**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**

**2019**

SUMÁRIO

[OBJETIVOS 3](#_Toc37712520)

[ELETRÔNICA 4](#_Toc37712521)

[**1.** Componentes principais 4](#_Toc37712522)

[**1.1.** Dispositivos sensores 4](#_Toc37712523)

[**1.2.** Alimentação do robô 5](#_Toc37712524)

[**1.3.** Dispositivos atuadores 6](#_Toc37712525)

[**2.** Componentes auxiliares 7](#_Toc37712526)

[**2.1.** Driver de motor 7](#_Toc37712527)

[**3.** O que é um microcontrolador? Pesquise sua função, principais componentes, pinos e comunicação (sugestão: pesquise por pinos analógicos/digitais e PWM). 9](#_Toc37712528)

[**4.** Para se orientar no seu trajeto é comum que robôs seguidores de linha utilizem sensores de refletância. No caso de um “array” (conjunto d e sensores), explique: 10](#_Toc37712529)

[**4.1.** Como o robô consegue determinar sua distância até a linha com base na leitura de um array? 10](#_Toc37712530)

[**4.2.** Como o robô ajusta sua posição com base nessa distância? (sugestão: pesquise sobre PID e os efeitos que os termos P, D e I geram na resposta do sistema). 11](#_Toc37712531)

[**5.** A partir de suas respostas, considerando que você possui um driver de motor como esse da fabricante Pololu, um Arduino Nano e os demais componentes pesquisados, em que pinos do microcontrolador você ligaria eles? Faça um esboço e justifique suas escolhas (você pode desenhar, criar uma tabela ou usar algum software para isso). 13](#_Toc37712532)

[PROGRAMAÇÃO 14](#_Toc37712533)

[**1.** Código 1052: Mês: 14](#_Toc37712534)

[**1.1.** Código-Fonte: 14](#_Toc37712535)

[**1.2.** Print URI: 14](#_Toc37712536)

[**1.** Código 1037: Intervalo I: 15](#_Toc37712537)

[**2.1.** Código-Fonte: 15](#_Toc37712538)

[**2.2.** Print URI: 15](#_Toc37712539)

[**2.** Código 1072: Intervalo II: 16](#_Toc37712540)

[**3.1.** Código-Fonte: 16](#_Toc37712541)

[**3.2.** Print URI: 16](#_Toc37712542)

[**3.** Código 2167: Falha do Motor: 17](#_Toc37712543)

[**1.3.** Código-Fonte: 17](#_Toc37712544)

[**1.4.** Print URI: 17](#_Toc37712545)

[REFERÊNCIAS 18](#_Toc37712546)

# 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:

****

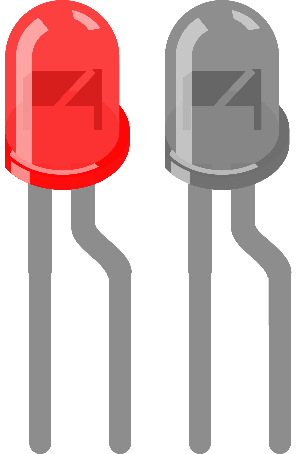
# ELETRÔNICA

## Componentes principais

### 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)



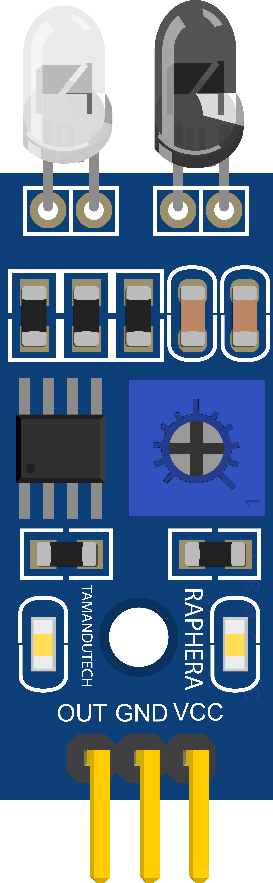
LED infravermelho

Fotodiodo

infravermelho

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:



LED infravermelho

Fotodiodo

infravermelho

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.

### 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.



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

* + - **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.

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.

### 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.



Pololu 30:1 6V HP Micro Metal Gearmotor

* + - **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).

