UFABC - UNIVERSIDADE FEDERAL DO ABC
TAMANDUTECH

Treinamento Segue Linha

GERAL - 1º ETAPA

Treinamento contendo conceitos básicos de eletrônica, e componentes dos robôs, para a categoria Segue Linha da Tamandutech.

Santo André - SP
2020

SUMÁRIO

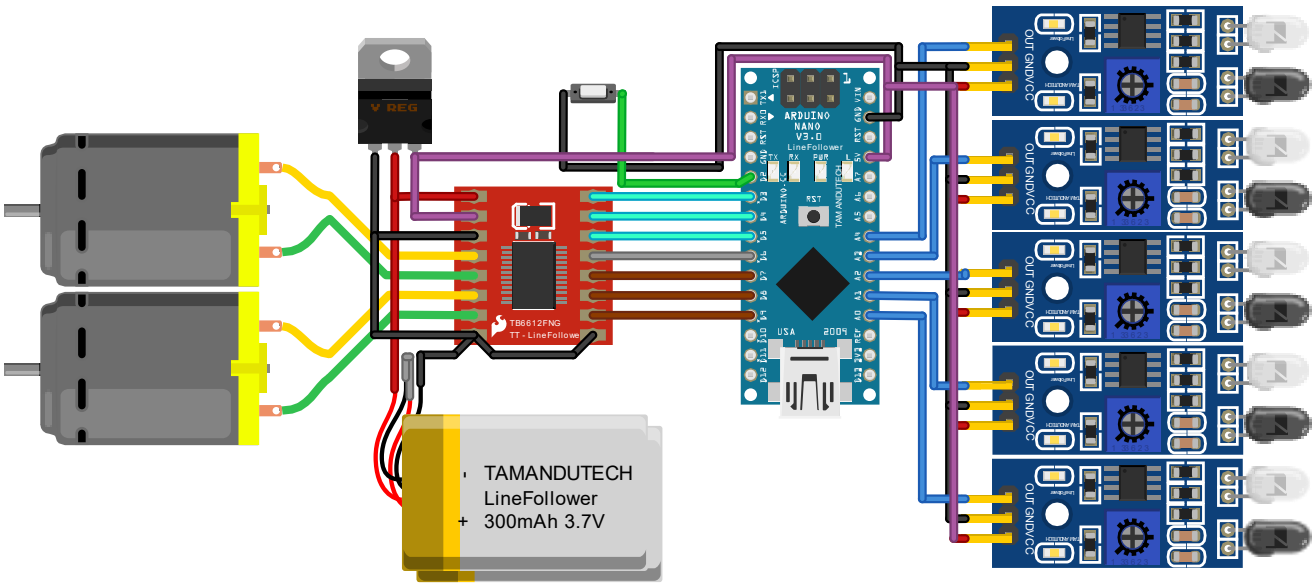
OBJETIVOS

Este treinamento tem como objetivo principal fornecer a descrição necessária de componentes e módulos eletrônicos para que você seja capaz de montar um circuito básico de um robô seguidor de linha.

Você será capaz de:

- Entender como um robô seguidor de linha se orienta (dispositivos sensores);
- Entender como um robô seguidor de linha se movimenta (dispositivos atuadores);
- Identificar os componentes, de um circuito básico, e sua funcionalidade;
- Como alimentar adequadamente o circuito;
- Montar seu circuito.

No final deste treinamento você deve compreender por completo o circuito básico abaixo:



ELETRÔNICA

1. Componentes principais

1.1. Alimentação do robô

A bateria irá depender do circuito disposto e do microcontrolador utilizado (caso existir um), como a maioria dos microcontroladores trabalha com tensão de 5V e a maioria dos componentes empregados para o circuito trabalham nessa faixa (5V ou menos) a bateria tem que possuir na sua capacidade mínima (quando estiver minimamente carregada) 5V ou mais.

O peso é importante

Portanto o ideal é utilizarmos uma bateria leve e pequena, com o mínimo necessário para comportar o circuito. Geralmente são utilizadas baterias com múltiplas células de Lítio, cada célula de Lítio tem em média 3,6V (2,8V descarregada e 4,2V carregada), utilizamos então baterias com no mínimo 2 células de Lítio para comportar os motores e módulos do robô.

A bateria ao abaixo é comumente utilizada em nossos robôs, e possui a seguinte descrição:

- **LI-PO**: a composição das células, Polímero de Lítio.
- **2S**: Descreve a quantidade de células que essa bateria possui, no caso apenas 2.
- **7,4V**: A tensão média, somada, das células. Como nossa bateria possui 2 (**2S**), 3,6V + 3,6V = 7,4V.

