

## Referência do Arquivo main.c

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

### Definições e Macros

```
#define F_CPU 16000000UL
```

### Funções

```
void adc_init (void)
uint16_t adc_read (uint8_t canal)
void timer0_init ()
void timer2_init ()
void recebeuDisparo ()
void moverTanque (char comando)
int main (void)
ISR (TIMER2_OVF_vect)
```

### Variáveis

```
volatile uint8_t overflow_count = 0
uint8_t leds [] = {2,3,4}
int vidas = 3
uint8_t vivo = 1
volatile uint8_t adc_flag = 0
volatile uint16_t adc_value = 0
int valor = 0
```

### Definições e macros

#### ◆ F\_CPU

```
#define F_CPU 16000000UL
```

### Funções

#### ◆ adc\_init()

```
void adc_init ( void )
```

```
15 □ {
16     ADMUX |= (1 << REFS0);
17     ADCSRA |= (1 << ADEN) | (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0); // prescaler 128
18 }
```

#### ◆ adc\_read()

```
uint16_t adc_read ( uint8_t canal )
```

```
20 {
21     ADMUX = (ADMUX & 0XF0) | (canal & 0x0F);
22     ADCSRA |= (1 << ADSC);
23     while (ADCSRA & (1 << ADSC));
24     return ADC;
25 }
```

### Definições e Macros

F\_CPU

### Funções

adc\_init

adc\_read

timer0\_init

timer2\_init

recebeuDisparo

moverTanque

main

ISR

### Variáveis

overflow\_count

leds

vidas

vivo

adc\_flag

adc\_value

valor

### Definições e macros

F\_CPU

### Funções

adc\_init

adc\_read

ISR

main

moverTanque

recebeuDisparo

timer0\_init

timer2\_init

### Variáveis

adc\_flag

adc\_value

leds

overflow\_count

valor

vidas

vivo

### ◆ ISR()

```
ISR ( TIMER2_OVF_vect )
```

```
212 {  
213     overflow_count++;  
214     if (overflow_count >= 61)  
215     {  
216         PORTB ^= (1 << PB3); // Pisca o Laser  
217         overflow_count = 0;  
218     }  
219 }
```

### ◆ main()

```
int main ( void )
```

```
170 {  
171     //CI L293D  
172     DDRD |= (1<<5)|(1<<6); // PWM 1,2 e PWM 3,4 , respectivamente  
173     DDRD |= (1<<7); //PIN 3A  
174     DDRB |= (1<<0); //PIN 4A  
175     DDRB |= (1<<1); //PIN 2A  
176     DDRB |= (1<<2); //PIN 1A  
177  
178     //Os pinos PB2 e PD7 são responsáveis por girar o tanque para FRENTE  
179     //Os pinos PB1 e PD0 são responsáveis por girar o tanque para TRÁS  
180  
181     //Laser  
182     DDRB |= (1<<3);  
183  
184     //LEDs para vida  
185     DDRC |= (1<<4)|(1<<3)|(1<<2);  
186  
187     //LDR  
188     DDRC &= ~(1 << PC5);  
189  
190     PORTC |= (1<<4)|(1<<3)|(1<<2);  
191  
192     //Serial.begin(9600);  
193     Serial.begin(38400);  
194     timer0_init();  
195     timer2_init();  
196     adc_init();  
197  
198     while(vivo == 1){  
199  
200         valor = adc_read(5);  
201         if (valor >= 1023) {  
202             recebeuDisparo();  
203         }  
204         if(Serial.available()){ //Faz a verificação dos dados recebidos através do HC-05  
205             char comando_recebido = Serial.read();  
206             moverTanque(comando_recebido);  
207         }  
208     }  
209 }
```

