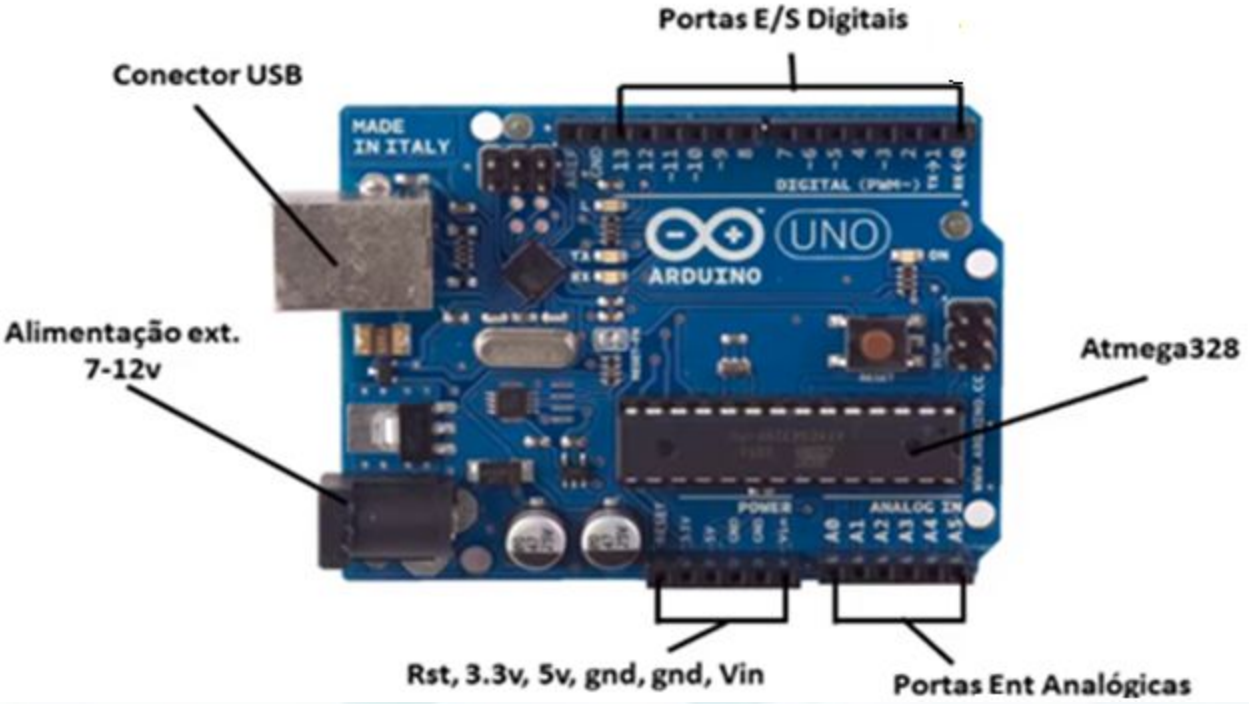
Arduíno

Um Arduino é um micro controlador de placa única e um conjunto de software para programá-lo. O hardware consiste em um projeto simples de hardware livre para o controlador, com um processador Atmel AVR e suporte embutido de entrada/saída. O software consiste de uma linguagem de programação padrão e do bootloader que roda na placa."

Em sintaxe: Arduino é um pequeno computador que pode ser programado para processar entradas e saídas entre o dispositivo e os componentes externos conectados a ele.

Arduíno – Conexões

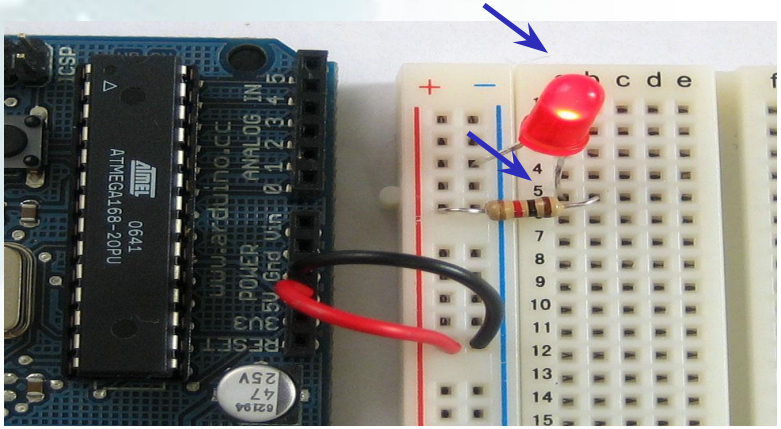


Arduíno – Componentes - Protoboard

O Arduino pode ser utilizado para desenvolver objetos interativos independentes, ou pode ser conectado a um computador, a uma rede, ou até mesmo à Internet para recuperar e enviar dados do Arduino e atuar sobre eles. Em outras palavras, ele pode enviar um conjunto de dados recebidos de alguns sensores para um site, dados estes que poderão, assim, ser exibidos na forma de um gráfico.

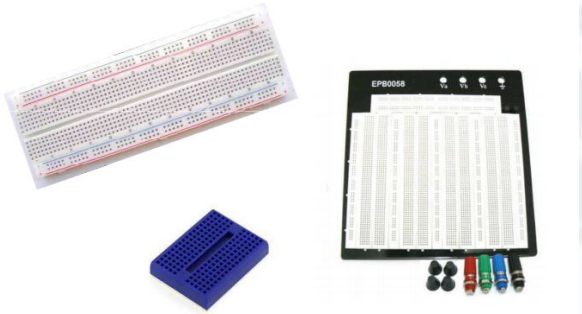
Chip Atmel Atmega – coração do Arduino.

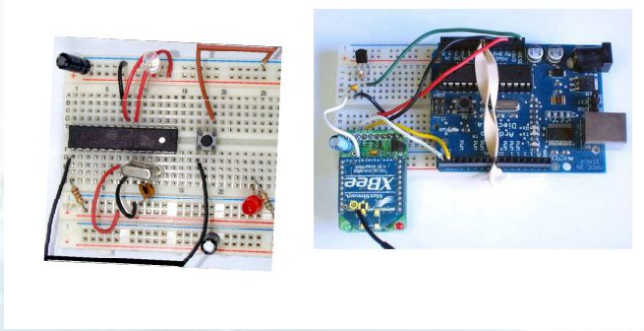
O Arduino pode ser conectado a LEDs, displays (mostradores) de matriz de pontos , botões, interruptores, motores, sensores de temperatura, sensores de pressão, sensores de distância, receptores GPS, módulos Ethernet ou qualquer outro dispositivo que emita dados ou possa ser controlado. Uma pesquisa na Internet revelará muitos outros projetos em que um Arduino foi utilizado para ler dados e controlar uma enorme quantidade de dispositivos.

