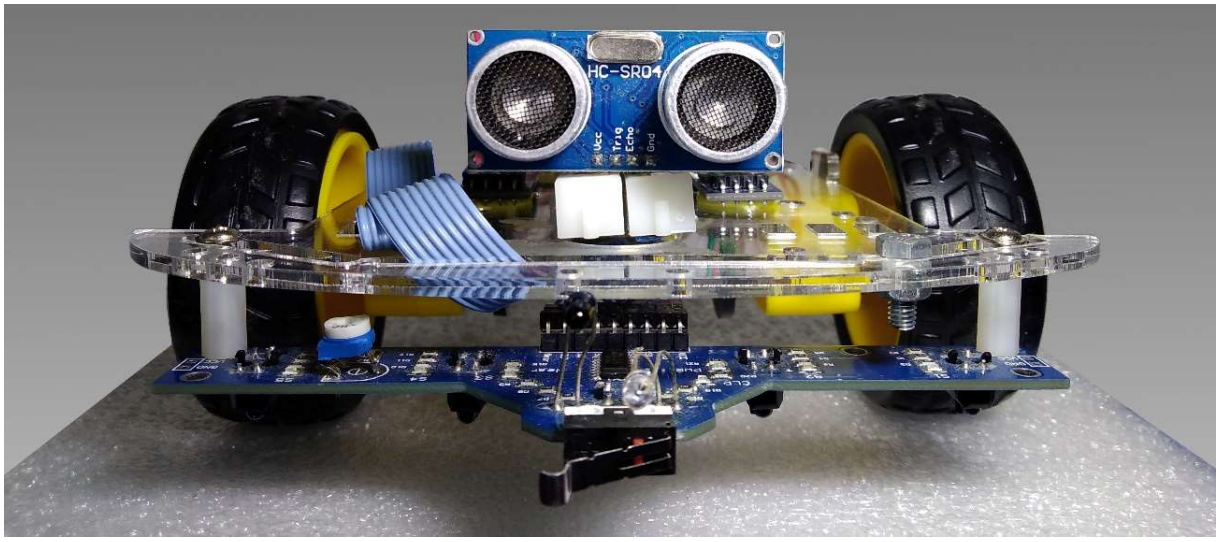
Manual Técnico
iModBot



Elaborado por:
Abel Teixeira – 2180522
Samuel Lourenço – 2180356

Verificado por:
Nelson Henriques – 2190514

Orientado por:
Carlos Neves
Luís Conde

Índice

Conteúdo

Introdução

O projeto iModBot@ipleiria.pt consiste num robô seguidor de linha, modular, de baixo custo.

É fornecida uma biblioteca escrita em C/C++ para facilitar o uso do mesmo.

Este robô pode ser programado através da aplicação Arduino IDE em linguagem C/C++ e também através de linguagem de programação por blocos utilizando o software *ArduBlocks*.

Neste documento serão descritos os componentes, as ligações dos mesmo e o uso da biblioteca.

Descrição dos componentes

❖ DOIT ESP32 DevKit V1

A placa de desenvolvimento DOIT ESP32 DevKit V1 irá controlar o robô e todos os seus periféricos. Esta placa é baseada no microcontrolador ESP32 desenvolvido pela Espressif Systems. Este microcontrolador possui uma velocidade de 80 até 240 MHz (dependendo da versão e/ou configuração), 4MB de memória flash e 520KB de SRAM.

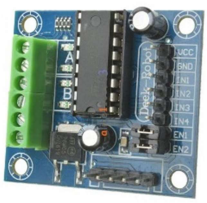
Tensão de entrada do módulo: 5 a 12V.

Tensão de operação: 3.3V.

Wi-Fi: IEEE 802.11 b/g/n/e/i.

Bluetooth BLE.

❖ Módulo de controlo dos motores, L293D



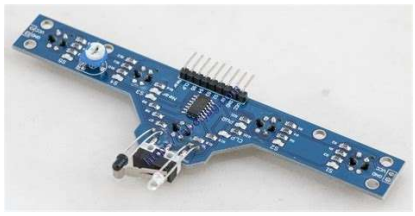
Este módulo servirá para controlar os motores.
No bloco terminal (VIN) pode ser conectada uma tensão de 7 até 35 V que irá alimentar o regulador 78M05 que por usa vez irá fornecer uma tensão de 5 V.
São necessários quatro pinos para controlar os dois motores, pode ser usada modulação de largura de pulsos (PWM) para controlar a velocidade dos motores.

Tensão de entrada no pino VIN: 7 a 35 V.

