Oficina de Robótica com Arduino

FAB LAB
JOINVILLE

Sumário

Softwares Utilizados

- Arduino IDE
- 123D Circuits

Laboratório 1 - Piscando um LED

- Partes
- Eletrônica
- Código
- O que aconteceu?
- Comandos aprendidos

Laboratório 2 - Acionando o LED com um botão

- Partes
- Eletrônica
- Código
- O que aconteceu?
- Comandos aprendidos

Laboratório 3 - Controlando o LED com um potenciômetro

- Partes
- Eletrônica
- Código
- O que aconteceu?
- Comandos Aprendidos

PopPet

- Eletrônica
- Código

Softwares Utilizados

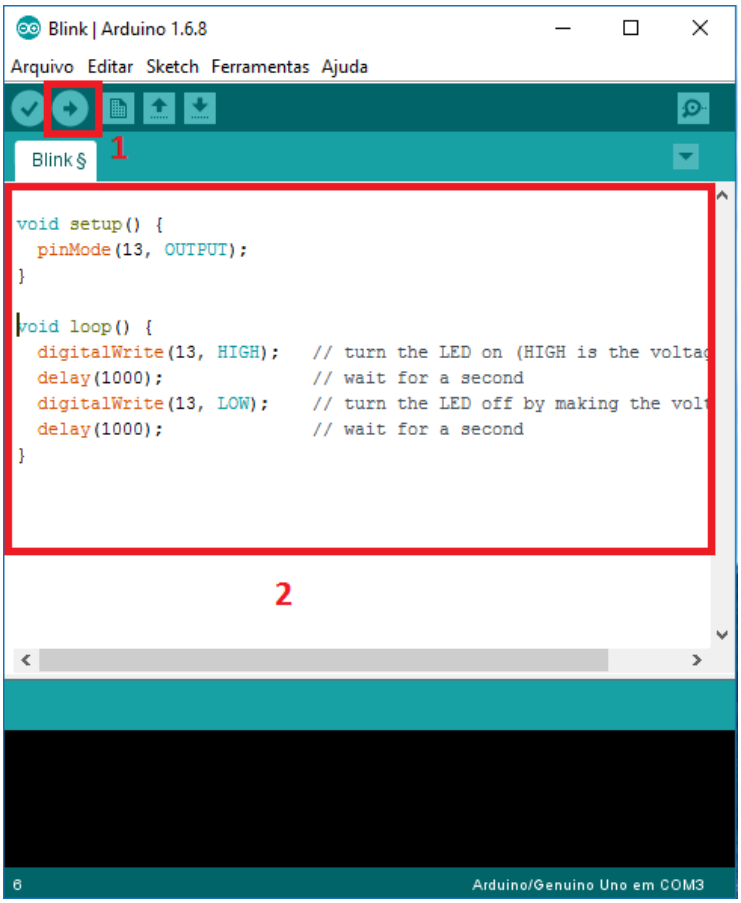
Para desenvolver os laboratórios deste guia utilizaremos dois sofwares, o Arduino IDE e o 123D Circuits.

Arduino IDE

O Arduino IDE permite que a placa Arduino seja programada de forma fácil, sem se preocupar com muitos detalhes que envolvem a compilação e a transferência do programa para a placa do Arduino. Para utilizar basta baixar o programa utilizando o link:

Download do Arduino IDE: https://www.arduino.cc/en/Main/Software

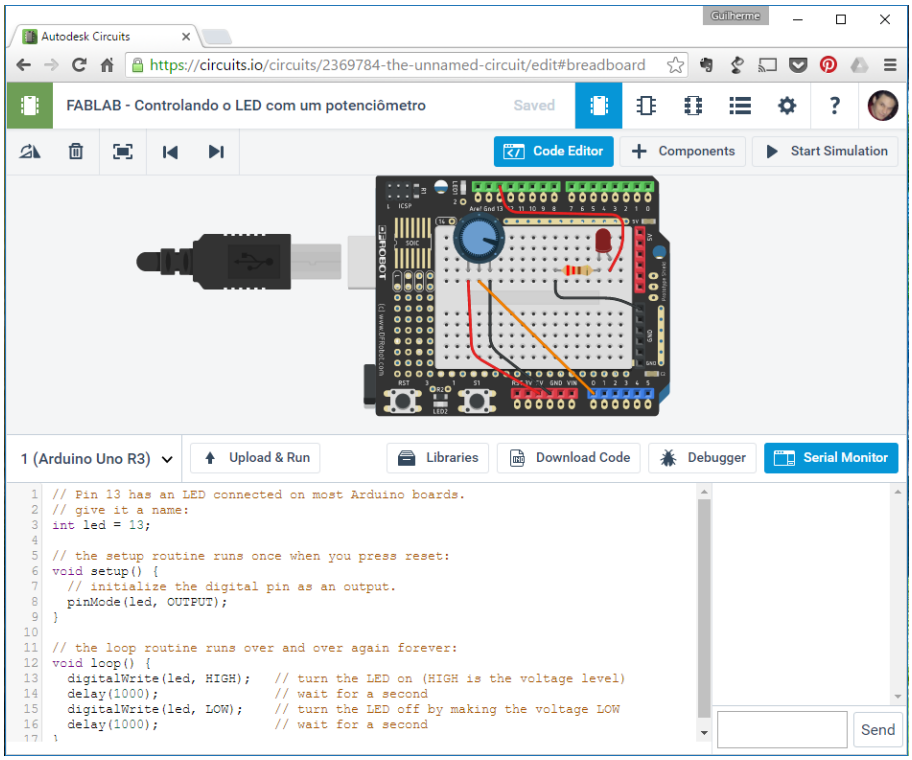
Ao abrir o programa pela primeira vez você verá uma tela como a imagem abaixo:



O botão 1 serve para compilarmos executarmos o código na placa Arduino. Na área 2 está o código que será executado.

123D Circuits

Outra ferramenta que utilizaremos é o site 123D Circuits (123d.circuits.io). É um site e você não precisa instalar nada no seu computador, basta entrar e se cadastrar. Neste site é possível montar os componentes eletrônicos, escrever o código e simular sua execução antes de montar todos esses passo na placa física. A imagem abaixo mostra um exemplo.



Todos os laboratórios serão disponibilizados no 123D Design para que você possa ver com calma as ligações, o código e possa executar a simulação para ver o seu funcionamento.

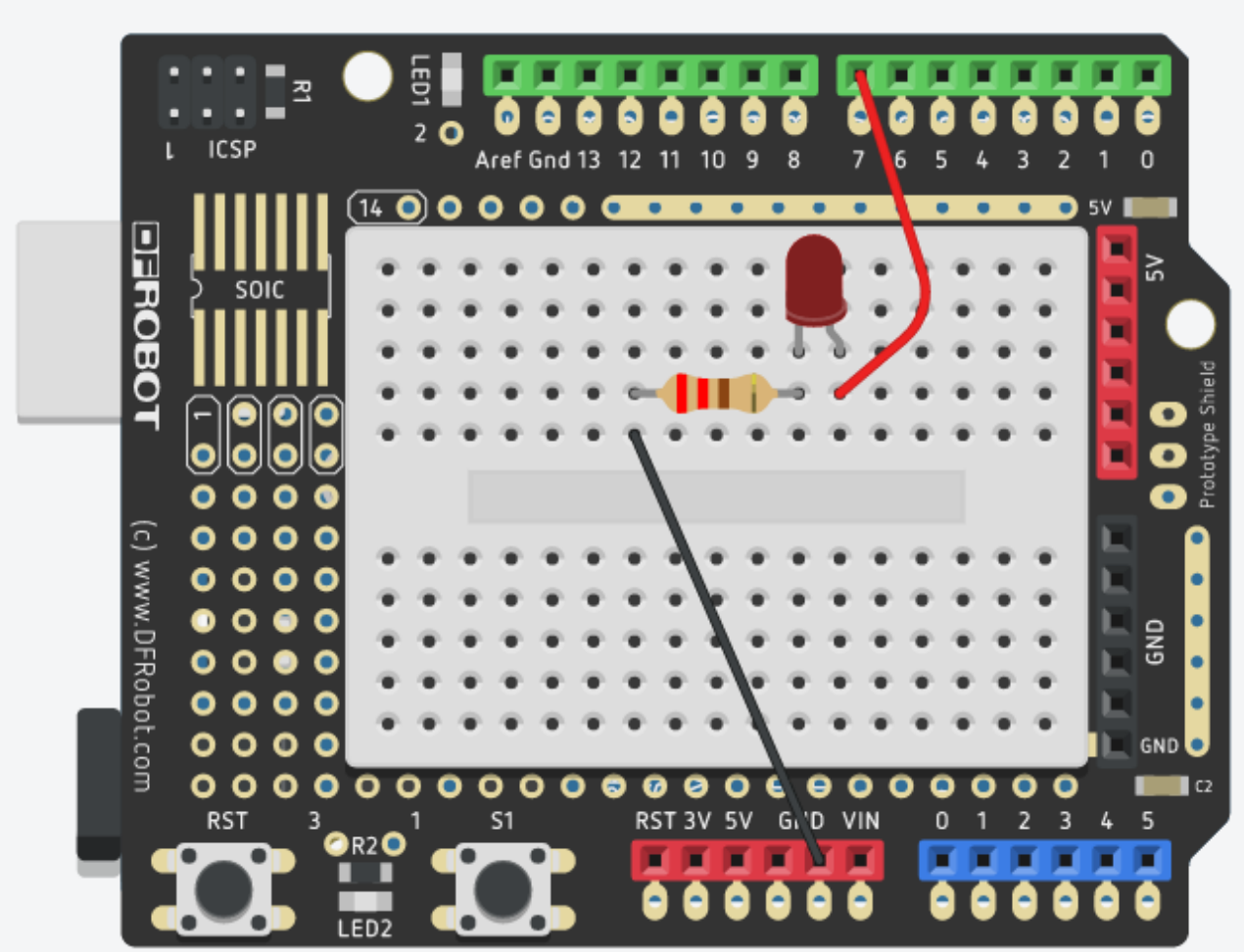
Laboratório 1 - Piscando um LED

Nesse laboratório controlaremos o acionamento de um LED utilizando o Arduino. Para ligar os componentes eletrônicos utilizaremos uma protoboard. A protoboard permite ligar os componentes sem a necessidade de soldá-los definitivamente.

Partes

1x LED (qualquer cor)
1x Resistor 220Ω
2x Jumpers (cabos)
1x Arduino
1x Protoboard

Eletrônica



Link para o labortatório:
http://bit.ly/fablab_lab1

Código

```
1. //Identifica em qual pino o LED foi
2. //ligado
3. int led = 7;
4.
5. void setup() {
6.  //Configura o pino 7 como saída
7.  pinMode(led, OUTPUT);
8. }
9.
10.
11. void loop() {
12.  //Liga o LED
13.  digitalWrite(led, HIGH);
14.
15.  //Espera 1 segundo
16.  delay(1000);
17.
18.  //Desliga o LED
19.  digitalWrite(led, LOW);
20.
21.  //Espera 1 segundo
22.  delay(1000);
23. }
```

O que aconteceu?

Para que o Arduino saiba o que fazer é necessário programá-lo. O programa escrito acima possui duas partes, a função **setup** e a **loop**. A função setup é executada uma única vez quando o Arduino é ligado. Já a função **loop** é executada repetidamente. O Arduino fica executando a função **loop** continuamente, quando ela termina, ela começa do início novamente.

Portas de saída
O Arduino possui diversas portas para se comunicar com o mundo externo, acender luzes, verificar o valor de sensores (ex: um sensor de temperatura). No programa anterior utilizamos a porta 7 para acionar um LED. No Arduino uma mesma porta pode funcionar como entrada ou saída, para isso precisamos informar o Arduino como a porta 7 deve ser tratada (entrada ou saída). Nesse caso a porta funciona como saída, então na função **setup** é executado o comando **pinMode(led, OUTPUT)**. Nesse comando foi informado o número da porta (led = 7) e o tipo (OUTPUT).

A função **loop** é o coração do programa, ela é executada repetidamente até que o Arduino seja desligado. No programa acima utilizamos o comando **digitalWrite(led, HIGH)** para ligar o LED e digitalWrite(led, LOW) para desligá-lo. Repare que a única diferença é o HIGH e LOW. O comando delay(2000) permite fazer o Arduino esperar um período de tempo. Esse período deve ser informado em milisegundos.

Comandos aprendidos

Comando	**Descrição**
pinMode(porta, OUTPUT);	Configura uma porta como saída.
digitalWrite(porta, HIGH);	Liga uma porta
digitalWrite(porta, LOW);	Desliga uma porta
delay(tempo);	Faz o Arduino esperar um período de tempo em milisegundos

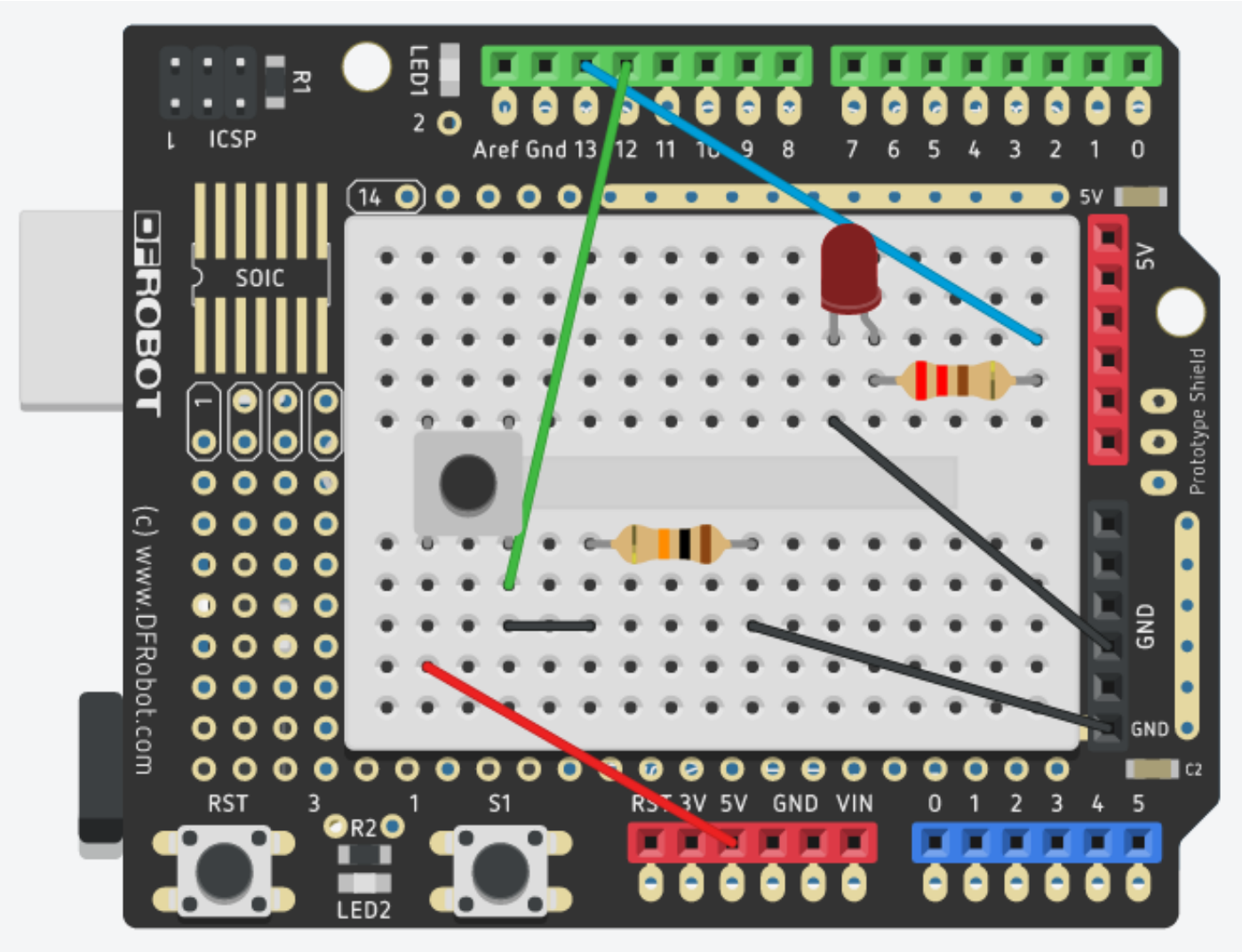
Laboratório 2 - Acionando o LED com um botão

Além das saídas digitais, o Arduíno também permite que sejam utilizados componentes de entrada, como sensores e botões, para que seja possível tomar decisões. Neste exercício vamos utilizar um botão para instruir o Arduino a acender um LED quando pressionado.

Partes

1x Arduino
1x Protoboard
1x Led
1x Resistor 220Ω
1x Botao
1x Resistor 10kΩ

Eletrônica



Link para o laboratório:
http://bit.ly/fablab_lab2

Código

```
1. //Cria uma variável led porta 13 para o LED
2. int led = 13;
3.
4. //Cria uma variável botao para a porta 12
5. int botao = 12;
6.
7. void setup() {
8.   //Configura a porta do LED como o
9.   //tipo de saída
10.  //(OUTPUT)
11.  pinMode(led, OUTPUT);
12.
13.  //Configura a porta do botao como o tipo de
14.  //entrada (INPUT)
15.  pinMode(botao, INPUT);
16. }
17.
18.
19. void loop() {
20.  //Captura se o botao está pressionado
21.  int acionado = digitalRead(botao);
22.
23.  //Verifica se o botão está pressionado (HIGH)
24.  if(acionado == HIGH){
25.    //Se estiver pressionado, liga o LED
26.    digitalWrite(led, HIGH);
27.  }else{
28.    //Se não estiver, desliga o LED
29.    digitalWrite(led, LOW);
30.  }
31. }
```

O que aconteceu?

Neste exercício, além, de configurarmos o LED como no exercício 1, utilizamos também um botão. Repare no comando **pinMode(botao, INPUT)** (linha 15), ele configura a porta **botao** (12) para que funcione como uma entrada digital.

Na função **loop** temos a lógica principal do nosso programa. Ele verifica se um botão está pressionado e, caso esteja, acende o LED. Para fazer essa verificação é utilizado o comando **digitalRead(botao)** (linha 21). Esse comando captura se o botão está pressionado (HIGH) ou não (LOW).

Para verificar se o botão está pressionado é necessário comparar o resultado obtido do comando digitalRead com HIGH ou LOW, essa verificação é feita utilizando o comando **if** (do português SE), na linha 24. Se a variável **acionado** for igual a HIGH (repare que usamos duas vezes o símbolo de igual), então ligamos o LED com o comando **digitalWrite(led, HIGH)** (linha 26). Se a variável **acionado** não for igual a HIGH o programa executará tudo que está dentro do **else**, linha 27, nesse caso, desligará o LED com o comando **digitalWrite(led, LOW)** (linha 29).

Comandos aprendidos

Comando	**Descrição**
pinMode(porta, INPUT);	Configura uma porta como entrada.
int resultado = digitalRead(porta);	Captura o valor de uma entrada digital
if(resultado == HIGH){ }	Verifica se o resultado da entrada digital é HIGH (Ex: botão pressionado)

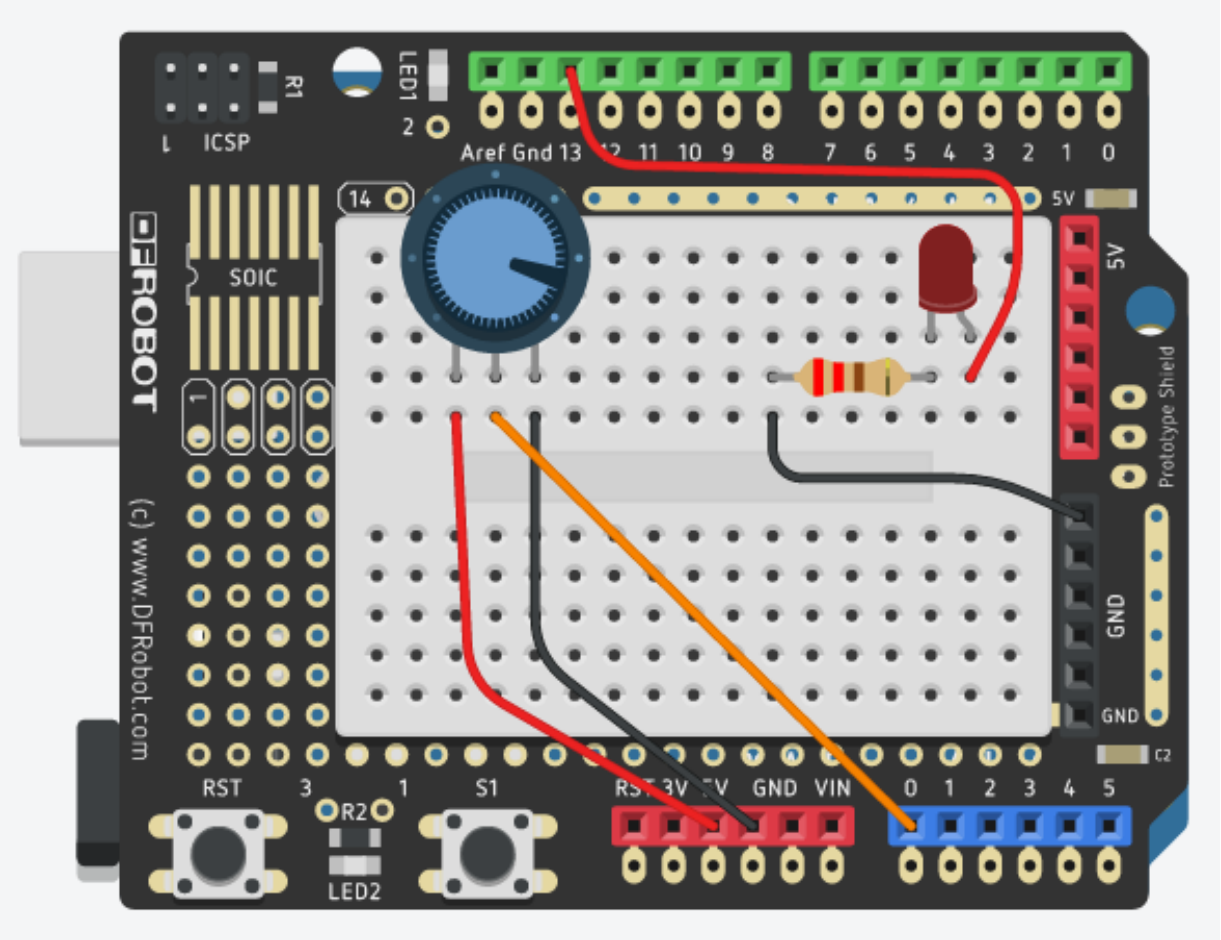
Laboratório 3 - Controlando o LED com um potenciômetro

Neste exercício controlaremos a velocidade em que o LED pisca a partir de um potenciômetro. O potenciômetro é um componente capaz de fornecer um valor analógico (uma variação entre 0v e 5v) em uma porta do Arduino e esse valor pode ser alterado rotacionando o botão do potenciômetro. Este valor será utilizado para determinar qual é o tempo de espera entre as piscadas do LED.

Partes

1x Arduino
1x Protoboard Shield
1x Resistor 220Ω
1x LED
1x Potenciômetro

Eletrônica



Link para o laboratório:
http://bit.ly/fablab_lab3

Código

```
1. //Cria uma variável para
2. //a porta do LED
3. int led = 13;
4.
5. //Cria uma variável para a
6. //porta do potenciômetro
7. int potenciometro = A0;
8.
9. //Cria uma variável para
10. //guardar o valor do
11. //potenciômetro
12. int valorPot = 0;
13.
14. void setup() {
15.   //Configura a porta do LED
16.  //como o tipo de saída
17.  pinMode(led, OUTPUT);
18.
19.  //Configura a porta
20.  //do potenciômetro
21.  //como o tipo de entrada
22.  pinMode(potenciometro, INPUT);
23.
24.  //Habilita a comunicação
25.  //com o computador
26.  Serial.begin(9600);
27. }
```

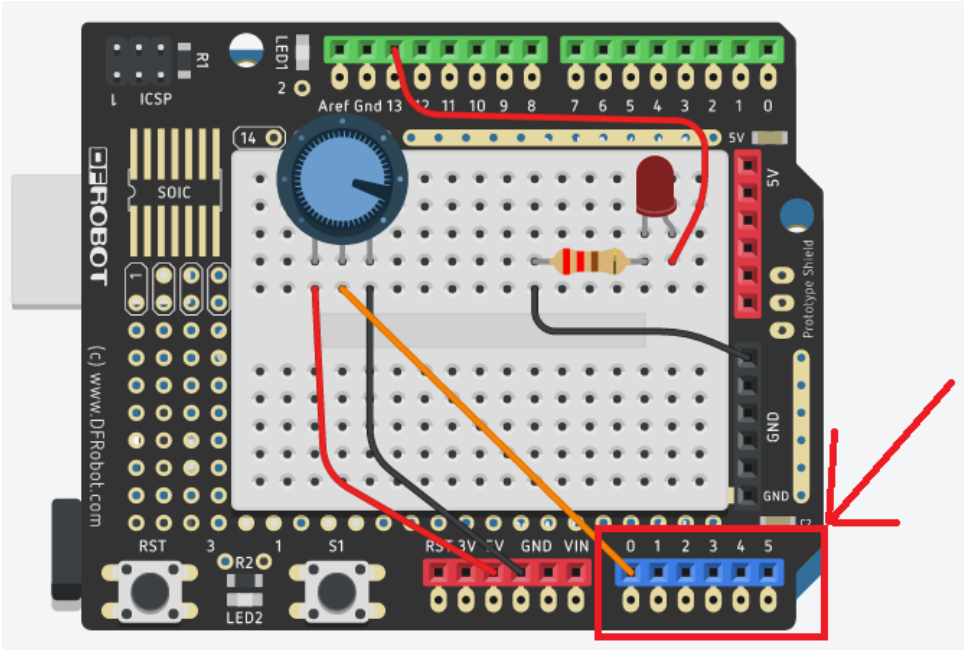
Código (continuação)

```
28. void loop() {
29.  //Captura o valor do
30.  //potenciômetro
31.  valorPot = analogRead(potenciometro);
32.
33.  //Mostra o valor do potenciômetro
34.  //na tela do computador
35.  Serial.print("Valor: ");
36.  Serial.println(valorPot);
37.
38.  //Liga o LED
39.  digitalWrite(led, HIGH);
40.
41.  //Espera uma quantidade de tempo
42.  //com base no valor que foi lido
43.  //pelo potenciômetro
44.  delay(valorPot);
45.
46.  //Desliga o LED
47.  digitalWrite(led, LOW);
48.
49.  //Espera uma quantidade de tempo
50.  //com base no valor que foi lido
51.  //pelo potenciômetro
52.  delay(valorPot);
53. }
```

O que aconteceu?

Neste exercício temos alguns elementos novos, mas não se preocupe, vamos ver com calma cada um deles. Nosso programa começa declarando uma variável **led** (linha 3) com o valor correspondente a porta em que o LED está ligado, a porta 13. Em seguida (na linha 7) é declarada a variável **potenciometro** com o valor correspondente a porta onde o potenciômetro está ligado. Repare que no me dessa porta agora é A0.

Além das portas de entrada e saída digitais, que funcionam apenas como ligada (HIGH) e desligada (LOW), o Arduino possui também a capacidade de ler dados analógicos. Os dados analógicos são valores que podem variar de 0v até 5v. Mas para ler dados analógicos precisamos utilizar algumas portas especiais do Arduino, as portas azuis destacadas na imagem abaixo:



Essas portas começam com a letra A (ex: A0, A1, A2, A3, A4 e A5). Por isso o potenciômetro está ligado na porta A0.

Na linha 12 declaramos uma variável do tipo inteiro **valorPot**. Guardaremos nessa variável o valor do potenciômetro em breve. Na função de **setup** é configurada a porta do LED como saída (linha 17) e é configurada a porta do portenciômetro como entrada utilizando o comando **pinMode(potenciometro, INPUT)**, na linha 22.

Na linha 26 temos um elemento novo, o **Serial.begin(9600)**. Como o Arduino não possui uma tela, fica difícil saber quais valores estão sendo lidos dos componentes, como o potênciômetro. Esse comando permite que o Arduino se comunique com o computador e mostre informações do Arduino no programa Serial Monitor no computador. Isso é muito útil para identificar problemas e saber o que o Arduino está fazendo.

Com as configurações feitas no **setup** vamos ver a lógica principal dentro do **loop**.

A primeira coisa que precisamos fazer é saber em qual posição está o potenciômetro. Isso é feito na linha 31. O comando `valorPot = analogRead(potenciometro);` obtém o valor da posição do potenciômetro e guarda na variável valorPot, essa posição pode variar entre 0 e 1023. Na linha 35 mostramos na tela do computador qual foi o valor lido. Na linha 39 o LED é ligado e na linha 44 utilizamos o valor lido do potenciômetro para fazer o Arduino aguardar. Como o valor ficará sempre entre 0 e 1023 o Arduino aguardará no máximo 1 segundo (1000ms). Nas linhas 47 e 52 o LED é desligado e é aguardado novamente até o ciclo se repetir.

Comandos Aprendidos

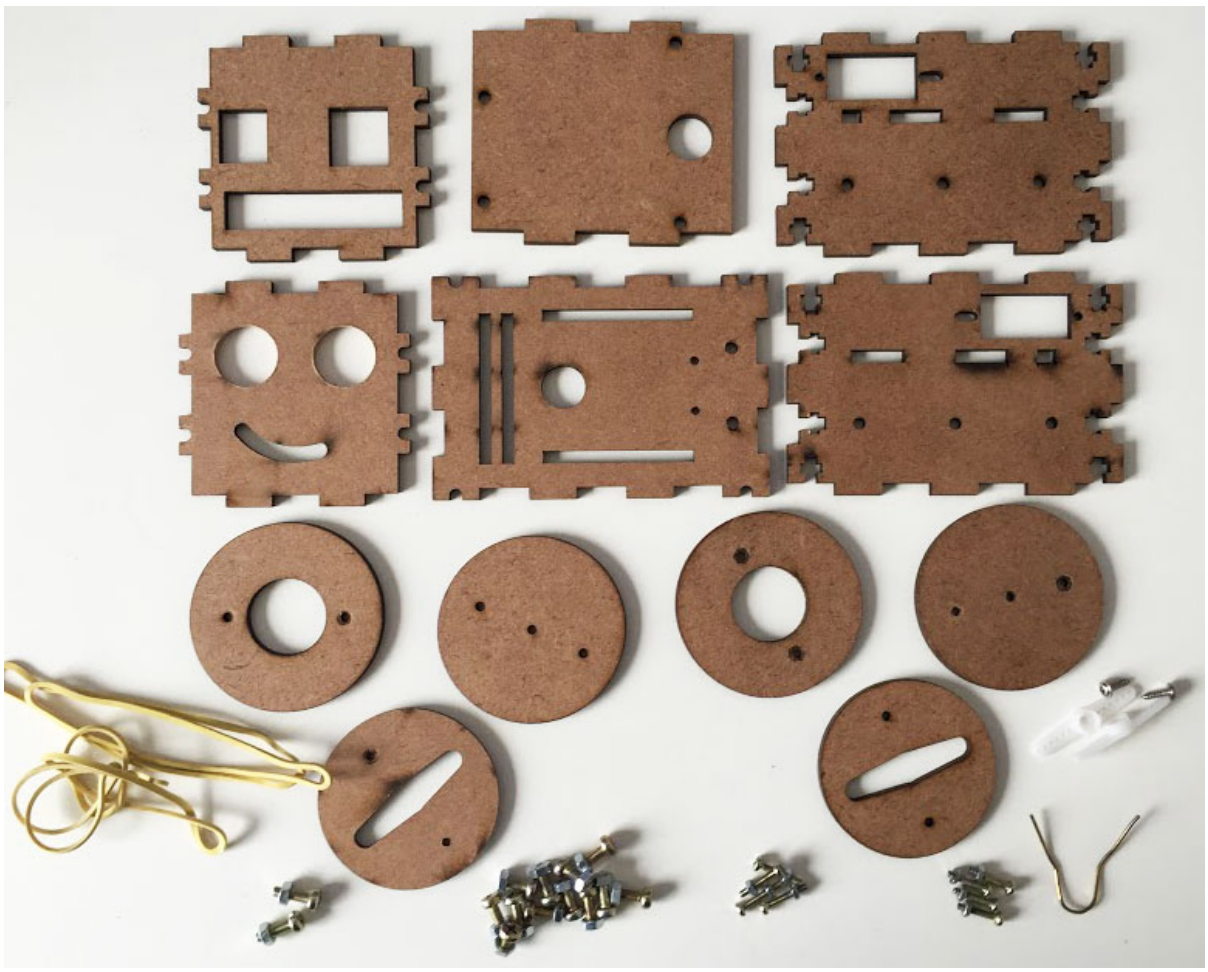
Comando	**Descrição**
Serial.begin(9600)	Configura a comunicação entre o Arduino e o computador
Serial.println(“mensagem”)	Mostra uma mensagem no Monitor Serial do computador
analogRead(porta)	Captura um valor analógico de uma porta (valor entre 0 e 1023)

PopPet

Agora que já conhecemos os fundamentos de como trabalhar com o Arduino vamos montar o nosso primeiro robô, o PopPet!
O robô consiste de 3 partes fundamentais:

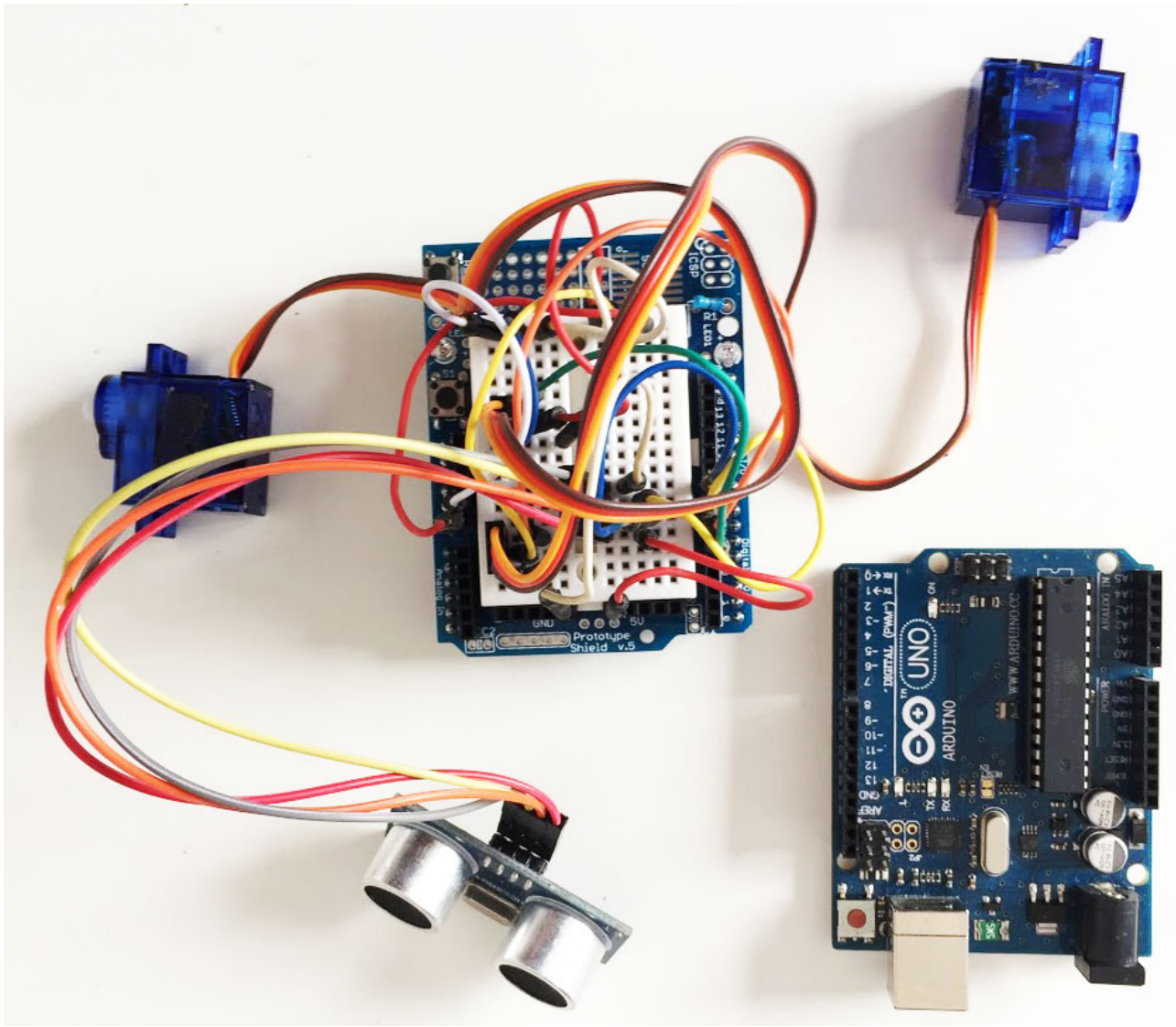
- Mecânica: chassi, rodas, suportes e parafusos
- Eletrônica: Arduino, protoboard, motores e sensor de ultrasom
- Programa/Código: Dará vida ao nosso robozinho.

A imagem abaixo mostra a parte mecânica do robô desmontada

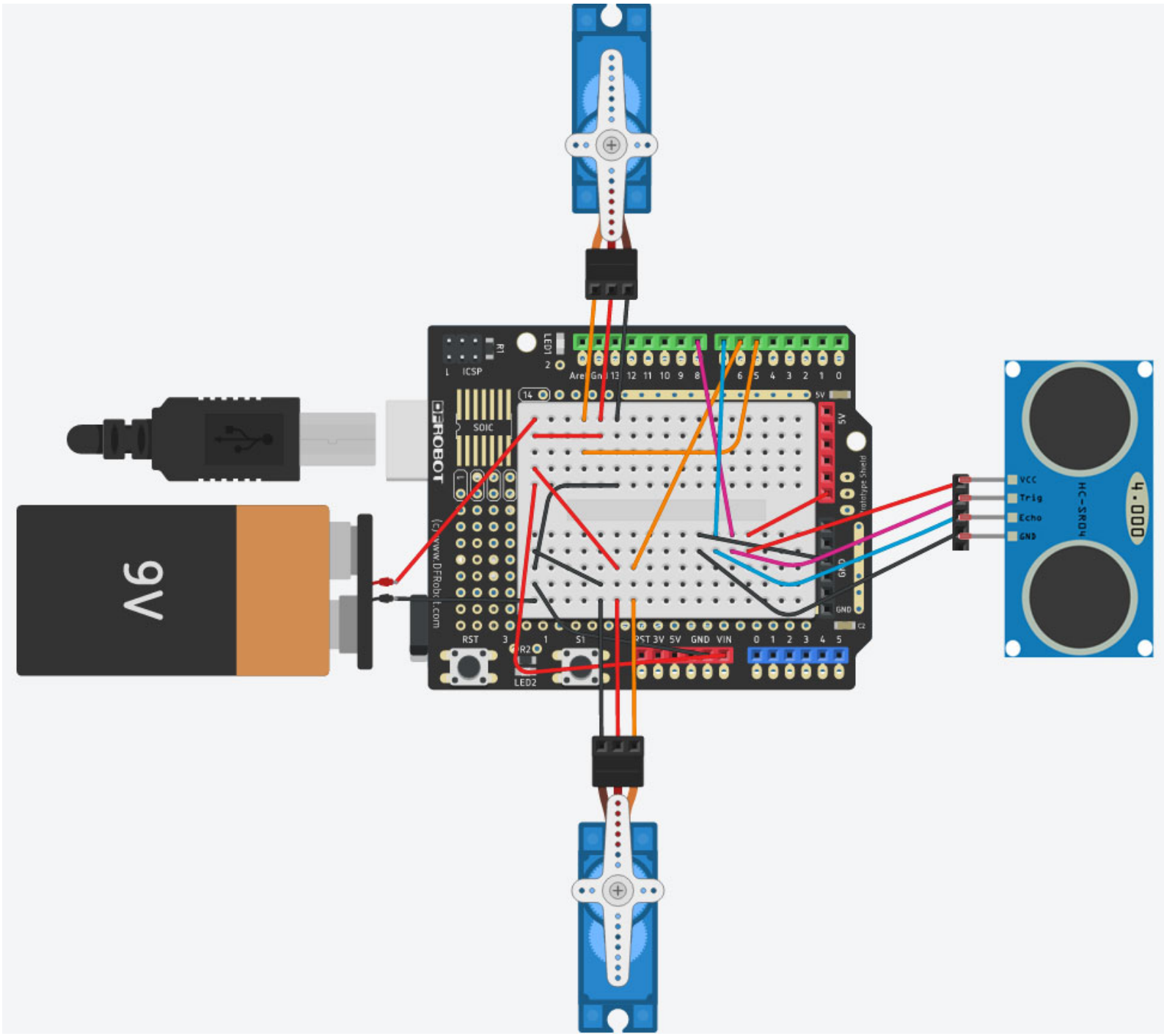


Não tem as partes mecânicas? Não se preocupe, você mesmo pode baixar o projeto e cortar ou imprimir.
Projeto: http://www.thingiverse.com/thing:1634750

Na imagem abaixo é possível ver os componentes eletrônicos do robô



Eletrônica



Código

Os códigos necessários para o funcionamento do robô podem ser obtidos em:
https://github.com/fablabjoinville/poppet/archive/master.zip

```
#include "PopPet.h"

//Cria um novo Pet
// 7 - Porta Echo
// 8 - Porta Trigger
// 6 - Porta Motor 1
// 5 - Porta Motor 2
PopPet pet(7,8,6,5);

 //Variável para
 //guardar a distancia
 int distancia = 0;

 void setup ()
 {
  //Configura a comunicação
  //com o computador
  Serial.begin(9600);

  //Inicializa o pet
  pet.Initialize();
 }

```

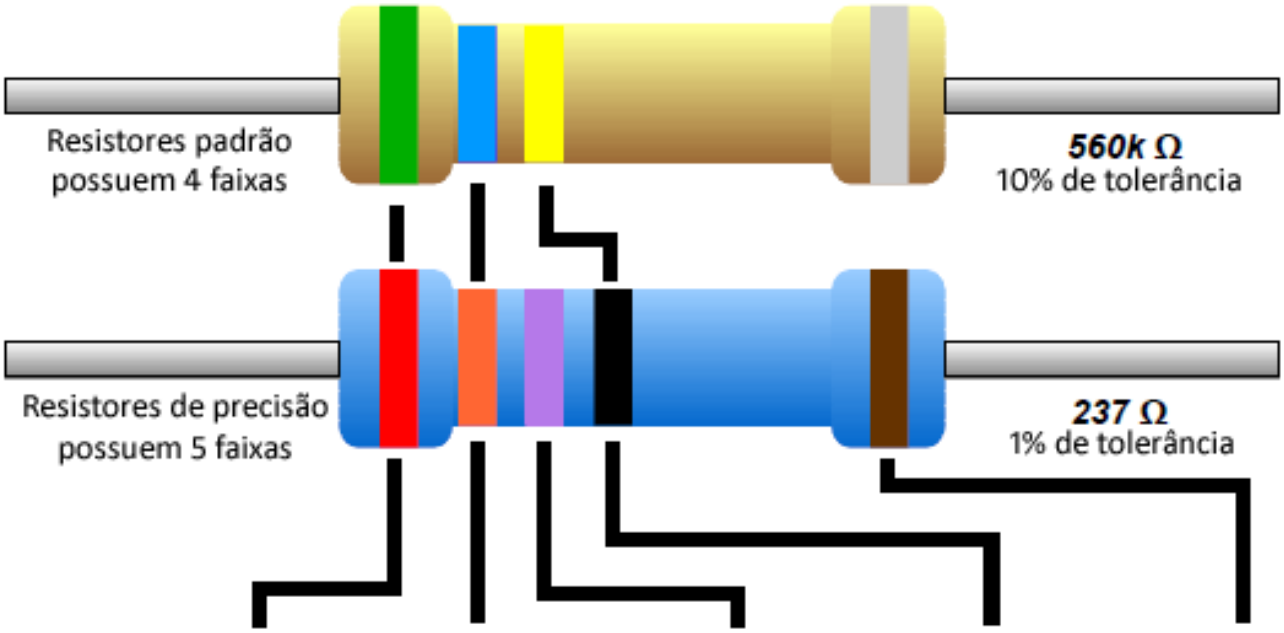
Código (continuação)

```
24. void loop()
25. {
26.   //Captura a distancia do
27.   //sensor ultrasonico
28.   distancia = pet.Distancia();
29.
30.   //Mostra a distância no
31.   //computador
32.   Serial.print("Distancia: ");
33.   Serial.println(distancia);

34.   //Se a distância for menor que 20cm
35.   if(distancia < 20){
36.     //Anda para a frente por 400ms
37.     pet.AndarFrente(400);
38.   }else{
39.     //Senão, para o robô
40.     pet.Parar();
41.    }
42. }
```

Código de Cores

A extremidade com mais faixas deve apontar para a esquerda



Cor	1ª Faixa	2ª Faixa	3ª Faixa	Multiplicador	Tolerância
Preto	0	0	0	x 1 Ω	
Marrom	1	1	1	x 10 Ω	+/- 1%
Vermelho	2	2	2	x 100 Ω	+/- 2%
Laranja	3	3	3	x 1K Ω	
Amarelo	4	4	4	x 10K Ω	
Verde	5	5	5	x 100K Ω	+/- .5%
Azul	6	6	6	x 1M Ω	+/- .25%
Violeta	7	7	7	x 10M Ω	+/- .1%
Cinza	8	8	8		+/- .05%
Branco	9	9	9		
Dourado				x .1 Ω	+/- 5%
Prateado				x .01 Ω	+/- 10%