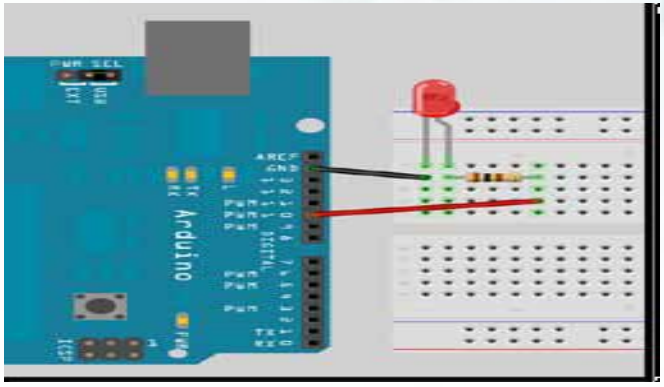


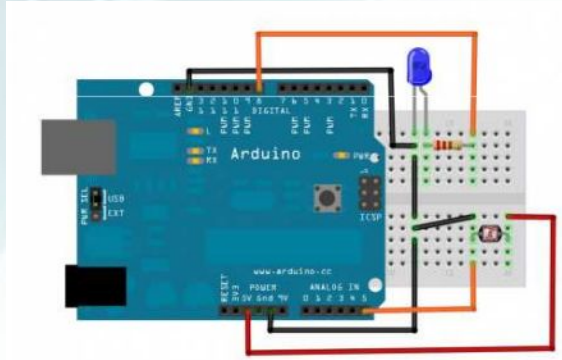
Arduíno – Componentes - Protoboard

Uma matriz de contato, ou placa de ensaio (ou protoboard, ou breadboard em inglês) é uma placa com furos e conexões condutoras para montagem de circuitos elétricos experimentais. A grande vantagem do protoboard na montagem de circuitos eletrônicos é a facilidade de inserção de componentes, uma vez que não necessita soldagem. As placas variam de 800 furos até 6000 furos, tendo conexões verticais e horizontais.









Arduíno – Programação

Para programar o Arduino (fazer com que ele faça o que você deseja) você utiliza o IDE do Arduino, um software livre no qual você escreve o código na linguagem que o Arduino compreende (baseada na linguagem C/C++ e pode até ser extendida por meio de bibliotecas C++). O IDE permite que você escreva um programa de computador sketches (rascunho, ou esboço)., que é um conjunto de instruções passo a passo, das quais você faz o upload para o Arduino. Seu Arduino, então, executará essas instruções, interagindo com o que estiver conectado a ele.

O hardware e o software são ambos de fonte aberta, o que significa que o código, os esquemas, o projeto etc. podem ser utilizados livremente por qualquer pessoa e com qualquer propósito.
Dessa forma, há muitas placas-clone e outras placas com base no Arduino disponíveis para compra, ou que podem ser criadas a partir de um diagrama. De fato, nada impede que você compre os componentes apropriados e crie seu próprio Arduino em uma matriz de pontos ou em sua PCB (Printed Circuit Board, placa de circuito impresso) feita em casa.

Como os projetos são de fonte aberta, qualquer placa-clone é 100% compatível com o Arduino e, dessa forma, qualquer software, hardware, shield (escudo), placas de circuito contendo outros dispositivos (por exemplo, receptores GPS, displays de LCD, módulos de Ethernet) etc. também será 100% compatível com o Arduino genuíno.

Versões de Arduino: Mini, Nano e Bluetooth do Arduino. Mega 2560, que oferece mais memória e um número maior de pinos de entrada/saída. Placas utilizam um novo bootloader, o Optiboot, que libera mais 1,5 kB de memória flash e permite uma inicialização mais rápida. Versão mais popular Uno (28 pinos ligados a um soquete C.I (circuito integrado).

Arduíno – Programação – Partes Básicas

A IDE foi desenvolvida com Java (necessita JVM), funciona em Windows. Mac OS X e Linux (pode precisar de driver) e utiliza GCC + GCC Avr para compilação, por sua vez a transferência para a placa é feita via USB pelo IDE e, pode ser baixada através do site www.arduino.cc

Métodos obrigatórios

```
void setup()
{
}
```

será executado uma única vez ao ligar a placa

```
void loop()
{
}
```

será executado infinitamente

Portas Digitais e Analógicas

Normalmente os componentes são ligados em portas digitais e analógicas por intermédio do código Arduino, utilizando as portas:

pinMode(<porta>,<modo>) □ **será configura uma porta digital para ser lida ou para enviarmos dados**

```
pinMode(13,OUTPUT);
pinMode(11,INPUT);
```

digitalWrite(<porta>, 0 ou 1) □ **envia 0 ou 1 para porta digital**

```
digitalWrite(11,1);
digitalWrite(11,0);
```

digitalRead(porta) □ **retorna um 0 ou 1 lido da porta**

```
int x;
x = digitalRead(11);
```

analogRead(<porta>) □ **retorna de 0 a 1023 com o valor da porta analógica**

```
int luz = analogRead(0);
```

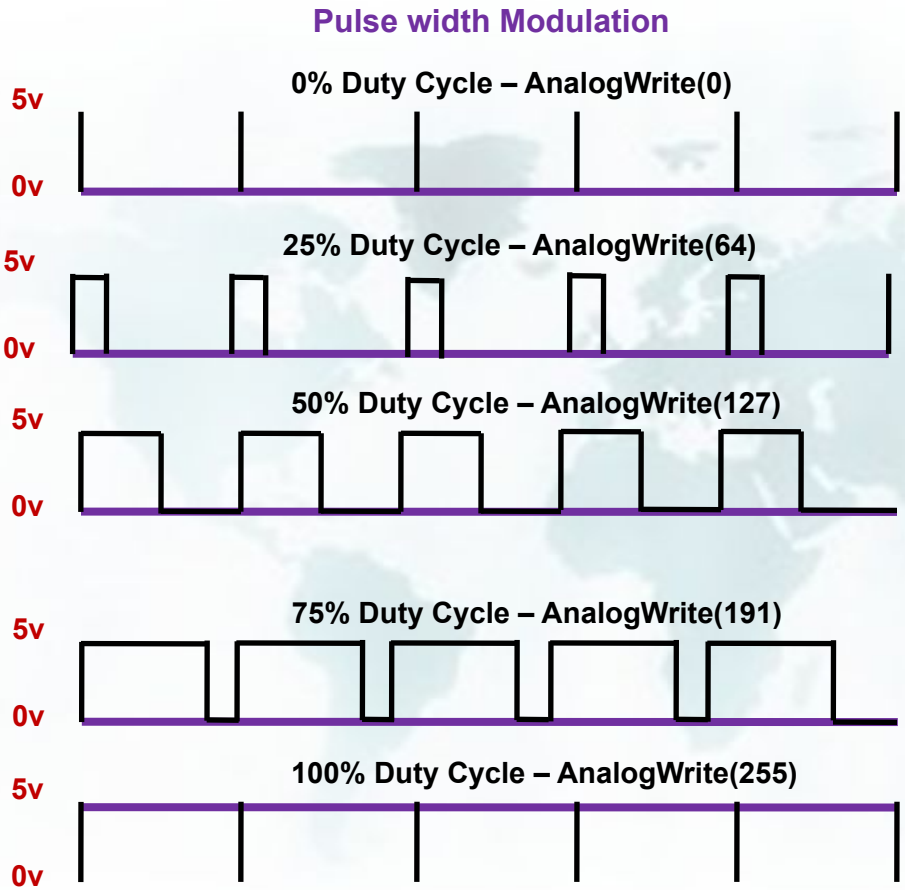
analogWrite(<porta>,<valor>) □ **escreve em uma porta PWM um valor de 0 a 255**

```
analogWrite(11,200);
```

