Ponto de Controle 2

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 II 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 4 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-se de 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.

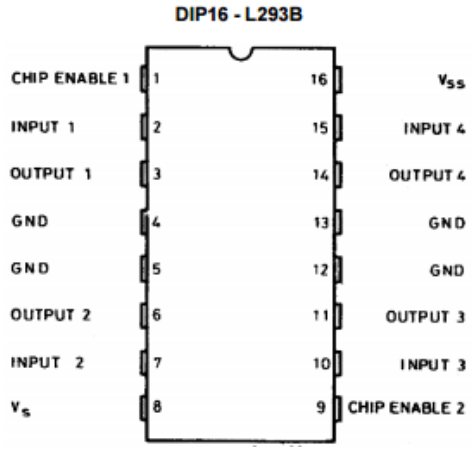

Figura 01. Pinagem do CI L293B

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.

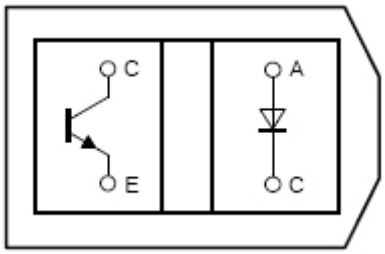

Figura 02. Vista superior do sensor TCRT5000.

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. 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**
BIT 0 (PORTA P1.0)	ENABLE 1
BIT 1 (PORTA P1.1)	ENABLE 2
BIT 2 (PORTA P1.2)	INPUT 1
BIT 3 (PORTA P1.3)	INPUT 2
BIT 4 (PORTA P1.4)	INPUT 3
BIT 5 (PORTA P1.5)	INPUT 4

Além disso, para utilização dos dois sensores, definiu-se que os bits 6 (PORTA P1.6) e 7 (PORTA P1.7) do registrador P1IN da MSP fossem conectados aos sensores 1 e 2, respectivamente.

Na função

```
int Motor_Direction(char string1[],char string2[])
```

foi definido o controle de direção dos motores. A função recebe dois argumentos, a primeira string chamada de string1 refere-se ao motor 1, e tem dois modos possíveis: **Right** e **Left**, que se refere a direção requerida para o motor, se é direita ou esquerda. A segunda string também possui esses mesmos dois modos, mas para o controle do motor 2. Na função `Motor_Direction`, existe quatro casos possíveis para direção:

- Right, Right;
- Right, Left;
- Left, Right.
- Left, Left.

Veja que a forma que ficou implementado a função foi utilizando a lógica *if-else-if*, que varre todas as condições possíveis. Um exemplo de condição está abaixo:

```
if (string1 == "Right" && string2 == "Right")
      {
        /*P1OUT = 00010111*/
        return  0x17;
       }
```

A função retorna 0x17, que significa 00010111_2.É possível verificar que conforme a Tabela 02, esta combinação de bits permite que os dois motores sejam acionados e girem no sentido horário. Portanto a função retorna um parâmetro para P1OUT no código principal.

Já a função

```
int Detectar_Sensor(void)
```

tem o propósito de selecionar o modo de direção do motor conforme o acionamento dos sensores. Neste caso, se o sensor do motor 1 for acionado, o motor 1 deve girar para a esquerda enquanto o motor 2 deve girar para a direita. Se o sensor do motor 2 for acionado, o motor 2 deve girar para a esquerda e o motor 1 manter-se girando para a direita. Isso serve para garantir que o robô irá virar na hora de detectar uma linha. Veja a utilização nesta linha

```
if(analogRead(A6) > 512)
   /*Se detectarmos uma linha no sensor1, deve-se
virar à esquerda*/
    {
      return      Motor_Direction("Left","Right");
//vire a esquerda
    }
```

A explicação da função **analogRead** vem logo a seguir.

- **Analogread**

Analogread () é uma função do energia que lê o valor de um pino analógico especificado. A placa LaunchPad tipicamente contém 8 canais, 10-bit conversor analógico para digital, isto significa que ela mapeará as tensões de entrada entre 0 e 3 volts em valores inteiros entre 0 e 1023. Assim é possível saber o valor de tensão em um pino através da simples divisão: 3 volts / 1024 unidades ou 0,0029 volts (2,9 mV) por unidade. A faixa de entrada e a resolução podem ser alteradas usando a função analogReference ().

