# Tacômetro Óptico

Wilton Miro Barros Júnior
FGA - Faculdade do Gama
UnB - Universidade de Brasília
Gama, Brasil
wiltonjrfla@gmail.com

Igor de Alcantara Rabelo
FGA - Faculdade do Gama
UnB - Universidade de Brasília
Gama, Brasil
rabelo.alcantara.igor@gmail.com

***Resumo*—Este documento visa mostrar a elaboração do projeto de um tacômetro óptico com emissor e receptor infravermelho que será controlado pelo microcontrolador MSP430 e terá a visualização da medição em um display LCD.**

***Keywords—tacômetro, óptico, infravermelho, LCD***

## I. Introdução

Os motores elétricos que são capazes de converter energia elétrica em energia mecânica e são utilizados em diversas máquinas que usamos no dia-a-dia[4]. Algumas vezes é necessário fazer testes de medições para saber se realmente o motor está em sua rotação ideal e o tacômetro é um dispositivo que é usado para obter o número de rotação de um motor. O laboratório de eletricidade da UnB-Gama possui esses motores elétricos que são usados para o aprendizado desde o manuseio até as configurações que são descritas pelo fabricante.

Esse projeto consiste em fazer um tacômetro usando sensores LED emissor e receptor infravermelho para medir a rotação do motor do laboratório da UnB-Gama e identificar através de um display de LCD16x2, se a rotação está adequada de acordo com as ligações que são usadas para o funcionamento adequado e também para verificar se está de acordo com o que foi proposto pelo fabricante. Será usado um microcontrolador Msp430g2553.

## II. Desenvolvimento

Para saber se o motor está realmente conforme o que está proposto pelo fabricante, planeja criar um tacômetro óptico que é um dispositivo capaz de medir a rotação do motor através de sensores LED's emissor e receptor infravermelho que ao incidir um feixe em uma fita refletiva fixada no eixo do motor que será capturada pelo emissor e assim através do processamento do microcontrolador msp430g2553 será obtido um resultado que será informado pelo display de LCD 16x2.

### *A. Descrição de Hardware*

Para a realização deste projeto foi utilizados os seguinte materiais.

Tabela 1.Lista de materiais

| Lista de materiais | |
|---|---|
| Item | Quantidade |
| MSP430g2553 LaunchPad | 1 |
| LED receptor infravermelho | 1 |
| LED emissor infravermelho | 1 |
| Jumpers | 19 |
| Protoboard | 2 |
| Display LCD 16x2 | 1 |
| Potenciômetro 10k | 1 |
| Ferro de solda | 1 |
| Solda | 1 |

O hardware consiste em dois LED's emissor e receptor infravermelho que são ligados no microcontrolador Msp430g2553. Ao ligar o LED emissor, um feixe infravermelho é lançado e esse feixe é refletido através de uma fita e volta para o receptor. No código que foi feito para o receptor, o LED receptor

foi ligado no pino A0, que é equivalente ao pino P1.0 da placa Msp430.

O LCD foi testado usando um exemplo da biblioteca do software energia que já possuía as pinagens corretas de conectar o display no MSP430. O display contém 16 entradas e os pinos que foram utilizados do msp430 foram: P2.0, P2.1, P2.2, P2.3, P2.4, P2.5, GND e VCC. Para a função RS e EN foram utilizados os pinos P2.0 e P2.1 que corresponde às entradas do display 4 e 6, para função DB4 a DB7 foram utilizadas P2.2, P2.3, P2.4, P2.5 que corresponde às entradas dos display 11 a 14, e enfim para alimentar o display foram utilizadas GND e VCC que correspondem às entradas do display 1,2,5,15 e 16. O funcionamento está de acordo com o diagrama de blocos do ANEXO E.

### *B. Descrição de Software*

Foi desenvolvido um código em linguagem C para msp430 usando a biblioteca msp430g2553.h no code composer studio 8.0.0. Foram utilizados 4 bits do display de lcd(D4 a D7).Existem outros 4 bits da placa de lcd que é o R/W, EN, RS, Contrast, Gnd e Vcc. O R/W é responsável pelo fluxo de dados entre a placa e o microcontrolador. Isso é, quando o R/W está ligado no Vcc ele está lendo os dados do sensor no display e quando R/W está no Gnd ele está escrevendo no display. O pino contraste é utilizado para aumentar ou diminuir o contraste do display de lcd através de um potenciômetro. Os pinos A e K são utilizados para iluminação do display.