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. Os conceitos PWM e Driver de motor serão descritos mais a frente neste documento.

## Componentes auxiliares

### 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:

* + Terminais básicos:



VM

VCC

GND

AO1

AO2

BO2

BO1

PWMA

AIN2

AIN1

STBY

BIN1

BIN2

PWMB

GND

GND

Driver Motor / Ponte H - TB6612FNG

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:

AO1

BO1

O1

Ex:

* + 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

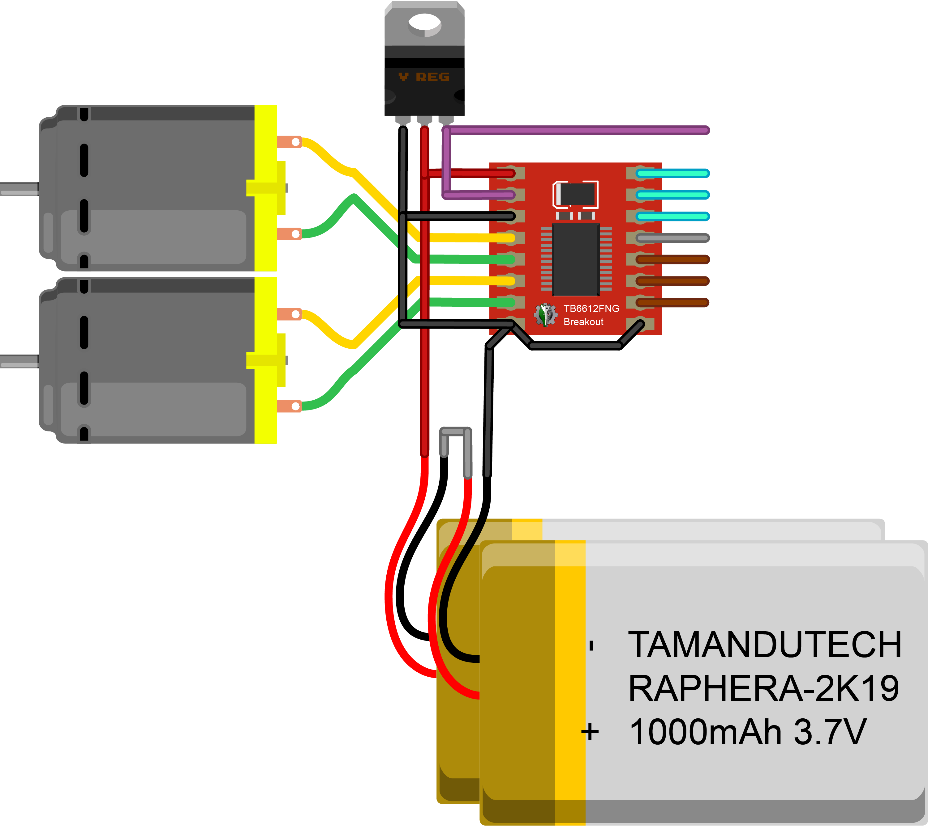
Os terminais do motor são divididos por:

* + - 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**.

CONTROLE MOTOR B



5V

STANDBY

CONTROLE MOTOR A

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

## O que é um microcontrolador? Pesquise sua função, principais componentes, pinos e comunicação (sugestão: pesquise por pinos analógicos/digitais e PWM).

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**

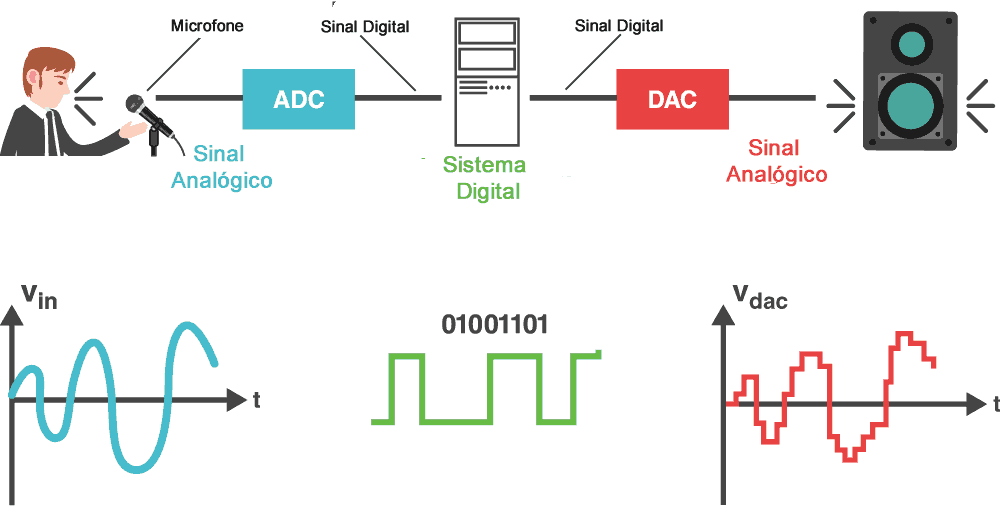
As portas digitais assumem níveis lógicos bem definidos, níveis representados em programa por 1 e 0 que assumem 5V/3,3V ou 0V, respectivamente.