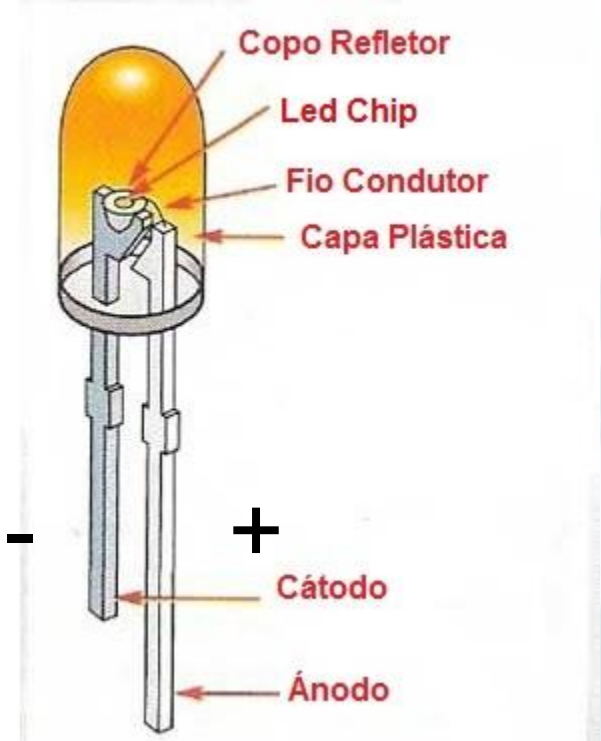
Arduíno – Portas Híbridas – PWM

Pulse width Modulation (Pulso com modulação).

Trata-se de uma porta digital, no entanto, é possuidora de modularização de zeros e uns, de modo que, ela consegue expressar uma idéia de potência.



Arduíno – Componentes



O **diodo emissor de luz**

- Conhecido pela sigla

LED (Light EmittingDiode)

- Sua funcionalidade básica é a emissão de luz
- Voltagem:
  - Vermelho 1.6 V
  - Verde 2.1 V
  - Amarelo 2.1 V
  - Laranja 2.2 V
  - Azul 4-5 V

Arduíno – Resistores - Identificação

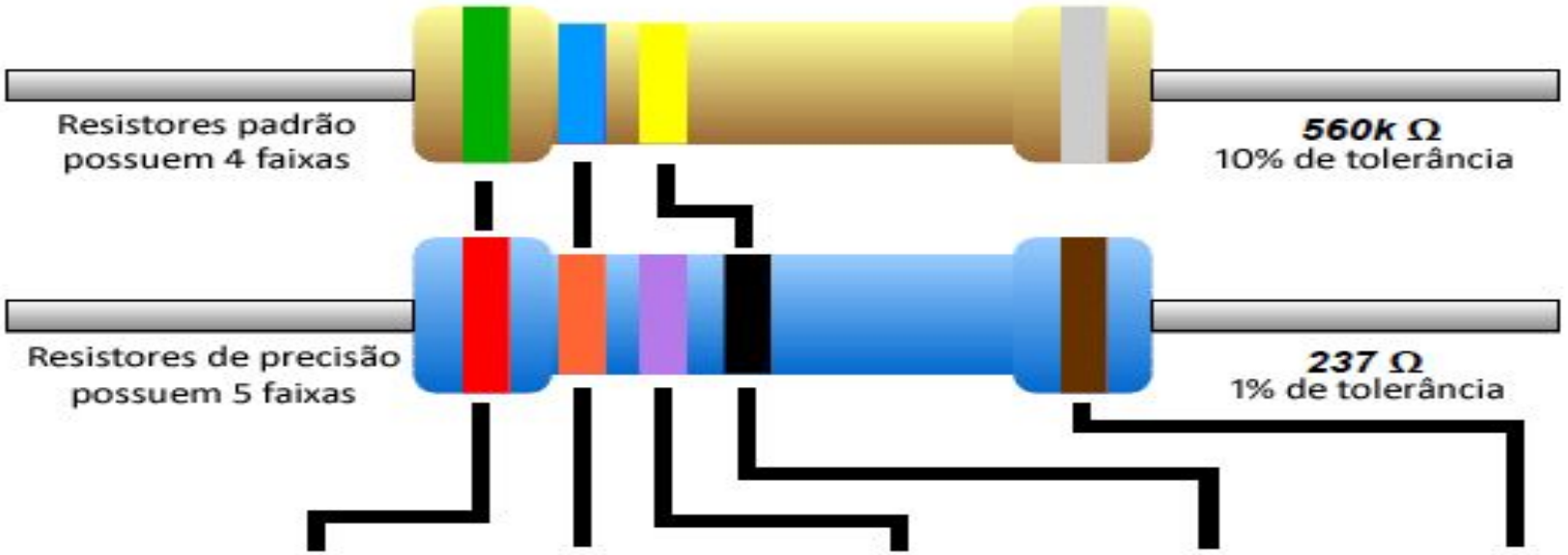
Funções:
O resistor tem por finalidade transformar a energia elétrica em energia térmica, assim como, diminuir a quantidade de corrente elétrica. Esta resistência elétrica é tida como a oposição à passagem de uma corrente elétrica **(medida em ohms)**

A identificação do valor da resistência de um resistor é de acordo as cores apresentadas em sua cápsula




Código de Cores Arduíno

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 x10 ^{0}	
Marrom	1	1	1	x 10 Ω x10 ^{1}	+/- 1%
Vermelho	2	2	2	x 100 Ω x10 ^{2}	+/- 2%
Laranja	3	3	3	x 1K Ω x10 ^{3}	
Amarelo	4	4	4	x 10K Ω x10 ^{4}	
Verde	5	5	5	x 100K Ω x10 ^{5}	+/- .5%
Azul	6	6	6	x 1M Ω x10 ^{6}	+/- .25%
Violeta	7	7	7	x 10M Ω x10 ^{7}	+/- .1%
Cinza	8	8	8	x10 ^{8}	+/- .05%
Branco	9	9	9	x10 ^{9}	
Dourado				x .1 Ω x10 $^{-1}$	+/- 5%
Prateado				x .01 Ω x10 $^{-2}$	+/- 10%

Arduíno – Resistores

Componentes necessários

De acordo com a tabela, para um resistor de 100 Ω você necessita de 1 na primeira faixa, que e marrom, seguido por um 0 na faixa seguinte, que e preta. Então, deve multiplicar isso por 101 (em outras palavras, adicionar um zero), o que resulta em marrom na terceira faixa. A faixa final indica a tolerância do resistor. Caso seu resistor tenha uma faixa dourada, ele tem uma tolerância de —5%; isso significa que o valor, de fato, do resistor varia entre 95 Ω e 105 Ω. Dessa forma, se você tem um LED que requer 2 V e 35 mA, necessitara de um resistor com uma combinação de faixas Marrom, Preto, Marrom.

Caso queira um resistor de 10 kΩ (ou 10 quilo-ohms), necessitara de uma combinação Marrom, Preto, Laranja (1, 0, +3 zeros).
Se necessitar de um resistor de 570 kΩ, quais serão as cores?
_____.