Devido a configuração do display lcd ser de 4 bits, é necessário enviar 4 bits em dois pacotes, sendo que os 4 bits mais significativos serão enviados primeiro e depois os 4 menos significativos. Com isso foi preciso realizar um tratamento para fazer o envio dos dois pacotes. Esse tratamento consistiu em realizar o deslocamento dos 4 bits mais significativos e depois os 4 bits menos significativos de forma que os bits ficassem disponíveis nas portas conectadas ao lcd para posterior envio.

Para realizar a exibição no display, foi conectado um botão na porta P1.3 da msp430g2553, que estava operando como entrada, que a informação de acionamento do botão. O registrador P1DIR deve estar habilitado em nível 0 para que a porta P1.3 seja considerada como entrada. Esses registradores são responsáveis pela habilitação de resistor pull-up/pull-down(P1REN), seleção de modo pull-up(P1OUT), habilitação da interrupção (P1IE), seleção do tipo de borda da interrupção(P1IES) e limpeza do flag de interrupção(P1IFG).O registrador de habilitação para interrupção é fundamental para que o funcionamento da interrupção seja adequado. Sem esse registrador a interrupção não é ativada mesmo que o botão de interrupção na porta P1.3 seja pressionado, a interrupção não funciona.

Após configurar os registradores da porta P1.3 e a interrupção, foi realizado o tratamento da interrupção gerada pelo acionamento do botão. Por último foi desenvolvido o código para transformar a variável int para char, porque o display lcd só aceita valores de uma variável char. Todos o código desenvolvido está descrito no apêndice C.

## III. Resultados

Após o realização do código, rodou-se o código na placa e observou-se aparecendo na primeira linha do LCD a palavra “tacometro” e na segunda linha, a palavra “digital”. A interrupção é habilitada quando o botão no pino 1.3 da placa Launchpad é pressionado e há uma contagem sobre quantas vezes o botão é pressionado.
Quando pressionado o botão, observa-se a tela limpa e depois aparece a frase “contador btn” na primeira linha e a palavra “pressionado” na segunda linha com a quantidades de vezes em caractere. Após a interrupção, volta a configuração original na tela do LCD.

## IV. Conclusão

Conclui-se que a parte do display de lcd do projeto do tacometro obteve resultado satisfatório após ter sido testado com o código em C usado no Code Composer 8.8.0. Foi possível usar a interrupção ao apertar o botão quando aparecia a mensagem “contador btn”. Esse botão causa a interrupção do programa. E a quantidade era mostrada no display lcd. E com isso também foi notado que foi possível desenvolver o código para a escrita de uma mensagem no display lcd.

## Referencias


[1] https://www.filipeflop.com/blog/controlando-um-lcd-16x2-com-arduino/

[2] http://www.instructables.com/id/Interfacing-16x2-LCD-with-msp430-launchpad-in-8-bi/