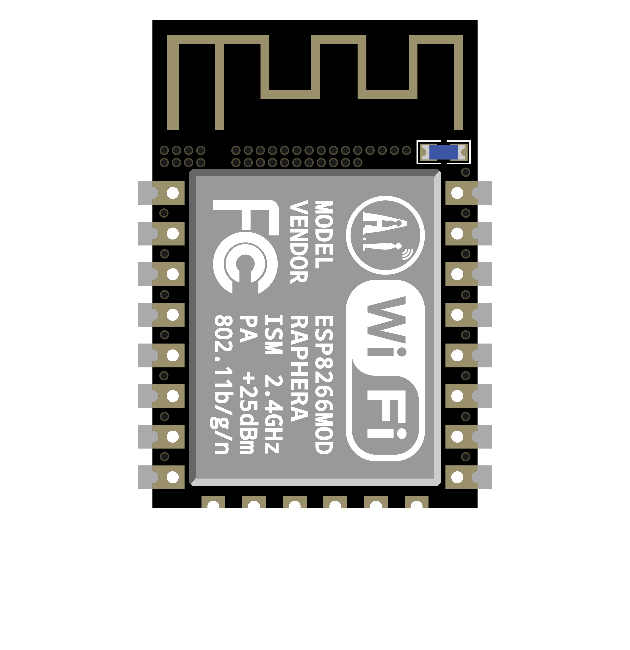
* **PORTAS ANLÓGICAS**

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.

## Para se orientar no seu trajeto é comum que robôs seguidores de linha utilizem sensores de refletância. No caso de um “array” (conjunto d e sensores), explique:

### Como o robô consegue determinar sua distância até a linha com base na leitura de um array?

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.

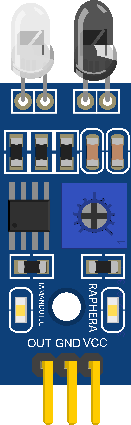
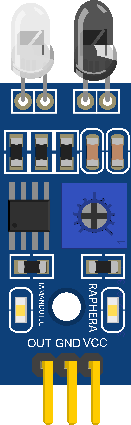
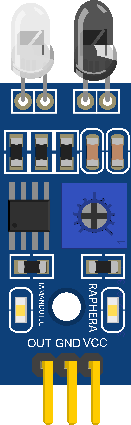
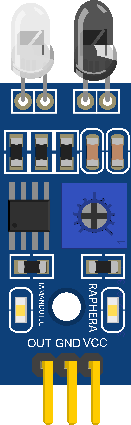
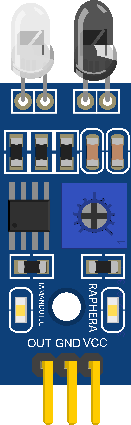
0

0

1

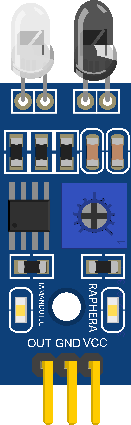
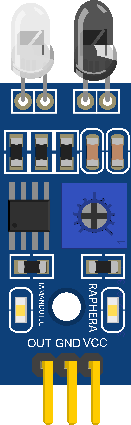
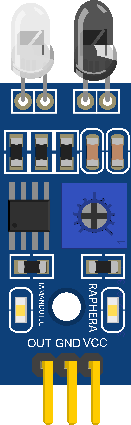
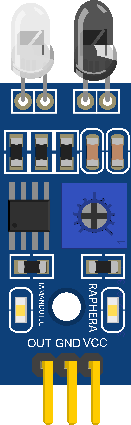
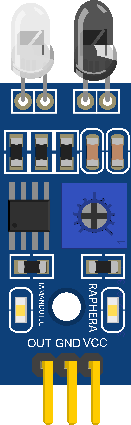
0

0



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:

### Como o robô ajusta sua posição com base nessa distância? (sugestão: pesquise sobre PID e os efeitos que os termos P, D e I geram na resposta do sistema).

