PORT
        DADOS_LCD = (DADOS_LCD & 0x0F)|(0xF0 & c);
        #else                               //compila código para os pinos de dados do LCD nos 4 LSB do PORT
        DADOS_LCD = (DADOS_LCD & 0xF0)|(c>>4);
        #endif

        pulso_enable();

        //segundo nibble de dados - 4 LSB
        #if (nibble_dados)          //compila código para os pinos de dados do LCD nos 4 MSB do PORT
        DADOS_LCD = (DADOS_LCD & 0x0F) | (0xF0 & (c<<4));
        #else                       //compila código para os pinos de dados do LCD nos 4 LSB do PORT
        DADOS_LCD = (DADOS_LCD & 0xF0) | (0x0F & c);
        #endif

        pulso_enable();

        if((cd==0) && (c<4))      //se for instrução de retorno ou limpeza espera LCD estar pronto
        _delay_ms(2);
}

//---------------------------------------------------------------------------------------
//Sub-rotina para inicialização do LCD com via de dados de 4 bits
//---------------------------------------------------------------------------------------
void inic_LCD_4bits(){      //sequência ditada pelo fabricando do circuito integrado HD44780
                            //o LCD será só escrito. Então, R/W é sempre zero.

        clr_bit(CONTR_LCD,RS);  //RS em zero indicando que o dado para o LCD será uma instrução
```

```
        clr_bit(CONTR_LCD,E);    //pino de habilitação em zero

        _delay_ms(20);           //tempo para estabilizar a tensão do LCD, após VCC ultrapassar 4.5 V (na prática
                                   pode ser maior).

        //interface de 8 bits
        #if (nibble_dados)
        DADOS_LCD = (DADOS_LCD & 0x0F) | 0x30;
        #else
        DADOS_LCD = (DADOS_LCD & 0xF0) | 0x03;
        #endif

        pulso_enable();          //habilitação respeitando os tempos de resposta do LCD
        _delay_ms(5);
        pulso_enable();
        _delay_us(200);
        pulso_enable();
        /*até aqui ainda é uma interface de 8 bits.
        Muitos programadores desprezam os comandos acima, respeitando apenas o tempo de
        estabilização da tensão (geralmente funciona). Se o LCD não for inicializado primeiro no
        modo de 8 bits, haverá problemas se o microcontrolador for inicializado e o display já o tiver sido.*/

        //interface de 4 bits, deve ser enviado duas vezes (a outra está abaixo)
        #if (nibble_dados)
        DADOS_LCD = (DADOS_LCD & 0x0F) | 0x20;
        #else
        DADOS_LCD = (DADOS_LCD & 0xF0) | 0x02;
        #endif

        pulso_enable();
        cmd_LCD(0x28,0);    //interface de 4 bits 2 linhas (aqui se habilita as 2 linhas)
                            //são enviados os 2 nibbles (0x2 e 0x8)
        cmd_LCD(0x08,0);    //desliga o display
        cmd_LCD(0x01,0);    //limpa todo o display
        cmd_LCD(0x0C,0);    //mensagem aparente cursor inativo não piscando
        cmd_LCD(0x80,0);    //inicializa cursor na primeira posição a esquerda - 1a linha
}


// Variaveis Globais
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 = 599000;
int min, seg1, seg0, ms1, ms0;

//--------------------------------------------------------------------------------
void setup(){
        DDRC = 0xFF;                    //porta C como saida
        DDRD = 0b00111111;              //porta D como saida
        PORTD = 0b11000000;             //pull-up no botao


        inic_LCD_4bits();

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

void loop(){
```

```
// ################ Realizando calculos ################

// 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;
	ms1 = (timer%1000)/100;
	ms0 = (timer%100)/10;

	// Tratando dos digitos da esquerda
	int leituraA = PIND & (1<<PD6);

	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 digitos da direita
	int leituraB = PIND & (1<<PD7);

		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;

}

// ################ INSERINDO OS DADOS NO LCD ################################

// Pontos do time A
cmd_LCD(0x80 ,0);

cmd_LCD(digA1+'0',1);
cmd_LCD(digA0+'0',1);

