
# Relatório do Mini Projeto

# Robot com Seguimento de Linha e Desvio de Obstáculos

**Tomás Lopes up201805155@up.pt**
**João Pedro up201805075@up.pt**
**17/02/2022**

**Arquiteturas de Computação Embarcada**

**Professor Paulo Costa**

# Conteúdo

# 1 Introdução

Um robot com seguimento de linha é uma máquina capaz de seguir um caminho predeterminado como, neste caso, uma linha escura numa superfície clara. Para desenvolver tal sistema precisamos de algo para determinar de forma eficaz a posição da linha e que reponda de forma proporcional a mudanças neste posicionamento garantindo que a trajetória da linha é assegurada. Esta aplicação pode ser usada nas mais diversas áreas, desde carros autónomos até automatização industrial. Neste projeto desenvolvemos um robot capaz de seguir uma linha, apresentando, além disso, desvio de obstáculos.

# 2 Montagem e Material

É importante começar em primeiro lugar por escolher os materiais que serão usados para construir um sistema capaz de executar as funções pretendidas.

Para este projeto vamos então necessitar de um microntrolador, capaz de executar todas as tarefas necessárias responsáveis pelo seguimento de linha, um sensor de linha, responsável por determinar a posição do robot na linha, motores e respetivo driver para permitir movimentação do robot, um sensor de distância para que possamos detetar possíveis obstáculos e alguma forma de alimentar todos estes componentes.

- **Sensor de Linha (5-ch ITR20001/T):** um bom sensor para determinar a posição do robot na linha são sensores infra-vermelho. Estes sensores são compostos por um díodo emissor(IR) e outro recetor(IR). e pemitem que seja introduzido um contraste entre a linha e o resto do ambiente senso que um deles reflete e o outro absorve a radiação. Para a nossa aplicação usamos um sensor compostos por um array de 5 sensores IR que permite ter uma melhor ideia do posicionamento do robot quando comparado ao standard que seria usar um destes de cada lado

- **Motores (2x SKU FIT0450):** para permitir a locomoção do robot precisamos de motores. Estes motores são DC e apresentam uma rotação máxima de cerca de 160 RPM que será mais que suficiente tendo em consideração as dimensões da nossa aplicação. Algo que decidimos foi também que estes motores tivessem um encoder o que permite determinar a direção e velocidade a que o robot se está a movimentar permitindo posteriormente aplicar técnicas de odometria para determinar parâmetros como velocidade, distância percorrida, etc...

- **Driver (TB6612FNG ):** com o objetivo de fornecer aos motores os sinais necessários para que possamos controlar a sua velocidade e direção de rotação usamos um driver de motor que deverá ser capaz de controlar dois motores em simultâneo

- **Sensor de Distância (VL53L0X):** a deteção de obstáculos pode ser feita recorrendo a sensores de distância. Neste caso optmamos por usar um sensor ToF (time of flight) que apresenta elevada precisão bem como é capaz de efetuar leituras rápidas quando comparado com sensores ultrasom já que neste caso um feixe de luz é utilizado para efetuar a medição

- **Alimentação (2x 18650 3.7V 2200mAh):** todos estes equipamentos requerem alimentação para funcionar. Assim, tendo em consideração os limites de tensão optamos por usar 2 pilhas lítio de 3.7V cada, ligadas em série

Lista de equipamentos:

- Sensor de Linha - 5-ch ITR20001/T
- Motor (2x) - SKU FIT0450
- Driver - TB6612FNG
- Sensor Distância - VL53L0X
- Alimentação (2x) - 18650 3.7V 2200mAh

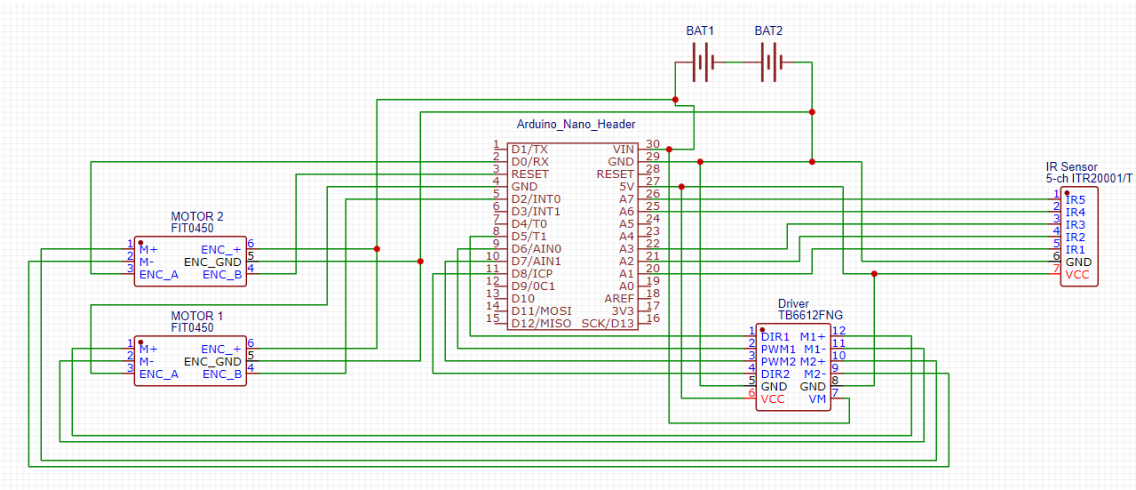


Figura 1: Diagrama da Montagem

# 3 Implementação

Tido o hardware, podemos avançar para o código que devemos implementar para que possamos dar uso a toda a informação recolhida dos sensores e que permite então seguimento de linha. Para isso devemos encontrar primeiro a melhor solução para nos encontrarmos na linha.

