IL Smart Traffic

Ícaro Leal da Cunha Lima
Larissa da Silva Lima

Versão
Terça, 1 de Dezembro de 2020

Documentação de Código

```
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
```

Definições e Macros

1 #define **F_CPU** 16000000UL
2 #define **BAUD** 9600
3 #define **MYUBRR** **F_CPU**/16/**BAUD**-1
4 #define **set_bit**(Y, bit_x) (Y|=(1<<bit_x))
5 #define **clr_bit**(Y, bit_x) (Y&=~(1<<bit_x))
6 #define **tst_bit**(Y, bit_x) (Y&(1<<bit_x))
7 #define **LED** PD5

Funções

8 void **USART_Init** (unsigned int ubrr)
Variavel que controla o tempo que o sinal vermelho permanece fechado.

9 void **USART_Transmit** (unsigned char data)
Envio de um frame de 5 a 8bits.

10 unsigned char **USART_Receive** (void)
Recepcao de um frame de 5 a 8bits.

11 **ISR** (USART_RX_vect)
12 **ISR** (INT0_vect)
13 **ISR** (ANALOG_COMP_vect)
14 void **SinalAberto** ()
15 void **FechaSinal** ()
16 void **SinalPedestre** ()
17 void **Barra** ()
18 int **main** (void)

Variáveis

19 uint8_t **i** = 0
20 uint8_t **botao** = 0
Variavel do laco.

21 int **tempo** = 3000
Variavel do botao.

Definições e macros

BAUD:main.cmain.c:BAUD#define BAUD 9600

clr_bit:main.cmain.c:clr_bit#define clr_bit(Y, bit_x) (Y&=~(1<<bit_x))

F_CPU:main.cmain.c:F_CPU#define F_CPU 16000000UL

LED:main.cmain.c:LED#define LED PD5

MYUBRR:main.cmain.c:MYUBRR#define MYUBRR F_CPU/16/BAUD-1

set_bit:main.cmain.c:set_bit#define set_bit(Y, bit_x) (Y|=(1<<bit_x))

tst_bit:main.cmain.c:tst_bit#define tst_bit(Y, bit_x) (Y&(1<<bit_x))

Funções

Barra:main.cmain.c:Barravoid Barra ()

Funcao que apaga sequencialmente a barra de LED enquanto durar o sinal vermelho

Os PORTB correspondem a posicao de cada LED da barra no circuito. 0 = LED apagado e 1 = LED aceso. O tempo dos delays eh controlado pelo valor enviado pelo usuario responsavel via monitor serial e sao divididos por 8, pois sao 8 LEDs na barra.

```
108             {
109
114     PORTC = 0b1111111;
115     PORTB = 0b00110000;
116     _delay_ms(tempo/8);
117     PORTC = 0b0111111;
118     _delay_ms(tempo/8);
119     PORTC = 0b0011111;
120     _delay_ms(tempo/8);
121     PORTC = 0b0001111;
122     _delay_ms(tempo/8);
123     PORTC = 0b0000111;
124     _delay_ms(tempo/8);
125     PORTC = 0b0000011;
126     _delay_ms(tempo/8);
127     PORTC = 0b0000001;
128     _delay_ms(tempo/8);
129     PORTC = 0b0000000;
130     _delay_ms(tempo/8);
131     PORTB = 0b00000000;
132     _delay_ms(tempo/8);
133 }
```

FechaSinal:main.cmain.c:FechaSinalvoid FechaSinal ()

Fecha o semaforo de veiculos e abre o de pedestres

```
85                  {
86      PORTB = 0b00000000; //Apaga sinal verde
87
88      PORTB = 0b00000100; //Acende sinal amarelo
89
90      SinalPedestre(); //Aciona o sinal para pedestres
91
92      PORTB = 0b00000000; //Apaga sinal amarelo
93
94      PORTB = 0b00010000; //Acende sinal vermelho
95      Barra(); //Aciona o funcionamento da barra de LEDs
96  }
```

ISR:main.cmain.c:ISRISR (ANALOG_COMP_vect)

Interrupcao do Comparador Analogico

```
70                      {
71
72      if(tst_bit(ACSR,ACO))//Verifica qual mudanca ocorreu na saida do comparador
73          set_bit(PORTD,LED);//O LED ficara ligado quando houver pouca luz (tensao
maior no terminal positivo do comparador)
74      else
75          clr_bit(PORTD,LED);//O LED ficara desligado quando houver  muita luz
(tensao menorr no terminal positivo do comparador)
76  }
```

ISR:main.cmain.c:ISRISR (INT0_vect)

Interrupcao por meio do botao fisico para pedestres

```
64              {
65      botao ^=1; //Inverte o estado do botao quando ele e pressionado
66      FechaSinal(); //Chama a funcao que faz o sinal fechar
67  }
```

