UNIVERSIDADE DE BRASÍLIA - UnB
Faculdade do Gama - FGA
Eletrônica Embarcada (120871)

PONTO DE CONTROLE 3
Projeto: Mini cortador de grama controlado por MSP430 (Minigrama)

Débora Janini Campos Guedes
Matrícula: 15/0008619
E-mail: deborajaninicg@gmail.com

Maysa Paula Lacerda Cardoso
Matrícula:11/0133315
E-mail: maysa41@hotmail.com

RESUMO

Este documento apresenta a proposta para projeto final da disciplina de eletrônica embarcada, sendo este um mini cortador de grama microcontrolado. O projeto a ser desenvolvido visa principalmente na segurança do usuário, mas também no seu conforto.

1. JUSTIFICATIVA

O investimento na criação de robôs autônomos móveis vem do antigo desejo e, por vezes, da necessidade de reprodução de atitudes humanas por meio de máquinas. Equipamentos dessa natureza substituem trabalhadores em ambientes insalubres ou perigosos, poupando recursos humanos que podem ser aproveitados em áreas mais complexas. Pode-se adicionar à vantagem do uso desses robôs o fato de eles serem capazes de realizar tarefas de maneira mais precisa, além de trabalharem quase que ininterruptamente,desde que seja provida alimentação elétrica necessária durante todo o período.

Levando isso em consideração, o projeto proposto, que é um mini cortador de grama foi pensado a fim de promover comodidade e principalmente segurança dos usuários. Sabe-se que cortadores de grama convencionais podem gerar diversos acidentes com a lâmina do cortador ou até mesmo com descargas elétricas; dessa forma, com um cortador de grama microcontrolado o usuário pode manter uma distância segura do cortador garantindo assim a sua salvaguarda e além disso, o usuário pode utilizar o mini cortador de onde quiser, já que este não precisa da presença física do usuário, garantindo um pouco mais de comodidade a ele.

Para realizar tal projeto será utilizado o microprocessador MSP430G2553 da Texas Instruments , que pode ser visto na figura 1.

Figura 1: MSP430

Sabe-se que a MSP430 é um microcontrolador de 16bit com uma arquitetura Von Neumann e ele serve para aplicações de baixa potência, que é o caso do projeto proposto; ele então será utilizado para controlar, por meio das portas, todos os motores e sensores a serem usados na construção.

2. OBJETIVOS

O objetivo deste projeto é desenvolver um cortador de grama autônomo, utilizando um microcontrolador da família MSP430 da Texas Instruments que seja capaz de cortar grama em uma área fechada e plana com autonomia energética e que seja capaz de desviar de obstáculos, a fim de trazer aos usuários mais comodidade e segurança na hora de cortar a grama do que cortadores convencionais.

3. REQUISITOS

- Capacidade de desviar de obstáculos
  (Esse é um requisito importante no qual serão utilizados sensores para que seja necessária a mínima interferência humana possível)

- Velocidade mínima do Minigrama
  (Para um bom funcionamento do Minigrama é necessário uma velocidade mínima, e ao analisar o robô desenvolvido na referência 2, essa velocidade seria de 0.35 km/h)

- Tempo mínimo de autonomia
  (Novamente para se ter um bom funcionamento é preciso que o robô possa andar sozinho por um mínimo de tempo, que foi determinado como 15 minutos, além de carregamento relativamente rápido)

- Garantia de segurança mínima
  (Para realizar esta garantia é necessário o desligamento do motor para possíveis acidentes como exposição da lâmina e diminuição da rotação da lâmina próximo a obstáculos)

Além dos requisitos técnicos do projeto, tem-se também os requisitos básicos do protótipo do projeto em si, sendo eles:

- Sensores IR :esses sensores ópticos reflexivos serão responsáveis para identificação de objetos reflexivos, entre outros;
- Sensores Ultrassônicos estes sensores serão responsáveis por encontrar obstáculos e mensurá-los de maneira rápida e precisa;
- Módulo Ponte H: esse módulo será utilizado para controle dos sentidos dos motores para que o robô cortador possa se movimentar livremente de acordo com a área e obstáculos encontrados;
- Módulo Relé: ele será utilizado para controlar o motor responsavel pelo corte da grama;
- Módulo Buzzer : servirá de alerta sonoro para o usuário
- Motores para as rodas do robô: serão responsáveis pela movimentação adequada;
- Motor de corte: motor de alta rotação para realizar o corte da grama
- Bateria de lítio de 12V/3A: será a font energética.
- Chassi: a estrutura e sustento do protótipo, que será dimensionado de acordo com os componentes e forma de movimentação designado em software.

