Ponto de Controle 3

Robô seguidor de linha

Leonardo Amorim de Araújo
15/0039921
Faculdade do Gama
Universidade de Brasília
St. Leste Projeção A - Gama Leste, Brasília - DF, 72444-240
Email: leonardoaraujodf@gmail.com

Josiane de Sousa Alves
15/0038895
Faculdade do Gama
Universidade de Brasília
St. Leste Projeção A – Gama Leste, Brasília – DF, 72444 – 240
Email: josianealves.18@gmail.com

RESUMO – Este documento apresenta o ponto de controle III do projeto final para a disciplina de Microprocessadores e Microcontroladores.

Palavras-chave — MSP 430; Robô seguidor de linha; Robô Autônomo; Microprocessador;

I. INTRODUÇÃO

Robôs autônomos são máquinas inteligentes capazes de realizar tarefas sem controle humano contínuo e explícito sobre seus movimentos. Estes podem sentir e obter informações sobre seus arredores, trabalhar e desviar de obstáculos. Um tipo de robô autônomo é o robô seguidor de linha que é capaz de identificar uma trilha de dimensões definidas e percorrer por ela. Esse tipo de robô pode ser usado para muitas aplicações, dentre elas, na indústria como os Veículos Guiados Automaticamente – AGVs; para filmagens em campos, quadras; guias em estabelecimentos comerciais de grandes proporções, etc.

II. OBJETIVOS

Construir um robô seguidor de linha programado na Lanchpad MSP 430 com o intuito de realizar tarefas de limpeza industrial.

III. DESCRIÇÃO

Com o intuito de auxiliar na limpeza industrial, especialmente em locais pequenos, de difícil acesso e locais que ofereçam risco à saúde, será construído um robô seguidor de linha que realiza um trajeto especificado, programado na MSP430. Este robô será capaz de se movimentar para frente, para trás, direita e esquerda e contará com 5 opções de velocidade, com isso o usuário poderá optar por uma limpeza mais superficial ou mais específica, dependo da necessidade no momento da limpeza.

Para que as funções acima funcionem corretamente, é necessário utilizar alguns componentes e funções que serão especificados a seguir.

HARDWARE

- **CI L293B – Dual Ponte H**

O CI L293B é um *PUSH-PULL FOUR CHANNEL DRIVER* que consiste de dois drivers de ponte H que controlam até dois motores DC. A ponte H permite que se controle a direção de giro dos motores e a sua velocidade utilizando o microcontrolador. A pinagem do CI é mostrada na Fig. 01.

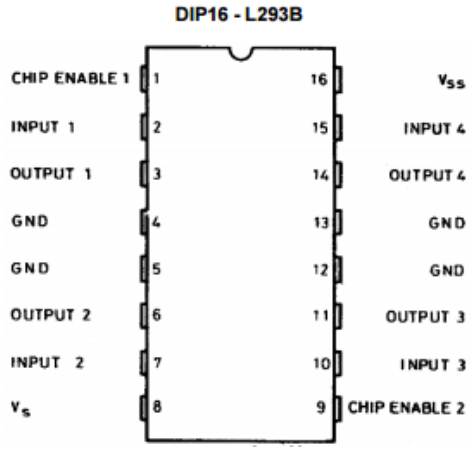

Figura 01. Pinagem do CI L293B

Os pinos 1 e 9 são chamados pinos de ENABLE pois habilitam o funcionamento do motor e devem estar em nível alto para que ocorra o funcionamento. Estes pinos podem ser controlados posteriormente para utilização do PWM para controle de velocidade do robô.

Os pinos 2, 3, 10, e 15 são os pinos que controlam o modo de giro do motor, que são três: direita, esquerda e freio.