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.

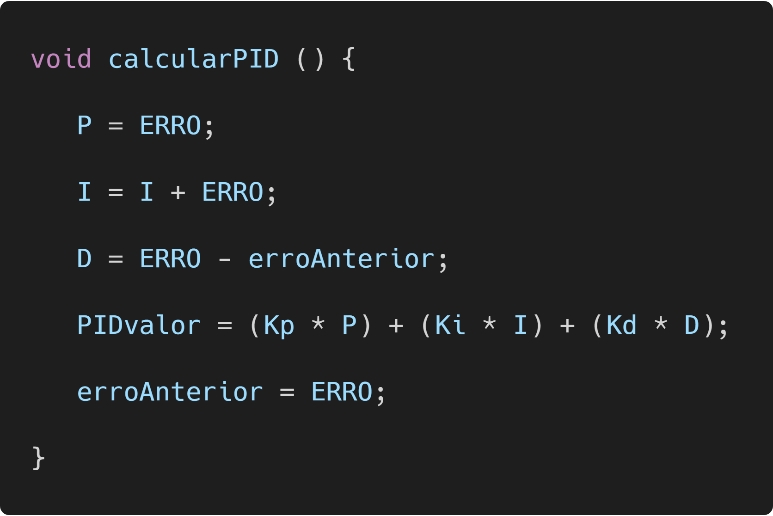
I = I + ERRO.

* Diferencial:

É a diferença entre o erro instantâneo do ponto de ajuste, e o erro a partir do instante anterior.

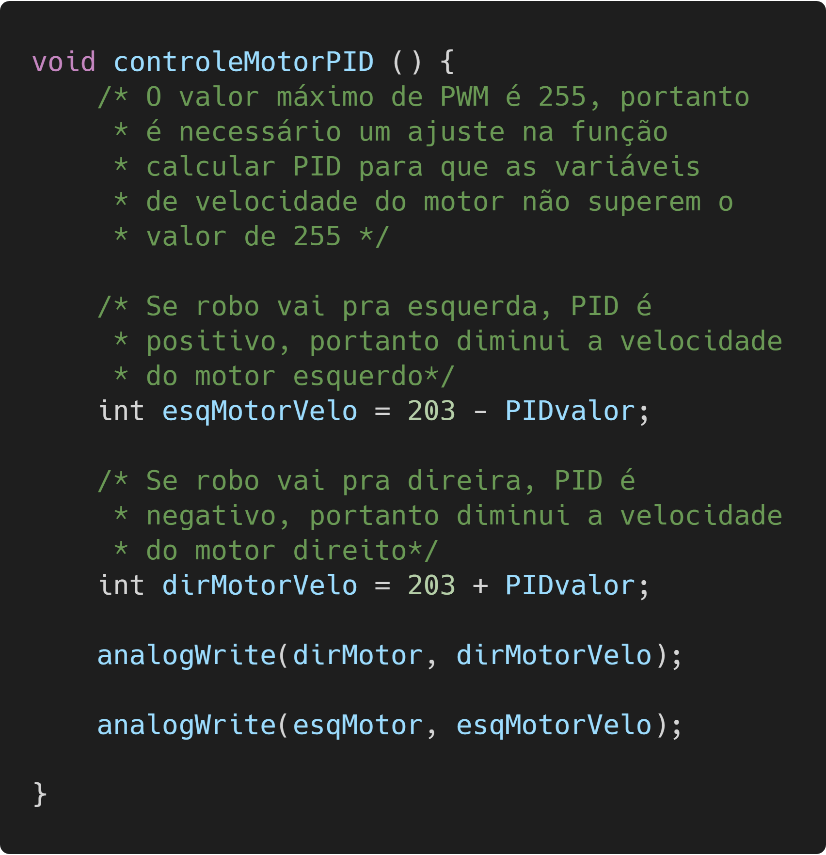
D = ERRO – erroAnterior.