Bateria nano-tech Li-Po 2S 7,4V

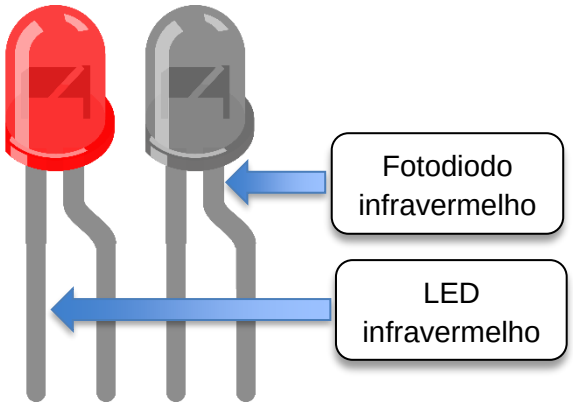
Atentar-se a tensão máxima suportada pelos componentes!

É necessário lembrar que sempre será necessário o uso de um regulador de tensão no circuito, as baterias variam sua tensão com o seu nível de carga, e na maioria das vezes a tensão entregue por elas excede a capacidade máxima suportada pelos componentes da placa. Abordaremos neste documento o uso de reguladores de tensão.

1.2. Dispositivos sensores

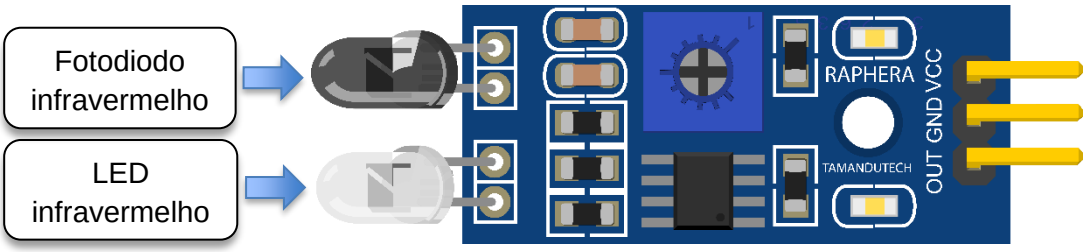
Para que nosso robô consiga detectar a linha devemos ser capazes de gerar um sinal que ele interprete, e esse sinal deve mudar de acordo com a superfície. No caso estamos tratando de variações de cor, mais especificamente cores bem contrastadas como uma linha preta em uma superfície de tonalidade branca. Sabendo disso, podemos utilizar da propriedade de reflexão da luz**.**

Conseguimos medir a quantidade de luz com um simples componente chamado Fotodiodo, tal componente na ausência de luz produz corrente mínima (quase nula) e na presença de luz produz quantidades maiores de corrente. (para esse caso utilizamos um sensível apenas a luz infravermelha)



Para que haja variação no fotodiodo devemos enviar a luz que será refletida ou absorvida pela superfície, utilizamos então um LED infravermelho.

Esses dois componentes são os básicos para montarmos um sensor de reflexão de luz infravermelha e geralmente são vendidos em módulos como a imagem abaixo:



Serão necessários, no mínimo, 2 desses módulos, cada um deles identifica uma borda da linha preta no chão, um a esquerda da linha e outro a direita da linha, dessa forma assim que o a luz infravermelha parar de ser refletida ao fotodiodo (quando a corrente do fotodiodo baixar) significa que o módulo atingiu a linha preta, pois nessa cor a maior parte da luz é absorvida. Com isso nosso robô pode fazer os ajustes necessários na direção para continuar seguindo a linha.

Desafio 1A: Procure por exemplos de sensores, pegue algumas imagens e informações que consideram úteis para um robô segue linha.
(Dica: Procure por "*IR Sensor Array*")

1.3. Dispositivos controladores

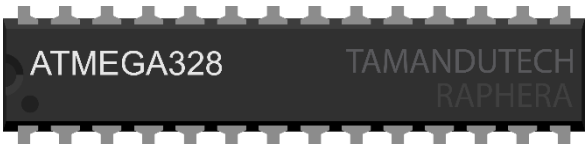
Os microcontroladores foram concebidos para aplicações embarcadas, são destinados a controles e por isso são muito comuns em projetos de automação e projetos mais simples onde o poder de processamento não é o foco. Possuem, geralmente, um núcleo de processador, memória e periféricos programáveis (saídas e entradas digitais e analógicas).