4. DESENVOLVIMENTO

O projeto final construído é composto tanto por hardware quanto por software que irá controlar todo o sistema por meio do microcontrolador MSP430G2553.

O hardware do protótipo consiste em sensores e microcontrolador como entrada e motores como saída, conforme o diagrama de blocos da figura 2.

.

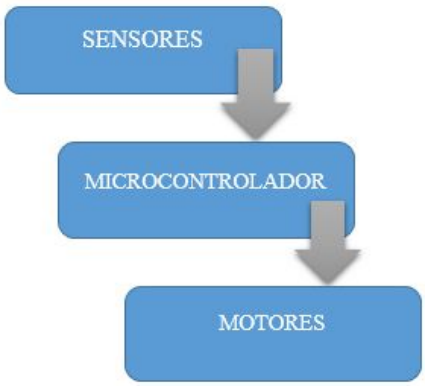

Figura 2: Diagrama de blocos do protótipo

4.1) Sensores

4.1.1) Sensor Ultrassônico

O sensor ultrassônico é muito utilizado para achar obstáculos e mensurá-los rapidamente com razoável precisão. No caso do projeto será usado o sensor ultrassônico HC-SR04, que é facilmente encontrado em lojas de componentes eletrônicos.

O sensor possui cerca de 15° de ângulo para detecção dos obstáculos e é capaz de medir distâncias de 2cm a 4m com ótima precisão. Ele possui um circuito pronto com emissor e receptor acoplados em 4 pinos (VCC, Trigger, ECHO e GND) para a devida medição. Na figura 3 tem-se uma imagem do módulo sensor com as suas pinagens.

Figura 3: Módulo sensor ultrassônico HC-SR04

Para começar a medição é necessário alimentar o módulo e colocar o pino Trigger em nível alto por mais de 10us. Assim, o sensor emitirá uma onda sonora que, ao encontrar um obstáculo, rebaterá de volta em direção ao módulo. Durante o tempo de emissão e recebimento do sinal, o pino ECHO ficará em nível alto. Logo, o cálculo da distância pode ser feito de acordo com o tempo em que o pino ECHO permaneceu em nível alto após o pino Trigger ter sido colocado em nível alto. O diagrama de tempos pode ser visto na figura 4.

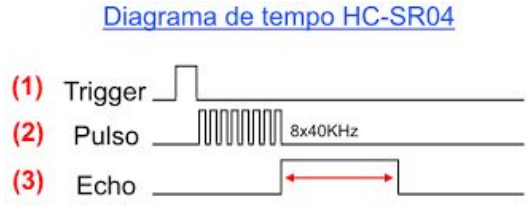

Figura 4: Diagrama de tempo do HC-SR04

Conforme o datasheet do sensor tem-se que a distância em metros é dado por:

$$d = \frac{Tempo\ de\ resposta\ do\ sensor * Velocidade\ do\ som\ (340\ m/s)}{2}$$

Equação 1: distância do sensor em função do eco

A velocidade do som poder ser considerada idealmente igual a 340 m/s, logo o resultado é obtido em metros se considerado o tempo em segundos. Na fórmula, a divisão por 2 deve-se ao fato de que a onda é enviada e rebatida, ou seja, ela percorre 2 vezes a distância procurada.

Funciona com uma alimentação de 5V DC, opera com corrente de 2mA, tem um ângulo de efeito de 15 graus, possui um alcance entre 2 cm e 4m e a sua precisão é de 3mm.

Para o projeto serão implementados 4 sensores ultrassônicos juntamente com um demux para o echo do sensor a fim de poupar memória e pinos do microcontrolador. Na

figura abaixo mostra-se um esquemático semelhante ao que será montado para esses sensores, os pinos podem ser alterados futuramente com o progresso do projeto.