Arduíno – Resistores - Identificação

Favor identificar qual é o valor da resistência dos resistores abaixo identificados de acordo com as cores apresentadas em suas cápsulas?







Arduíno – Módulos - Shields

Internet

Xbee(wireless)

Motor control

Bluetooth

Acelelometro

Internet

Expansor de saídas

Navigations (robos)

Detector de gás

Game / Joystic

Memória SD

Wave (música)

GPS

ETC

Sensor de Luminosidade

Teclado

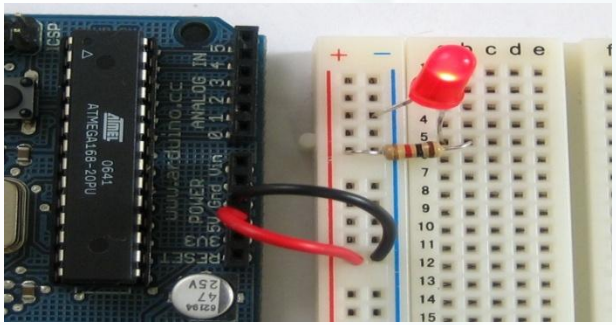
GPS Shield

Piscar – Led – Experimento 1

Para realizar mos o experimento a seguir qual será o valor que será utilizado pelo resistor?
O Arduino utiliza-se de uma saída equivalente a mais ou menos 5 volts. Do mesmo modo que o led vermelho é possuidor de uma voltagem de 1.6 volts e, uma corrente aproximadamente 20 mA.
Em sendo assim, ao aplicarmos a lei da física conhecida como lei de Ohm obteremos:

- **U = R . i**
- **R = U / i**
- **R = (5,0 – 1,6) / 0,020**
- **R = 3,4 / 0,020**
- **R = 170 omhs**

Pois bem, o resistor a ser utilizado terá um valor de 170 a 1k ohms



Para realizarmos o experimento **1** serão utilizados os seguintes materiais: arduino, protoboard, fios de conexão, Led's e um resistor de 220 ohms.

- **Programa (exp. 1)**

```
void setup() {
      pinMode(13, OUTPUT);
      }

void loop() {
      digitalWrite(13, HIGH); //HIGH = 1 : LIGA LED
      delay(500);
      digitalWrite(13, LOW); //LOW = 0 : DESLIGA LED
      delay(500);
          }
```

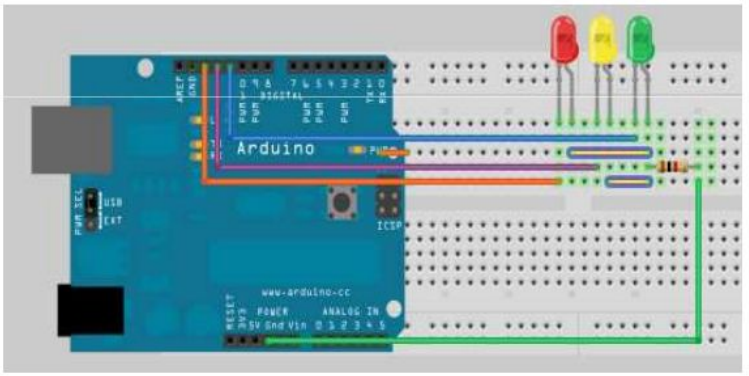
Obs: Finalmente para testarmos a eficiência de nossa experiência iremos substituir a função "delay(500)" por "delay(1000)". Observem o que acontece?
Da mesma forma, também iremos alterar o programa acima para piscar conforme o ritmo solicitado:lento, lento, rápido, rápido

Piscar – 3 Led's – Experimento 2

Para realizar mos o experimento a seguir qual será o valor que será utilizado pelo resistor?
O Arduino utiliza-se de uma saída equivalente a mais ou menos 5 volts. Do mesmo modo que o led vermelho é possuidor de uma voltagem de 1.6 volts e, uma corrente aproximadamente 20 mA.
Em sendo assim, ao aplicarmos a lei da física conhecida como lei de Ohm obteremos:

- **U = R . i**
- **R = U / i**
- **R = (5,0 – 1,6) / 0,020**
- **R = 3,4 / 0,020**
- **R = 170 omhs**

Pois bem, o resistor a ser utilizado terá um valor de 170 a 1k ohms



Para realizarmos o experimento **2** serão utilizados os seguintes materiais: arduino, protoboard, fios de conexão, 3 Led's e um resistor de 220 ohm e , terá a função de fazer os 3 led's piscarem seqüencialmente.

Programa (exp. 2)

```
void setup() {
        pinMode(13, OUTPUT);
        pinMode(12, OUTPUT);
        pinMode(11, OUTPUT);
        }

void loop() {
        digitalWrite(13, HIGH); //HIGH = 1 : LIGA LED
        delay(500);
        digitalWrite(13, LOW); //LOW = 0 : DESLIGA LED
        delay(500);
            }
```

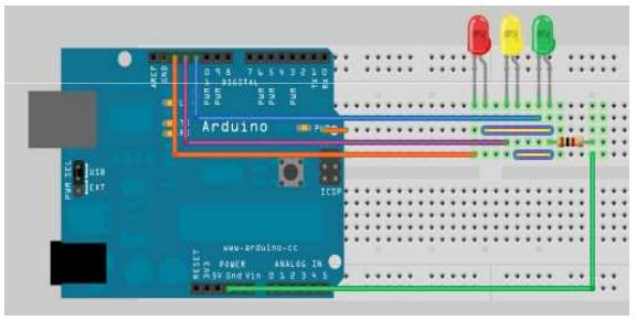
Obs: Finalmente para testarmos a eficiência de nossa experiência iremos substituir a função "delay(500)" por "delay(1000)". Observem o que acontece?
Da mesma forma, também iremos alterar o programa acima para piscar conforme o ritmo solicitado:lento, lento, rápido, rápido

Piscar – 3 Led's – Experimento 2

Para realizarmos o experimento **2** serão utilizados os seguintes materiais: arduino, protoboard, fios de conexão, 3 led's e um resistor de 220 ohm que, por sua vez, terá a função de fazer os 3 led's piscarem seqüencialmente.

Programa (exp. 2)

```
void setup() {
        pinMode(13, OUTPUT);
        pinMode(12, OUTPUT);
        pinMode(11, OUTPUT);
        }

void loop() {
        digitalWrite(11, HIGH);
        delay(500);
        digitalWrite(11, LOW);
        delay(500);
        digitalWrite(12, HIGH);
        delay(500);
        digitalWrite(12, LOW);
        delay(500);
        digitalWrite(13, HIGH);
        delay(500);
        digitalWrite(13, LOW);
        delay(500);

    }
```



Para testarmos mais uma vez a eficiência de nossa experiência iremos substituir na função "void setup()" os seguintes parametros.

Da mesma forma, também iremos alterar o programa ao lado para piscar conforme o ritmo solicitado.

