Mini Cortador de Grama Controlado por MSP430 (Minigrama)

Débora Janini Campos Guedes
Eletrônica Embarcada (Engenharia Eletrônica)
Faculdade do Gama (FGA-UnB)
Gama, DF
deborajaninicg@gmail.com

Maysa Paula Lacerda Cardoso
Eletrônica Embarcada(Engenharia Eletrônica)
Faculdade do Gama (FGA-UnB)
Gama, DF
maysa41@hotmail.com

***Resumo*— Este relatório descreve como foi realizado e contruído todo o projeto final de eletrônica embarcada. O porjeto consiste em um mini cortador de grama microcontrolado pela MSP430G2553.O projeto desenvolvido buscou, além de por o conhecimento da disciplina em prática, a segurança do usuário e também o seu conforto e comodidade.**

Keywords—MSP430, Automação, Sensor, Motores e Corte

I. INTRODUÇÃO

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.

A. Justificativa

Levando em consideração esse desejo de reproduzir ações humanas por robôs, 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.

Fig. 1. MSP430s

A segurança é algo de fato importante, como dito os cortadores de grama convencionais podem gerar diversos acidentes e de acordo com [1] mais de 55 mil lesões por ano nos EUA são causados por cortadores de grama e, mesmo que não haja uma estatística oficial no Brasil, provavelmente esse número é semelhante. Dentre o número de acidentes citados, muitos resultam em perda de dedos ou parte do pé e aproximadamente 75 pessoas por ano acabam falecendo devido a esses acidentes.

Outro benefício é, claramente, a não necessidade de um operador. Ou seja, menor risco a pessoa, menor custo em serviços de jardinagem também e menos tempo gasto em tal atividade, afinal muitas pessoas não posssuem tempo suficiente para cortar as gramas de suas casas, logo automatizar esse serviço acaba sendo uma solução.

B. Objetivo

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 os cortadores convencionais.

C. Requisitos

Neste tópico é apresentado todos os requisitos pensados que o projeto deveria atender, tantos os requisitos técnicos gerais, quanto os de segurança e os do protótipo do Minigrama.

1) Requisitos Técnicos Gerais

- Capacidade de desviar de obstáculos:

  Esse é um dos principais requisitos; e para o obter foram utilizados sensores para que houvesse a mínima interferência humana possível.

- Velocidade mínima do Minigrama

  A fim de obter um bom funcionamento foi necessário pensar em uma velocidade mínima e de acordo com [2] essa velocidade seria de 0.35 Km/h.

- Tempo mínimo de autonomia

  Novamente, para se ter um bom funcionamento é necessário que o robô possa andar sozinho por um mínimo de tempo, que foi determinado de 15 minutos, além de ter um carregamento relativamente rápido, assim, foi pensado em baterias recarregáveis.

- Altura mínima da grama

  Esse requisito trata mais da estética que a grama deve possuir após o corte. A altura do corte depende muito da espécie da grama, entretanto a menor altura entre elas é de 3cm [3], que foi a altura usada como referência no projeto.

2) Requisitos de segurança

- Desligamento automático do motor

  Para garantir segurança, o motor de corte (com a lâmina) é desligado assim que o Minigrama for levantado enquanto estiver em funcionamento ou sofrer algum tipo de exposição da lâmina.

- Ausência de manuseio durante corte

  Um requisito que veio da justificativa do projeto é a ausência da necessidade de que uma pessoa fique perto do cortador enquanto ele está em funcionamento, evitando, dessa forma, acidentes.

3) Requisitos de Hardware do protótipo

- Sensores Ultrassônicos

  Estes sensores são os responsáveis para encontrar os obstáculos e os mensurar de maneira rápida e precisa para que o robô possa seguir em direção distinta ao do obstáculo.

- Módulo Ponte H

  A Ponte H é responsável pelo controle dos motores de tração a fim de fazer a livre movimentação do robô de acordo com a área em que ele se encontra.

- Módulo Relé

  O relé é o módulo responsável pelo controle do motor de corte.

- Motores de Tração

  São os motores controlados pela Ponte H e que determinam a movimentação do robô.

- Motor de Corte

  É o motor DC responsável pelo corte da Grama.

- Baterias Recarregáveis

  Elas são a fonte energética do protótipo, as quais alimentam os três motores e a placa MSP.

