Faculdade de Engenharia da Universidade do Porto

Mestrado Integrado em Engenharia Electrotécnica

e de Computadores

Projeto final: Robô Seguidor de Linha

Ana Sofia Teixeira – 201806466

Jéssica Ferreira de Oliveira – 201806863

24 de janeiro 2021

Índice

Introdução

Este projeto tem como objetivo principal a construção de um robô de configuração diferencial que seja capaz de seguir uma linha preta presente no chão.

Com este projeto abordou-se temas como as interrupções e conversores A/D. O conversor AD converte uma tensão de input analógica para um valor digital de 10 bits por sucessivas aproximações. O valor mínimo representa o GND (ground) e o valor máximo representa a tensão de referência menos 1 LSB (less significant bit).

O robô possui um sistema de direção e tomada de decisões operado pelo microcontrolador Atmega328p, a partir de informações que são recolhidas por sensores periféricos.

O sistema para a deteção da linha conta com o uso de sensores de luz infravermelha para detetar o sinal refletido a partir de um díodo emissor apropriado. O robô irá seguir a trajetória correta uma vez que o preto absorve a luz e o branco irá refletir.

O movimento do robô é realizado através de motores DC alimentados por uma bateria, implementando-se o timer no modo PWM de forma a conseguirmos controlar os motores.