Corrente fornecida pelo 78M05: 500mA.

Nota: Forneça alimentação ao módulo apenas através do pino VIN. O pino **VCC é uma saída** e está conectado à saída do regulador 78M05 (presente na placa do módulo) que fornece 5 V.

❖ Placa de sensores IR 74HC14



A placa de sensores infravermelhos possui seis emissores e seis recetores infravermelhos para possibilitar a deteção de linhas. Os recetores infravermelhos estão ligados a um inversor com histerese 74HC14 que irá evitar ruído nas ligações do mesmo à placa de desenvolvimento DOIT ESP32 DevKit V1. O sensor "*Near*" pode obter falsos-positivos devido à luz natural/ambiente.

A tensão de entrada (VCC) é de 2 a 6 V.

❖ Encoder Óticos

Os sensores óticos servem para obter um feedback da rotação das rodas podendo deste modo obter a distância percorrida e saber se uma está a rodar mais rapidamente que a outra.

Tensão de operação: 3.3 a 5 V.

❖ HC-SR04

O sensor ultrassónico HC-SR04 permite detetar obstáculos com antecedência.

Tensão de operação: 5 V.

Consumo de corrente típico: 15mA.

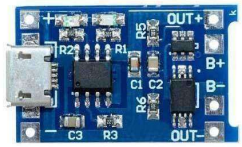
Frequência de funcionamento: 40 MHz.

Alcance: desde 2cm até 4m.

Angulo de medição: 15º.

Tempo do sinal de gatilho: 10us nível TTL.

❖ Módulo de carga TP4056



O módulo baseado no circuito integrado TP4056 permite carregar baterias de iões lítio. Possui um LED vermelho que indica que a bateria está a ser carregada e um LED azul que indica que o carregamento foi concluído.

Tensão de entrada: 4,5 a 8V.

Carrega a bateria até 4,2V ao máximo de 1A.

A potência de saída é a potência da bateria, tendo um máximo de 3 amperes limitado por um MOSFET.

❖ Módulo elevador de tensão MT3608

O módulo baseado no circuito integrado MT3608 efetua uma elevação de tensão ajustável.

A eficiência máxima é cerca de 93%, quanto maior for a diferença entre tensão de entrada/saída menor será a eficiência.

Tensão de entrada: 2 a 24 V.

Tensão de saída ajustável: 5 a 28V.

Corrente máxima fornecida (pico): 2A.

❖ Bateria de lítio de iões

Para o projeto foi usada uma bateria de lítio de iões.

Este tipo de baterias são normalmente acessíveis e possuem bastante autonomia.

Recursos do ESP32 usados pela biblioteca

A biblioteca iModBot usa os seguintes recursos do ESP32:

- 28 748 bytes de memória Flash (2,2%);
- Timer nº 0;
- Canais PWM nº 0, 1, 2 e 3;
- Canal nº 0 do periférico “Pulse Counter” nº 0;
- Canal nº 0 do periférico “Pulse Counter” nº 1.

Descrição das funções da biblioteca

Nota: A contagem de pulsos dos enconder óticos é efetuada por hardware usando um módulo denominado de “pulse counter”. A configuração deste módulo encontra-se dentro das bibliotecas “E32_PC0” e “E32_PC1”. Quando a biblioteca “RobotOnLine” é usada estas outras duas bibliotecas serão incluídas automaticamente. Não é necessário incluir as bibliotecas “E32_PC0” e “E32_PC1” no programa que use a biblioteca “RobotOnLine”, o ambiente Arduino IDE poderá incluir as mesmas por defeito, estas inclusões podem ser apagadas.

- **Void editWheelPin(uint8_t ,uint8_t)**

  Permite alterar os pinos onde são ligados os sensores óticos.

  Se esta função for usada, a mesma deve ser chamada antes da função “begin()”.

  Por defeito os pinos usados para os sensores óticos são o 26 e 27.

  Os parâmetros são *encoder* da esquerda, *encoder* da direita respetivamente.

- **Void editMotorPin(uint8_t ,uint8_t , uint8_t ,uint8_t ,)**

  Permite alterar onde são ligadas as conceções para controlar os motores.

  Por defeito os pinos usados são o 2, 4, 16 e 17.

  Os parâmetros são IN1, IN2, IN3, IN4 respetivamente.

- **Void editUltrasonicPin(uint8_t ,uint8_t)**

  Esta função permite alterar os pinos que conectam ao sensor ultrassónico HC-SR04.

  Por defeito os pinos usados são o 14 e o 12.

