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  }
```

ISR()

ISR (TIMER2_OVF_vect)

```
{
    overflow_count++;
    if (overflow_count >= 61)
    {
        PORTB ^= (1 << PB3); // Pisca o Laser
        overflow_count = 0;
    }
}
```

main()

int main (void)

```
            {
    //CI L293D
    DDRD |= (1<<5)|(1<<6); // PWM 1,2 e PWM 3,4 , respectivamente
    DDRD |= (1<<7); //PIN 3A
    DDRB |= (1<<0); //PIN 4A
    DDRB |= (1<<1); //PIN 2A
    DDRB |= (1<<2); //PIN 1A

  //Os pinos PB2 e PD7 são responsáveis por girar o tanque para FRENTE
    //Os pinos PB1 e PD0 são responsáveis por girar o tanque para TRÁS

    //Laser
    DDRB |= (1<<3);

    //LEDs para vida
    DDRC |= (1<<4)|(1<<3)|(1<<2);

    //LDR
    DDRC &= ~(1 << PC5);

    PORTC |= (1<<4)|(1<<3)|(1<<2);

  //Serial.begin(9600);
    Serial.begin(38400);
    timer0_init();
    timer2_init();
    adc_init();

    while(vivo == 1){

        valor = adc_read(5);
        if (valor >= 1023) {
            recebeuDisparo();
    }
    if(Serial.available()){ //Faz a verificação dos dados recebidos através do HC-05
      char comando_recebido = Serial.read();
      moverTanque(comando_recebido);
    }
    }
}
```

moverTanque()

`void moverTanque ( char comando )`

```
                            {

    switch(comando){
        case 'F': // frente
            OCR0A = 255; //PD6
            OCR0B = 255; //PD5
            PORTB &= ~(1<<0); //Polaridade para trás
            PORTB &= ~(1<<1); //Polaridade para trás
            PORTD |= (1<<7);// Polaridade para frente
            PORTB |= (1<<2);// Polaridade para frente
            break;
        case 'B': // ré
            OCR0A = 255;
            OCR0B = 255;
            PORTD &= ~(1<<7);// Polaridade para frente
            PORTB &= ~(1<<2);// Polaridade para frente
            PORTB |= (1<<0);  //Polaridade para trás
            PORTB |= (1<<1); //Polaridade para trás
            break;
        case 'L': // esquerda
            OCR0A = 0;
            OCR0B = 255;
            PORTD &= ~(1<<7);// Polaridade para frente
            PORTB &= ~(1<<0); //Polaridade para trás
            PORTB &= ~(1<<1); //Polaridade para trás
            PORTB |= (1<<2);// Polaridade para frente
            break;
        case 'R': // direita
            OCR0A = 255;
            OCR0B = 0;
            PORTB &= ~(1<<1); //Polaridade para trás
            PORTB &= ~(1<<0); //Polaridade para trás
            PORTB &= ~(1<<2);// Polaridade para frente
            PORTD |= (1<<7); // Polaridade para frente
            break;
        case 'G': // diagonal sup esq
            OCR0A = 127;
            OCR0B = 255;
            PORTB &= ~(1<<0); //Polaridade para trás
            PORTB &= ~(1<<1); //Polaridade para trás
            PORTD |= (1<<7); // Polaridade para frente
            PORTB |= (1<<2); // Polaridade para frente
            break;
        case 'H': // diagonal sup dir
            OCR0A = 255;
            OCR0B = 127;
            PORTB &= ~(1<<0); //Polaridade para trás
            PORTB &= ~(1<<1); //Polaridade para trás
            PORTD |= (1<<7); // Polaridade para frente
            PORTB |= (1<<2); // Polaridade para frente
            break;
        case 'I': // diagonal inf esq
            OCR0A = 127;
            OCR0B = 255;
            PORTB |= (1<<0); //Polaridade para trás
            PORTB |= (1<<1); //Polaridade para trás
            PORTD &= ~(1<<7); // Polaridade para frente
            PORTB &= ~(1<<2); // Polaridade para frente
            break;
        case 'J': // diagonal inf dir
            OCR0A = 255;
            OCR0B = 127;
            PORTB |= (1<<0);
            PORTB |= (1<<1);
            PORTD &= ~(1<<7);
            PORTB &= ~(1<<2);
            break;
        case 'S':
            OCR0A = 0;
            OCR0B = 0;
            PORTB &= ~(1<<0);
            PORTB &= ~(1<<1);
            PORTD &= ~(1<<7);
            PORTB &= ~(1<<2);
            break;
    }
}
```

◆ `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
```