

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