// Temporizador
```

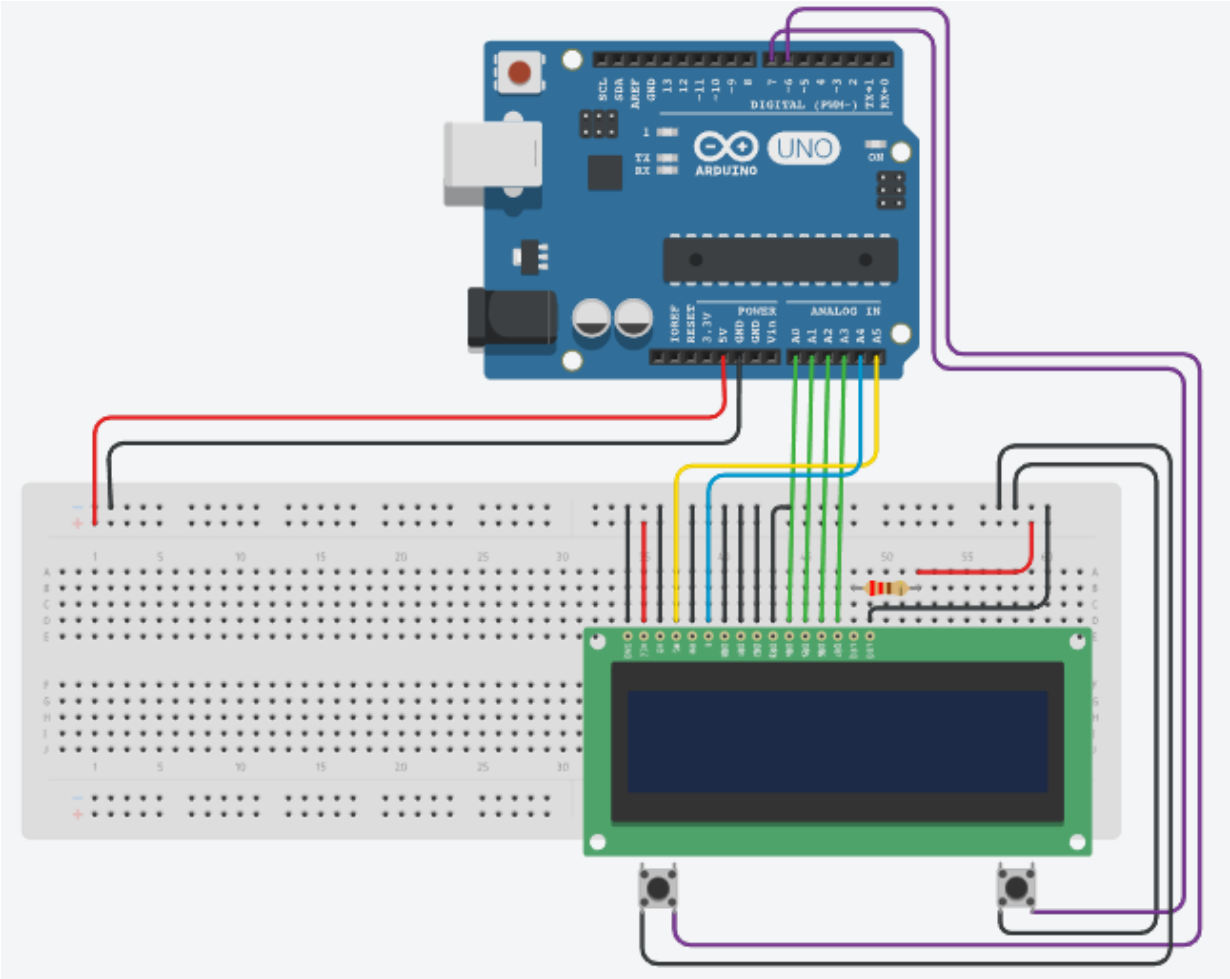
```
        cmd_LCD(0x84 ,0);

        cmd_LCD('0',1);
        cmd_LCD(min+'0',1);
        cmd_LCD(':',1);
        cmd_LCD(seg1+'0',1);
        cmd_LCD(seg0+'0',1);
        cmd_LCD(':',1);
        cmd_LCD(ms1+'0',1);
        cmd_LCD(ms0+'0',1);

        // Pontos do time B
        cmd_LCD(0x8E ,0);

        cmd_LCD(digB1+'0',1);
        cmd_LCD(digB0+'0',1);

}
```



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

Valor desta atividade na média: 0.8