## 3.1 Posicionamento na Linha

A solução mais usada e implementada na livraria *QTRSensors.h* envolve atribuir um peso à leitura de cada sensor, somando em seguida todos os valores:


Figura 2: Determinação da Posição na Linha

Desta forma sabemos que se a leitura for 2000, o carro está centrado mas se for menor ou maior são necessários ajustes. Estes ajustes poderão ser introduzidos recorrendo a um controlo PID.

## 3.2 Controlo PID

O controlo PID implementado é bastante básico sendo que inclui um termo de erro (P), dado pela diferença entre a posição ideal na linha (2000) e posição atual na mesma, outro termo que envolve a diferença entre o erro atual e o erro anterior (D) e um termo que corresponde a um somatório do erro (I). Cada um destes termos é depois multiplicado por uma constante.

$$PID = P * Kp + I * Ki + D * Kd \tag{1}$$

Estabelecendo uma velocidade base para o carro, podemos adicionar este valor a um motor e subtrair ao outro a cada iteração do loop. Com isto o robot irá automaticamente ajustar-se de forma a que se matenha no topo da linha sendo agora o maior desafio determinar as constantes certas a usar. Na nossa experiência estes valores foram: Kp = 10, Ki = 0 e Kd = 2.

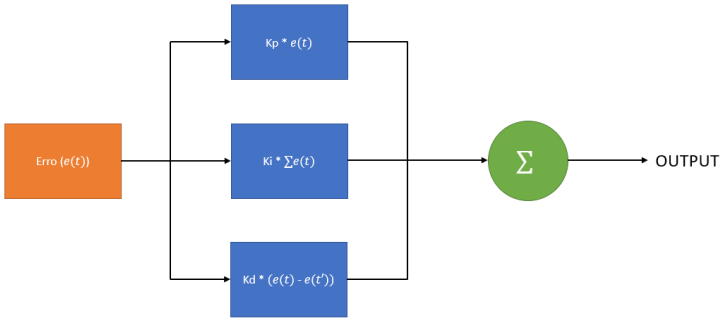


Figura 3: Diagrama do Controlo PID

É importante mencionar nesta fase que devemos ter cuidado no que toca à velocidade do robot espcialmente visto que os motores podem rodar em ambas as direções. Ao não ser cautelosos o robot poderá alterar quase instantâneamente a sua direção mantendo uma velocidade elevada o que poderá não só danificar o motor como provocar um overload nos motores que fará com que o robot perca controlo.

## 3.3 Deteção de Obstáculos

Outro pont importante adicionar a este sistema é deteção de objetos que é um processo relativamente simples, basta adquirir a distância medida pelo sensor VL53L0X e verificar se esta é menor do que um determinado threshold. Se for menor podemos então parar o robot ou densenvolver um sistema que permita que o robot contorne os obstáculos.

## 3.4 Contorno de Obstáculos

Como mencionado anteriormente, pretendemos agora que o robot contorne obstáculos que se encontrem no seu caminho. Isto poderá ser feito de uma forma simples caso o obstáculo se encontre numa reta:

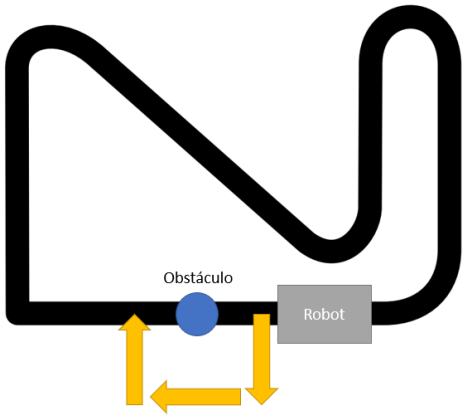


Figura 4: Contorno de Obstáculo

Este processo consiste em rodar o robot 90 graus para o lado esquerda, seguir em frente, rodar 90 graus para o lado direito, seguir em frente, rodar 90 graus para o lado direito e seguir em frente até encontrar a linha passando novamente ao controlo PID. Estes movimentos são facéis de obter já que recorrendo à odometria conseguimos obter a distância percorrida e o ângulo do robot com bastante precisão (devemos garantir que existe tração suficiente, caso contrário estas medições serão incorretas).

## 3.5 Cruzamentos

Um último pormenor que poderíamos adicionar a este sistema é a capacidade de decidir o caminho que deve tomar em cruzamentos e interseções. Para isso, usamos o nosso conhecimento prévio da pista bem como os dados da odometria para dizer no código que em certos troços pretendemos ignorar um dos caminhos. Isto não foi no entanto implementado por causa de um falha que tivemos num dos encoder que tornou impossível recorrer à odometria.