Para que possam se comunicar com o mundo real através de sensores e controlar circuitos é necessário que as portas I/O tenham algumas características como por exemplo:

- **PORTAS DIGITAIS**
  As portas digitais assumem níveis lógicos bem definidos, níveis representados em programa por 1 e 0 que assumem 5V/3,3V ou 0V, respectivamente.

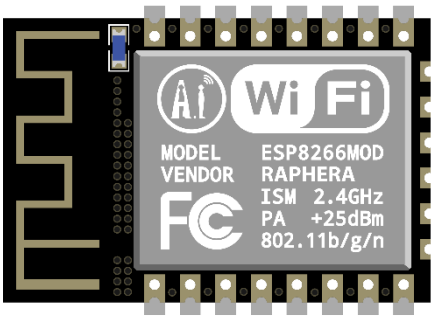
- **PORTAS ANLÓGICAS**
  As portas analógicas fazem a leitura de sinais/tensões que variam ao longo do tempo, essas portas funcionam com um conversor ADC que faz a representação dessa variação em bits (no ATMega328p são 1024 níveis).

- **PWM**
  Os microcontroladores são bons em ler e mandar sinais lógicos definidos, para resolver o problema de enviar um sinal variado foi implementada a técnica de PWM (Pulse Width Modulation), que pulsa rapidamente um sinal digital com o intuito de variar a tensão ao longo do tempo.



Um exemplo comum de microcontrolador é o ATMega328p, encontrado principalmente em placas Arduino UNO, possuí características interessantes para qualquer projeto como memória Flash de 32kb, 23 pinos I/O (6 deles analógicos e 17 digitais com 6 PWM, sendo 2

desses digitais utilizados normalmente para o cristal externo e 1 utilizado para o botão de reset).



Outro exemplo é o ESP8266 ESP12-E, um pouco mais moderno e com mais recursos, encontrado em placas de desenvolvimento NodeMCU, possuí características melhores se comparado ao ATMega328p como módulo wireless, uma frequência de CPU (Clock) 2x maior de fábrica, podendo chegar até 4x maior do que a capacidade máxima suportada pelo ATMega328p e uma capacidade para armazenamento de até 4MB (depende da versão do microcontrolador). Em contrapartida possui apenas uma porta analógica (entrada) e trabalha com tensões de 3,3V o que, para alguns circuitos, vai demandar mais trabalho para trabalhar com componentes que operam a 5V.

Desafio 1B: Com o crescimento do mercado da internet das coisas, temos hoje um grande número de microcontroladores com Wi-Fi ou Bluetooth integrado, como você imagina o uso de recursos de conectividade para um robô segue linha? Há alguma vantagem?

Desafio 1C: Dentre os microcontroladores disponíveis no mercado, pesquise um que para você seja o ideal no uso de um robô segue linha.

1.4. Dispositivos atuadores

Para que o robô possa se mover, utilizamos 2 motores DC, de preferência leves e pequenos com caixa de redução. Temos ao lado um exemplo de motor comumente utilizado pela equipe e possui a seguinte descrição:

- **30:1**: Proporção de redução da caixa de redução.
- **6V**: Tensão segura máxima suportada pelo motor.
- **HP**: High-Power, suporta maiores quantidades de corrente, e com isso fornece um torque maior em relação ao **LP** (Low-Power), ou **MP** (Medium-Power).

Pololu 30:1 6V HP Micro Metal Gearmotor

Atentar-se a tensão máxima suportada pelo motor!

Motores desse tipo suportam uma tensão de operação maior, geralmente essa sobrecarga nos fornece maiores mais RPM (rotação por minuto) e um torque maior, porém nos custa vida útil do motor, sendo reduzida bruscamente ou até levando o motor a queimar instantaneamente dependendo das situações. Por isso, é importante trabalhar dentro da sua zona de segurança e, caso necessário maior toque e mais RPM alterar para um motor com menor proporção da caixa de redução como um **10:1** (https://www.pololu.com/product/999).

Como controlamos a quantidade de RPM e torque do motor?