Materiais utilizados

1. Arduino Uno
2. Motor DC
3. DC motor driver 2 canais
4. Sensor Infravermelho
5. Estrutura impressa em 3D

Figura 1 - Motor DC

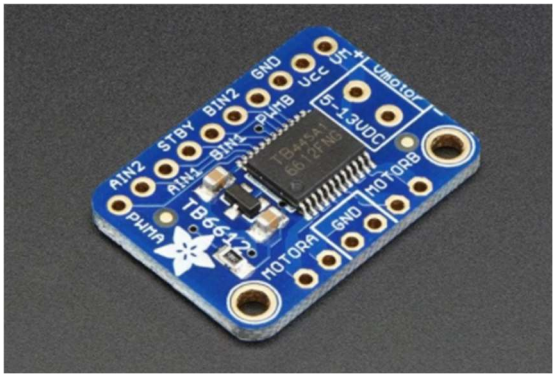


Figura 2 - DC motor driver- 2 canais

Figura 3 – Sensor infravermelho

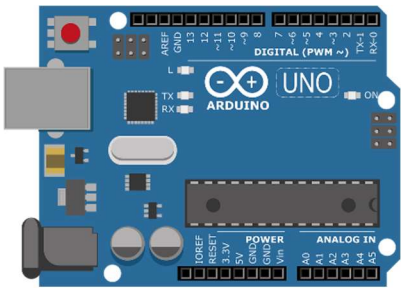


Figura 4 – Arduino Uno

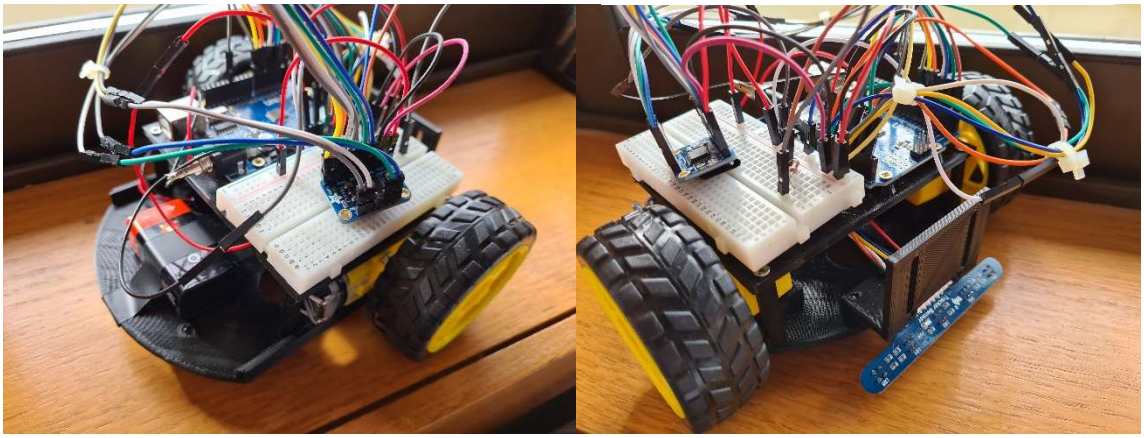


Figura 5 – Robô finalizado

Procedimento

1. Funcionamento do Robô

O principal objetivo deste projeto é fazer um robô capaz de seguir uma linha, recorremos a um array de sensores para que este pudesse detetar a linha e tomar as devidas ações.

O robô utiliza um sensor infravermelho de 5 canais (5CH-ITR20001/T) que realiza a deteção da linha, um microcontrolador Atmega328P integrado a uma placa Arduino Uno que recebe os valores de leitura dos sensores, realiza os cálculos necessários e envia sinais PWM para a ponte H (TB6612FNG), que recebe esses sinais realiza seus cálculos lógicos e envia sinais aos motores, que irão movimentar o robô de acordo com os sinais recebidos.

2. Controlo de motores: Modo PWM (Pulse Width Modulator)

Para controlar a velocidade das rodas, gerou-se uma onda quadrada através do modo PWM. Utilizou-se o timer0 e o timer 2 no modo Fast PWM. O timer 0 associado ao motor A e o timer 2 associado ao motor B. Escolheu-se o modo non-inverting, ou seja, elimina-se os valores OC0A e OC2B (iguala-se ao valor BOTTOM) quando a contagem dos timers for igual ao valor OC0A e OCR2B.

```
void pwmA_motor() {
  TCCR0A |= 1<<(COM0A1) | 1<<(WGM00) | 1<<(WGM01);
  TCCR0B |= 1<<(CS00);
}
int pwmA(int v)  {
  OCR0A = v;
  return 0;
}
void pwmB_motor() {
  TCCR2A |= 1<<(COM2B1) | 1<<(WGM20) | 1<<(WGM21);
  TCCR2B |= 1<<(CS20);
}
int pwmB(int v) {
  OCR2B = v;
  return 0;
}
```

Os valores do OCR0A e OCR2B, responsáveis pelo andamento do robô, são definidos tendo por base a correção da posição à saída do controlador PID:

```
int speedA = base_power - power_error;
if(speedA>240) speedA=240;
if(speedA<0)  speedA=0;
int speedB = base_power + power_error;
if(speedB>240) speedB=240;
if(speedB<0)  speedB=0;
pwmA(speedA);
pwmB(speedB);
```

3. Controlo Proporcional Integral e Derivativo (PID)

Um controlador PID é uma técnica de controlo de processos que une três ações: derivativa, integral e proporcional.

Conseguimos perceber pela imagem que a variável que é controlada pelo PID, tem o seu valor de saída sempre a ser comparado com um SetPoint que, no nosso trabalho denominamos como "*Base_Power*".

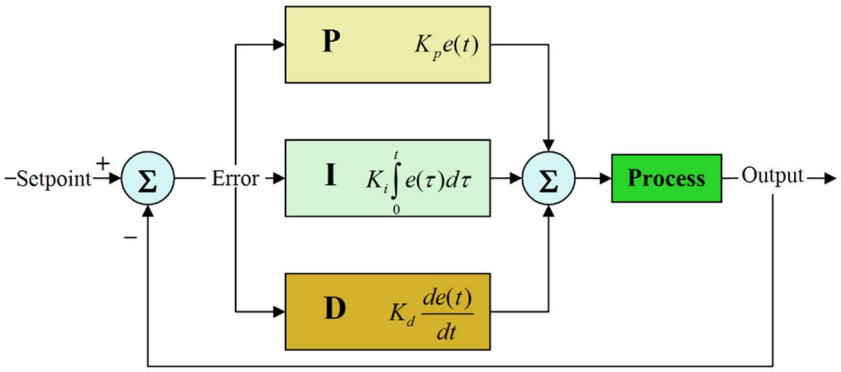


Figura 6 - Controlo PID

$$PID = Kp \times e(t) + Ki \times \int e(t)\,dt + Kd \times \frac{de(t)}{dt}$$

Onde Kp, Ki e Kd são constantes que foram definidas no projeto por métodos de tentativa/erro.