- Chassi

  O chassi é a estrutura e sustento do protótipo, que foi dimensionado de acordo com os componentes e forma de movimentação designado pelo software.

4) Relação requisitos e Hardware

A relação entre os requisitos e o hardware do protótipo pode ser explicitada pela tabela 1, que mostra qual é o módulo de hardware necessário para atingir determinado requisito.

TABELA I. REQUISITOS X HARDWARE

TÓPICO	REQUISITO	MÓDULO/HARDWARE
OPERACIOANAL	DESVIO DE OBSTÁCULO	SENSOR ULTRASSÔNICO
	VELOCIDADE MÍNIMA	PONTE H/ MOTORES DE TRAÇÃO
	ALTURA MÍNIMA	RELÉ/ MOTOR DE CORTE/ CHASSI
ENERGÉTICO	AUTONOMIA DE FUNCIONAMENTO	BATERIA
	CARREGAMENTO RÁPIDO	BATERIA
SEGURANÇA	DESLIGAMENTO AUTOMÁTICO DO MOTOR	SENSOR ULTRASSÔNICO
	AUSÊNCIA DE MANUSEIO	PRÓPRIO PROJETO

II. DESENVOLVIMENTO

A fim de cumprir o estabelecido na justificativa e atingir os requisitos for realizada a pesquisa do que seria necessário para a construção do Minigrama.

O protótipo contruído é composto por hardware quanto por software, o qual controla todo o sistema por meio do microcontrolador MSP430G2553.

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

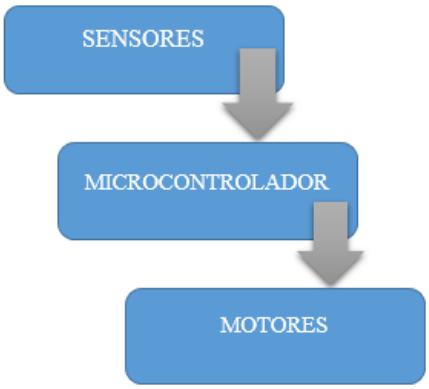

Fig. 2. Diagrama de Blocos do Protótipo

A. *Hardware*

1) Módulos

1.1) Sensor Ultrassônico

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

O sensor possui cerca de 15° de ângulo de detecção dos obstáculos e é capaz de medir distâncias de 2cm a 4m com ótima precisão. Ele possui um circuito pronto com o 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.

Fig. 3. Módulo Sensor Ultrassônico HC-SR04

.

Para começar a medição é necessário alimentar o módulo e coloca 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 feio 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 tempo pode ser visto na figura 4.

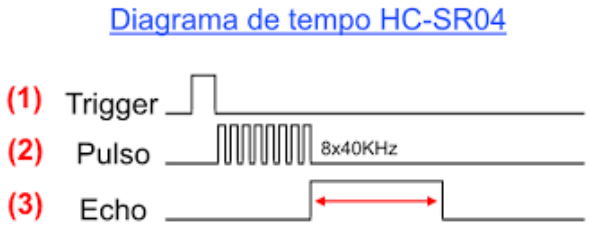

Fig. 4. Diagrama de Tempo do HC-SR04

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

$$d = \frac{Tempo\ de\ resposta\ do\ sensor * Velocidade\ do\ som}{2} \quad (1)$$

A velocidade do som pode ser considerada idealmente igual a 340m/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.

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

Para o projeto foram implementados 4 sensores ultrassônicos de acordo com o esquemático abaixo ligados a um demux que seleciona qual sensor tem prioridade de acordo com os valores de distância lidos. Dos 4, um é o sensor frontal, dois laterais e um (que fica embaixo do chassi) é o de segurança.

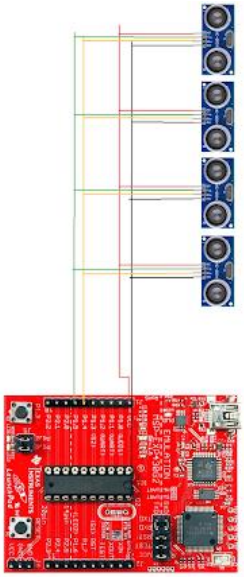

Fig. 5. Esquemático de Ligação dos Sensores Ultrassônicos

1.2) Módulo Relé

O relé é uma chave eletromecânica, que sob determinada tensão, fecha o contato 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 movimentos 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

Fig. 6. Módulo Relé

1.3) 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.