Demora cerca de 100ms para ler uma entrada analógica, então a taxa de leitura máxima é cerca de 10.000 vezes por segundo.

É possível então observar que a função analogread é importante para o projeto em questão, pois será útil para a leitura do valor de tensão no sensor óptico.

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, e conectando o mesmo ao código mostrado no Anexo A através da MSP430, foi possível simular as seguintes funções do robô:

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

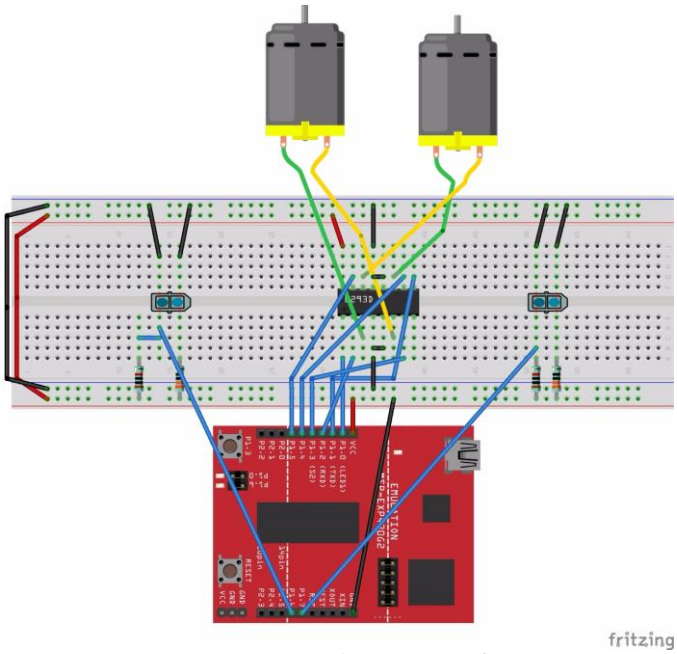

Figura 03. Simulação de algumas funções do robô seguidor de linha.

Dado o sucesso obtido nas simulações, conclui-se que é possível realizar a efetiva montagem do robô.

V. Conclusão

Com o trabalho realizado até o presente momento e os conhecimentos adquiridos na disciplina de Microprocessadores e Microcontroladores, é possível concluir que o robô seguidor de linha para auxílio na limpeza industrial é um projeto factível e coerente com o conteúdo apresentado na disciplina, uma vez que além de ser uma solução para um problema real, agrega conhecimento à dupla, pois é a oportunidade de colocar em prática os conceitos aprendidos.

VI. ANEXOS

A) Código:

```
#include<msp430g2553.h>
int Motor_Direction(char string1[],char string2[])
{
if (string1 == "Right" && string2 == "Right")
  {
    /*P1OUT = 00010111*/
    return  0x17;
  }
  else if (string1 == "Right" && string2 == "Left")
  {
    /*P1OUT = 00100111*/
    return  0x27;
  }
  else if (string1 == "Left" && string2 == "Left")
  {
    /*P1OUT = 00101011*/
    return  0x2B;
  }

  else if (string1 == "Left" && string2 == "Right")
  {
    /*P1OUT = 00011011*/
    return  0x1B;
  }
}
int Detectar_Sensor(void)
{
  if(analogRead(A6) > 512)
  /*Se detectarmos uma linha no sensor1, deve-se virar a esquerda*/
  {
    return Motor_Direction("Left","Right"); //vire a esquerda
  }
  else if(analogRead(A7) > 512)
  {
  /*Se detectarmos uma linha no sensor2, deve-se virar a direita*/
    return Motor_Direction("Right","Left"); //vire a direta
  }
  else
  {
    return  Motor_Direction("Right","Right"); //siga reto
  }
}

int main(void)
{
  WDTCTL = WDTPW | WDTHOLD;
  P1OUT = 0;
  P1DIR |= 0x3F; /*P1DIR = 00111111*/
  for(;;)
  {
    P1OUT = Detectar_Sensor();
  }

}
```

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.