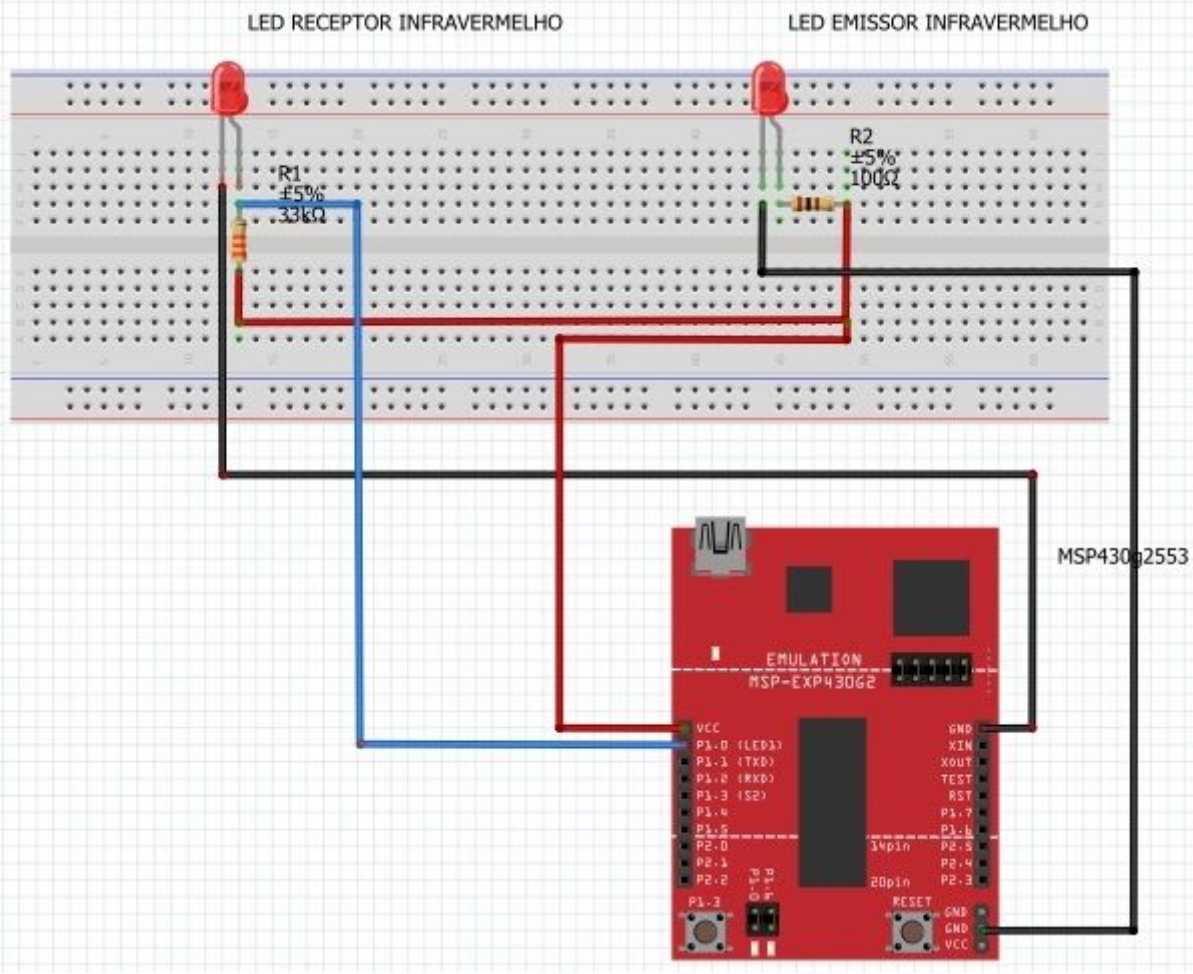
[3] https://www.circuitvalley.com/2011/12/16x2-char-lcd-with-ti-msp430-launch-pad.html

[4] https://brasilescola.uol.com.br/fisica/eletricidade-acionamento-motores-eletricos.htm