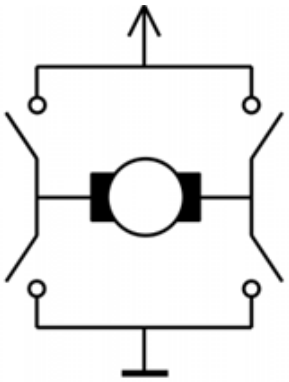

Fig. 7. Preceito do funcionamento da Ponte H

A ponte H possui as seguintes especificações:

- Tensão de operação: 4~35V.
- Chip: ST L298N (Datashhet).
- Controle de 2 motores DC ou motor de passo
- Corrente de operação máxima: 2ª por canal ou 4ª máximo.
- Corrente lógica: 0~36mA.
- Tensão lógica: 5v.
- 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.

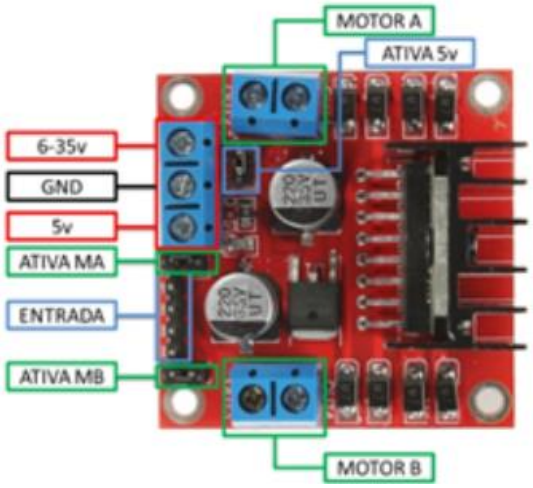

Fig. 8. Módulo Ponte H

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

Fig. 9. 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.

2) Microcontrolador

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

A placa MSP é responsável por controlar todos os periféricos a ela ligados, sendo, então, estes todos os sensores suprecitados. 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.

3) Motores

Para o controle das rodas que compõem o Minigrama são utilizados dois motores DC 12V e para o corte da grama um motor DC, também 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 responsável pelo corte da grama é o motor de tração de impressora C9000-6005 12V e pode ser visto na figura abaixo.

Fig. 10. Motor de Corte

Já o motor escolhido para fazer o controle das rodas foi um motor de vidro elétrico de carro da Rosch, que pode ser visto na figura 11.

Fig. 11. Motor de tração das rodas

Assim, o esquemático abaixo mostra de forma simples a ligação entre os componentes:

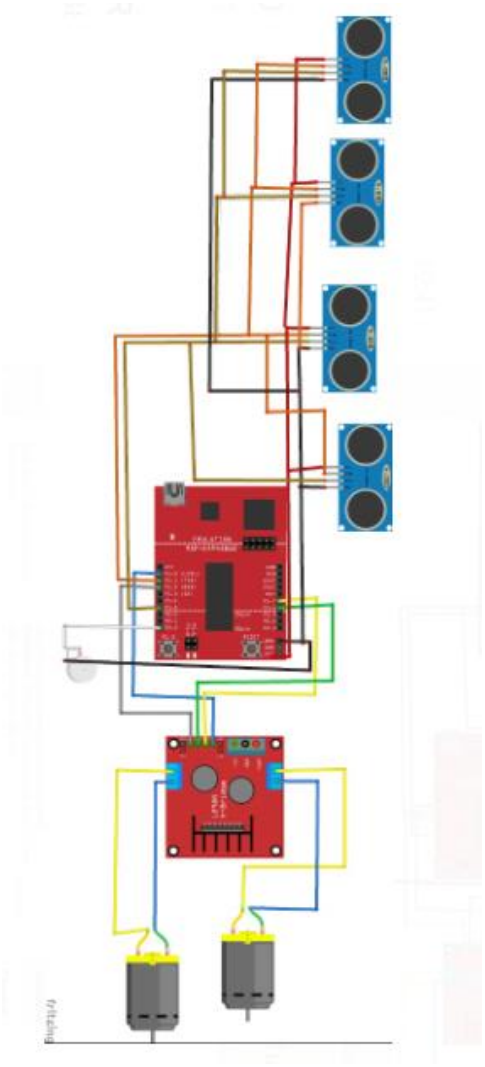

Fig. 12. Motor de tração das rodas

B. *Dimensionamento Energético*

Para a construção do Minigrama foi necessário que a fonte energética que alimenta o robô fosse capaz de fornecer toda a energia necessária para que os três motores realizem as funções devidas. A tabela 2 mostra a dimensão energética para esses motores.