Programa (exp. 2)

```
void setup() {
        pinMode(13, OUTPUT);
        pinMode(12, OUTPUT);
        pinMode(11, OUTPUT);
            }

void loop() {
        digitalWrite(11, HIGH);
        delay(500);
        digitalWrite(11, LOW);
        digitalWrite(12, HIGH);
        delay(500);
        digitalWrite(12, LOW);
        talWrite(13, HIGH);
        ay(500);
        talWrite(13, LOW);
        }
```

Classe Serial

Este tipo de classe denominada serial permite ao arduino comunicar-se com o computador por intermédio de uma porta serial (via USB)

Instruções

São observadas dois principais tipos de instruções :

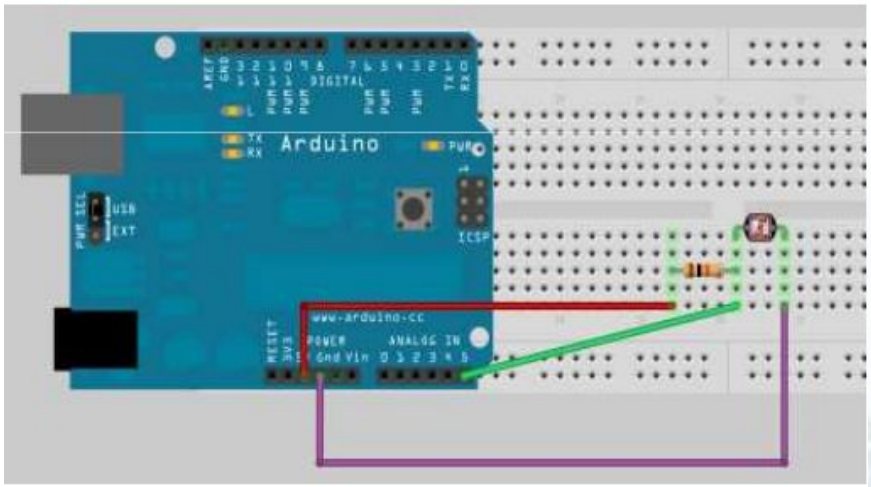
Serial.begin (9600) □ **inicia a comunicação inicia a comunicação do Arduino com o computador utilizando a velocidade 9600 bits por segundo**

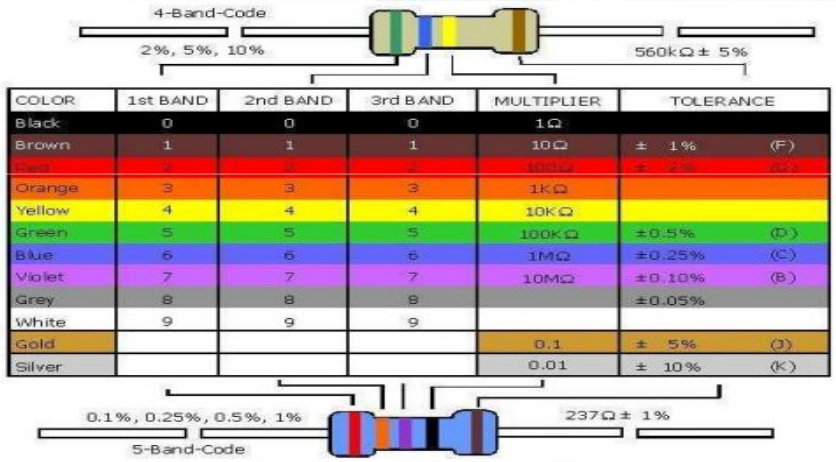
Serial.println (x,DEC) □ **envia o valor de x como decimal para o computador**

LDR (Resistor Dependente da Luz

Um resistor dependente da luz, é um transdutor de entrada (sensor) que converte a luz em valores de resistência. Sua composição química contém de sulfeto de cádmio (CdS) ou seleneto de cádmio (CdSe).
Existe um aspecto de suma importância que diz respeito à sua resistência, pois a mesma diminui na medida que a luminosidade fica intensa, assim como, quando da existência de pouca luminosidade, aumenta consideravelmente sua resistência,

Luz Ambiente – Experimento 3





COLOR	1st BAND	2nd BAND	3rd BAND	MULTIPLIER	TOLERANCE	
Black	0	0	0	1Ω		
Brown	1	1	1	10Ω	± 1%	(F)
Red	2	2	2	100Ω	± 2%	(G)
Orange	3	3	3	1KΩ		
Yellow	4	4	4	10KΩ		
Green	5	5	5	100KΩ	±0.5%	(D)
Blue	6	6	6	1MΩ	±0.25%	(C)
Violet	7	7	7	10MΩ	±0.10%	(B)
Grey	8	8	8		±0.05%	
White	9	9	9			
Gold				0.1	± 5%	(J)
Silver				0.01	± 10%	(K)

Para realizarmos o experimento **3** serão utilizados os seguintes materiais: arduino, protoboard, fios de conexão, LDR e um resistor de 10 K.

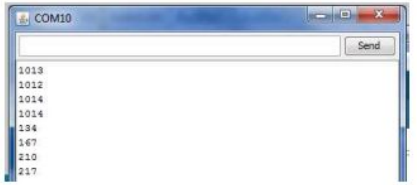
Programa (exp. 3)

```
void setup()
{
    Serial.begin(9600); // inicia comunicação com computador
}

void loop()
{
   int luz = analogRead(5); // faz leitura da luminosidade
   Serial.println(luz); // envia valor da luminosidade ao computador
   delay(500);
}
```

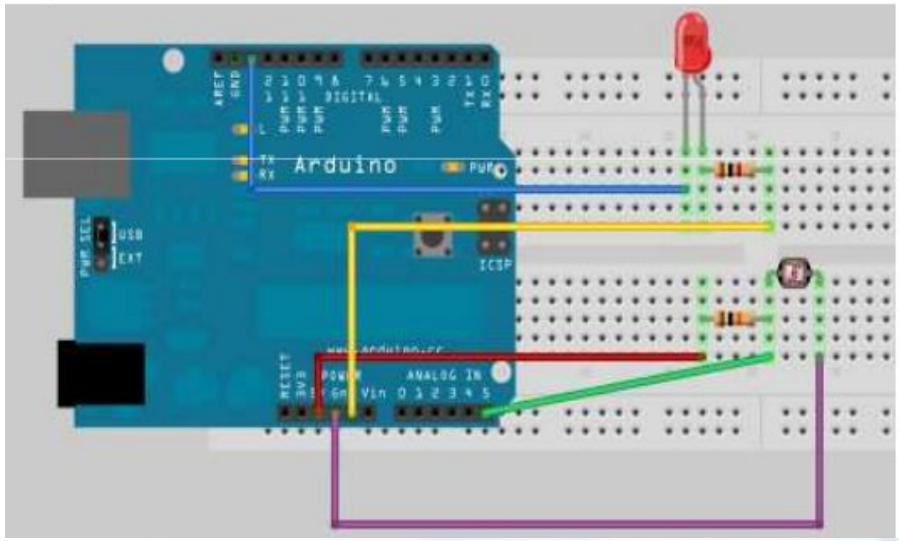
Obs: A função deste experimento será a de medir a luminosidade do ambiente e enviar para o computador através da porta serial. Para executar o programa será necessário acionar o botão ***"Serial Monitor"*** do IDE.