Inicialmente as constantes Ki e Kd foram igualadas a zero, e a constante Kp foi regulada até obtermos uma resposta rápida do robô a seguir a linha, isto é, pretendíamos que o SetPoint fosse alcançado com rapidez o que provoca oscilações.

Uma vez determinada esta constante, foi-se aumentando Ki até diminuir as oscilações de forma razoável.

Tendo então Kd e Ki definidos, foi-se ajustando Kd até o robô seguisse a linha de forma satisfatória.

A ação proporcional (P) é aproximadamente proporcional à posição do robô em relação à linha. Isto é, se o robô estiver centrado na linha então o valor proporcional ira ser 0. Se estiver à esquerda da linha então será um número positivo e à direita da linha, será negativo. Isto é calculado a partir do valor "pos" calculado da seguinte forma:

$$pos = \frac{(-2000 \times value0 + -1000 \times value1 + 0 \times value2 + 1000 \times value3 + 2000 \times value4)}{value0 + value1 + value2 + value3 + value4}$$

Este valor é então uma estimativa feita usando uma média ponderada dos índices do sensor (valueN, sendo N=0,1,2,3 ou 4) multiplicados por 1000 e subtraindo no mesmo momento 2000. Os valores intermédios significarão que a linha está entre dois sensores. Ficamos então com:

$$proportional = pos$$

A ação integral (I) tem como função registar o histórico do movimento do robô, isto é, uma soma de todos os valores do termo proporcional que foram guardados desde o início do funcionamento.

$$integral = last\_integral + proportional$$

A ação derivativa (D) é, por fim, a taxa de variação do valor proporcional. No projeto calculamos este valor como sendo a diferença entre os dois últimos valores da componente proporcional.

$$derivative = proportional - last\_proportional$$

Após isto temos uma variável ("*power_error*") que irá combinar estes valores com as constantes definidas, é este valor que é usado para determinar a velocidade dos motores.

$$power\_error = proportional \times Kp + Integral \times Ki + Derivative \times Kd$$

Por fim, temos a nossa função PID definida da seguinte forma:

```
int PID(int posicao,int* last_proportional, int* last_integral, int* last_derivative)  {
  int proportional = posicao;
  int integral = *last_integral + proportional;
  int derivative = proportional - *last_proportional;
  *last_integral = integral;
  *last_proportional = proportional;
  *last_derivative = derivative;
  int power_error = proportional * Kp + integral * Ki + derivative * Kd;
  return power_error;
}
```

4. Sensor infravermelho para deteção da linha (Conversor AD)

Para poder localizar a linha preta do circuito em relação ao carro, utilizou-se uma linha de sensores infravermelhos constituída por cinco LEDs e fotodíodos. A linha de sensores possui cinco outputs analógicos que foram conectados aos pinos PC0 a PC4.

Este sensor fica colocado na parte dianteira do robô, com os sensores apontados para o chão de forma a fornecer uma leitura precisa da reflexão da luz sobre a superfície.

Quando é detetada uma superfície com uma maior capacidade refletora (uma superfície branca por exemplo) o output resultante de cada sensor é de valor mais elevado do que quando é detetada uma superfície preta (por exemplo), cuja capacidade refletora é significativamente mais baixa.