TABELA II. DIMENSÃO ENERGÉTICA

MOTOR	CORRENTE	TENSÃO
MOTOR DE TRAÇÃO DA RODA ESQUERDA	1A	12V
MOTOR DE TRAÇÃO DA RODA DIREITA	1A	12V
MOTOR DE CORTE	800MA	12V

Para atender essa necessidade foram utilizadas duas baterias de lítio recarregáveis.

Fig. 13. Baterias recarregáveis

C. *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:

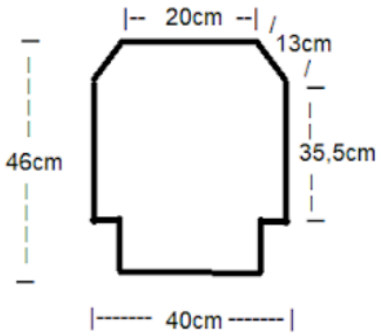

Fig. 14. Dimensões da Base do Chassi

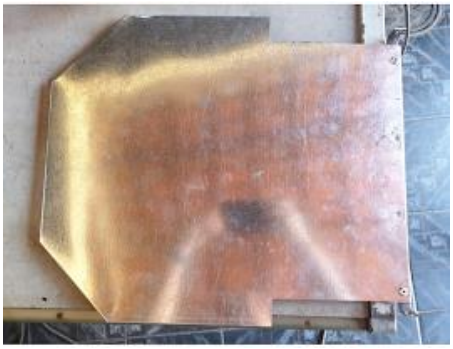

Fig. 15. Base do Chassi Cortada

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:

Fig. 16. Tecnil e Motor de Tração

Fig. 17. Chassi Montado 1

Fig. 18. Chassi Montado 2

Fig. 19. Chassi Montado com Furo para Motor de Corte

D. Funcionamento do Código

O código começa a rodar após o comando de iniciar do usuário e funciona da seguinte forma:

1º- O robô realiza a leitura do sensor frontal e caso a distância com o obstáculo seja maior que 30 centímetros ele corta a grama, caso seja menor ele fará a leitura dos sensores direito e esquerdo.

2º- Após a leitura dos sensores laterais ele toma a decisão também conforme a distância do obstáculo, caso a distância seja menor que 30cm, ele dá ré e refaz a leitura, caso a distância seja maior ele vira para direita ou esquerda (o que tiver a maior distância de um obstáculo) e daí, novamente realiza o corte da grama.

3º- Após realizar o corte ele volta a ler o sensor frontal e realiza novamente as etapas acima descritas.

Entretanto caso após a segunda ré o robô não consiga ainda virar para algum lado, ele para e deve-se reiniciar o sistema.

Além dessa lógica principal, que pode ser melhor entendida pelo fluxograma abaixo, tem-se também o do quarto sensor, que caso a distância seja menor que a determinada, ele deve parar o motor de corte para evitar acidentes.

