|  |  |  |
| --- | --- | --- |
|  | **Tutorial – Movimento iModBot@ipleiria.pt** |  |

# Introdução

O robô iModBot é um veículo elétrico de pequenas proporções controlado pelo microcontrolador ESP32. Este microcontrolador pode ser programado usando a linguagem C/C++ e o software gratuito Arduino IDE.

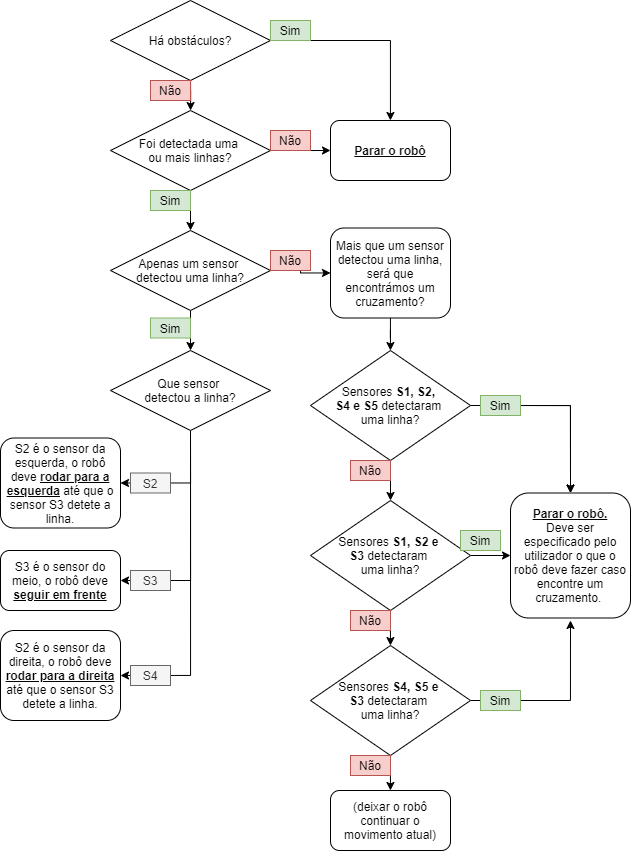
Este tutorial vai focar-se na função de seguir automaticamente um trajeto.

Para efetuar a tarefa de seguir um trajeto autonomamente é necessário verificar constantemente se há algum obstáculo a obstruir o caminho, caso haja é necessário parar os motores.

De seguida é necessário verificar os sensores infravermelho de modo a determinar o percurso que o robô deve seguir.

Embora pareca um processo simples há varias coisas a ter em conta.

O seguinte fluxograma descreve os procedimentos para executar a tarefa de seguir um trajeto (linha) autónomamente.



|  |
| --- |
| **Seguir trajeto – exemplo 1** |
| *Exemplo:*  #include <**iModBot**.h>  **iModBot** robot; // Criar um objeto chamado "robot"  void setup()  {   robot.begin(); // Configura os pinos.     robot.beginAutoDrive(); // Inicia a função de condução autónoma.  }  void loop()  {   int info = 0; // Variável para guardar o numero devolvido pela função autoDrive().   int command = 0; // variável para lembrar que comandos já foram executados.   info = robot.autoDrive(0); // Aqui copiamos o valor devolvido pela função para a variável info.   /\* Sempre que usarem a função autoDrive() têm de enviar um valor, estes são as instruções aceites:       0 -> Enviar 0 ou qualquer outro número (até 255) não listado se não quiser dar instruções ao robô.       1 -> Enviar 1 para colocar o robô a rodar para a direita até encontrar uma linha.       2 -> Enviar 2 para colocar o robô a rodar para a esquerda até encontrar uma linha.       3 -> Enviar 3 para dizer ao robô para seguir em frente.       4 -> Enviar 4 para dizer ao robô para recuar, tenha em atenção que o robô pode-se desalinhar            ligeiramente devido à roda traseira.   \*/  } |
| **Descrição** A biblioteca “RobotOnLine\_Rev2” dispõe de funções que permitem facilmente implementar a tarefa de seguir um trajeto de forma autónoma. **Sintaxe** beginAutoDrive(); // Inicia a função de condução autónoma.  autoDrive(byte ); // Realiza tarefas necessárias para a condução autónoma. **Parâmetros** autoDrive(numero )  a variável número pode ter os seguintes valores:  0 sem efeito.  1 robô roda para a direita até encontrar uma linha.  2 robô a rodar para a esquerda até encontrar uma linha.  3 robô segue em frente.  4 robô recua.  A função autoDrive(byte) pode devolver um dos seguintes valores:  1 multiplas linhas encontradas.  2 obstaculo encontrado.  3 não foram encontradas linhas. **Respostas** Nenhum |