A Tabela 01 mostra como deve ser utilizado o CI e os resultados esperados:

Tabela 01. Modo de utilização do CI L293B

Pino 1	Pino 2	Pino 7	Função
ALTO	BAIXO	ALTO	Girar no sentido horário
ALTO	ALTO	BAIXO	Girar no sentido anti-horário
ALTO	BAIXO	BAIXO	Parado
ALTO	ALTO	ALTO	Parado
BAIXO	Sem aplicabilidade		

Os pinos apresentados na Tabela 01 controlam o primeiro motor. Para o segundo motor a função é a mesma, mas os pinos 1, 2 e 7 equivalem respectivamente aos pinos 9, 10 e 15. O motor 1 tem seu polo positivo ligado no pino 3 e o negativo no pino 2. O mesmo para o motor 2 que deve ser ligado nos pinos 11 e 14. O pino 16 deve ser conectado no Vcc da MSP e o pino Vs na fonte de tensão do motor. Os GNDs da MSP e da fonte que alimentam o motor são ligados no mesmo nó, assim como os pinos denominados GND no CI.

No projeto, o CI L293B será utilizado para controlar a direção do robô, conforme o acionamento dos sensores, e o modo de limpeza, alternando entre as velocidades dos motores.

- **Sensor Óptico Reflexivo TCRT5000**

Este sensor, cuja a vista superior é mostrada na Fig. 02, é composto por dois componentes que funcionam em conjunto, um fototransistor e um led infravermelho envoltos em uma estrutura de plástico.

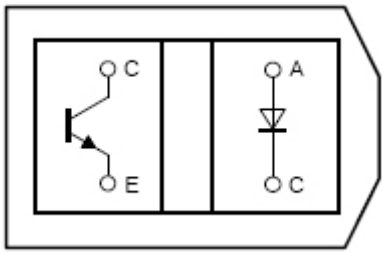

Figura 02. Vista superior do sensor TCRT5000.

O LED emite um feixe de luz infravermelha numa frequência não visível a olho nu, e o fototransistor tem a função de capturar o feixe de luz emitido pelo LED. O funcionamento do fototransistor é bem semelhante ao de um transistor, possui um coletor, um emissor, e a base, que é ativada ao receber um feixe de luz infravermelha.

O sensor possui quatro pinos, sendo dois do LED e dois do fototransistor. São eles:

-Anodo do led (A);
-Catodo do led (C);
-Coletor do fototransistor (C);
-Emissor do fototransistor (E).

Ao aplicar-se uma tensão no LED, ele emite o feixe de luz infravermelha, quando esse feixe de luz encontra algum objeto, essa luz é refletida na base do transistor, fazendo com que haja condução de corrente entre o coletor e o emissor do mesmo.

O LED emite um feixe de luz infravermelha que é refletida pelo objetivo que estiver a frente do sensor e que é, por sua vez, detectada pelo fototransistor. Quanto mais claro for o objeto, melhor será o funcionamento do sensor, pois mais luz será refletida. E é aproveitando-se dessa condição que o sensor será útil para o robô seguidor de linha, pois, no momento em que o sensor detectar uma superfície de cor preta, no caso a linha que ele deve seguir, significa que sua trajetória acabou e o robô está prestes a desviar do caminho. Nesse momento será ativado um comando (vire à direita, por exemplo) para que o robô volte ao trajeto original.

SOFTWARE

Antes de entrar em detalhes sobre o código, é importante destacar como ficou a definição da porta P1OUT da MSP para controle dos motores através do CI L293B. Algumas mudanças foram acrescentadas com relação ao Ponto de Controle 2. A Tabela 02 mostra como ficou estruturado.

Tabela 02. Correspondência entre os bits em P1OUT e no CI L293B

Bit em P1OUT	**Porta do L293B**	**Especificação**
BIT 0 (PORTA P1.0)	INPUT 1	SAÍDA
BIT 1 (PORTA P1.1)	INPUT2	SAÍDA
BIT 2 (PORTA P1.2)	INPUT3	SAÍDA
BIT 3 (PORTA P1.3)	Botão de Controle de Velocidade dos Motores	ENTRADA
BIT 4 (PORTA P1.4)	INPUT 4	SAÍDA
BIT 5 (PORTA P1.5)	Sensor 1 (se for possível utilizar nesta porta o conversor AD)	ENTRADA
BIT 6 (PORTA P1.6)	ENABLE1 e ENABLE2 (saída com PWM)	SAÍDA
BIT 7 (PORTA P1.7)	Sensor 2 (se for possível utilizar nesta porta o conversor AD)	ENTRADA

Os sensores foram definidos nas portas P1.5 e P1.7, mas podem ser mudados ao descobrirmos como utilizar o conversor AD da MSP430.