Serial Monitor

Resultado da operação



Controle de LED com Luz Ambiente – Experimento 4



Para concluirmos nosso aprendizado em relação a LED's e luz ambiente, favor elaborar um circuito com 1 LDR e 2 LED's (1 vermelho e 1 verde). Assim que estiver escuro, acender o LED vermelho e quando estiver claro, acender o LED verde.

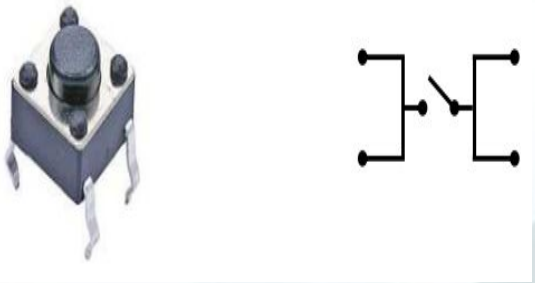
Para realizarmos o experimento **3** serão utilizados os seguintes materiais: arduino, protoboard, fios de conexão, LDR , resistor de 10 K, LED um resistor de 220 ohm.

Programa (exp. 3)

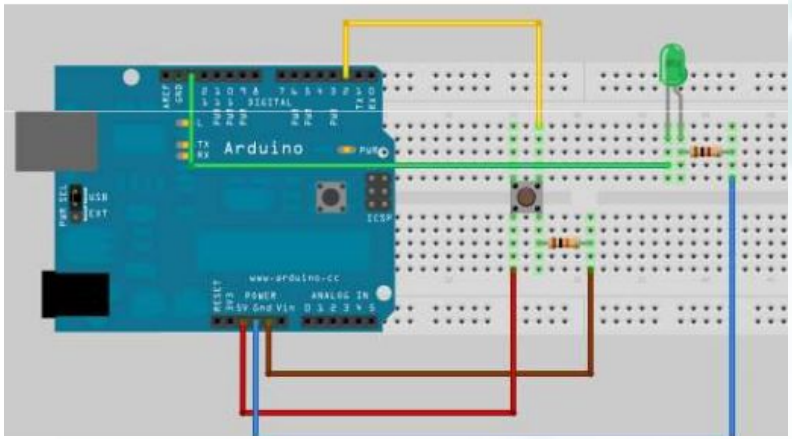
```
void setup()
{
    Serial.begin(9600); // inicia comunicação com computador
    pinMode(13,OUTPUT);
}

void loop()
{
   int luz = analogRead(5); // faz leitura da luminosidade
   Serial.println(luz); // envia valor da luminosidade ao computador
   if (luz>500)
     digitalWrite(13,HIGH);
     else
     digitalWrite(13,LOW);
     delay(500);
 }
```

Obs: A função deste experimento será a de acender o LED quando o ambiente estiver escuro e, apagar o LED quando o ambiente estiver claro.

Chave com Táctil (Push Button) - Experimento 5



Para darmos prosseguimento ao nosso aprendizado iremos obter mais detalhes de informações sobre o Push Button também conhecido por Chave de Toque. Sua principal ação incide sobre o fechamento de um contato quando pressionada.



Para realizarmos o experimento **5** serão utilizados os seguintes materiais: arduino, protoboard, botão táctil, resistor de 10 K, LED um resistor de 220 ohm.

Programa (exp. 3)

```
void setup()
{
    Serial.begin(9600); // inicia comunicação com computador
    pinMode(2, INPUT);
    pinMode(13, OUTPUT);
}

void loop()
{
   int sensorValue = digitalRead(2);
   Serial.println(sensorValue);
   if (sensorValue==1)
     digitalWrite(13,HIGH);
     else
     digitalWrite(13,LOW);
     delay(500);
}
```

Obs: A função deste experimento será a de acender o LED quando se o botão for ativado.

Arduino – BUZZER – Experimento 6

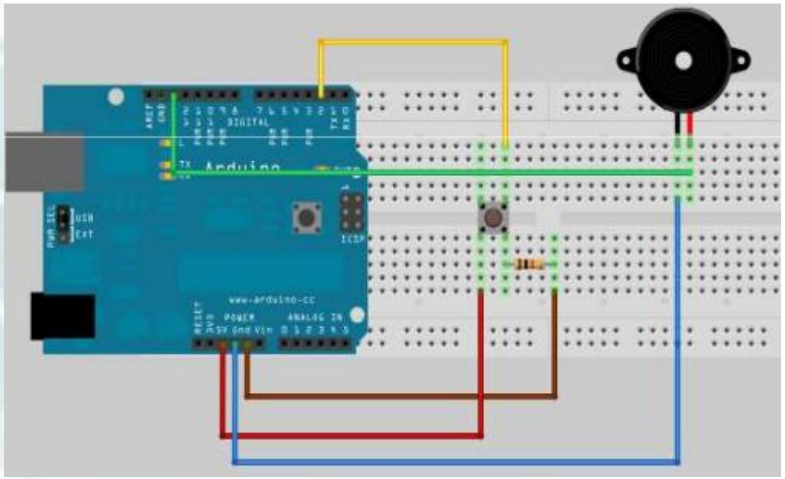
BUZZER (Buzina)

O buzzer é um equipamento, de dimensões pequena , cuja composição é formada de 2 camadas de metal e uma
camada interna de cristal piezoelétrico (como um lanche). Assim que é alimentado com uma fonte de sinal, vibra de acordo com a mesma frequência recebida, funcionando como uma sirene ou alto-falante.
É interessante notar que existem várias versões e tamanhos.
Algo importante a saber é, que todos os dispositivos sonoros de alarmes (como os de automóvel) utilizam um buzzer como fonte de sonoridade.
Interessante notar que a sua vantagem em relação aos altos-falantes comuns é o baixíssimo consumo de energia em relação à potência sonora, podendo ser facilmente alimentado com pequenas baterias.

Para realizarmos o experimento **6** serão utilizados os seguintes materiais: arduino, protoboard, fios de conexão, Botão Táctil , resistor de 10 K e uma Buzina.

Programa (exp. 6)

```
void setup()
{
     Serial.begin(9600); // inicia comunicação com computador
     pinMode(2, INPUT);
     pinMode(13, OUTPUT);
}

void loop()
{
   int sensorValue = digitalRead(2);
   Serial.println(sensorValue);
   if (sensorValue==1)
      digitalWrite(13,HIGH);
      else
      digitalWrite(13,LOW);
      delay(100);
 }
```



Para concluirmos nosso aprendizado em relação a sons, favor elaborar um circuito com 1 LDR e 1 BUZZER, para quando estiver escuro, tocar a Buzina.

Arduino – Música & BUZZER – Experimento 7

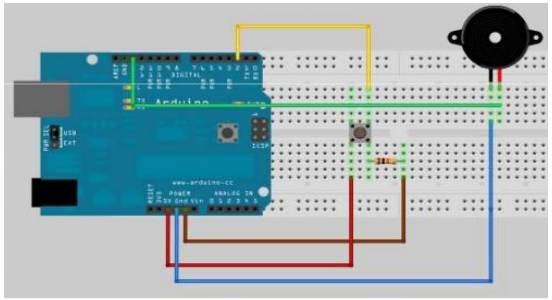
Para realizarmos o experimento 7 serão utilizados os seguintes materiais: arduino, protoboard, fios de conexão e uma BUZZER (Buzina).