  Os parâmetros são *trig*, *echo* respetivamente.

- **Void editSensorPin(uint8_t ,uint8_t , uint8_t ,uint8_t , uint8_t ,uint8_t , uint8_t)**

  Permite alterar os pinos onde são estabelecidas as ligações à placa frontal que possui vários sensores de infravermelho.

  Caso use a função de condução autónoma (seguir um trajeto de forma autónoma) da biblioteca, deve chamar esta função antes da função “beginAutoDrive();”

  Por defeito os pinos usados são 25, 33, 32, 35, 34, 39, 36.

  Os parâmetros são S1, S2, S3, S4, S5, CLP, *Near* respetivamente.

- **Void setStopDistance(byte)**

Esta função é destinada à funcionalidade de condução autónoma e permite definir a distância à qual o robô para. Quando encontra um obstáculo.
Por defeito a distância é 9cm. O número especificado de 0 a 255 é interpretado em centímetros,

- **Void begin()**
  Esta função necessita de ser executada uma vez para iniciar a biblioteca e configurar os pinos e periféricos usados.

- **Void beginAutoDrive()**
  Esta função é destinada à funcionalidade de condução autónoma e necessita de ser chamada uma vez para configurar as interrupções associadas aos sensores S2, S3, S4 e CLP. Quando esta função for chamada o robô irá verificar qual o valor de PWM mínimo para colocar ambas as rodas em movimento.

- **Void endAutoDrive()**
  Esta função é destinada à funcionalidade de condução autónoma e necessita de ser chamada uma vez para desassociar as interrupções previamente configuradas pela função **beginAutoDrive()**.

- **byte autoDrive(byte)**
  Esta função é destinada à funcionalidade de condução autónoma e necessita de ser chamada constantemente para efetuar várias tarefas.
  Os valores devolvidos são os seguintes:
    - "0", nada a reportar;
    - "1", múltiplas linhas encontradas;
    - "2", obstáculo encontrado;
    - "3", não foram encontradas quaisquer linhas.

  O valor inserido é a instrução a efetuar para quando é encontrada mais que uma linha ou algum obstáculo:

    - "1", rodar para a direita;
    - "2", rodar para a esquerda;

- "3", se não houver obstáculo seguir em frente;
- "4", retroceder.

- **void disableCLP()**

  Esta função é destinada à funcionalidade de condução autónoma, necessita de ser chamada uma vez para ignorar as leituras do sensor CLP.

- **void disableNear()**

  Esta função é destinada à funcionalidade de condução autónoma e necessita de ser chamada uma vez para ser ativada. A sua função é desativar a leitura do sensor *Near*.

- **void disableUltrasonic()**

  Esta função é destinada à funcionalidade de condução autónoma, necessita de ser chamada uma vez para ignorar as leituras do sensor de ultrassons HC-SR04.

- **void noLineDelay(uint)**

  Esta função é destinada à funcionalidade de condução autónoma. A sua função é especificar o tempo que o robô demora a parar após deixar de detetar qualquer linha. Por defeito o valor é 700ms, o valor inserido é interpretado em milissegundos.

- **void setSpeeds(byte ,byte ,byte)**

  Esta função permite definir o valor *dutty-cycle* (8 bits) usado para as velocidades máxima, média e mínima respetivamente. Por defeito os valores usados são 255, 200 e 180.

- **byte distance()**

  Esta função devolve o valor medido (em centímetros) pelo sensor de ultrassons HC-SR04. Devolve um valor entre (e incluindo) 0 a 255.

- **void turnLeft(uint16_t)**

  Esta função permite fazer o robô rodar x graus para a esquerda.
  Se for enviado um 0 o robô irá tentar rodar 90 graus. Se for enviado um valor acima de 360 o robô irá apenas rodar 360 graus.

Esta função irá ser efetuada usando a velocidade mínima necessária para mover as rodas, isto para evitar que uma (ou ambas) as rodas derrapem.

Quando esta função for chamada o robô irá verificar qual o valor de PWM mínimo para colocar ambas as rodas em movimento.

- **void turnRight(uint16_t)**

  Esta função permite fazer o robô rodar x graus para a direita.
  Se for enviado um 0 o robô irá tentar rodar 90 graus. Se for enviado um valor acima de 360 o robô irá apenas rodar 360 graus.

  Esta função irá ser efetuada usando a velocidade mínima necessária para mover as rodas, isto para evitar que uma (ou ambas) as rodas derrapem.

  Quando esta função for chamada o robô irá verificar qual o valor de PWM mínimo para colocar ambas as rodas em movimento.