No anexo deste documento encontra-se o código que utilizamos. A primeira função é chamada de *Motor_Direction*, que foi definida em Assembly e tem o objetivo de controlar a direção em que os motores giram. Esta função só controla as portas de INPUT do CI L293B, fazendo inversão de direção dos motores ou parando estes.

A função recebe dois argumentos do tipo char, que se referem à direção que se requer do motor. Por exemplo, se na função main () for definido que se deseja que o robô siga em frente, então passa-se como argumentos à função Motor_Direction ('R','R'), que significa Right – Right. Deve-se utilizar a Tabela ASCII, já que estamos trabalhando em Assembly. Logo em hexadecimal, R = 0x52. Como utilizamos as letras R – RIGHT (direita), L – LEFT (esquerda) e S – STOP (parar), temos os equivalentes na Tabela ASCII:

L – 0x4C
R – 0x52
S – 0x53

A função compara os valores nos registradores R12 (que se refere ao char c1 na função main()) e R13 (que se refere ao char c2 na função main()) e retorna um inteiro que refere-se a direção que se deseja do motor para ser enviado à porta P1.

A segunda função é chamada de *Control_Speed()*. O objetivo desta é acionar os pinos de ENABLE1 e ENABLE2 do CI L293B, usando PWM, onde é possível controlar a velocidade do robô. Esta aplicação é necessária para que o usuário possa escolher qual o tipo de limpeza deseja. Definimos que são 6 velocidades possíveis, onde em uma destas o robô permanece parado. A função recebe como argumento um inteiro *Speed* que se refere a uma das 6 velocidades requeridas e não retorna nenhum valor, somente configura o modo de comparação dos *Timers* 1 e 0 da MSP. Para este modo na MSP430G2553, EQU0 está na Porta P1.6, conforme o DATASHEET e será resetado em TACCR1 e setado TACCR0 (OUTMOD_7). O controle de velocidade é selecionado pelo botão da porta P1.3 na MSP430, e será discutido em detalhes a seguir. Um exemplo pode ser visto abaixo, caso o usuário deseje a velocidade 4.

```
else if(Speed == 4)
  {
   TACTL = TACLR;
   TACCR0 = 10000; //100 hz
   TACCR1 = 8750; //87.5% of the Duty Cycle
   TACCTL1 = OUTMOD_7;
   TACTL = TACTL = TASSEL_2 + ID_0 + MC_1;
  }
```

Na função main() que chama todas as funções principais do código, ficaram definidos os usos da porta P1.0 da MSP, além das definições para interrupção e para o modo de baixo consumo LPM0. Isto pode ser visto no código no Anexo A deste trabalho. No caso deste projeto, a interrupção será utilizada para quando o usuário pressionar o botão da MSP na porta P1.3 e desejar mudar a velocidade. No momento em que esta ação ocorrer, a interrupção da porta P1 será ativada e teremos, a execução da instrução conforme abaixo

```
#pragma vector = PORT1_VECTOR
__interrupt void Port_1(void)
{

  volatile unsigned int i = 60000;

  //Controle de Velocidade dos Motores
  if((P1IN&BTN)==0)
  {
   if(com!=5)
   {
    com++;
    Control_Speed(com);
    while(i--);
   }
   else
   {
    com = 0;
    Control_Speed(0);
    while(i--);
   }
  }
}
```

A variável **i** acima serve para realizar o deboucing do botão, que pode ser visto no final do código com o while(i--). Verificando-se que o botão foi pressionado, a variável global **com**, que começa inicializada em zero, irá selecionar o modo de velocidade que se deseja para o robô, e quando esta atingir 5 e o botão for pressionado mais uma vez, volta a ser 0.

IV. RESULTADOS

Com o intuito de verificar a viabilidade do projeto pretendido pela dupla, alguns testes foram feitos com a MSP430, o CI L293B e os sensores ópticos reflexivos. Montando-se o circuito da Fig. 03 na protoboard, foi possível simular as seguintes funções do robô:

- Movimento em linha reta;
- Desvio para a direita;
- Desvio para a esquerda;
- Parada.