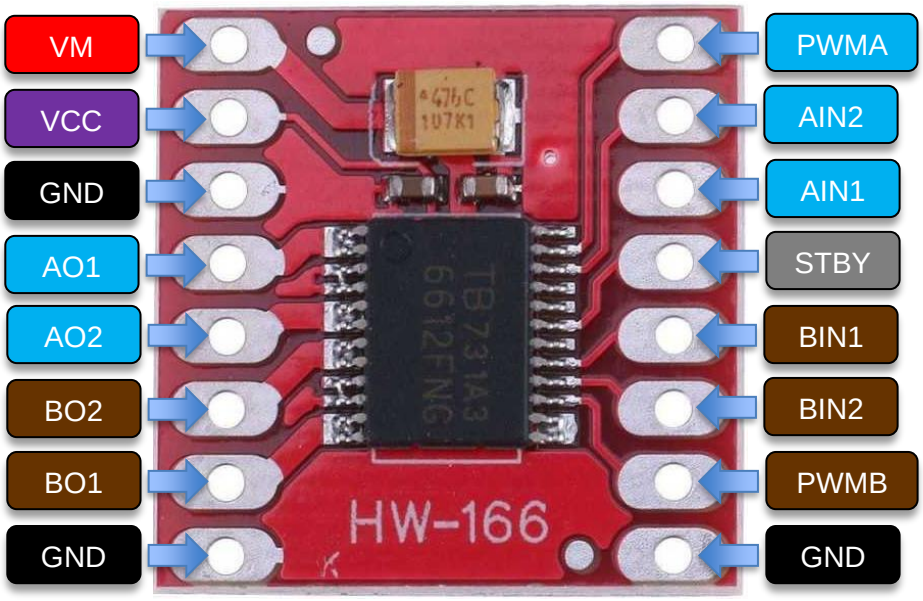
Para controlar motores dessa natureza precisamos alterar o nível de tensão em sua faixa de segurança (0V – 6V). Em projetos microcontrolados, ou que utilizem placas de desenvolvimento que contenham um microcontrolador (como por exemplo: Arduino e NodeMCU) conseguimos alterar o nível de tensão com um recurso do microcontrolador chamado **PWM**, e um **Driver de motor** que nos permita alimentar o motor através da bateria.

2. Componentes auxiliares

2.1. Driver de motor

Um driver de motor serve como uma interface entre o microcontrolador (ou circuitos que forneçam correntes baixas) e um motor, além de fornecer um ganho de corrente ele permite um controle mais completo do motor, nos permitindo até alterar o sentido da corrente para alterar o sentido de rotação do motor, além do controle de velocidade (com PWM).

Um driver comumente utilizado, e fácil de encontrar é o abaixo:



Driver Motor / Ponte H - TB6612FNG

- Terminais básicos:

VM	Alimentação do motor
VCC	Alimentação do CI (TB6612)
GND	Terra, negativo da bateria
STBY	Controle de atividade do CI (TB6612)

Para que serve o terminal STBY?

O terminal STBY serve para controle de atividade do CI, o TB6612 só vai “trabalhar” quando estiver com sinal alto (5V ou 3V3 dependendo da alimentação) em seu terminal STBY. Caso contrário nada será feito e os motores não movimentarão.

Este módulo possui dois canais de controle, significa que comporta 2 motores DC, portanto existem terminais semelhantes em ambos os canais:

Ex: AO1 = BO1 = O1

- Terminais de controle do motor:

IN1	Controle Lógico 1 do Motor
IN2	Controle Lógico 2 do Motor
PWM	Controle PWM do Motor
O1	Terminal 1 do Motor
O2	Terminal 2 do Motor

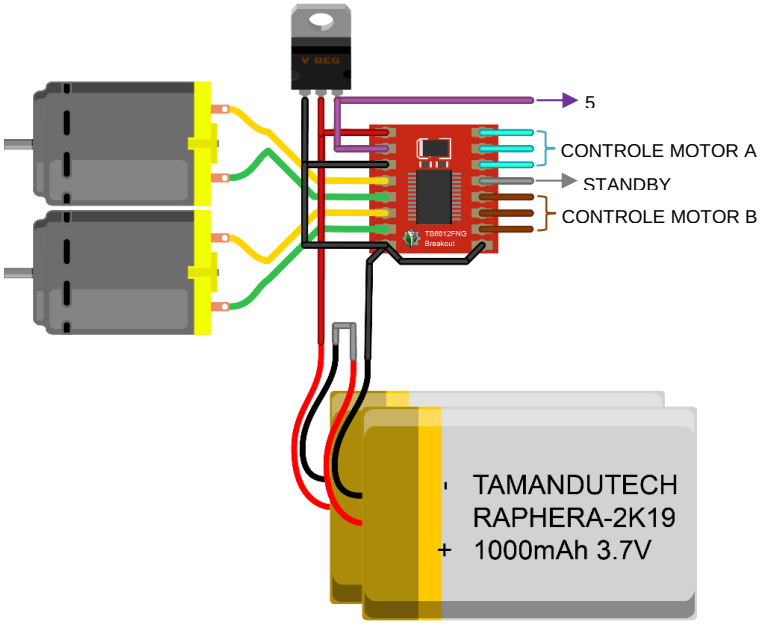
Os terminais do motor são divididos por:

- Controle de sentido de rotação: **IN1** e **IN2**
- Controle de RPM: **PWM**
- Saída (conexão com o motor): **O1** e **O2**

Para que servem os terminais de controle lógico?