- **void steerLeft(byte)**

  Esta função faz com que o robô vire ligeiramente à esquerda sem parar de seguir em frente.

- **void steerRight(byte)**

  Esta função faz com que o robô vire ligeiramente à direita sem parar de seguir em frente.

- **void rotateLeft(byte)**

  Esta função faz o robô rodar para a esquerda com a velocidade determinada pelo *Dutty-Cycle* (Modelação de Largura de Pulso) inserido. Um valor *Dutty-Cycle* muito baixo pode não ser suficiente para iniciar o movimento.

- **void rotateRight(byte)**

  Esta função faz o robô rodar para a direita com a velocidade determinada pelo *Dutty-Cycle* (Modelação de Largura de Pulso) inserido. Um valor *Dutty-Cycle* muito baixo pode não ser suficiente para iniciar o movimento.

- **void stopMotors()**

  Esta função simplesmente para ambos os motores.

- **void forward(byte)**

  Faz o robô seguir em frente com a velocidade determinada pelo *Dutty-Cycle* (Modelação de Largura de Pulso) inserido. Um valor *Dutty-Cycle* muito baixo pode não ser suficiente para iniciar o movimento. Se o valor inserido for 0 o robô irá parar.

- **void reverse(byte)**

  Faz o robô recuar com a velocidade determinada pelo *Dutty-Cycle* (Modelação de Largura de Pulso) inserido. Um valor *Dutty-Cycle* muito baixo pode não ser suficiente para iniciar o movimento. Se o valor inserido for 0 o robô irá parar.

- **void leftWheel(short)**

  Esta função controla o motor esquerdo individualmente. Pode ser inserido um valor de -255 a 255, um valor negativo fará a roda esquerda recuar e vice-versa. Se for inserido o valor 0 o motor irá parar.

- **void rightWheel(short)**

  Esta função controla o motor direito individualmente. Pode ser inserido um valor de -255 a 255, um valor negativo fará a roda direita recuar e vice-versa. Se for inserido o valor 0 o motor irá parar.

- **void clearEncoderCount()**

  A contagem dos pulsos lidos pelos *encoder* óticos é feita por hardware usando um módulo do ESP32 denominado “Pulse Counter”, esta função coloca a contagem de ambos os *encoder* a zero.

- **int16_t getLeftEncoderCount()**

  Esta função devolve o número de pulsos contados pelo *encoder* esquerdo.

- **int16_t getRightEncoderCount()**

  Esta função devolve o número de pulsos contados pelo *encoder* direito.

- **byte checkSensors()**

  Esta função efetua uma leitura digital de cada pino presente na placa de sensores IR na frente do robô. Devolve um byte cujos bits significam o seguinte:

  - bit 0: É “1” se o sensor "Near" detetar algo.

- bit 1: É “1” se o botão CLP for pressionado, isto normalmente acontece quando o robô colide com algo.
- bit 2: É “1” se o sensor mais à direita detetar uma linha.
- bit 3: É “1” se o sensor se o sensor da direita detetar uma linha.
- bit 4: É “1” se o sensor do meio detetar uma linha.
- bit 5: É “1” se o sensor da esquerda detetar uma linha.
- bit 6: É “1” se o sensor mais à esquerda detetar uma linha.

- **bool readS1 (); bool readS2(); bool readS3; bool readS4(); bool readS5();**

  Permite ler o estado dos sensores S1, S2, S3, S4 e S5. Devolve o valor “0” se for detetada uma linha preta.

- **bool readNear();**

  Devolve 1 se for detetado um obstáculo pelo sensor Near;

- **bool readCLP();**

  Devolve o valor “1” se o botão (*switch*) for pressionado.

- **Void enSpeedAdj();**

  Ativa a função de ajuste de velocidade das rodas com base na velocidade especificada, seja esta em em rotações por segundo (RPS), rotações por minuto (RPM) ou metros por segundo (M/S), pelo utilizador.

- **Float getRightRPS();**

  Devolve um valor de tipo *float* que representa a velocidade da roda direita em rotações por segundo.

- **Float getLeftRPS();**

  Devolve um valor de tipo *float* que representa a velocidade da roda esquerda em rotações por segundo.

- **Float getRightRPM();**

  Devolve um valor de tipo *float* que representa a velocidade da roda direita em rotações por minuto.

- **Float getLeftRPM();**

  Devolve um valor de tipo *float* que representa a velocidade da roda esquerda em rotações por minuto.