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



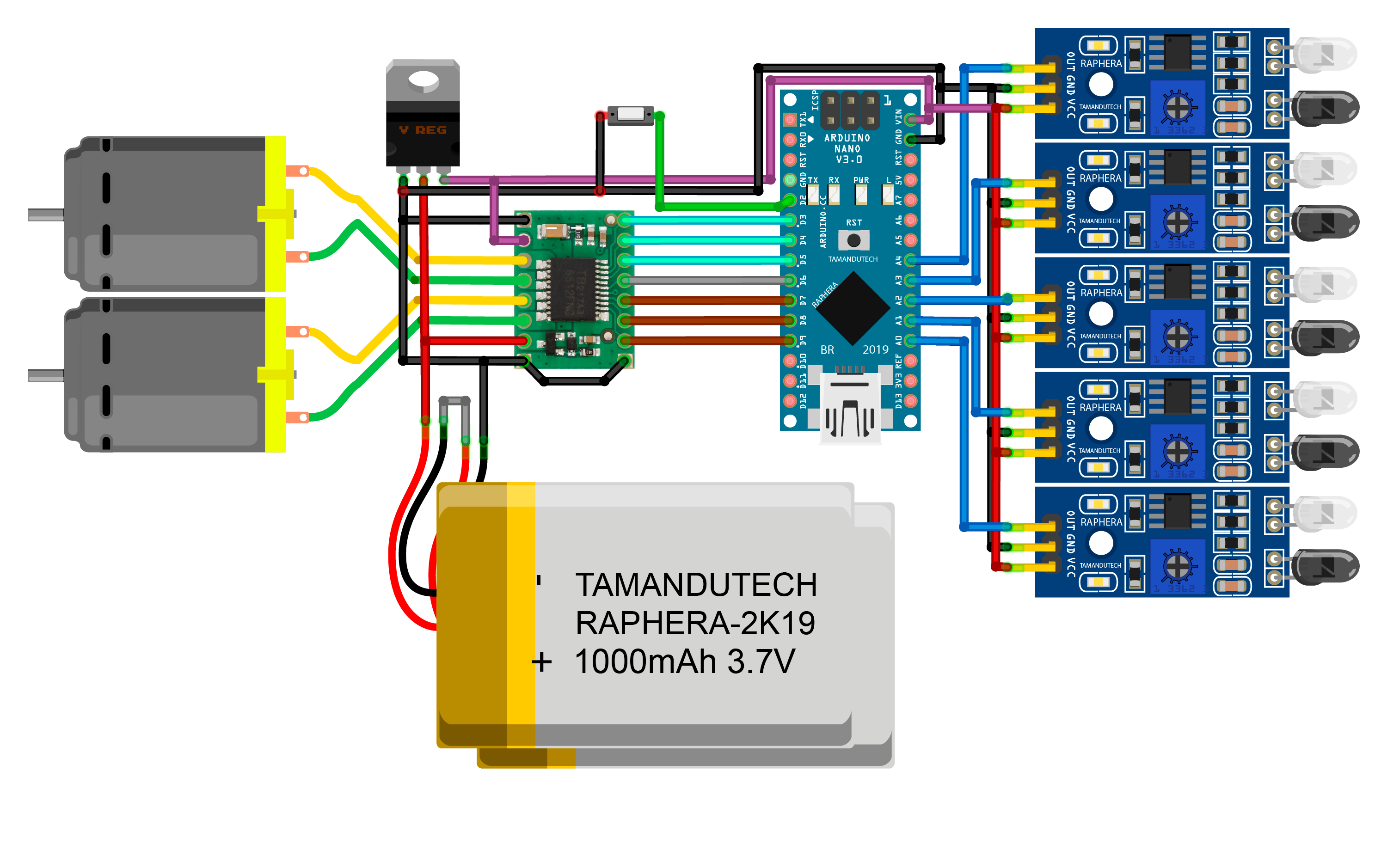
Equação PID

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



## A partir de suas respostas, considerando que você possui um driver de motor como esse da fabricante Pololu, um Arduino Nano e os demais componentes pesquisados, em que pinos do microcontrolador você ligaria eles? Faça um esboço e justifique suas escolhas (você pode desenhar, criar uma tabela ou usar algum software para isso).