|  |
| --- |
| **Seguir trajeto – exemplo 2** |
| *Exemplo:*  #include <**iModBot**.h>  **iModBot** robot;  void setup()  {  **Serial**.begin(115200);   robot.begin();  }  void loop()  {   if((robot.readS1() == 1) && (robot.readS2() == 1) && (robot.readS3() == 0) && robot.readS1() == 1) &&  (robot.readS2() == 1))   {  robot.forward(110);  while((robot.readS1() == 1) && (robot.readS2() == 1) && (robot.readS3() == 0) && robot.readS1() == 1) &&  (robot.readS2() == 1))  {  ;  }  }    else if((robot.readS1() == 1) && (robot.readS2() == 0) && (robot.readS3() == 1) && robot.readS1() == 1) &&  (robot.readS2() == 1))   {  robot.rotateLeft(110);  while((robot.readS1() == 1) && (robot.readS2() == 0) && (robot.readS3() == 1) && robot.readS1() == 1) &&  (robot.readS2() == 1))  {  ;  }  }  else if((robot.readS1() == 0) && (robot.readS2() == 1) && (robot.readS3() == 1) && robot.readS1() == 1) &&  (robot.readS2() == 1))   {  robot.rotateLeft(110);  while((robot.readS1() == 0) && (robot.readS2() == 1) && (robot.readS3() == 1) && robot.readS1() == 1) &&  (robot.readS2() == 1))  {  ;  }  }  else if((robot.readS1() == 1) && (robot.readS2() == 1) && (robot.readS3() == 1) && robot.readS1() == 0) &&  (robot.readS2() == 1))   {  robot.rotateRight(110);  while((robot.readS1() == 1) && (robot.readS2() == 1) && (robot.readS3() == 1) && robot.readS1() == 0) &&  (robot.readS2() == 1))  {  ;  }  }  else if((robot.readS1() == 1) && (robot.readS2() == 1) && (robot.readS3() == 1) && robot.readS1() == 1) &&  (robot.readS2() == 0))   {  robot.rotateRight(110);  while((robot.readS1() == 1) && (robot.readS2() == 1) && (robot.readS3() == 1) && robot.readS1() == 1) &&  (robot.readS2() == 0))  {  ;  }  }  else if((robot.readS1() == 1) && (robot.readS2() == 0) && (robot.readS3() == 0) && robot.readS1() == 1) &&  (robot.readS2() == 1))   {  robot.rotateLeft(110);  while((robot.readS1() == 1) && (robot.readS2() == 0) && (robot.readS3() == 0) && robot.readS1() == 1) &&  (robot.readS2() == 1))  {  ;  }  }  else if((robot.readS1() == 1) && (robot.readS2() == 1) && (robot.readS3() == 0) && robot.readS1() == 0) &&  (robot.readS2() == 1))   {  robot.rotateRight(110);  while((robot.readS1() == 1) && (robot.readS2() == 1) && (robot.readS3() == 0) && robot.readS1() == 0) &&  (robot.readS2() == 1))  {  ;  }  }  else if((robot.readS1() == 1) && (robot.readS2() == 1) && (robot.readS3() == 0) && robot.readS1() == 0) &&  (robot.readS2() == 0))   {  robot.rotateRight(110);  while((robot.readS1() == 1) && (robot.readS2() == 1) && (robot.readS3() == 0) && robot.readS1() == 0) &&  (robot.readS2() == 0))  {  ;  }  }    else if((robot.readS1() == 0) && (robot.readS2() == 0) && (robot.readS3() == 0) && robot.readS1() == 1) &&  (robot.readS2() == 1))   {  robot.rotateLeft(110);  while((robot.readS1() == 0) && (robot.readS2() == 0) && (robot.readS3() == 0) && robot.readS1() == 1) &&  (robot.readS2() == 1))  {  ;  }  }  else if(((robot.readS1() == 0) && (robot.readS2() == 0) && (robot.readS3() == 0) && robot.readS1() == 0) &&  (robot.readS2() == 0)) || (robot.readCLP() == 1) ||  (robot.distance() <= 10))  {  robot.stopMotors();  }  } |
| **Descrição** Este exemplo permite ao robô seguir um trajeto sem recorrer à função de condução autónoma disponibilizada pela biblioteca “RobotOnLine\_Rev2”  Note que há ciclos while() após cada instrução que afeta o movimento do robô, isto serve para impedir que a instrução seja repetida indefinidamente. **Sintaxe** **Parâmetros**  state = variável que recebe o valor da função, 1 ou 0, consoante o estado do pino. **Respostas** Nenhum |