- **Float getRightMS();**

Devolve um valor de tipo *float* que representa a velocidade da roda direita em metros por segundo.

- **Float getLeftMS();**

  Devolve um valor de tipo *float* que representa a velocidade da roda esquerda em metros por segundo.

- **Void forwardRPS(float);**

  Permite colocar o robô a mover-se em frente à velocidade especificada (em rotações por segundo RPS).

- **Void reverseRPS(float);**

  Permite colocar o robô a retroceder à velocidade especificada (em rotações por segundo RPS).

- **Void rotateRightRPS(float);**

  Permite colocar o robô a rodar para a direita (sentido horário) à velocidade especificada (em rotações por segundo RPS).

- **Void rotateLeftRPS(float);**

  Permite colocar o robô a rodar para a esquerda (sentido anti-horário) à velocidade especificada (em rotações por segundo RPS).

- **Void forwardRPM(float);**

  Permite colocar o robô a mover-se em frente à velocidade especificada (em rotações por minuto RPM).

- **Void reverseRPM(float);**

  Permite colocar o robô a retroceder à velocidade especificada (em rotações por minuto RPM).

- **Void rotateRightRPM(float);**

  Permite colocar o robô a rodar para a direita (sentido horário) à velocidade especificada (em rotações por minuto RPM).

- **Void rotateLeftRPM(float);**

  Permite colocar o robô a rodar para a esquerda (sentido anti-horário) à velocidade especificada (em metros por segundo m/s).

- **Void forwardMS(float);**

  Permite colocar o robô a mover-se em frente à velocidade especificada (em metros por segundo m/s).

- **Void reverseMS(float);**

Permite colocar o robô a retroceder à velocidade especificada (em metros por segundo m/s).

- **Void rotateRightMS(float);**

  Permite colocar o robô a rodar para a direita (sentido horário) à velocidade especificada (em metros por segundo m/s).

- **Void rotateLeftMS(float);**

  Permite colocar o robô a rodar para a esquerda (sentido anti-horário) à velocidade especificada (em metros por segundo m/s).

- **Void moveAt(byte, byte, float);**

  Função responsável por colocar o robô em movimento para a direção e velocidade especificadas.

  O primeiro parâmetro (da esquerda para a direita) é a direção, podem ser enviados os seguintes valores:

  - "1", coloca o robô a seguir em frente;
  - "2", coloca o robô a rodar para a direita;
  - "3", coloca o robô a recuar;
  - "4", coloca o robô a rodar para a esquerda;
  - Um valor diferente que se encontre de 0 a 255 irá parar ambos os motores.

  O segundo parâmetro especifica o tipo de unidade, podem ser enviados os seguintes valores:

  - "1", em rotações por segundo (RPS);
  - "2", em rotações por minuto (RPM);
  - "3", em metros por segundo (M/S).

  O terceiro parâmetro é a velocidade relativa à unidade especificada no segundo parâmetro.