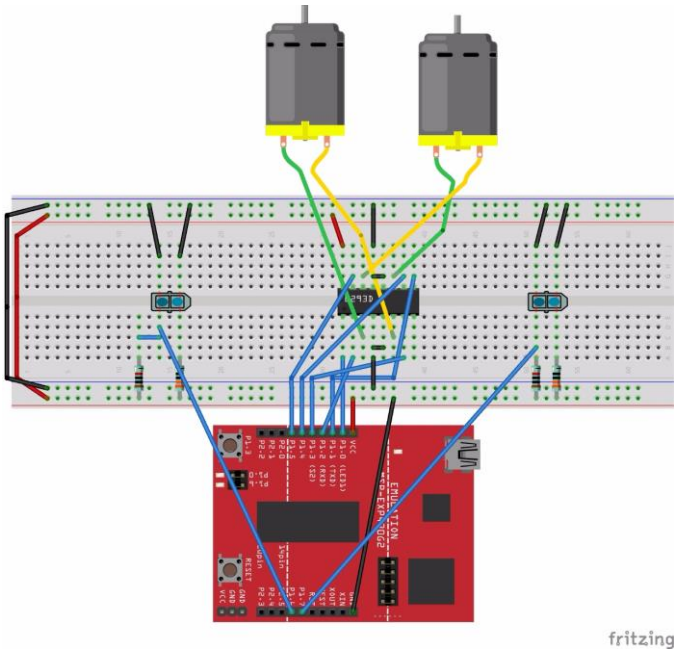

Figura 03. Simulação de algumas funções do robô seguidor de linha.

Dado o sucesso obtido nas simulações, deu-se início a construção do robô utilizando-se um chassi de acrílico com dois motores DC de 6V e dois sensores ópticos reflexivos acoplados.

A Fig. 04 mostra o robô construído.

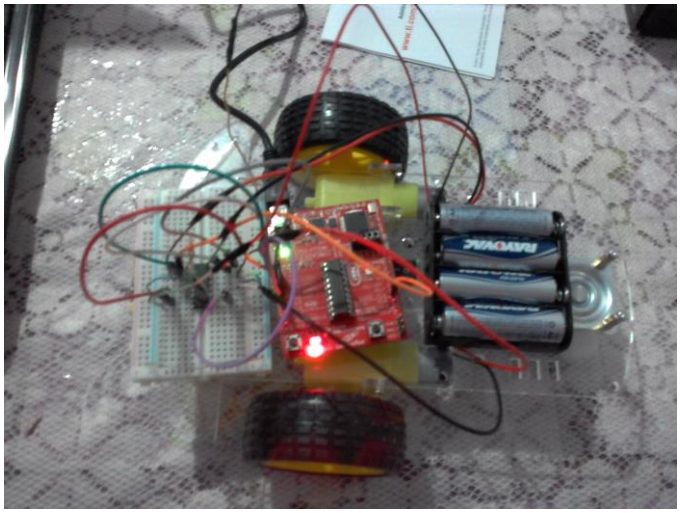

Figura 04a. Robô seguidor de linha – vista superior.

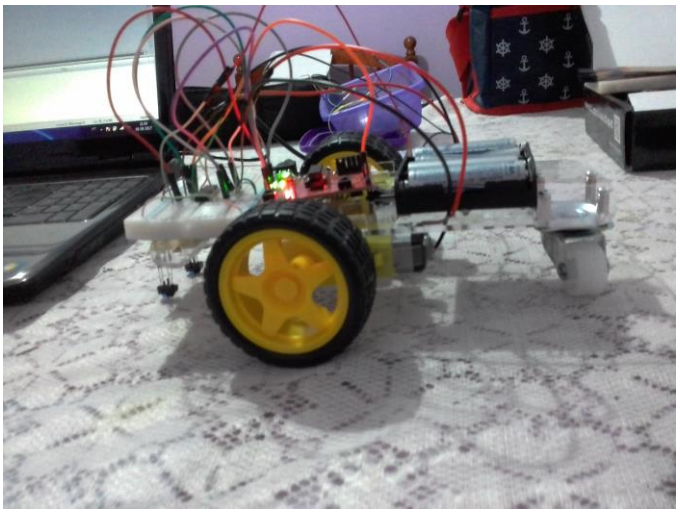

Figura 04b. Robô seguidor de linha – vista lateral.