### ◆ moverTanque()

```
void moverTanque ( char comando )
```

```
90         {
91     switch(comando){
92         case 'F': // frente
93             OCR0A = 255; //PD6
94             OCR0B = 255; //PD5
95             PORTB &= ~(1<<0); //Polaridade para trás
96             PORTB &= ~(1<<1); //Polaridade para trás
97             PORTD |= (1<<7); // Polaridade para frente
98             PORTB |= (1<<2); // Polaridade para frente
99             break;
100        case 'B': // ré
101            OCR0A = 255;
102            OCR0B = 255;
103            PORTD &= ~(1<<7); // Polaridade para frente
104            PORTB &= ~(1<<2); // Polaridade para frente
105            PORTB |= (1<<0); //Polaridade para trás
106            PORTB |= (1<<1); //Polaridade para trás
107            break;
108        case 'L': // esquerda
109            OCR0A = 0;
110            OCR0B = 255;
111            PORTD &= ~(1<<7); // Polaridade para frente
112            PORTB &= ~(1<<0); //Polaridade para trás
113            PORTB &= ~(1<<1); //Polaridade para trás
114            PORTB |= (1<<2); // Polaridade para frente
115            break;
116        case 'R': // direita
117            OCR0A = 255;
118            OCR0B = 0;
119            PORTB &= ~(1<<1); //Polaridade para trás
120            PORTB &= ~(1<<0); //Polaridade para trás
121            PORTB &= ~(1<<2); // Polaridade para frente
122            PORTD |= (1<<7); // Polaridade para frente
123            break;
124        case 'G': // diagonal sup esq
125            OCR0A = 127;
126            OCR0B = 255;
127            PORTB &= ~(1<<0); //Polaridade para trás
128            PORTB &= ~(1<<1); //Polaridade para trás
129            PORTD |= (1<<7); // Polaridade para frente
130            PORTB |= (1<<2); // Polaridade para frente
131            break;
132        case 'H': // diagonal sup dir
133            OCR0A = 255;
134            OCR0B = 127;
135            PORTB &= ~(1<<0); //Polaridade para trás
136            PORTB &= ~(1<<1); //Polaridade para trás
137            PORTD |= (1<<7); // Polaridade para frente
138            PORTB |= (1<<2); // Polaridade para frente
139            break;
140        case 'I': // diagonal inf esq
141            OCR0A = 127;
142            OCR0B = 255;
143            PORTB |= (1<<0); //Polaridade para trás
144            PORTB |= (1<<1); //Polaridade para trás
145            PORTD &= ~(1<<7); // Polaridade para frente
146            PORTB &= ~(1<<2); // Polaridade para frente
147            break;
148        case 'J': // diagonal inf dir
149            OCR0A = 255;
150            OCR0B = 127;
151            PORTB |= (1<<0);
152            PORTB |= (1<<1);
153            PORTD &= ~(1<<7);
154            PORTB &= ~(1<<2);
155            break;
156        case 'S':
157            OCR0A = 0;
158            OCR0B = 0;
159            PORTB &= ~(1<<0);
160            PORTB &= ~(1<<1);
161            PORTD &= ~(1<<7);
162            PORTB &= ~(1<<2);
163            break;
164    }
165 }
166 }
```

◆ recebeuDisparo()

```
void recebeuDisparo ( )
```

```
40         {
41     vidas--;
42     if(vidas!=0){
43         vivo = 0;
44         PORTC &= ~(1 << leds[vidas]);
45         //Realiza o 180º graus
46         OCR0A = 255;
47         OCR0B = 255;
48         PORTB &= ~(1<<2);
49         PORTB &= ~(1<<0);
50         PORTD |= (1<<7);
51         PORTB |= (1<<1);
52         _delay_ms(2000); // delay até realizar o giro de 180º
53         //Encerra o giro, desliga os motores e permanece inativo
54         PORTB &= ~(1<<3);
55         TIMSK2 &= ~(1 << TOIE2);
56         OCR0A = 0;
57         OCR0B = 0;
58         PORTB &= ~(1<<2);
59         PORTB &= ~(1<<1);
60         PORTB &= ~(1<<0);
61         PORTD &= ~(1<<7);
62         _delay_ms(3000);
63         vivo = 1;
64         TIMSK2 |= (1 << TOIE2);
65     }
66 } else{
67     //Todas as vidas foram encerradas, logo não pode funcionar mais
68     vivo = 0;
69     PORTB &= ~(1<<3);
70     TIMSK2 &= ~(1 << TOIE2);
71     PORTC &= ~(1 << leds[vidas]);
72     //Realiza o 180º graus
73     OCR0A = 255;
74     OCR0B = 255;
75     PORTB &= ~(1<<2);
76     PORTB &= ~(1<<0);
77     PORTD |= (1<<7);
78     PORTB |= (1<<1);
79     _delay_ms(2000); // delay até realizar o giro de 180º
80     //Encerra o giro, desliga os motores e permanece inativo
81     OCR0A = 0;
82     OCR0B = 0;
83     PORTB &= ~(1<<2);
84     PORTB &= ~(1<<1);
85     PORTB &= ~(1<<0);
86     PORTD &= ~(1<<7);
87 }
88 }
```

#### ◆ timer0\_init()

```
void timer0_init ( )
```

```
27         {
28     TCCR0A = (1 << WGM01) | (1 << WGM00);
29     TCCR0A |= (1 << COM0A1) | (1 << COM0B1);
30     TCCR0B = (1 << CS01) | (1 << CS00);
31 }
```

#### ◆ timer2\_init()

```
void timer2_init ( )
```

```
33         {
34     TCCR2A = 0x00;
35     TCCR2B = (1 << CS22) | (1 << CS21) | (1 << CS20);
36     TIMSK2 = (1 << TOIE2);
37     sei();
38 }
```

## Variáveis

#### ◆ adc\_flag

```
volatile uint8_t adc_flag = 0
```

◆ adc\_value

```
volatile uint16_t adc_value = 0
```

◆ leds

```
uint8_t leds[] = {2,3,4}
```

```
8 {2,3,4};
```

◆ overflow\_count

```
volatile uint8_t overflow_count = 0
```

◆ valor

```
int valor = 0
```

◆ vidas

```
int vidas = 3
```

◆ vivo

```
uint8_t vivo = 1
```