ISR:main.cmain.c:ISRISR (USART_RX_vect)

```
47                  {
48      char recebido; //char que eh enviada pelo usuario no monitor serial
49      recebido = UDR0; //char armazenada no registrador UDR0
50
51      if(recebido == 'a'){ //ajusta o tempo do sinal vermelho para um maior
intervalo de tempo ('a'umenta)
52          tempo = 5000;
53      }
54      if(recebido == 'p'){ //ajusta o tempo do sinal vermelho para o intervalo de
tempo padrao ('p'adrao)
55          tempo = 3000;
56      }
57      if(recebido == 'd'){ //ajusta o tempo do sinal vermelho para um menor
intervalo de tempo ('d'iminui)
58          tempo = 1000;
59      }
60      USART_Transmit(recebido);
61  }
```

main:main.cmain.c:mainint main (void)

```
135             {
136
137      /* GPIO */
138      DDRB = ~(0); //Define todas as portas B como saida
139      DDRC = ~(0); //Define todas as portas C como saida
```

```
140
141     /* Interrupcao, PWM e Comparador Analogico*/
142     DDRD = 0b00101000; //Define a PD2 como entrada para o botao que gera a
interrupcao e PD5 e PD6 como saidas para o PWM gerado pelo Timer0
143     PORTD = 0b11011111;//Habilita o pull-up das portas D
144
145     EICRA= 0b00000010; //Interrupcao externa INT0 na borda de descida
146     EIMSK= 0b00000001; //Habilita a interrupcao externa INT0
147
148     /* TIMER */
149     TCCR0A = 0b10100011; //PWM nao invertido nos pinos OC0A e OC0B
150     TCCR0B = 0b00000011; //liga TC0, prescaler = 64, fpwm =
f0sc/(256*prescaler) = 16MHz/(256*64) = 976Hz
151     OCR0A = 0; //registrador do Timer0
152     OCR0B = 0; //registrador do Timer0
153
154     /*Comparador Analogico*/
155     DIDR1 = 0b00000011; //desabilita as entradas digitais nos pinos AIN0 e AIN1
156     ACSR = 1<<ACIE; //habilita interrup. por mudanca de estado na saida do
comparador
157
158     sei();//habilita todas as interrupcoes
159
160     /* USART */
161     USART_Init(MYUBRR);//Inicializa a USART
162
163     while (1){
164         SinalAberto(); //inicializa o programa com o sinal aberto para os
veiculos e fechado para os pedestres
165     }
166 }
```

SinalAberto:main.cmain.c:SinalAbertovoid SinalAberto ()

Abre o sinal para motoristas e fecha para pedestres

```
79                  {
80     PORTB = 0b00000010; //Sinal verde
81     OCR2B = 255; //Pedestre vermelho ligado
82     OCR2A = 0;   //Pedestre verde desligado
83 }
```

SinalPedestre:main.cmain.c:SinalPedestrevoid SinalPedestre ()

Muda gradativamente o estado do semaforo para pedestres, de vermelho para verde

```
98                   {
99     for(i=0;i<25;i++){ //Laco finito para executar a oscilacao do PWM ate que um
LED brilhe ate 250 e o outro apague totalmente
100         OCR2B = OCR2B - 10; //Decrementa em 10 unidades o valor do LED
vermelho, para que ele apague gradativamente
101         OCR2A = OCR2A + 10; //Incrementa em 10 unidades o valor do LED verde,
para que ele acenda gradativamente
102         _delay_ms(100); //Tempo de espera entre as iteracoes do laco
103     }
104     OCR2B = 0; //Seta o valor do PWM do sinal verde para 0, redundancia para
garantir o bom funcionamento
105     OCR2A = 255; //Seta o valor do PWM do sinal vermelho para 255, redundancia
para garantir o bom funcionamento
106 }
```

USART_Init:main.cmain.c:USART_Initvoid USART_Init (unsigned int *ubrr*)

Variavel que controla o tempo que o sinal vermelho permanece fechado.

```
24                                  {// Inicializacao da USART
25     UBRR0H = (unsigned char)(ubrr>>8); //Ajusta a taxa de transmissao
26     UBRR0L = (unsigned char)ubrr;
27     UCSR0B = (1<<RXCIE0)|(1<<RXEN0)|(1<<TXEN0); //Habilita o transmissor e o
receptor
28     UCSR0C = (1<<USBS0)|(3<<UCSZ00); //Ajusta o formato do frame: 8 bits de
dados e 2 de parada
29
30     DDRC = 0xFF; //Define a porta C como saida
31 }
```

USART_Receive:main.cmain.c:USART_Receiveunsigned char USART_Receive (void)

Recepcao de um frame de 5 a 8bits.

```
42 {
43     while(!(UCSR0A & (1<<RXC0))); //Espera o dado ser recebido
44     return UDR0; //Le o dado recebido e retorna
45 }
```

USART_Transmit:main.cmain.c:USART_Transmitvoid USART_Transmit (unsigned char *data*)

Envio de um frame de 5 a 8bits.

```
35 {
36     while(!( UCSR0A & (1<<UDRE0)));//Espera a limpeza do registr. de transmissao
37     UDR0 = data; //Coloca o dado no registrador e o envia
38 }
```

Variáveis

botao:main.cmain.c:botaouint8_t botao = 0

Variavel do laco.

i:main.cmain.c:iuint8_t i = 0

tempo:main.cmain.c:tempoint tempo = 3000

Variavel do botao.