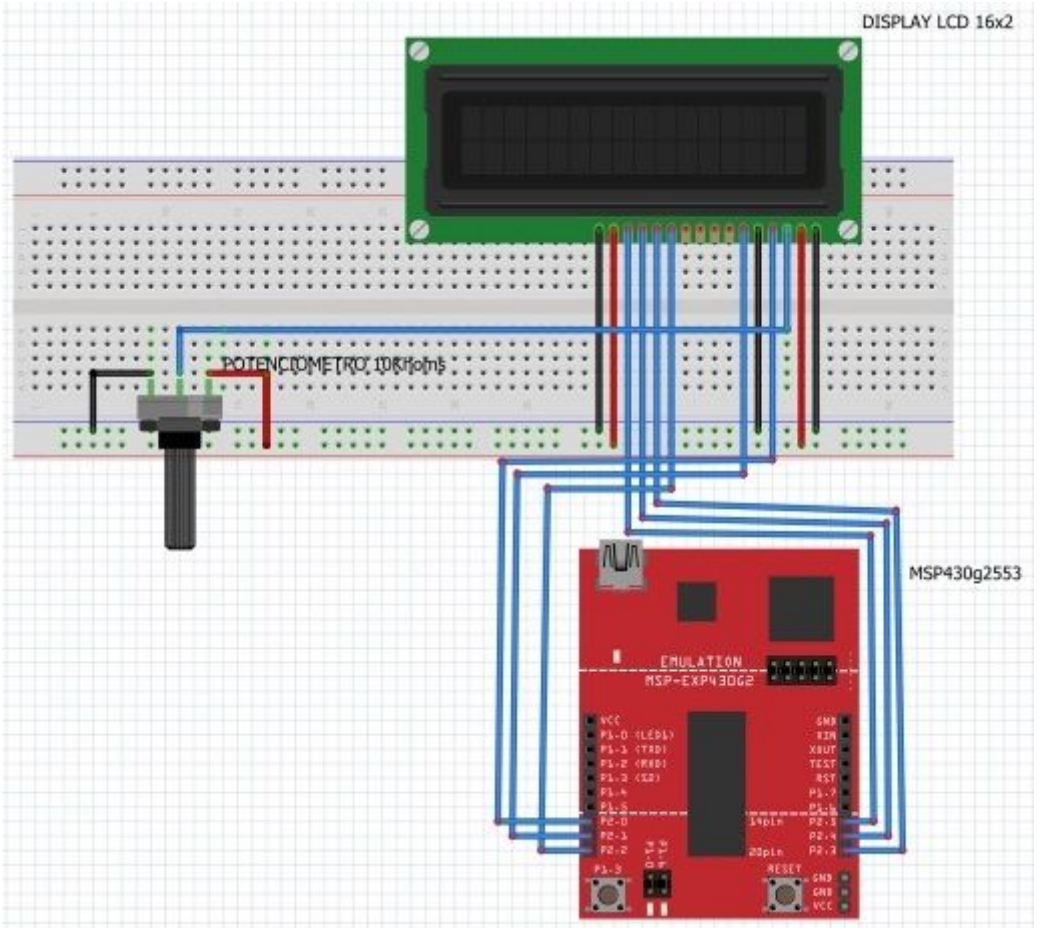
[5] Davies, J., MSP430 Microcontroller Basics, Elsevier, 2008