Programa (exp. 7)

```
int speakerPin = 13;
int length = 15; // número da nota
char notes[] = "ccggaagffeeddc "; // um espaço representa um resto
int beats[] = { 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 4 };
int tempo = 300;
void playTone(int tone, int duration) {
for (long i = 0; i < duration * 1000L; i += tone * 2)
{
digitalWrite(speakerPin, HIGH);
delayMicroseconds(tone);
digitalWrite(speakerPin, LOW);
delayMicroseconds(tone);
}
 }
void playNote(char note, int duration) {
char names[] = { 'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C' };
int tones[] = { 1915, 1700, 1519, 1432, 1275, 1136, 1014, 956
}; // tocar o tom corresponde ao nome da nota
```

Programa (exp. 7) – continuação

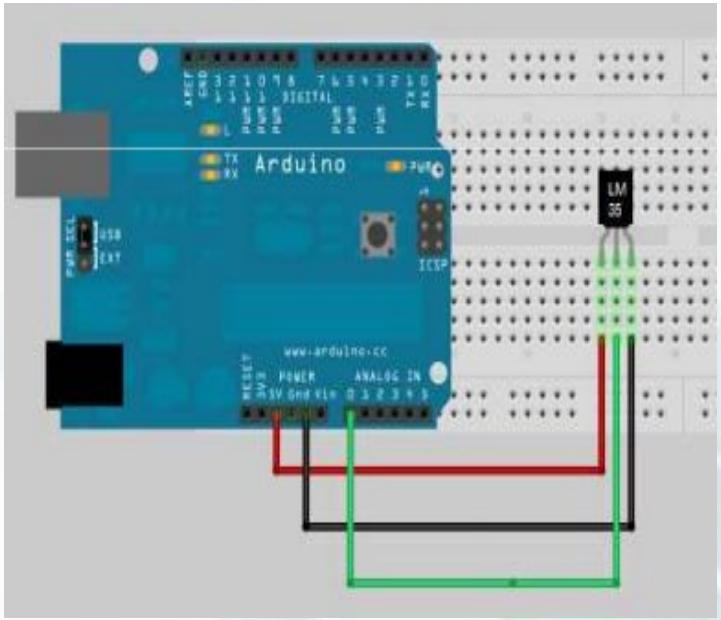
```
for (int i = 0; i < 8; i++) {
if (names[i] == note) {
playTone(tones[i], duration);
} } }
void setup() {
pinMode(speakerPin, OUTPUT);  }

void setup() {
pinMode(speakerPin, OUTPUT);
}

void loop() {
for (int i = 0; i < length; i++) {
if (notes[i] == ' ') {
delay(beats[i] * tempo); // rest
} else {
playNote(notes[i], beats[i] * tempo);
}
// pause between notes
delay(tempo / 2);
}
}
```

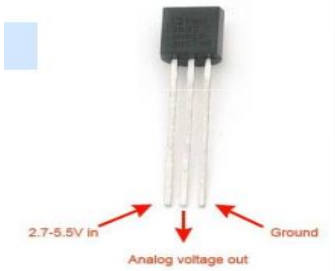


LM35 - Sensor deTemperatura Experimento 7

LM35

O LM35 Sensor de Temperatura, é um sensor de temperatura linear e trabalha com temperaturas entre -55 e 150 graus centigrados.
Tem por função efetuar a leitura da temperatura e enviar para
a porta serial.





Para realizarmos o experimento **8** serão utilizados os seguintes materiais: arduino, protoboard, fios de conexão e uma sensor de temperatura LM35.

Programa (exp. 8)

```
void setup()
{
    Serial.begin(9600); // inicia comunicação com computador
}

void loop()
{
  int t = analogRead(0);
  Serial.println(t);
  delay(1000);
 };
```

Programa alterado (exp. 8)

```
void setup()
{
Serial.begin(9600);
}

void loop()
{
int t = analogRead(0);
float tc = ( 5 * t * 100.0 / 1024);
Serial.print("Temperatura: ");
Serial.print(tc);
Serial.println(" graus celsius");
delay(1000);
 };
```

Enviando Dados - do Computador para Arduino

LM35

Anteriormente em alguns exercícios, utilizamos a classe Serial para enviar dados do Arduino para o computador através da porta serial.

É sabido que, a classe Serial também pode ser utilizada para receber dados enviados pelo
computador ao Arduino.

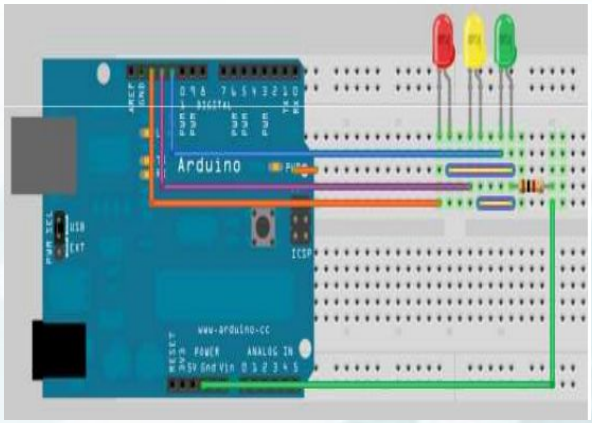
Para isso, serão utilizadas duas instruções:

Serial.available() que indica a quantidade de dados disponíveispara leitura e, Serial.read() que lê o próximo valor do buffer enviado pela serial, se retornar -1 não possui nenhum valor para ser lido no buffer.

Tem a função de Ligar/desligar Leds pelo Computador

Led1 (‘1’=Liga, ‘A’=Desliga)
Led2 (‘2’=Liga, ‘B’=Desliga)
Led3 (‘3’=Liga, ‘C’=Desliga)

Digite os valor abaixo e clique em Send: – 1 – B– 2 – B– 1A1A – 1A1B1C – 123ABC123ABC – 123CBA123CBA



Programa (exp.9)

Digite os valor abaixo e clique em Send:
– 1
– B
– 2
– B
– 1A1A
– 1A1B1C
–123ABC123ABC
– 123CBA123CBA

Para realizarmos o experimento **9** serão utilizados os seguintes materiais: arduino, protoboard, fios de conexão, 3 LED’s, resistor 220 ohms.

Programa (exp.9)

```
void setup() {
Serial.begin(9600);
pinMode(13,OUTPUT);
pinMode(12,OUTPUT);
pinMode(11,OUTPUT);
}
void loop() {
int x = Serial.available();
Serial.println(x);
if(x > 0) {
int dado = Serial.read();
if (dado=='1') digitalWrite(13,HIGH);
if (dado=='2') digitalWrite(12,HIGH);
if (dado=='3') digitalWrite(11,HIGH);
if (dado=='A') digitalWrite(13,LOW);
```

• Transfira o programa e clique no botão Serial Monitor.
Continua...