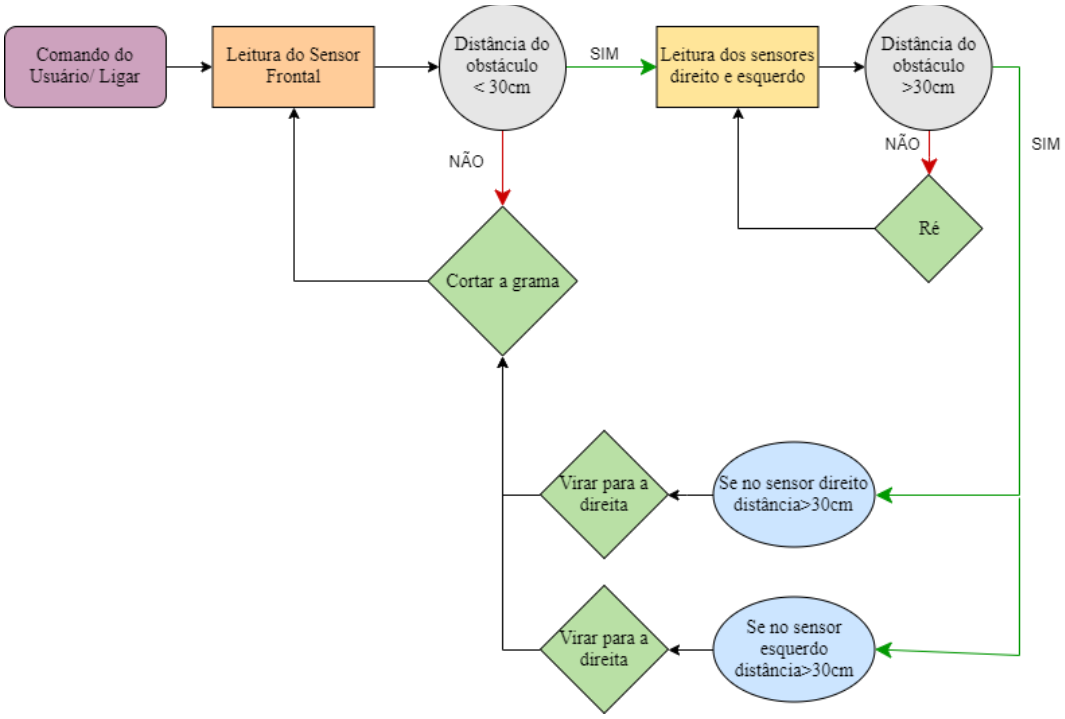

Fig. 20. Fluxograma da lógica de movimentação definido em código

OBS: O código segue em apêndice.

III. RESULTADOS

Os testes dos resultados foram realizados por etapas durante o desenvolvimento do projeto a fim de averiguar quais problemas poderiam ocorrer em cada componente. Primeiro foram realizados testes separados para cada componente no Energia IDE, e depois foram feitos os testes do sensores diretamente no Code Composer já usando a placa micocontroladora MSP430G2553. Esses primeiros testes dos sensores foram bem sucedidos, mostrando que de fato os sensores estavam fazendo a leitura dos obstáculos nas distâncias determinadas.

Depois disso foram realizados os testes juntando os sensores com motores DC pequenos a fim de visualizar se a lógica programada está funcionando de acordo com o esperado, e também obteve-se sucesso nestes teste.

Após os testes com motores simples, foram realizados os testes com os motores do projeto em si, já na estrutura do chassi do projeto desenvolvido, o maior problema que se teve nesses testes foi com a Ponte H, pois mesmo que ela suportasse as correntes exigidas por cada motor, ela esquentou muito e acabou queimando, então para solucionar esse problema foi adicionado um cooler para realizar o resfriamento da Ponte H e com isso, pode-se finalmente ver que o robô estava funcionando de acordo com a lógica programada desenvolvida.

Já os códigos foram sofrendo pequenas alterações ao longo do projeto conforme ele ia se desenvolvendo e evoluindo com a integração dos componentes até que finalmente teve-se o código final mostrado na seção anterior “II D – Funcionamento do código”.

O projeto funciona de acordo com o fluxograma mostrado anteriormente. Os motores controlam a velocidade de uma roda em relação a outra para poder virar à esquerda ou À direita e mantém as velocidades das duas iguais caso vá para a frente ou de ré. Além disso, o código do quarto sensor ultrassônico, o sensor de segurança que fica embaixo, também funcionou como devido, parando o motor de corte quando a distância era maior do que a determinada para corte, o que ajuda a evitar acidentes.

Haveria inicialmente, um sensor infravermelho para auxiliar na detecção de obstáculos, entretanto os sensores ultrassônicos já estavam sendo suficientes para o projeto proposto. Além disso, seria necessário fazer algumas adições ao código, aumentando a complexidade do projeto, talvez não houvesse tempo suficiente para a data limite e também sobrecarregaria a placa MSP.