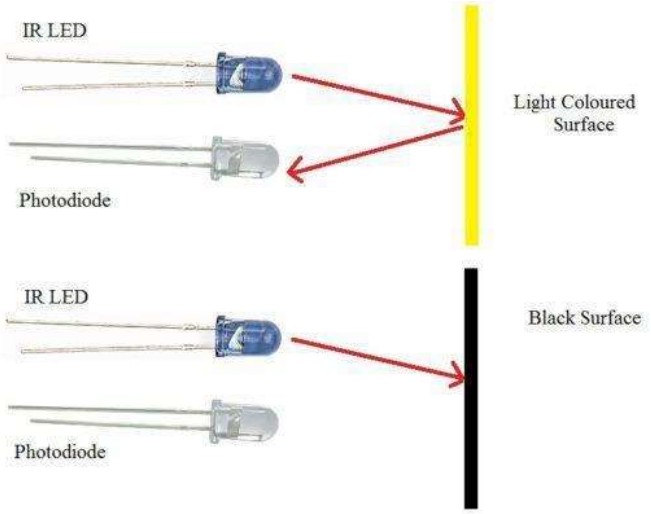


Figura 7 - Funcionamento base do sensor infravermelho

Desta forma, pode-se obter a posição da linha preta através de uma normalização do valor lido por cada sensor da linha e, consecutivamente, de uma média pesada.

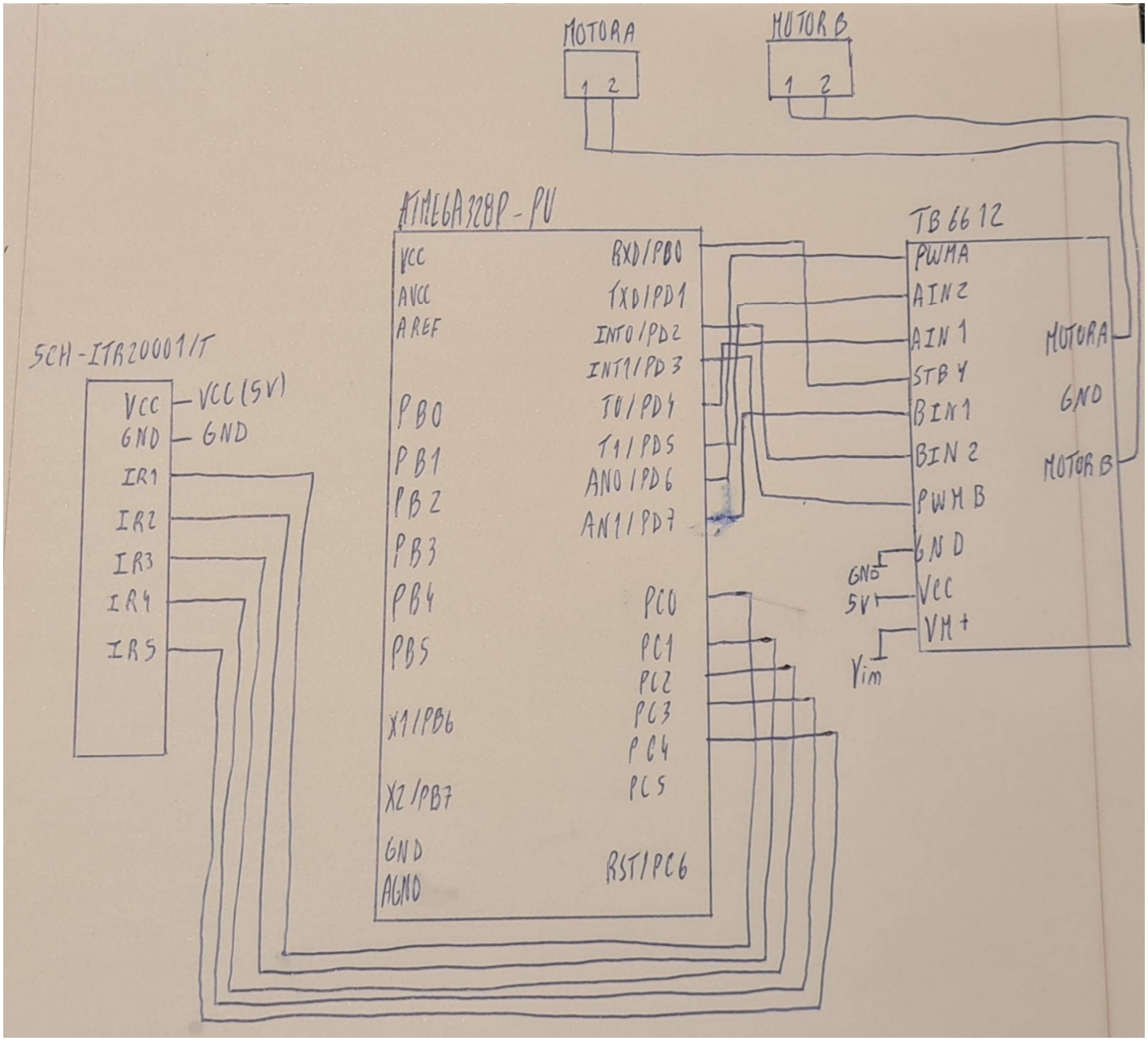
Posteriormente, foi feita uma normalização de cada valor lido por cada sensor com o intuito de reduzir o efeito dos fatores ambientais que os possam afetar e, por fim, foi feita uma média pesada de forma a transformar os valores normalizados provenientes de cada sensor num valor que represente a distância do centro do carro à linha preta do circuito.

```
int  value0 =1024- ADCREAD(0);
int  value1 =1024- ADCREAD(1);
int  value2 =1024-ADCREAD(2);
int  value3 = 1024-ADCREAD(3);
int  value4 = 1024-ADCREAD(4);
```