Com o robô construído, e utilizando-se o código mostrado no Anexo A, verificou-se o funcionamento do controle de velocidade dos motores através do botão da MSP. Conforme mostrado no código, o motor possui 5 velocidades, funcionando com 50% da capacidade total, 62,5%, 75%, 87,5% e 100%, além do repouso, 0%.

Apesar de já estarem acoplados ao chassi, os sensores ópticos ainda não estão sendo utilizados, pois os conhecimentos adquiridos até o presente momento na disciplina de Microprocessadores e Microcontroladores não são suficientes para a realização da conversão A/D dos dados lidos pelos sensores.

V. Conclusão

Com o trabalho realizado até o presente momento e o conteúdo apresentado em Microprocessadores e Microcontroladores, fica cada vez mais evidente a possibilidade de sucesso na realização do projeto. Assim sendo, a dupla visa, para o próximo ponto de controle, estar capacitada para utilizar os sensores ópticos reflexivos, realizando testes com o robô andando em trajetórias demarcadas com faixas pretas.

É possível concluir ainda, que embora não haja tempo hábil e fuja do contexto, os conhecimentos adquiridos na disciplina, capacitam a dupla para construir um robô seguidor de linha com mais funções do que as apresentadas nessa proposta de projeto, como por exemplo, programar o robô para realizar a limpeza personalizada de um ambiente com vários cômodos (i.e. limpeza mais rápida no cômodo 1, mais lenta no cômodo 2, etc.).

VI. Anexos

A) Código:

a. Função em Assembly Motor_Direction

```
#include "msp430g2553.h"
        NAME  Motor_Direction
        EXTERN          rand
```

```
;=========================================
=======
; Motor_Direction
;=========================================
=======
        PUBLIC Motor_Direction          ; Declare symbol
to be exported
        RSEG   CODE                     ;      Code      is
relocatable


Motor_Direction:
;============================
;       Definition: The function's objective is to controls the
motors direction
;     when the sensors TCRT5000 change their values passing
under a black line.
;       It controls the robot movement turning to the left or to
the right.
;
; - Registers Convention
;       R12 = c1
;       R13 = c2
;       The function returns the value on R12
;
; - Table ASCII Convention
;       R = 0x52 (meaning: RIGHT)
;       L = 0x4C (meaning: LEFT)
;       S = 0x53 (meaning: STOP)
;
; - IC L293B configuration
;
;       RIGHT RIGHT |  RIGHT LEFT  | LEFT RIGHT  |
STOP STOP
;     INPUT1 = 1  | INPUT1 = 0  | INPUT1 = 1 | INPUT1 =
1
;     INPUT2 = 0  | INPUT2 = 1  | INPUT2 = 0 | INPUT2 =
1
;     INPUT3 = 1  | INPUT3 = 1  | INPUT3 = 0 | INPUT3 =
1
;     INPUT4 = 0  | INPUT4 = 0  | INPUT4 = 1 | INPUT4 =
1
;
; - Corresponding BITs in P1
;     P1.0 - INPUT1
;     P1.1 - INPUT2
;     P1.2 - INPUT3
;     P1.4 - INPUT4
;================================
;================================
;if (c1 == 'R' && c2 == 'R')
;================================
if_1_Motor_Direction:     CMP.B  #0x52, R12
        JNE     if_3_Motor_Direction
        CMP.B  #0x52, R12
        JNE     if_2_Motor_Direction
; return the value 00101 for right right
        MOV.w #0x5,     R12
        RET
;================================
;else if (c1 == 'R' && c2 == 'L')
;================================
if_2_Motor_Direction:
; return the value 10001 for right left
     MOV.W   #0x11, R12
        RET
;================================
;  else if (c1 == 'L' && c2 == 'R')
;================================
if_3_Motor_Direction:
     CMP.B      #0x4C, R12
        JNE     if_4_Motor_Direction
; return the value 00110 for right left
        MOV.W#0x6,     R12
        RET
;================================
;  else if (c1 == 'S' && c2 == 'S')
;================================
if_4_Motor_Direction:
; return the value 10111 for stop
        MOV.W#0x17, R12
        RET

     END
```

b. Função Control_Speed()

```
#include<msp430g2553.h>
#define BTN BIT3
volatile unsigned int com=0;

void Control_Speed(int Speed)
{
   if(Speed == 0)
   {
    TACTL = TACLR;
    TACCR0 = 10000; //100 hz
    TACCR1 = 0;
    TACCTL1 = OUTMOD_7;
    TACTL = TACTL = TASSEL_2 + ID_0 + MC_1;
   }
   else if(Speed == 1)
   {
    TACTL = TACLR;
    TACCR0 = 10000; //100 hz
    TACCR1 = 5000; //50% of the Duty Cycle
    TACCTL1 = OUTMOD_7;
    TACTL = TACTL = TASSEL_2 + ID_0 + MC_1;
   }
   else if(Speed == 2)
   {
    TACTL = TACLR;
    TACCR0 = 10000; //100 hz
    TACCR1 = 6250; //62.5% of the Duty Cycle
```

```
    TACCTL1 = OUTMOD_7;
    TACTL = TACTL = TASSEL_2 + ID_0 + MC_1;
   }
   else if(Speed == 3)
   {
    TACTL = TACLR;
    TACCR0 = 10000; //100 hz
    TACCR1 = 7500; //75% of the Duty Cycle
    TACCTL1 = OUTMOD_7;
    TACTL = TACTL = TASSEL_2 + ID_0 + MC_1;

   }
   else if(Speed == 4)
   {
    TACTL = TACLR;
    TACCR0 = 10000; //100 hz
    TACCR1 = 8750; //87.5% of the Duty Cycle
    TACCTL1 = OUTMOD_7;
    TACTL = TACTL = TASSEL_2 + ID_0 + MC_1;
   }
   else if(Speed == 5)
   {
    TACTL = TACLR;
    TACCR0 = 10000; //100 hz
    TACCR1 = 10000; //100% of the Duty Cycle
    TACCTL1 = OUTMOD_7;
    TACTL = TACTL = TASSEL_2 + ID_0 + MC_1;
   }
}
```

c. Função Principal main ()

```
#include<msp430g2553.h>
#define BTN BIT3
volatile unsigned int com=0;

extern int Motor_Direction(char c1,char c2);

int main(void)
{
  WDTCTL = WDTPW | WDTHOLD;
  BCSCTL1 = CALBC1_1MHZ;        //MCLK e SMCLK @ 1MHz
  DCOCTL = CALDCO_1MHZ;         //MCLK e SMCLK @ 1MHz
  P1OUT = 0;/*Inicializando as saidas em 0*/
  P1DIR |= 0x57; /*P1DIR = 01010111*/
  P1DIR &= ~BTN;
  P1OUT |= BTN; /*Definindo o resistor de pull-up e pull-down*/
  P1REN |= BTN; //Definindo como pull-down
  P1SEL |= BIT6; //Usando a funcao TA0.1 na porta P1.6
  P1SEL2 &= ~BIT6;//Usando a funcao TA0.1 na porta P1.6
  P1IES |= BTN; //Habilita Interrupcao na borda de descida
  P1IE |= BTN; // Habilita Interrupcao na porta P1.0
  Control_Speed(0); //Inicializar os motores parados
  P1OUT |= Motor_Direction('R','R'); // Motores girando para a direita
  _BIS_SR(GIE + LPM0_bits);
}

#pragma vector = PORT1_VECTOR
__interrupt void Port_1(void)
{

   volatile unsigned int i = 60000;

   //Controle de Velocidade dos Motores
   if((P1IN&BTN)==0)
   {
    if(com!=5)
    {
     com++;
     Control_Speed(com);
     while(i--);
    }
    else
    {
     com = 0;
     Control_Speed(0);
     while(i--);
    }
   }
}
```

REFERÊNCIAS

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

[2] Laboratório de Garagem, Carrinho seguidor de linha que desvia de obstáculos com plataforma Zumo e Arduino. Disponível em: <http://labdegaragem.com/profiles/blogs/tutorial-carrinho-seguidor-de-linha-que-desvia-de-obstaculos-com-> Acesso em 04 de Abril de 2017.

[3] Apostila: Oficina seguidor de linha. Vieira, Gabriel Meneses.

[4] McRoberts, Michael. Arduino Básico. [tradução Rafael Zanolli]. São Paulo: Novatec Editora, 2011.