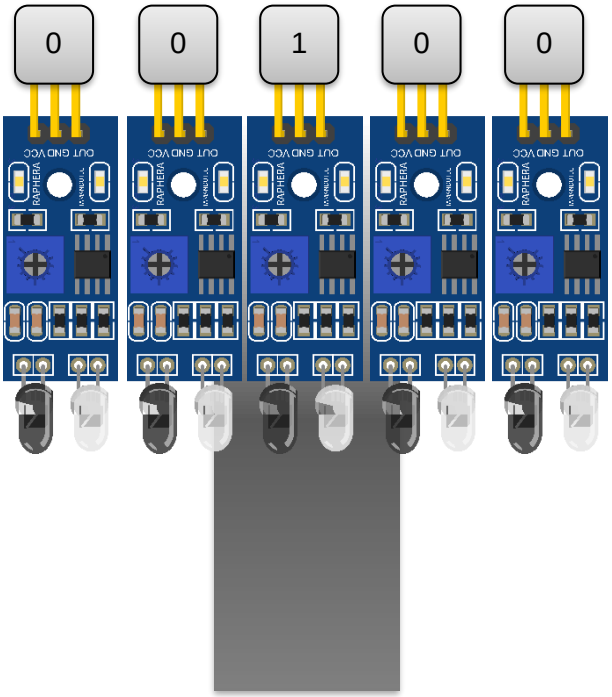
Servem basicamente para que possamos inverter a corrente de passagem entre **O1** e **O2**, e com isso inverter a rotação do motor. O sentido da corrente se dá quando aplicamos nível alto em um **IN** e nível baixo em outro **IN**, invertendo os níveis invertemos também o sentido da corrente na saída em **O1** e **O2**.

O circuito ao lado mostra como podemos fazer a conexão com um driver de motor (TB6612FNG):



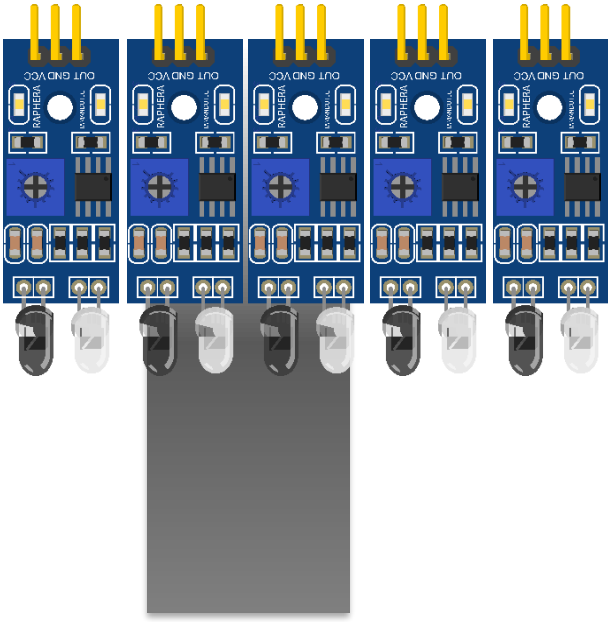
ORIENTAÇÃO

1. Distância da linha central



O array de sensores nos permite dimensionar o erro do robô, podemos criar variáveis de erro para cada estado lógico que os sensores produzirem em conjunto.

Nesse estado, por exemplo, onde o robô está centralizado e os sensores estão com estado “00100”, podemos considerar como erro 0.



Assim que o robô for um pouco para a direita teremos o estado lógico “01100” e podemos considerar como erro -1.

S1	S2	S3	S4	S5	ERRO
0	0	0	0	1	4
0	0	0	1	1	3
0	0	0	1	0	2
0	0	1	1	0	1
0	0	1	0	0	0
0	1	1	0	0	-1
0	1	0	0	0	-2
1	1	0	0	0	-3
1	0	0	0	0	-4

Os estados lógicos possíveis com o conjunto de 5 sensores com essas distâncias formando um array estão descritos na tabela ao lado.