Vale salientar, que mesmo não sendo um resultado direto do projeto, que durante a reta final o protótipo tomou uma queda de aproximadamente 1 metro enquanto fazia o furo do chassi para motor de corte, essa queda acabou por danificar alguns componentes, como o demux e a roda de tração direita; ou seja, uma queda em um robô, caso comerciável é algo que pode danificar o produto, assim como diversos outros produtos disponíveis e que infelizmente não cabem garantia, pois não entraria em defeitos de fabricação.

IV. CONCLUSÃO

A sociedade moderna está cada vez mais precisando e desejando automatizar serviços, muitas vezes por comodidade, mas também por segurança e com isso, a reprodução de atividades humanas por meio de máquinas tem se tornado cada vez maior. Tendo isso em consideração o projeto proposto visa automatizar o corte de grama, evitando

assim a necessidade de manuseio e os riscos de acidentes, que nos EUA chegam a 55 mil casos de lesões e 75 mortes por ano.

O protótipo foi construído com 4 sensores ultrassônicos que mandam os sinais de distância calculadas para o microcontrolador (MS430G2553) como entradas e esse, de acordo com a lógica do código desenvolvido, manda a resposta para a ponte H que controla os motores de tração, que seriam as saídas do sistema, garantindo a movimentação do Minigrama em determinada direção dependendo dos obstáculos no caminho.

Os testes realizados foram altamente satisfatórios, pois foi possível atingir o objetivo proposto e alcançar os requisitos determinados ao projeto e o Minigrama pode enxergar os obstáculos e se desviar deles, indo para a esquerda, direita ou dando ré, além de parar o sensor de corte quando o quarto sensor ultrassônico (sensor de segurança) observa uma distância maior do que a de corte da grama.

REFERÊNCIAS

[1] J.A.V. Sanhudo, "O que voce deve saber antes de cortar a grama". Disponível em: http://www.clinicadope.net/index.php/dicas/21-se-voce-esta-pensando-em-cortar-grama-deveria-ler-isto . Acesso em: 09 set 2019.

[2] R. Pedrão, "Goat (cortador de grma autônomo)", Instituto Mauá de Tecnologia. 2014. Disónível em: https://www.youtube.com/watch?v=R-s7Zx80uL8 . Acesso em 08 set 2019.

[3] S. Teixeira, "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 de 2019.

[4] J. Davies, "MSP430 MICROCONTROLLER BASICS". Elsevier Ltd. Burlington, MA, EUA; Oxford, UK.

[5] 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 . Acesso em: 11 out. 2019.

[6] Datasheet do sensor ultrassônico. Disponível em: https://datasheet4u.com/datasheet-pdf/Cytron/HC-SR04/pdf.php?id=1291829 . Acesso em: 10 out 2019.

[7] Datasheet da ponte h. Disponível em: https://www.alldatasheet.com/datasheet-pdf/pdf/22440/STMICROELECTRONICS/L298N.html. Acesso em 10 out 2019.