# 4 Problemas

Ao testar as implementações notamos algums problemas que comprometeram o funcionamento do robot. Um deles veio da falta de tração que o robot apresentava, especialmente tendo em consideração a superfície da pista o que tornava díficil que este executa-se curvas apertadas. A solução para este problema foi adicionar algum peso ao robot colocando um pisa papéis no topo.
Mais à frente quando começamos a tentar usar os encoders apercebemo-nos também que o encoder de um dos motores não funcionava corretamente o que nos impediu de obter dado de odometria que seria usada para controlar o robot de forma mais eficaz bem como para o guiar em interseções.

# 5 Resultados

## 5.1 Seguimento de Linha

O seguimento de linha através do controlo PID mostrou-se um sucesso com o robot a ser capaz de afetuar um grande número de manobras, incluíndo o que é possivelmente o maior desafio que são as curvas apertadas que são conseguidas dado que as rodas são livres de rodar para a frente e para trás fazendo com que o robot consiga rodar em torno de si mesmo. A maior tarefa foi encontrar as devidas constantes. Neste caso acabamos por obter um valor Kp de 10 ao qual adicionamos Kd de 2 que introduziu algum amortecimento evitando um comportamento em zig-zag por parte do robot em torno da linha. No nosso caso optamos por utilizar Ki nulo sendo que nunca favoreceu o comportamento do sistema. Notamos que apesar do sucesso, as constantes do controlo PID poderia ser optimizadas.

## 5.2 Desvio de Obstáculos

Ao desenvolver o desvio de obstáculos apercebemo-nos que o encoder de um dos motores não funcionava de modo adequado já que a leitura da rotação aparentou apresentar um valor máximo. Isto foi reforçado por um barulho proveniente deste mesmo motor o que nos levou a concluir que possivelmente as engrenagens não estariam nas melhores condições. Isto levou a que a opção da odometria recorrendo aos encoders tivesse de ser descartada. Tentamos no entanto fazer uma conversão do sinal PWM usado para obter a sua rotação o que não teve também sucesso já que o mesmo sinal PWM induz comportamentos bastante diferentes com a variação da tensão das pilhas o que tornou o método da odometria não viável. Decidimos então seguir com uma solução que envolve dizer manualmente o sinal a enviar para os motores e descobrimos o tempo necessário para cada manobra. Reconhecemos que esta solução está longe de ser a mais adequada mas apresentou resultados decentes nos nossos testes.

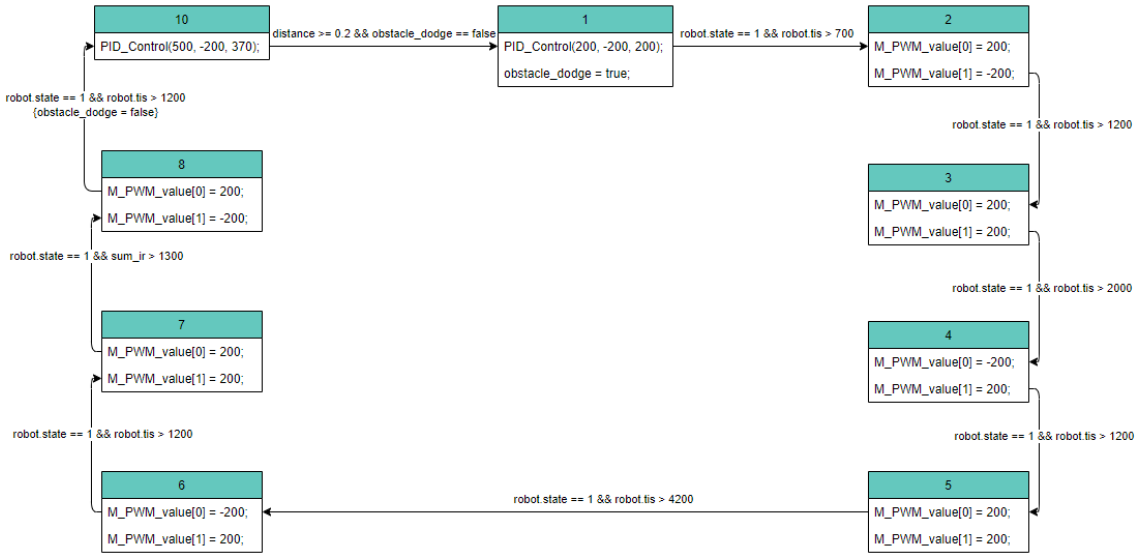


Figura 5: Máquina de Estados para Desvio de Obstáculos

# 6 Conclusão

Neste trabalho implementamos um controlo PID num sistema que permitiu que fizessemos com sucesso o seguimento de uma linha. Tentamos extender este projeto ao adicionar desvio de obstáculos que apesar dos precalços e do sistema ”rudimentar”implementado apresentou também resultados bastante aceitáveis. Não conseguimos contudo que o sistema se soubesse guiar em cruzamentos dado que não encontramos outra solução à falta de odometria.