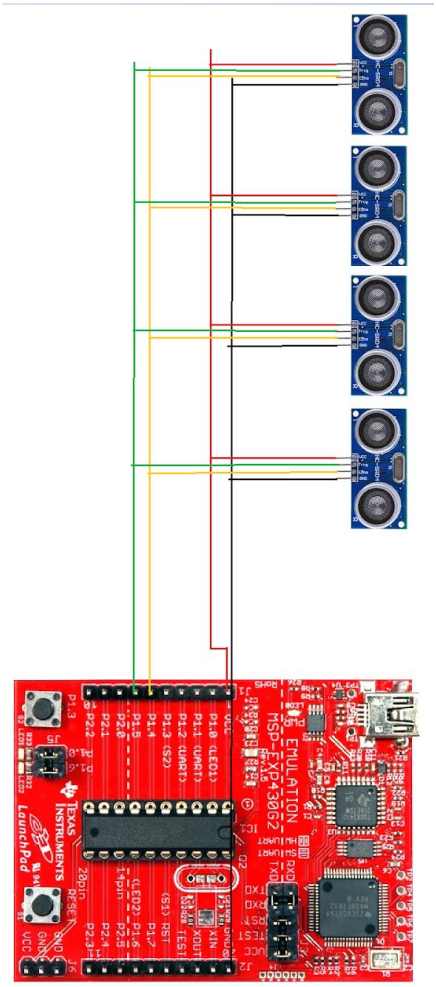

Figura 05 - Esquemático de ligassão dos sensores ultrassônicos

4.1.2) Módulo Buzzer

O módulo buzzer é utilizado para gerar um alerta sonoro, como um alarme, de maneira simples. Para seu funcionamento correto é necessário apenas alimentar, de forma adequada, o VCC e o GND do módulo e controlar o áudio pea entrada I/O do próprio módulo.

Figura 6: Módulo buzzer passivo

4.1.3) Módulo Relé

O relé é uma chave eletromecânica, que sob determinada tensão, fecha o contato da eletromagneticamente por meio da corrente elétrica que percorre as espiras da bobina do relé.

O módulo relé é composto pelas seguintes partes básicas:

- Eletroímã (bobina): constituído por um fio de cobre que envolve um núcleo de ferro, oferecendo um caminho de baixa relutância para o fluxo magnético;
- Armadura Fixa: atua como suporte do relé;
- Armadura móvel: desloca-se devido à atração do campo eletromagnético induzido no núcleo, sendo este movimento responsável pelos movimento dos contatos;
- Conjunto de contatos: NA e NF;
- Mola de rearme;
- Terminais: variam de acordo com a aplicação.

Considerando as características do módulo relé, ele foi escolhido para controlar o motor responsável pelo corte da grama, pois para controlar esse motor é necessário somente a regulação de tensão para ligar e desligar o motor, não sendo então necessário controlar a velocidade e nem o sentido do motor, o que faz desnecessário o uso de uma

ponte H para tal, sendo esta explicada a seguir.

Figura 7: Módulo Relé

4.1.4) Módulo Ponte H

A ponte H pode controlar até dois motores DC, permitindo o controle não somente do sentido de rotação do motor, como também da sua velocidade, por meio dos pinos PWM.

A ponte H possui quatro transistores responsáveis por direcionar a corrente para o acionamento do motor. Esse funcionamento elétrico é semelhante a letra “H”, por isso o nome. Porém este módulo tem uma peculiaridade, sendo esta a habilidade de aplicar ambas polaridades direcionais para o motor. A figura 7 mostra a principal economia da ponte H com base no exemplo dos interruptores: se dois interruptores diagonais estão fechados, o motor começa a funcionar; a direção da rotação do motor depende, em diagonal, que os motores se fechem. Na ponte H real os interruptores são substituídos por transistores que são selecionados de acordo com a corrente do motor e com a tesão.

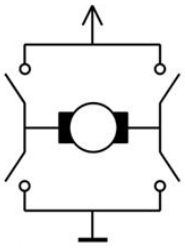

Figura 8: Preceito do Funcionamento da Ponte H

A ponte H possui as seguintes especificações:

- Tensão de operação: 4~35V
- Chip: ST L298N (Datasheet)
- Controle de 2 motores DC ou um motor de passo
- Corrente de operação máxima: 2A por canal ou 4A máximo
- Tensão lógica: 5V
- Corrente lógica: 0~36mA
- Limites de temperatura: -20~135°C
- Potência máxima: 25W
- Dimensões: 43x43x27mm
- Massa: 30g

A figura 8 mostra o módulo Ponte H e em seguida o seu respectivo funcionamento.