[ORIENTAÇÃO] **Desafio 1:** Descreva os estados lógicos possíveis para 8 sensores (como na tabela acima).

2. Atribuição de valor ao erro

O PID (Proporcional, Derivativo e Integral) é utilizado para controlar uma quantidade física e tornar ela igual a um valor pré-especificado quando necessário.

No nosso exemplo o valor pré-determinado é o erro 0, e sempre que esse valor é desviado desejamos retornar a ele, mas de maneira proporcional a quantidade do erro gerado com o desvio. Então como termo:

- **Proporcional**:
  Responsável pela magnitude da mudança, vai determinar a proporção do ajuste, no caso do exemplo esse é a variável erro.
  P = ERRO.

- **Integral**:
  É a soma de todos os valores de erro anterior, responsável pela magnitude da mudança necessária para atingir o ponto de ajuste.
  I = I + ERRO.

- **Diferencial**:
  É a diferença entre o erro instantâneo do ponto de ajuste, e o erro a partir do instante anterior.
  D = ERRO – erroAnterior.

Criamos então uma função para calcular o PID a todo ciclo:

Equação PID

```
void calcularPID () {
   P = ERRO;
   I = I + ERRO;
   D = ERRO - erroAnterior;
   PIDvalor = (Kp * P) + (Ki * I) + (Kd * D);
   erroAnterior = ERRO;
}
```

Com o valor do PID podemos utilizar em nossa função de controle do motor para ajuste de posição:

```
void controleMotorPID () {
    /* O valor máximo de PWM é 255, portanto
     * é necessário um ajuste na função
     * calcular PID para que as variáveis
     * de velocidade do motor não superem o
     * valor de 255 */

    /* Se robo vai pra esquerda, PID é
     * positivo, portanto diminui a velocidade
     * do motor esquerdo*/
    int esqMotorVelo = 203 - PIDvalor;

    /* Se robo vai pra direira, PID é
     * negativo, portanto diminui a velocidade
     * do motor direito*/
    int dirMotorVelo = 203 + PIDvalor;

    analogWrite(dirMotor, dirMotorVelo);

    analogWrite(esqMotor, esqMotorVelo);

}
```

[ORIENTAÇÃO] **Desafio 2**: A taxa de cálculo do valor do PID deve ser constante ou isso é indiferente? (Em outras palavras, devo calcular o valor em um período preciso como por exemplo de 1 em 1 segundo, ou 100 em 100 milissegundos, ou isso é indiferente para consistência do erro obtido?) Dica: Pesquise vídeos de explicação sobre o PID.

MÃO NA MASSA (ou no computer)

Com base no que você viu nesse documento de referência, planeje seu próprio circuito para um robô seguidor de linha com o microcontrolador ESP32. Fica a seu critério escolher os outros componentes como driver do motor, sensores, regulador de tensão, entre outros...

Somos uma comunidade aberta para melhorias e caso conheça/encontre um microcontrolador que considera mais adequado para o circuito, por favor, fique à vontade desde que apresente a justificativa para tal, mostre para nós o porquê da escolha, venda seu peixe 😉.

Para construção do circuito utilize o software que achar melhor, caso não tenha um em mente, recomendamos o uso do Fritzing por sua facilidade (apenas para execução do desafio, tendo em mente que para construção dos nossos robôs utilizamos e recomendamos o uso do CircuitMaker).

- Link para download do Fritzing:
  https://github.com/fritzing/fritzing-app/releases

- Link para download do CircuitMaker:
  https://circuitmaker.com/installer-download

REFERÊNCIAS

A. TUTORIAL: ROBÔ SEGUIDOR DE LINHA COM CONTROLE PID E AJUSTES POR APLICATIVO ANDROID – Laboratório de Garagem – Disponível em: http://labdegaragem.com/profiles/blogs/tutorial-rob-seguidor-de-linha-com-controle-pid-e-ajustes-por.

B. ARDUINO LINE FOLLOWER WITH PID AND 90 DEGREE TURNS – Instructables – Disponível em: https://www.instructables.com/id/Arduino-Line-Follower/.

C. O QUE É PWM E PARA QUE SERVE? – Citisystems – Disponível em: https://www.citisystems.com.br/pwm/.

D. ESP8266 OVERVIEW – Espressif – Disponível em: https://www.espressif.com/en/products/hardware/esp8266ex/overview.

E. ATMEGA328P – Microchip – Disponível em: https://www.microchip.com/wwwproducts/en/ATmega328p.