Esquema elétrico recomendado

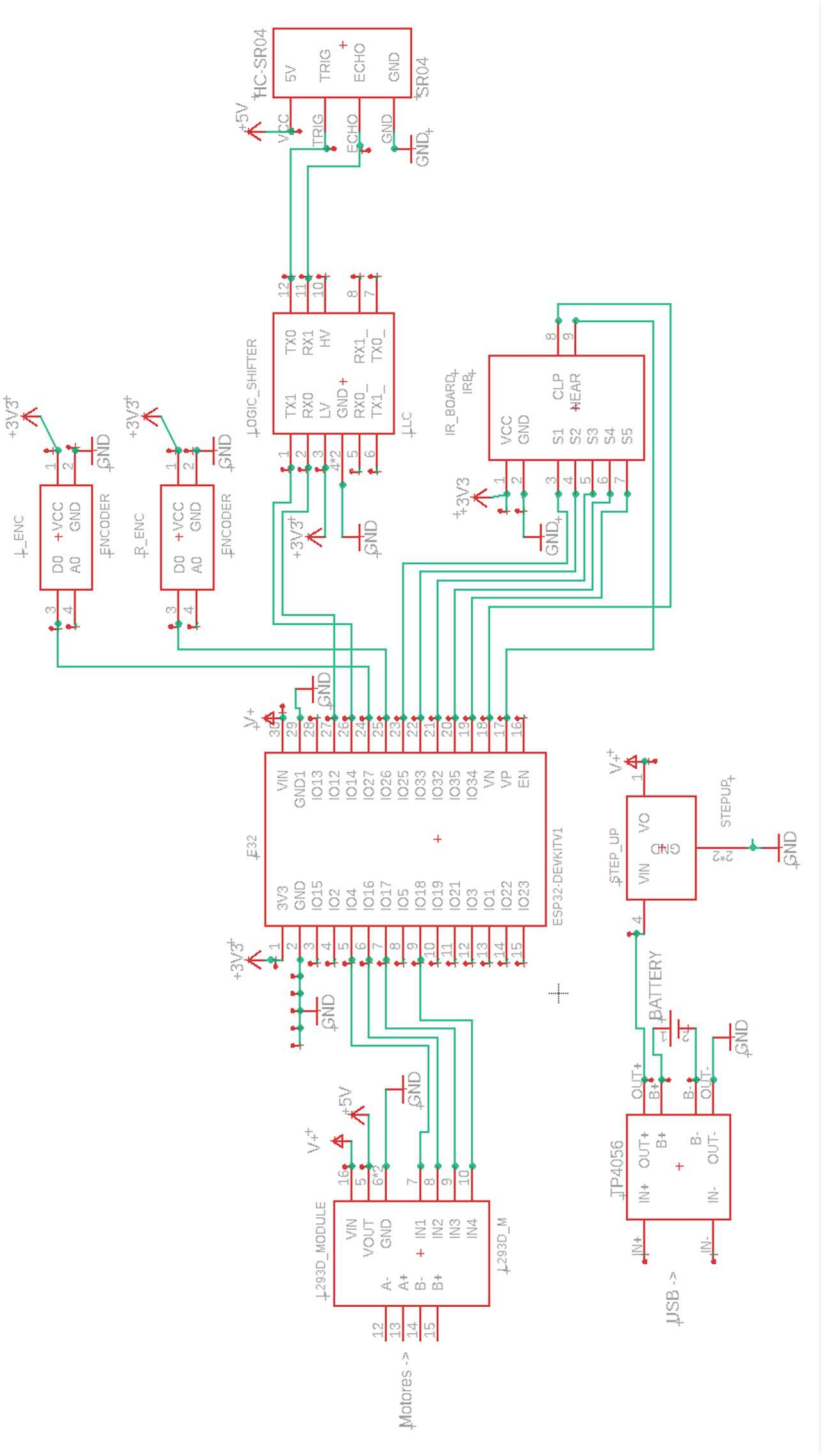


Figura 0.1 – Esquema

O esquema demonstrado na Figura 0.1 indica o esquema elétrico recomendado para o bom funcionamento do robô didático.