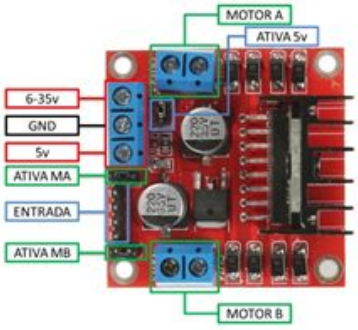

Figura 9: Módulo Ponte H

a) **(Motor A) e (Motor B)** se referem aos conectores para ligação de 2 motores DC ou 1 motor de passo.

b) **(Ativa MA)** e (**Ativa MB)** – são os pinos responsáveis pelo controle PWM dos motores A e B. Se estiver com jumper, não haverá controle de velocidade, pois os pinos estarão ligados aos 5v. Esses pinos podem ser utilizados em conjunto com os pinos PWM do microcontrolador.

c) **(Ativa 5v) e (5v)** – Este Driver Ponte H L298N possui um regulador de tensão integrado. Quando o driver está operando entre 6-35V, este regulador disponibiliza uma saída regulada de +5v no pino (5v) para um uso externo (com jumper), podendo alimentar por exemplo outro componente eletrônico. Portanto não alimente este pino (5v) com +5v do microcontrolador se estiver controlando um motor de 6-35v e jumper conectado, isto danificará a placa. O pino (5v) somente se tornará uma entrada caso esteja controlando um motor de 4-5,5v (sem jumper), assim poderá usar a saída +5v do microcontrolador.

d) **(6-35V) e (GND) –** Aqui será conectado a fonte de alimentação externa quando o driver estiver controlando um motor que opere entre 6-35v. Por exemplo se estiver usando um motor DC 12v, basta conectar a fonte externa de 12v neste pino e (GND).

e) **(Entrada) –** Este barramento é composto por IN1, IN2, IN3 e IN4. Sendo estes pinos responsáveis pela rotação do Motor A (IN1 e IN2) e Motor B (IN3 e IN4).

A tabela em figura abaixo mostra a ordem de ativação de um motor A por meio dos pinos IN1 e IN2. O mesmo esquema pode ser aplicado aos pinos IN3 e IN4, responsáveis por um motor B.

MOTOR	IN1	IN2
HORÁRIO	5v	GND
ANTI-HORÁRIO	GND	5v
PONTO MORTO	GND	GND
FREIO	5v	5v

Figura 10: Ativação dos motores

Para pode usar a ponte H nos motores, estes terão que ser alimentados por uma fonte externa, pois a placa não é capaz de fornecer nos pinos toda a potência necessária para o controle deles.

4.1.5) Módulo sensor IR

Este módulo é um sensor óptico reflexivo e é altamente usado dentro da robótica. O módulo IR a ser usado é o TCRT5000, que possui acoplado um sensor infravermelho (emissor) e um fototransistor (receptor). Ele auxiliará na detecção de objetos. O módulo sensor IR possui também um potenciômetro que auxilia na distÂncia de detecção de objetos, podendo variar de 8mm a 25mm.

O funcionamento do sensor é dada por meio de uma alimentação de 5V na entrada VCC e o GND com o terra, no caso o terra da MSP430; a saída é a resposta digital do sensor, sendo 1 para objeto detectado e 0 para não detectado.

Figura 11: Módulo sensor óptico TCRT5000

4.2) O Microcontrolador

O microcontrolador a ser usado no projeto é, como já dito, o MSP430G2553, que pode ser visto na Figura 1. Já foi explicitado que o MSP430 da Texas Intruments é um microcontrolador de 16bits com uma arquitetura Von Neumann e é usado em aplicações de baixa potência.

A placa MSP será responsável por controlar todos os periféricos a ela ligados, sendos, então, estes todos os sensores supracitados. Com isso, pode-se dizer que este microcontrolador é o cérebro do projeto,

responsável por todo o controle e ações que o Minigrama irá tomar.

4.3) Os motores de tração

Para o controle das rodas que irão compor o robô Minigrama serão utlizados dois motores DC 12V.

Os motores DC possuem ímã permanente e têm uma construção bastante simples. Assim como o seu controle também é bastante elementar. A sua velocidade não é necessariamente determinada pelo sinal de controle, pois depende de vários fatores, principalmente do valor binário aplicado na alimentação. A relação entre binário e velocidade de um motor de corrente contínua ideal é linear, o que significa que quanto maior é a carga sobre o eixo menor é a velocidade do veio e maior é a corrente através da bobina.

Normalmente, a velocidade do motor é dependente da tensão aplicada nos terminais do motor. Se o motor alimentar uma tensão nominal, ele roda a uma velocidade nominal. Se a tensão dada ao motor é reduzida, a velocidade do motor e o binário são reduzidos também. Este tipo de controle de velocidade é também chamado de controle analógico. Isto pode ser implementado, por exemplo, usando um transistor ou um reostato (dispositivo utilizado para variar a resistência de um circuito).