[6] MSP430 Assembly Language Tools v18.1.0.LTS - User’s Guide

ANEXO A - Esquemático emissor e receptor infravermelho



ANEXO B - Esquemático display LCD 16x2



# ANEXO C - Código do display LCD 16x2 em C

```
#include <msp430g2553.h>
//LCD control pin definitions
#define DATA_REG P1OUT=BIT5                    //SET RS HIGH (Registro de dados)
#define INST_REG P1OUT = (~BIT5)               //SET RS LOW (Registro de instruções)
#define ENABLE_PIN_HIGH P2OUT |= BIT0          //SET ENABLE HIGH
#define ENABLE_PIN_LOW P2OUT &= (~BIT0)        //SET ENABLE LOW

//VARIABLE DECLARATION

int counter = 0;
char char_counter[5];

//Implementação da função itoa (converte valores em strings para mostrar no LCD)

char *itoa(int from, char to[])
{
      char const digit[] = "0123456789";
      char* p = to;
      int shifter;

      if(from < 0 ){
            *p++ = '-';
            from *= -1;
}
shifter = from;

do{
      ++p;
      shifter = shifter/10;
}while(shifter);

*p = '\0';

do{
      *--p = digit[(from % 10)];
      from = from / 10;
} while(from);

 return to;
}
void configure_clocks()
{
      WDTCTL = WDTPW + WDTHOLD; //PARAR WATCHDOG PARA EVITAR A REINICIALIZAÇÃO DO
TEMPO
       DCOCTL = CALDCO_1MHZ;         //DCO = 1MHZ
      BCSCTL1 = CALBC1_1MHZ;         //BC1 = 1 MHZ
      BCSCTL1 = 0x00;                 //0
      BCSTL3   =  0x00;               //0
 }

  void delay_us(unsigned int us)
 }
      __delay_cycles(1);                  //FREQUENCIA 1MHZ = PERIODO 1us
      us--;
```

```
 }
}
 void delay_ms(unsigned int ms)
 {
     while(ms)
          {
          __delay_cycles(1000);      //FREQUENCIA DE 1 MHZ = PERIODO DE 1 us => (1 ms = 1000 us)
          ms--;
          {
  }
void data_write(void)
     {
          ENABLE_PIN_HIGH;
          delay_ms(5);
          ENABLE_PIN_LOW;
     }
void send_data(unsigned char data)
{
     unsigned char higher_nibble = 0x3c & (data >> 2);
     unsigned char lower_nibble = 0x3c & (data <<  2);

     delay_us(200);
     DATA_REG;

     P2OUT = (P2OUT & 0xc3) | (higher_nibble);     //ENVIAR PRIMEIRO 4 bits
     data_write();
     P2OUT = (P2OUT & 0xc3) | (higher_nibble);     //ENVIAR POR ÚLTIMO 4 bits
     data_write();
}

void send_string(char *s)
{
     while(*s)
     {
          send_data(*s);
          s++;
     }
}

void send_command(unsigned char cmd)
{
     unsigned char higher_nibble = 0x3C & (cmd >> 2);
     unsigned char lower_nibble = 0x3C  & (cmd << 2);

     INST_REG;
     P2OUT = (P2OUT & 0xC3) | (higher_nibble); //ENVIAR PRIMEIRO 4 bits
     data_write();

     P2OUT = (P2OUT & 0xC3) | (lower_nibble); //ENVIAR POR ÚLTIMO 4 bits
     data_write();
}
void lcd_init()
{
     P1DIR = 0x20;          //Set P1.5 COMO SAÍDA
     P1OUT = 0x00;          //Set P1 SAÍDAS BAIXAS
     P2DIR = 0x3D;          //Set P2.0 P2.2 P2.3 P2.4 P2.5 COMO SAIDA
     P2OUT = 0x00;          //Set P2 SAÍDAS BAIXAS
```

```
delay_ms(15);
send_command(0x33); //CÓDIGO DE INICIALIZAÇÃO

delay_ms(200);
send_command(0x32); //CÓDIGO DE INICIALIZAÇÃO

delay_ms(40);
send_command(0x28); //2 linhas e 5x7 matriz(4 bit mode)

delay_ms(40);
send_command(0x0E); //Display on, cursor on

delay_ms(40);
send_command(0x01); //Clear display screen

delay_ms(40);
send_command(0x06); //INCREMENTAR cursor (shift cursor para a direita)

delay_ms(40);
send_command(0x80); //Definir o cursor para começar na 1ª linha

int main(void)
{
    configure_clocks();             //start clock
    lcd_init();                     //start LCD

    send_string("TACOMETRO");       //ESCREVER NA PRIMEIRA LINHA
    send_command(0xC0);             //DEFINIR O CURSOR PARA O INÍCIO DA SEGUNDA LINHA
    send_string("DIGITAL")          //ESCREVER NA SEGUNDA LINHA
    send_command(0x0C);

    P1REN |= BIT3;                  //Enable pull-up/down resistor in P1.3
    P1OUT |= BIT3;                  //SELECIONAR pull-up MODO in P1.3
    P1IE |= BIT3;                   //Enable interrupt in P1.3
    P1IES = BIT3;                   //Hi-Lo edge in P1.3
    P1IFG &= ~BIT3;                 //Clear interrupt flag in P1.3

    __bis_SR_register(GIE);         //Enable global interrupt

    while(1)
    {}
}


#pragma vector = PORT1_VECTOR
__interrupt void Port_1(void){
counter++;                          //Contador incremental do botão pressionado
itoa(counter, char_counter);        //Converter contagem de inteiros para char char_count
send_command(0x01);                 //LIMPAR A TELA
send_command(0x80);                 //Definir o cursor para começar a primeira linha
send_string("Contador btn:");       //ESCREVER NA PRIMEIRA LINHA
send_command(0xC0);                 //Definir o cursor para o início da segunda linha
send_command("Pressionado: ");      //Escreva na segunda linha
send_string(char_counter);          //Escreva char_counter na segunda linha
send_string(" x");                  //Escreva na segunda linha
send_command(0x0C);                 //Exibir ligado, cursor desligado
__delay_cycles(50000);              //Delay
P1IFG &= ~BIT3;                     //Clear interrupt flag in P1.3
```

ANEXO D - Diagrama de blocos