Segue abaixo as ligações:



No Arduino para cada motor no driver é necessário 1 PWM, esse é o nosso fator limitante, o restante das conexões pode ser feito em qualquer uma das outras portas do Arduino. A soma das tensões da bateria resulta em uma tensão de no máximo 8,4V e no mínimo 6V, portanto foi necessário a inclusão de um regulador de tensão para alimentar o Arduino e os outros componentes. Foi incluído um pushbutton para que dê um “Start” no programa e o robô comece a andar. O array de sensores foi feito dessa forma pois no software utilizado para desenhar o circuito não havia nenhum módulo confeccionado com todos os sensores em apenas uma placa.

# PROGRAMAÇÃO

## Código 1052: [Mês](https://www.urionlinejudge.com.br/judge/en/problems/view/1052):

### Código-Fonte:

#include <stdio.h>  
  
char months[12][15] = {"January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"};  
  
int main() {  
 int month;  
 scanf("%d", &month);  
 printf("%s\n", months[month - 1]);  
 return 0;  
}

### Print URI:

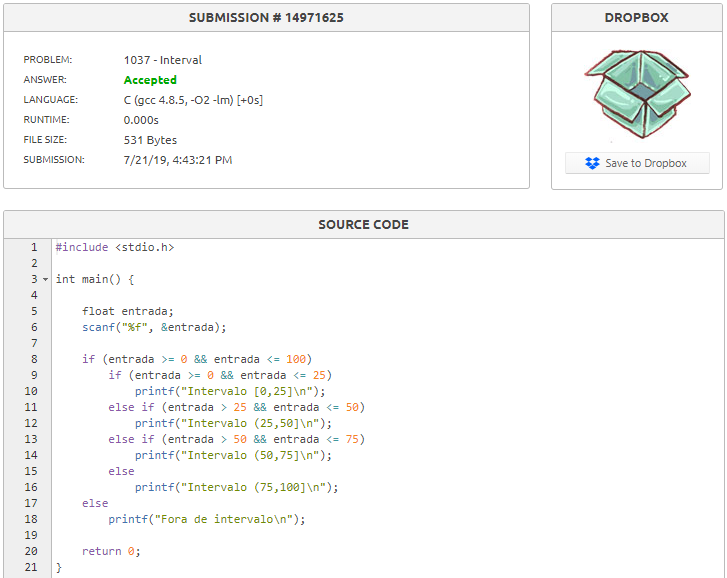


## Código 1037: [Intervalo I](https://www.urionlinejudge.com.br/judge/en/problems/view/1037):

### Código-Fonte:

#include<stdio.h>  
  
int main() {  
  
 float entrada;  
 scanf("%f", &entrada);  
  
 if (entrada >= 0 && entrada <= 100)  
 if (entrada >= 0 && entrada <= 25)  
 printf("Intervalo [0,25]\n");  
 else if (entrada > 25 && entrada <= 50)  
 printf("Intervalo (25,50]\n");  
 else if (entrada > 50 && entrada <= 75)  
 printf("Intervalo (50,75]\n");  
 else  
 printf("Intervalo (75,100]\n");  
 else  
 printf("Fora de intervalo\n");  
  