Motores DC são controlados por microcontroladores, por meio da utilização de modulação de largura de impulso (PWM - pulse with modulation), por transistores de comutação rápida on - off. A potência do motor total é algo entre parado e velocidade máxima. O tempo de todo o período de PWM quando o transistor é aberto, chamado ciclo de trabalho, que é medido em percentagem. 0% significa que o transistor está constantemente fechado e não conduz, 100% significa que o transistor está aberto e conduz. A frequência de PWM deve ser suficientemente elevada para evitar a vibração do eixo do motor. Em baixas frequências o motor produz um ruído e, portanto, é principalmente utilizada a modulação de frequência acima de 20 kHz.

O motor escolhido para fazer esse controle das rodas foi uma motor de vidro elétrico de carro da Rosch visto na figura abaixo.

Figura 12: Motor DC 12V

As figuras 13 a 15 a seguir mostram algumas outras características do motor

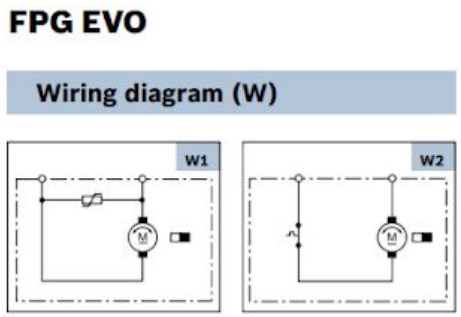

Figura 13: Conexões do Motor

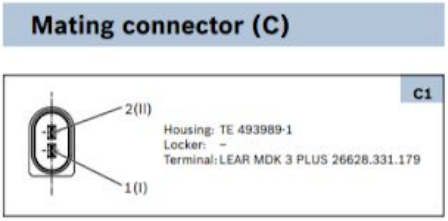

Figura 14: Conector de acoplamento

4.4) Funcionamento dos Softwares

Foi desenvolvido código para testar os sensores que serão utilizados no projeto do Minigrama. Esse código pode ser visualizado em apêndice, junto com os comentário

explicitando o que está sendo feito. Ele também estará presente no trello e github.

4.5) Estrutura

A estrutura física do projeto foi o ponto focal dessa terceira etapa,afinal sem ela não seria possível dar continuidade no projeto, pois para aperfeiçoamento dos sensores é necessária montagem e testes.

A base do chassi foi construída usando metal de calha reciclado, que foi um material leve e relativamente barato, o que e ideal para o projeto. Esta plataforma base tem uma área aproximada de 40cm X 46 cm , mas não tem sua forma retangular, mas sim como a figura seguinte:

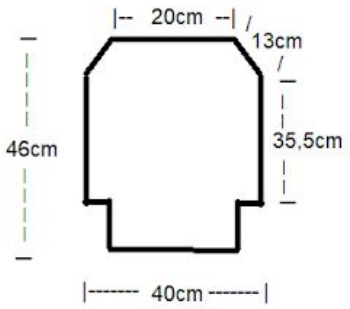

Figura 15: Dimensões da base do chassi

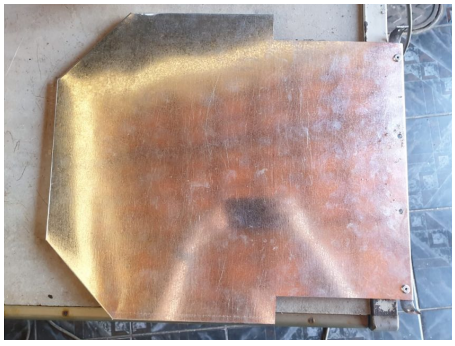

Figura 16: Base do chassi cortada

Na parte frontal da base foi parafusada uma roda boba de 6cm de diâmetro e na parte traseira onde há os dois espaços retangulares foram colocadas as duas rodas de tração junto com os motores a serem controlados pela ponte H.

Essa rodas traseiras foram pensadas de acordo com o projeto, nisso, elas precisariam ser resistentes, as rodas foram, então, construídas em tecnil com 5 cm de raio e 1 cm de largura. Assim, a estrutura montada ficou como nas figuras a seguir:

Figura 17: Tecnil e motor de tração

Figura 18: Chassi montado 1

Figura 19: Chassi montado 2

4.6) Dimensionamento energético

Para a construção do Minigrama é necessário que a fonte energética que

alimentará o robô seja capaz de fornecer toda a energia necessária para que os três motores (os de tração e o de corte) realizem as funções devidas.