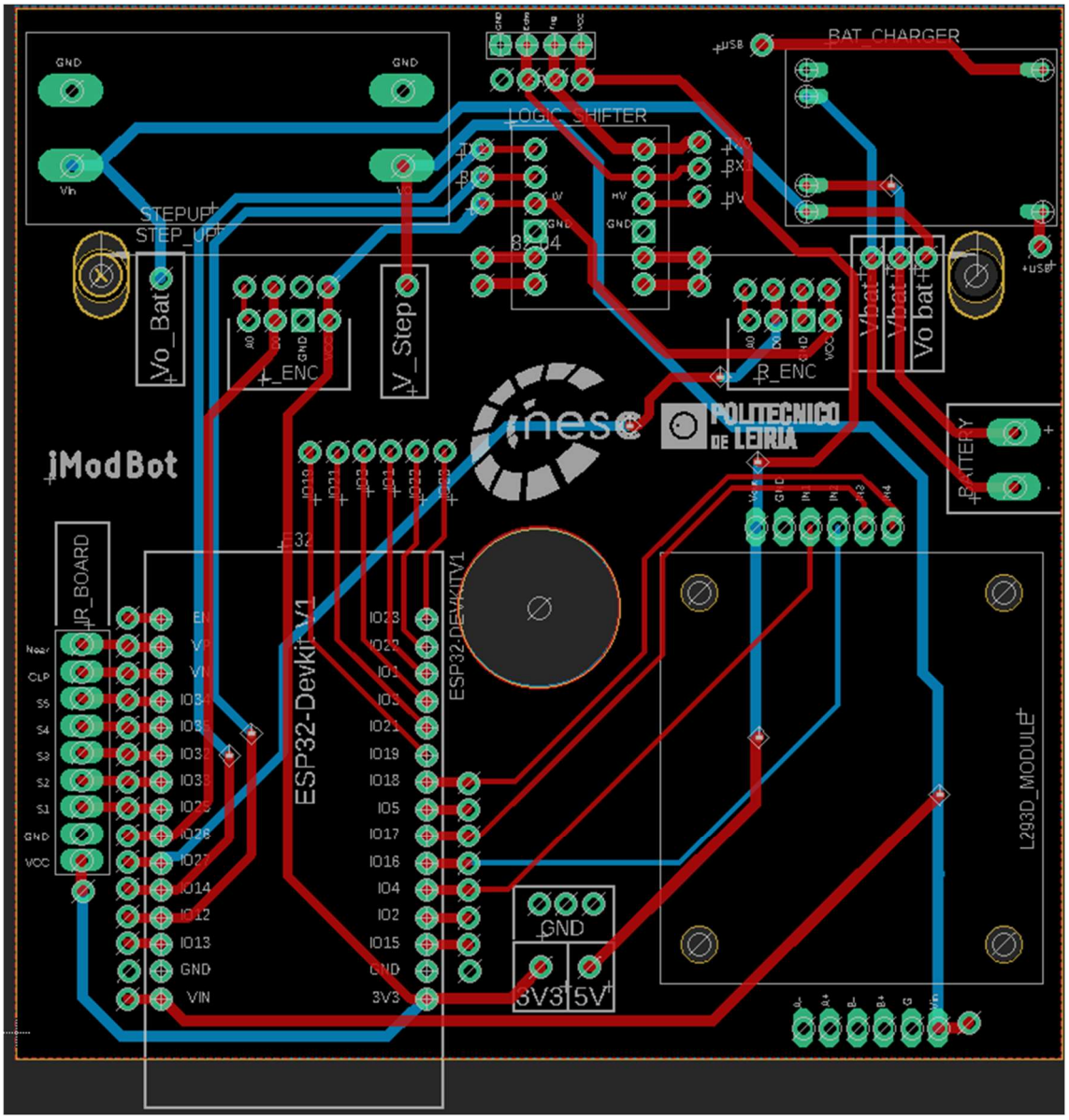


Figura 0.2 - Placa de Circuito Impresso

A Figura 0.2 exemplifica um possível *layout* da placa de circuito impresso criada a partir do esquema da Figura 0.1.

Descrição do funcionamento da condução autónoma da biblioteca

Para efetuar a tarefa de seguir um trajeto de forma autónoma a biblioteca “RobotOnLine” recorre a duas funções principais, sendo uma delas associada a várias interrupções externas.

Função ”think()”

A função _think() é executada sempre que houver uma interrupção externa nos pinos conectados aos sensores S2, S3, S4 e CLP. Esta função, quando executada, desativa temporariamente as interrupções externas de modo a poder operar sem o risco de ocorrer um “loop” indeterminado.

Resumidamente a função “_think()” verifica se o robô embateu e a posição do trajeto (linha). Se não for encontrada nenhuma linha esta função irá copiar o valor da função “millis()” (esta função devolve o valor em milissegundos decorrido desde o inicio do programa) para uma variável global. Esta variável global será usada como uma “flag” e o valor da mesma será usado pela função “byte autoDrive(byte)” para parar o movimento do robô após um *delay* especificado. Quando o robô efetua uma curva poderão existir breves momentos em que a linha não seja detetada por nenhum dos sensores, daí a existência deste atraso na paragem do robô.

A função “_think()” também verifica se o robô chegou a um cruzamento. Para tal é que uma das seguintes combinações de sensores seja cumprida:

- S3 e S1 e S2 (sensores do meio e da esquerda);
- S3 e S4 e S5 (sensores do meio e da direita);
- S1 e S2 e S4 e S5 (sensores da direita e da esquerda).

Basicamente os sensores das bordas da placa são verificados de modo a ter a certeza de que o robô se encontra num cruzamento. Este processo foi implementado de modo a que o utilizador possa especificar o que o robô deve fazer quando este chega a um cruzamento.

A função “_think()” recorre a uma variável global denominada “_LLP” (abreviado de *Last Line Position*) para saber a ultima posição da linha, para evitar repetir instruções e para passar informação à função “byte autoDriver(byte)”.

Função “byte autoDrive(byte)”

A função “byte autoDrive(byte)” deve ser chamada o mais brevemente possível, pois esta é complementar à função “_think()” e executa várias tarefas. Esta função recebe e devolve valores, os valores devolvidos têm o seguinte significado:

- “0”, nada a reportar;
- “1”, múltiplas linhas encontradas;
- “2”, obstáculo encontrado;
- “3”, não foram encontradas linhas.