 return 0;  
}

### Print URI:

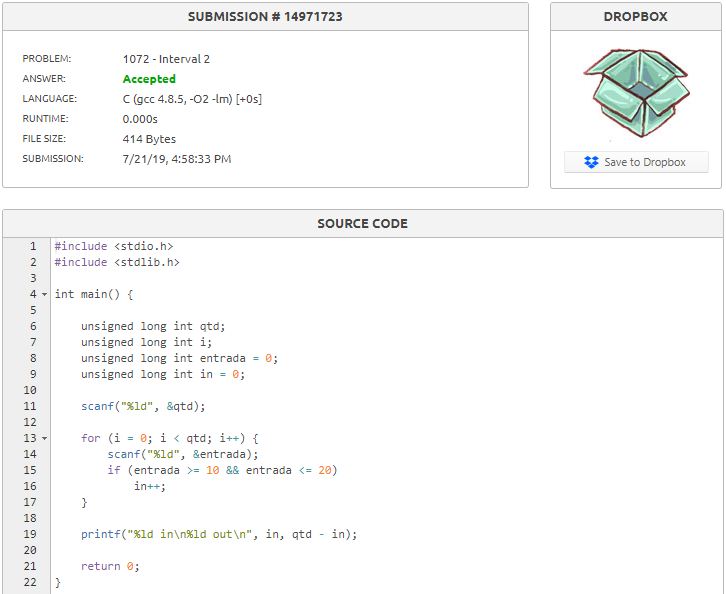


## Código 1072: [Intervalo II](https://www.urionlinejudge.com.br/judge/en/problems/view/1072):

### Código-Fonte:

#include <stdio.h>   
  
int main() {  
  
 unsigned long int qtd;  
 unsigned long int i;  
 unsigned long int entrada = 0;  
 unsigned long int in = 0;  
  
 scanf("%ld", &qtd);  
  
 for (i = 0; i < qtd; i++) {  
 scanf("%ld", &entrada);  
 if (entrada >= 10 && entrada <= 20)  
 in++;  
 }  
  
 printf("%ld in\n%ld out\n", in, qtd - in);  
  
 return 0;  
}

### Print URI:

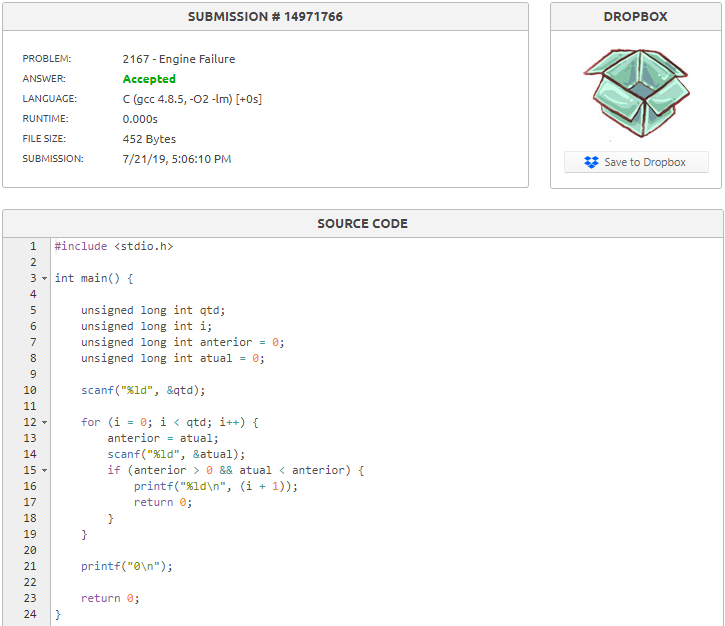


## Código 2167: [Falha do Motor](https://www.urionlinejudge.com.br/judge/en/problems/view/2167):

### Código-Fonte:

#include <stdio.h>  
  
int main() {  
  
 unsigned long int qtd;  
 unsigned long int i;  
 unsigned long int anterior = 0;  
 unsigned long int atual = 0;  
  
 scanf("%ld", &qtd);  
  
 for (i = 0; i < qtd; i++) {  
 anterior = atual;  
 scanf("%ld", &atual);  
 if (anterior > 0 && atual < anterior) {  
 printf("%ld\n", (i + 1));  
 return 0;  
 }  
 }  
  
 printf("0\n");  
  
 return 0;  
}

### Print URI:



# REFERÊNCIAS

* 1. 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>.
  2. ARDUINO LINE FOLLOWER WITH PID AND 90 DEGREE TURNS – Instructables – Disponível em: <https://www.instructables.com/id/Arduino-Line-Follower/>.
  3. O QUE É PWM E PARA QUE SERVE? – Citisystems – Disponível em: <https://www.citisystems.com.br/pwm/>.
  4. ESP8266 OVERVIEW – Espressif – Disponível em: <https://www.espressif.com/en/products/hardware/esp8266ex/overview>.
  5. ATMEGA328P – Microchip – Disponível em:  [<https://www.microchip.com/wwwproducts/en/ATmega328p>](http://saeedsolutions.blogspot.com/2013/07/8051-dac-using-dac0808-code-proteus.html).