Motor	Corrente	Tensão
Motor de tração da roda esquerda	1A	12V
Motor de tração da roda direita	1A	12V
Motor de corte (DC)	800mA	12V

Tabela 1: Dimensão energética

Visto a corrente total necessária é de quase 3A, foi necessário optar então por uma bateria que pudesse alimentar todo o pROjeto, sendo ela então uma bateria de 3A e tensão de 12V. Já para alimentar a LaunchPad foi concluído que terá uma fonte de alimentação própria para não sobrecarregar a bateria dos motores.

5. REVISÃO BIBLIOGRÁFICA

1. DAVIES, John. MSP430 MICROCONTROLLER BASICS. Elsevier Ltd. Burlington, MA, EUA; Oxford, UK.
2. BOSCH, Catálogo de Motores Elétricos 2017-2018. Disponível em: <http://br.bosch-automotive.com/media/parts/download_2/motores_eletricos/Catlogo_Motores_Eltricos_2017-2018.pdf> . Acessado em 11 out. 2019
3. DATASHEET do SENSOR ULTRASSÔNICO. Disponível em: <https://datasheet4u.com/datasheet-pdf/Cytron/HC-SR04/pdf.php?id=1291829> Acessado em 10 out. 2019
4. DATASHEET do SENSOR IR. Disponível em: <https://pdf1.alldatasheet.com/datasheet-pdf/view/252411/VISHAY/TCRT5000.html> Acessado em 10 out. 2019
5. DATASHEET da PONTE H. Disponível em <https://www.alldatasheet.com/view.jsp?Searchword=L298n&gclid=CjwKCAjwusrtBRBmEiwAGBPgE34bqFVo3os6G6NF-Ny_Yu6DZGpR7Iwn4W9H8-BkK5pKCF-qflsWSBoCTV0QAvD_BwE> Acesso em: 10 out. 2019.

6. REFERÊNCIAS

1. Em que altura devo cortar a grama? 2018. Disponível em: https://www.cpt.com.br/cursos-jardinagem/artigos/em-que-altura-devo-cortar-a-grama. Acesso em: 11 set. 2019.
2. PEDRÃO, Rodrigo. GOAT (Cortador de Grama Autônomo) - Instituto Mauá de Tecnologia - Projeto TG. Disponível em: https://www.youtube.com/watch?v=R-s7Zx80uL8 . Acesso em: 08 set. 2019.

LINKS

Trello:
https://trello.com/invite/b/IsrEyxcj/ff349a757fe3ba2e29e57d4533d30657/minigrama

GitHub:
https://github.com/deborajanini/minigrama
(Sendo organizado de acordo com o andamento do projeto)

APÊNDICE

```
#include <msp430g2553.h>
//Motores
#define MOTORA1 BIT0 // P1
#define MOTORA2 BIT7 // P1
#define BTN BIT0//P2
#define TECY BIT1 //P2
#define TECP BIT7 //P2
#define BATEU BIT6//P2
#define MOTORB1 BIT6 // P1
#define MOTORB2 BIT5 // P2
#define LED1 BIT0
#define LED2 BIT6
#define LEDS LED2

//

#define MOTORCORTE BIT2 //P2 // LIGADO EM NIVEL LÓGICO BAIXO
#define LIGAMOTORA1 (P1OUT |= MOTORA1)
#define DESLIGAMOTORA1 (P1OUT &= ~MOTORA1)
#define LIGAMOTORA2 (P1OUT |= MOTORA2)
#define DESLIGAMOTORA2 (P1OUT &= ~MOTORA2)
#define LIGAMOTORB1 (P1OUT |= MOTORB1)
#define DESLIGAMOTORB1 (P1OUT &= ~MOTORB1)
#define LIGAMOTORB2 (P2OUT |= MOTORB2)
#define DESLIGAMOTORB2 (P2OUT &= ~MOTORB2)
#define LIGAMOTORCORTE (P2OUT &= ~MOTORCORTE)
#define DESLIGAMOTORCORTE (P2OUT |= MOTORCORTE)
#define TESTSECURE (P1IN&SECURESWITCH == 0)
//Seletora do multiplexador
#define SEL1 BIT4 //P2
#define SEL0 BIT3 //P2

//Sensores ultrasssonico
#define triggerC BIT1
#define triggerD BIT2
#define triggerE BIT4
#define triggerB BIT0 ///P2
#define echo BIT5

//LEDs para teste
#define LED0 BIT0
#define LED1 BIT6
#define LED2 BIT5

//Direções/
#define CENTRO 0
#define DIREITA 1
#define ESQUERDA 2
#define EMBAIXO 3


int miliseconds;

long sensor;


typedef struct sensoresUltrassonicos
  {
    int Medicao[4];// Medições para entrada do filtro média móvel
    int valorFinal; // Resultado do filtro média móvel

  }sensoresUltrassonicos;

```

```
65 //Procedimentos
66 void filtro_Media_movel(sensoresUltrassonicos *sensor);
67 void viraParaDireita();
68 void viraParaEsquerda();
69 void ParaTudo();
70 void Reiniciar();
71 void ViraParaTras();
72 void vira180graus();
73 void SegueCortandoGrama();
74 void RecorecoCortandoGrama();
75 void InicizalizaPortas();
76 //Funções
77 int MedeFrente();
78 int MedeDireita();
79 int MedeEsquerda();
80 int MedeEmbaixo();
81
82
83
84
85 int main(void)
86 {
87     InicizalizaPortas();
88     DESLIGAMOTORCORTE;
89     __delay_cycles(10000000);              // for 10us
90     ViraParaTras();
91     __delay_cycles(1000000);
92     LIGAMOTORA1;
93     DESLIGAMOTORA2;
94     LIGAMOTORB1;
95     DESLIGAMOTORB2;
96     LIGAMOTORCORTE;
97     int Frente = 0 , Direita = 0 ,Esquerda = 0 , Embaixo = 0;
98     int Reco = 1;
```

main.c | main.asm | main.c

```
91     __delay_cycles(1000000);
92     LIGAMOTORA1;
93     DESLIGAMOTORA2;
94     LIGAMOTORB1;
95     DESLIGAMOTORB2;
96     LIGAMOTORCORTE;
97     int Frente = 0 , Direita = 0 ,Esquerda = 0 , Embaixo = 0;
98     int Reco = 1;
99     while(1)
100    {
101
102
103
104        while((P2IN&BTN)==0)
105            ParaTudo();
106
107        if((P2IN&BATEU)==0)
108            vira180graus();
109        if((P2IN&TECY)==0)
110            Reco = -1;
111        if((P2IN&TECP)==0)
112            Reco = 0;
113
114
115
116
117        Frente = MedeFrente();
118        if(Frente < 30)
119        {
120            Esquerda =  MedeEsquerda();
121            Direita = MedeDireita();
122            if((Direita >Esquerda) && (Direita > 30))
123                viraParaDireita();
124            else if((Direita < Esquerda) && (Esquerda > 30))
```

```
123                 viraParaDireita();
124             else if((Direita < Esquerda) && (Esquerda > 30))
125                 viraParaEsquerda();
126             else
127             {
128                 ViraParaTras();
129                 Direita = MedeDireita();
130                 Esquerda =  MedeEsquerda();
131                 if(Direita > 30)
132                     viraParaEsquerda();
133                 else if(Esquerda > 30)
134                     viraParaEsquerda();
135                 else
136                     ParaTudo();
137 
138             }
139         }
140 
141         else
142         {
143 
144             if(Reco == -1)
145                 RecorecoCortandoGrama();
146             else
147                 SegueCortandoGrama();
148 
149 
150         }
151 
152 
153     }
154 
155 }
156 
```

```
261 int MedeFrente()
262 {
263     int nroMedicao = 0;
264     sensoresUltrassonicos distancia = {.Medicao = {0,0,0,0}, .valorFinal = 0};
265 
266     P2OUT &= ~SEL1;
267     P1OUT &= ~SEL0;
268     __delay_cycles(100);
269     while(nroMedicao <= 3)
270     {
271       __delay_cycles(10000);              // for 10us
272       P1IE &= ~0x01;             // disable interupt
273       P1DIR |= triggerC;           // trigger pin as output
274       P1OUT |= triggerC;           // generate pulse
275       __delay_cycles(10);               // for 10us
276       P1OUT &= ~triggerC;                 // stop pulse
277       P1DIR &= ~echo;          // make pin P1.2 input (ECHO)
278       P1IFG = 0x00;                     // clear flag just in case anything happened before
279       P1IE |= echo;            // enable interupt on ECHO pin
280       P1IES &= ~echo;          // rising edge on ECHO pin
281     distancia.Medicao[nroMedicao] = sensor/58;
282     // Send_String("Distancia Centro:")
283    // Send_Int(distancia[CENTRO].valorFinal);
284     //Send_String("\n");
285 
286     nroMedicao++;
287 
288 }
289     filtro_Media_movel(&distancia);
290     return distancia.valorFinal;
291 }
292 
```

```
int MedeDireita()
{
    int nroMedicao = 0;
    sensoresUltrassonicos distancia = {.Medicao = {0,0,0,0}, .valorFinal = 0};

    P2OUT &= ~SEL1;
    P1OUT |= SEL0;
    __delay_cycles(100);
    while(nroMedicao <= 3)
    {
        __delay_cycles(10000);              // for 10us

      P1IE &= ~0x01;       // disable interupt
      P1DIR |= triggerD;          // trigger pin as output
      P1OUT |= triggerD;          // generate pulse
      __delay_cycles(10);             // for 10us
      P1OUT &= ~triggerD;                 // stop pulse
      P1DIR &= ~echo;         // make pin P1.2 input (ECHO)
      P1IFG = 0x00;                   // clear flag just in case anything happened before
      P1IE |= echo;           // enable interupt on ECHO pin
      P1IES &= ~echo;         // rising edge on ECHO pin

    distancia.Medicao[nroMedicao] = sensor/58;
    //Send_String("Distancia Direita:");
    //Send_Int(distancia[DIREITA].valorFinal);
    //Send_String("\n");

    nroMedicao++;

    }
    filtro_Media_movel(&distancia);
    return distancia.valorFinal;

}
```

```
int MedeEsquerda()
{

    int nroMedicao = 0;
    sensoresUltrassonicos distancia = {.Medicao = {0,0,0,0}, .valorFinal = 0};


    P2OUT |= SEL1;
    P1OUT &= ~SEL0;
    __delay_cycles(100);
    while(nroMedicao <= 3)
    {
        __delay_cycles(10000);              // for 10us

      P1IE &= ~0x01;       // disable interupt
      P1DIR |= triggerE;          // trigger pin as output
      P1OUT |= triggerE;          // generate pulse
      __delay_cycles(10);             // for 10us
      P1OUT &= ~triggerE;                 // stop pulse
      P1DIR &= ~echo;         // make pin P1.2 input (ECHO)
      P1IFG = 0x00;                   // clear flag just in case anything happened before
      P1IE |= echo;           // enable interupt on ECHO pin
      P1IES &= ~echo;         // rising edge on ECHO pin

    distancia.Medicao[nroMedicao] = sensor/58;
    //Send_String("Distancia Direita:");
    //Send_Int(distancia[DIREITA].valorFinal);
    //Send_String("\n");

    nroMedicao++;

    }
    filtro_Media_movel(&distancia);
    return distancia.valorFinal;
```

```
}
int MedeEmbaixo()
{
    int nroMedicao = 0;
    sensoresUltrassonicos distancia = {.Medicao = {0,0,0,0}, .valorFinal = 0};

    P2OUT |= SEL1;
    P2OUT |= SEL0;
    __delay_cycles(100);
    while(nroMedicao <= 3)
    {
        __delay_cycles(10000);              // for 10us

      P1IE &= ~0x01;       // disable interupt
      P2DIR |= triggerB;          // trigger pin as output
      P2OUT |= triggerB;          // generate pulse
      __delay_cycles(10);             // for 10us
      P2OUT &= ~triggerB;                 // stop pulse
      P1DIR &= ~echo;          // make pin P1.2 input (ECHO)
      P1IFG = 0x00;                   // clear flag just in case anything happened before
      P1IE |= echo;           // enable interupt on ECHO pin
      P1IES &= ~echo;         // rising edge on ECHO pin

      __low_power_mode_3();    //entra no modulo lpm3 (ACLK)

    distancia.Medicao[nroMedicao] = sensor/58;
    //Send_String("Distancia Direita:");
    //Send_Int(distancia[DIREITA].valorFinal);
    //Send_String("\n");

    nroMedicao++;

    }
    filtro Media movel(&distancia);
```

```
    P2OUT |= TECP;

    CCTL0 = CCIE;                             // CCR0 interrupt enabled
    CCR0 = 1000;                    // 1ms at 1mhz
    TACTL = TASSEL_2 + MC_1;                  // SMCLK, upmode
    CCTL0 = CCIE;                             // CCR0 interrupt enabled
    _BIS_SR(GIE);

}
#pragma vector=PORT1_VECTOR
__interrupt void Port_1(void)
{
    if(P1IFG&echo)  //is there interrupt pending?
        {
          if(!(P1IES&echo)) // is this the rising edge?
          {
            TACTL|=TACLR;   // clears timer A
            miliseconds = 0;
            P1IES |= echo;  //falling edge
          }
          else
          {
            sensor = (long)miliseconds*1000 + (long)TAR;    //calculating ECHO lenght

          }
    P1IFG &= ~echo;             //clear flag
    }
}
#pragma vector=TIMER0_A0_VECTOR
__interrupt void Timer_A (void)
{
  miliseconds++;
}
```