$$posicao = \frac{(0 * (long)value0 + 1000 * (long)value1 + 2000 * (long)value2 + 3000 * (long)value3 + 4000 * (long)value4)}{(value0 + value1 + value2 + value3 + value4)}$$

Onde o value0, value1, value2, value3 e value4 correspondem às variáveis onde estão armazenados os valores normalizados de cada sensor e os valores 0, 1000, 2000, 3000 e 4000 correspondem ao peso de cada posição, sendo o valor central, idealmente, 2000.

Esquemático Elétrico



Conclusão

Este projeto serviu como consolidação de conhecimentos, uma vez que são usados vários temas abordados nas aulas desta unidade curricular e permitiu assim pôr em prática os mesmos.

Neste projeto foi então desenvolvido um robô seguidor de linha que funcionou em todas as situações de teste com uma velocidade boa, exceto quando se encontrava numa situação de "curva e contracurva" apertada, fenómeno que era possível contornar fazendo ligeiras alterações ao código de controlo dos motores DC.

Implementamos um controlador PID e observamos diretamente as diferenças quando as constantes Kp, Kd e Ki variavam. O controlador proporcional permitiu ao carro mudar a direção, os controladores integral e derivativo fizeram com que os movimentos do carro fossem mais suaves.

O microcontrolador possui ainda entradas e saídas de dados não utilizadas, o que permite ainda a expansão do número de sensores ou outros dispositivos. O Arduino também possui memória disponível para esta mesma expansão.

Algo que foi possível perceber também foi a necessidade de automatizar ainda mais o processo de controlo, por exemplo, implementar o envio de forma remota dos valores de velocidade ou dos parâmetros do PID cada vez que se sentisse a necessidade de alterar estes valores, usando por exemplo, uma aplicação para smartphone que comunicaria com o robô através de um módulo Bluetooth.

Uma outra aplicação extra, poderia ter sido a implementação de um controlo de start&stop através de um controlo remoto.

Concluímos assim, que este projeto poderá servir como base para outros projetos mais complexos e dinâmicos, já que existe bastante potencial de evolução.

Webgrafia

[1] https://www.botnroll.com/en/controllers/1957-adafruit-tb6612-12a-dcstepper-motor-driver-breakout-board-.html

[2] https://cdn-shop.adafruit.com/datasheets/TB6612FNG_datasheet_en_20121101.pdf

[3] https://www.botnroll.com/en/infrared/2586-tracker-sensor-infrared-line-tracking.html

[4] https://www.waveshare.com/wiki/Tracker_Sensor

[5] http://sistemaolimpo.org/midias/uploads/c9265a4069821ae565485c39b07d0902.pdf

[6] https://www.pololu.com/docs/0J18/19