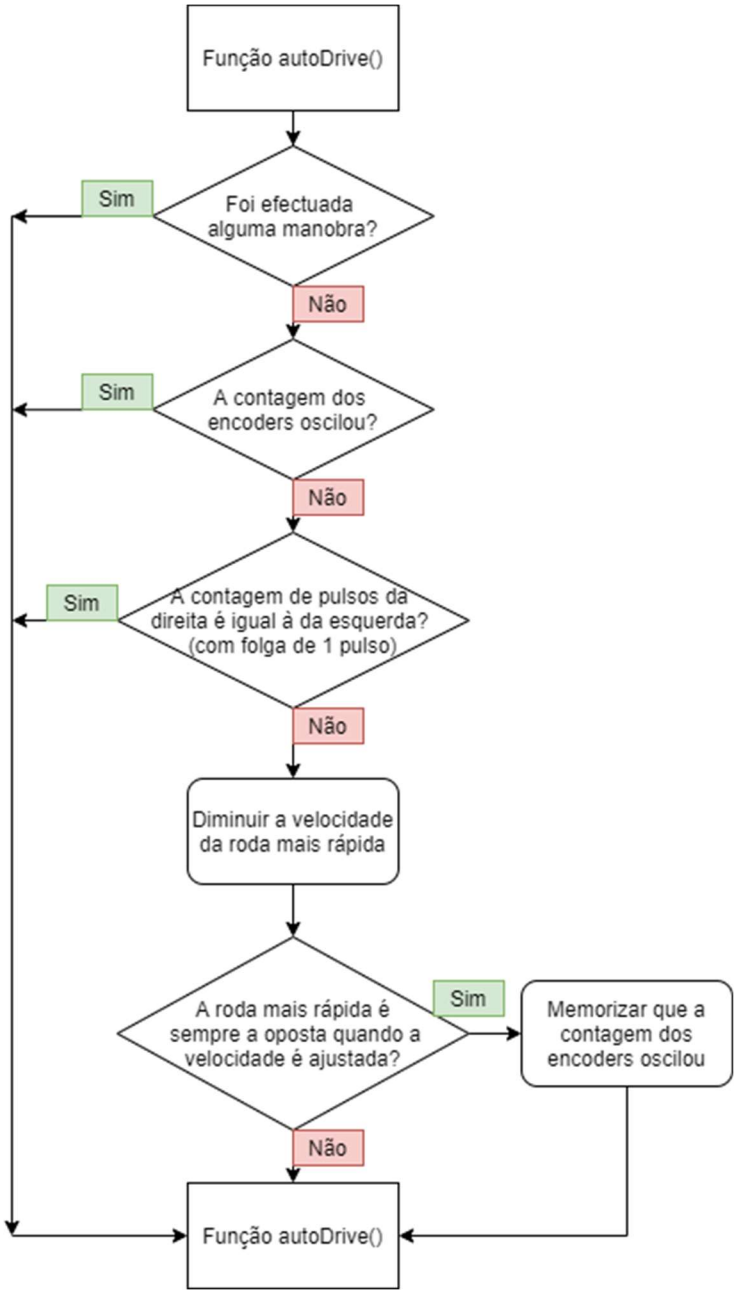
Caso seja encontrado um obstáculo ou múltiplas linhas (valor devolvido “1” ou “2”) podem ser enviados os seguintes valores para a função:

- “1”, o robô irá rodar para a direita até encontrar uma linha;
- “2”, o robô irá rodar para a esquerda até encontrar uma linha;
- “3”, caso não haja nenhum obstáculo, seguir em frente;
- “4”, retroceder.

É necessário enviar sempre um valor para a função, caso não seja intenção do utilizador dar uma instrução ao robô recomenda-se que envie o valor “0”. Apenas os números 1, 2, 3 e 4 dão instruções ao robô, qualquer outro número (até 255) não terá nenhum efeito.

A leitura dos sensores de ultrassons (HC-SR04) e sensor de infravermelho (*Near)* é efetuada nesta função.

A função “byte autoDrive(byte)” é responsável por fazer o ajuste da velocidade das rodas do robô, esta operação decorre de acordo com o seguinte fluxograma:


Função autoDrive()
Foi efectuada alguma manobra?
Sim
Não
A contagem dos encoders oscilou?
Sim
Não
A contagem de pulsos da direita é igual à da esquerda? (com folga de 1 pulso)
Sim
Não
Diminuir a velocidade da roda mais rápida
A roda mais rápida é sempre a oposta quando a velocidade é ajustada?
Sim
Memorizar que a contagem dos encoders oscilou
Não
Função autoDrive()

O seguinte fluxograma descreve o funcionamento de ambas as funções: