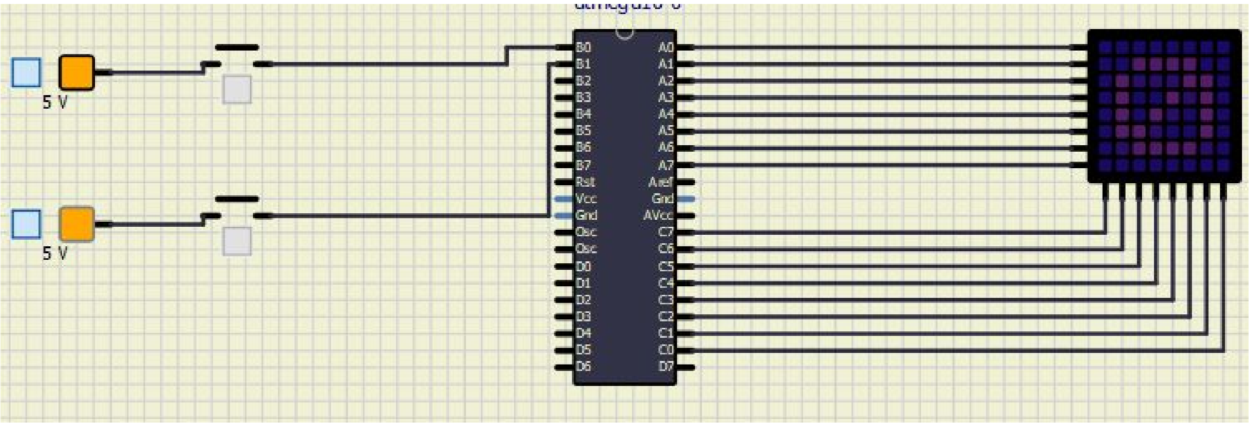
Projeto Intermediário - 2 - Controle de Leds por Contagem e Botões

Tutorial:

Componentes

2 Tensão Fixa 5v
2 Botão
1 Matriz de Led
1 Microcontrolador Atmega8-18

Simulação Software SimulIDE



Montagem do Projeto no SimulIDE

Conexões dos componentes:

Botão 1	Porta D1
Botão 2	Porta D2
Led 1	Porta A0
Led 2	Porta A1
Led 3	Porta A2
Led 4	Porta A3
Led 5	Porta A4
Led 6	Porta A5

Led 7	Porta A6
Led 8	Porta A7
Led 9	Porta C0
Led 10	Porta C1
Led 11	Porta C2
Led 12	Porta C3
Led 13	Porta C4
Led 14	Porta C5
Led 15	Porta C6
Led 16	Porta C7

<u>Programação em C Software CODEVision</u>

Bibliotecas utilizadas:

<stdio.h>
<delay.h>
<mega16.h>
<io.h>

```
/*******************************************************
This program was created by the CodeWizardAVR V3.43
Automatic Program Generator
© Copyright 1998-2021 Pavel Haiduc, HP InfoTech S.R.L.
http://www.hpinfotech.ro

Project :
Version :
Date    : 06/03/2021
Author  :
Company :
Comments:
```

```
Chip type               : ATmega16
Program type            : Application
AVR Core Clock frequency: 14,745600 MHz
Memory model            : Small
External RAM size       : 0
Data Stack size         : 256
****************************************************/
#define F_CPU 16000000UL;
#include <mega16.h>
#include <delay.h>
#include <io.h>

// Declare your global variables here
int cont;

// Standard Input/Output functions
#include <stdio.h>

char a_p[]={
        1,2,4,8,16,32,64,128
};

char a_n[][8]={
{0x00,0x3c,0x46,0x4a,0x52,0x62,0x3c,0x00},
{0x00,0x3c,0x46,0x4a,0x52,0x62,0x3c,0x00},
{0x00,0x3c,0x46,0x4a,0x52,0x62,0x3c,0x00}
};




int i =0,j=0, k=0;

void main(void)
{
// Declare your local variables here
cont = 0;


// Input/Output Ports initialization
// Port A initialization
// Function: Bit7=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out Bit1=Out Bit0=Out
```

```
DDRA=0XFF;
// State: Bit7=0 Bit6=0 Bit5=0 Bit4=0 Bit3=0 Bit2=0 Bit1=0 Bit0=0


// Port B initialization
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
DDRB=(0<<DDB7) | (0<<DDB6) | (0<<DDB5) | (0<<DDB4) | (0<<DDB3) | (0<<DDB2) |
(0<<DDB1) | (0<<DDB0);
// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
PORTB=(0<<PORTB7) | (0<<PORTB6) | (0<<PORTB5) | (0<<PORTB4) | (0<<PORTB3) |
(0<<PORTB2) | (0<<PORTB1) | (0<<PORTB0);

// Port C initialization
// Function: Bit7=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out Bit1=Out Bit0=Out
DDRC=0XFF;
// State: Bit7=0 Bit6=0 Bit5=0 Bit4=0 Bit3=0 Bit2=0 Bit1=0 Bit0=0


// Port D initialization
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
DDRD=(0<<DDD7) | (0<<DDD6) | (0<<DDD5) | (0<<DDD4) | (0<<DDD3) | (0<<DDD2) |
(0<<DDD1) | (0<<DDD0);
// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
PORTD=(0<<PORTD7) | (0<<PORTD6) | (0<<PORTD5) | (0<<PORTD4) | (0<<PORTD3) |
(0<<PORTD2) | (0<<PORTD1) | (0<<PORTD0);

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=0xFF
// OC0 output: Disconnected
TCCR0=(0<<WGM00) | (0<<COM01) | (0<<COM00) | (0<<WGM01) | (0<<CS02) | (0<<CS01) |
(0<<CS00);
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer1 Stopped
// Mode: Normal top=0xFFFF
// OC1A output: Disconnected
// OC1B output: Disconnected
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=(0<<COM1A1) | (0<<COM1A0) | (0<<COM1B1) | (0<<COM1B0) | (0<<WGM11) |
(0<<WGM10);
TCCR1B=(0<<ICNC1) | (0<<ICES1) | (0<<WGM13) | (0<<WGM12) | (0<<CS12) | (0<<CS11) |
(0<<CS10);
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer2 Stopped
// Mode: Normal top=0xFF
// OC2 output: Disconnected
ASSR=0<<AS2;
TCCR2=(0<<PWM2) | (0<<COM21) | (0<<COM20) | (0<<CTC2) | (0<<CS22) | (0<<CS21) |
(0<<CS20);
TCNT2=0x00;
OCR2=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=(0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) | (0<<OCIE1A) | (0<<OCIE1B) | (0<<TOIE1) |
(0<<OCIE0) | (0<<TOIE0);

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=(0<<ISC11) | (0<<ISC10) | (0<<ISC01) | (0<<ISC00);
MCUCSR=(0<<ISC2);

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud Rate: 9600
UCSRA=(0<<RXC) | (0<<TXC) | (0<<UDRE) | (0<<FE) | (0<<DOR) | (0<<UPE) | (0<<U2X) |
(0<<MPCM);
UCSRB=(0<<RXCIE) | (0<<TXCIE) | (0<<UDRIE) | (1<<RXEN) | (1<<TXEN) | (0<<UCSZ2) |
(0<<RXB8) | (0<<TXB8);
UCSRC=(1<<URSEL) | (0<<UMSEL) | (0<<UPM1) | (0<<UPM0) | (0<<USBS) | (1<<UCSZ1) |
(1<<UCSZ0) | (0<<UCPOL);
UBRRH=0x00;
UBRRL=0x5F;

// Analog Comparator initialization
// Analog Comparator: Off
// The Analog Comparator's positive input is
// connected to the AIN0 pin
// The Analog Comparator's negative input is
// connected to the AIN1 pin
ACSR=(1<<ACD) | (0<<ACBG) | (0<<ACO) | (0<<ACI) | (0<<ACIE) | (0<<ACIC) | (0<<ACIS1) |
(0<<ACIS0);
SFIOR=(0<<ACME);

// ADC initialization
// ADC disabled
ADCSRA=(0<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIF) | (0<<ADIE) | (0<<ADPS2) |
(0<<ADPS1) | (0<<ADPS0);

// SPI initialization
// SPI disabled
SPCR=(0<<SPIE) | (0<<SPE) | (0<<DORD) | (0<<MSTR) | (0<<CPOL) | (0<<CPHA) |
(0<<SPR1) | (0<<SPR0);

// TWI initialization
// TWI disabled
TWCR=(0<<TWEA) | (0<<TWSTA) | (0<<TWSTO) | (0<<TWEN) | (0<<TWIE);



while (1)


    {
    if (PINB.0)
```

```
{
    cont = cont + 1;
    printf("Numero de pessoas no interior: = %d.\r\n",cont);

};

delay_ms(500); // Aguarda 500 milisegundos

if (PINB.1)
{
  if(cont == 0)
  {
    printf("Ambiente vazio \r\n");
    }

    else
    {
   cont = cont - 1;
   printf("Numero de pessoas no interior: = %d.\r\n",cont);
   }


};

if(cont >= 3){



  for (k=0;k<120;k++)  //2*50=100
                {
                        PORTA=a_p[i];
                        PORTC=~a_n[j][i];
                        delay_ms(2);
                        i+=1;
                        if(i>8)
                        {
                                i=0;
                        }
                }
                j+=1;
                if(j>2)
                {
                        j=0;
```

```
                      i=0;
              }

              }



 }

}
```