APÊNDICE A

```
#include <msp430g2553.h>



//Motores
#define MOTORA1 BIT0 // P1
#define MOTORA2 BIT7 // P1

#define MOTORB1 BIT6 // P1
#define MOTORB2 BIT2 // P1
//

#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 (P1OUT |= MOTORB2)
#define DESLIGAMOTORB2 (P1OUT &= ~MOTORB2)
#define LIGAMOTORCORTE (P2OUT |= MOTORCORTE)
#define DESLIGAMOTORCORTE (P2OUT &= ~MOTORCORTE)

//Seletora do multiplexador
#define SEL1 BIT4
#define SEL0 BIT3

//Sensores ultrasssonico
#define triggerC BIT1
#define triggerD BIT3
#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;


//Procedimentos
void filtro_Media_movel(sensoresUltrassonicos *sensor);
void viraParaDireita();
void viraParaEsquerda();
void ParaTudo();
void Reiniciar();
void ViraParaTras();
void SegueCortandoGrama();

void InicializaPortas();
//Funções
int MedeFrente();
int MedeDireita();
int MedeEsquerda();
int MedeEmbaixo();




int main(void)
{
    InicializaPortas();
    __delay_cycles(10000000);             // for 10us
    LIGAMOTORA1;
    DESLIGAMOTORA2;
    LIGAMOTORB1;
    //DESLIGAMOTORB2;
    LIGAMOTORCORTE;
    volatile int Frente = 0 , Direita = 0 ,Esquerda = 0 , Embaixo = 0;
    while(1)
    {

        Frente = MedeFrente();
        if(Frente < 30)
```

```
        {
            Direita = MedeDireita();
            Esquerda =  MedeEsquerda();
            if(Direita > 30)
                viraParaDireita();
            else if(Esquerda > 30)
                viraParaEsquerda();
            else
            {
                ViraParaTras();
                Direita = MedeDireita();
                Esquerda =  MedeEsquerda();
                if(Direita > 30)
                    viraParaDireita();
                else if(Esquerda > 30)
                    viraParaEsquerda();
                else
                    ParaTudo();

            }
        }

        else
        {
            SegueCortandoGrama();

        }




    }


}

void viraParaDireita()
{
    DESLIGAMOTORB1;
    DESLIGAMOTORB2;
    LIGAMOTORA1;
    DESLIGAMOTORA2;
    __delay_cycles(3000000);
    SegueCortandoGrama();

}

void viraParaEsquerda()
{

    DESLIGAMOTORA1;
    DESLIGAMOTORA2;
    LIGAMOTORB1;
    DESLIGAMOTORB2;
    __delay_cycles(3000000);
    SegueCortandoGrama();



}
void ParaTudo()
```

```
{
    DESLIGAMOTORA1;
    DESLIGAMOTORA2;
    DESLIGAMOTORB1;
    DESLIGAMOTORB2;
    DESLIGAMOTORCORTE;


}
void Reiniciar()
{
    LIGAMOTORA1;
    DESLIGAMOTORA2;
    LIGAMOTORB1;
    DESLIGAMOTORB2;
    LIGAMOTORCORTE;

}
void ViraParaTras()
{
    DESLIGAMOTORA1;
    DESLIGAMOTORB1;
    LIGAMOTORB2;
    LIGAMOTORA2;
    __delay_cycles(3000000);
    SegueCortandoGrama();

}

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

    P2OUT &= ~SEL1;
    P1OUT &= ~SEL0;

    while(nroMedicao <= 3)
    {
     __delay_cycles(10000);              // for 10us
      P1IE &= ~0x01;            // disable interupt
      P1DIR |= triggerC;          // trigger pin as output
      P1OUT |= triggerC;          // generate pulse
      __delay_cycles(10);             // for 10us
      P1OUT &= ~triggerC;                 // 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 Centro:");
   // Send_Int(distancia[CENTRO].valorFinal);
    //Send_String("\n");

    nroMedicao++;

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

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

    P2OUT &= ~SEL1;
    P1OUT |= SEL0;

    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;

}

void SegueCortandoGrama()
{
    LIGAMOTORA1;
    DESLIGAMOTORA2;
    LIGAMOTORB1;
    DESLIGAMOTORB2;
    LIGAMOTORCORTE;

}


int MedeEsquerda()
{

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


    P2OUT |= SEL1;
    P1OUT &= ~SEL0;

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

    P2OUT |= SEL1;
    P1OUT |= SEL0;

    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;

}



void filtro_Media_movel(sensoresUltrassonicos *sensor)
```

```
{
    sensor->valorFinal= (sensor->Medicao[0]+ sensor->Medicao[1]+ sensor->Medicao[2] +
sensor->Medicao[3])/4;
}

void InicizalizaPortas()
{
    WDTCTL = WDTPW + WDTHOLD;
    BCSCTL1 = CALBC1_1MHZ;
    DCOCTL = CALDCO_1MHZ;

    P2DIR |= SEL1;
    P2OUT &= ~SEL1;
    P1DIR |= SEL0;
    P1OUT &= ~SEL0;

// Motores
    P2SEL &= ~(BIT6 +BIT7);
    P2SEL2 &= ~(BIT6 +BIT7);

    P1DIR |= MOTORA1;
    P1OUT &= ~MOTORA1;
    P1DIR |= MOTORA2;
    P1OUT &= ~MOTORA2;
    P1DIR |= MOTORB1;
    P1OUT &= ~MOTORB1;
    P1DIR |= MOTORB2;
    P1OUT &= ~MOTORB2;
    P2DIR |= MOTORCORTE;
    P2OUT &= ~MOTORCORTE;


    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++;
}
```