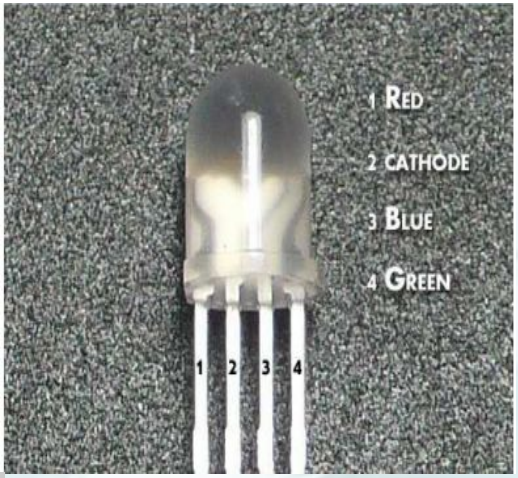
```
if (dado=='A') digitalWrite(13,LOW);
if (dado=='B') digitalWrite(12,LOW);
if (dado=='C') digitalWrite(11,LOW);}
delay(500);
}
```

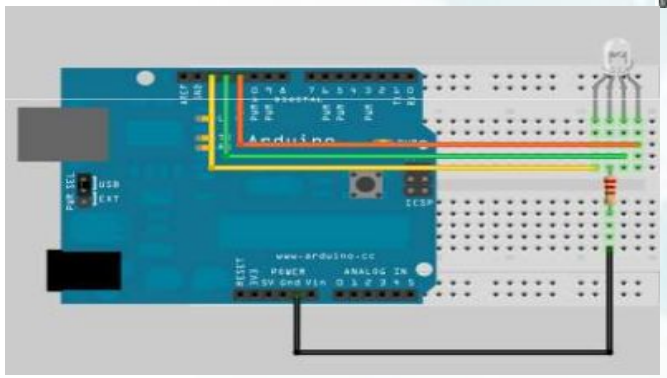
LED RGB

São denominados Led's RGB, aqueles led's que podem emitir 3 cores: **vermelho, verde e azul (RGB).**

Deste modo, as cores podem ser emitidas isoladamente ou em conjunto utilizando-se da porta PWM, onde pode-se controlar a intensidade de cada cor.

Este tipo de led, tem por função alternar as cores emitidas, ou seja, elas podem ser combinadas.





Para realizarmos o experimento **10** serão utilizados os seguintes materiais: arduino, protoboard, fios de conexão, LED's, RGB resistor 220 ohms.

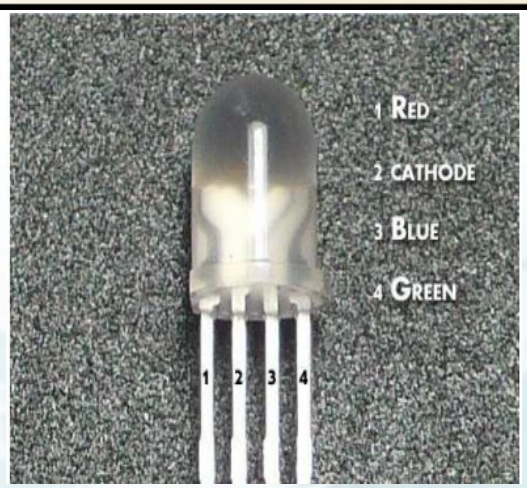
Programa (exp.10)

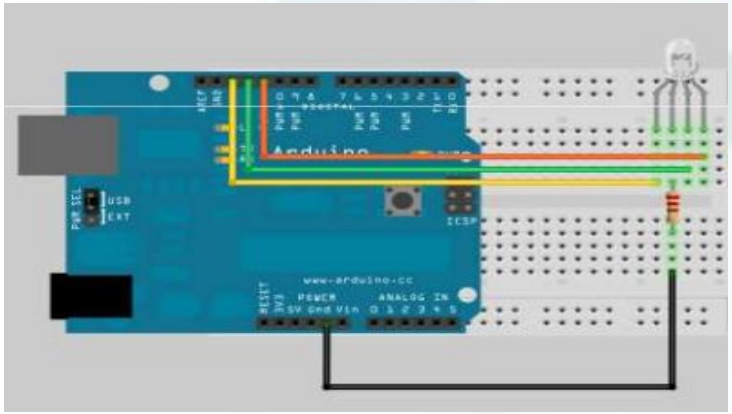
```
void setup()
{
pinMode(13,OUTPUT);
pinMode(12,OUTPUT);
pinMode(11,OUTPUT);
}
void loop()
 {
digitalWrite(13,HIGH); // vermelho
delay(500);
digitalWrite(13,LOW);
digitalWrite(12,HIGH); // verde
delay(500);
digitalWrite(12,LOW);
digitalWrite(11,HIGH); // azul
delay(500);
digitalWrite(11,LOW);
delay(1000);
}
```

LED RGB

Favor alterar o programa anterior para gerar outras cores RGB:

- Amarelo: Vermelho + Verde
- Magenta: Vermelho + Azul
- Ciano: Verde + Azul





Para realizarmos o experimento **10a** serão utilizados os seguintes materiais: arduino, protoboard, fios de conexão, LED's, RGB resistor 220 ohms.

Programa (exp.10a)

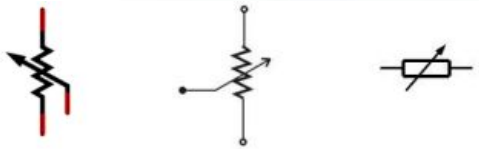
```
void setup()
{

}
void loop()
 {

 }
```

POTENCIOMETRO

São denominados potenciômetros , aqueles dispositivos eletromecânicos possuidores de resistores de valores fixos onde são movidos por intermédio de contatos deslizantes, provocando variações. É de praxe que eles sejam possuidores de 3 terminais, um Vcc, outro GND e um de sinal (analógico).

Potenciômetro de 10k:

Representação



FUNÇÃO MAP

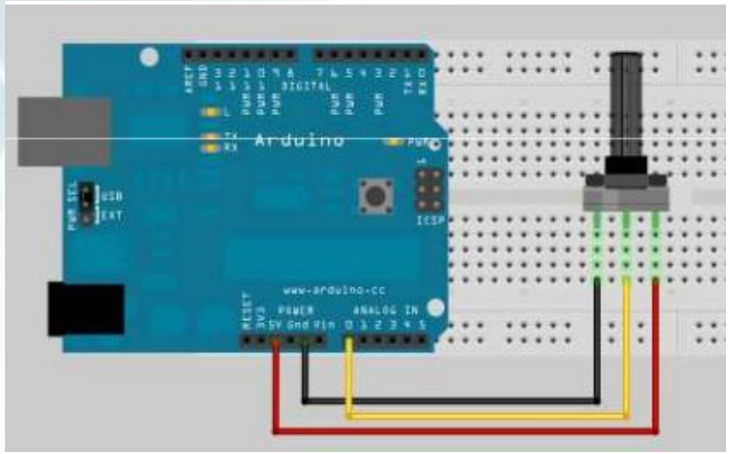
Esta função **"map"** tem o poder de converter uma determinada faixa de valores de entrada em uma também determinada faixa de valores de saída. Observe o seguinte caso: para um determinado potenciômetro gerando valores de entrada entre 0 e 1023 (10bits) é desejado que haja uma conversão para
uma saída analógica entre 0 e 255 (8 bits).

Sintaxe

map(entrada, in_min, in_max, out_min, out_max)

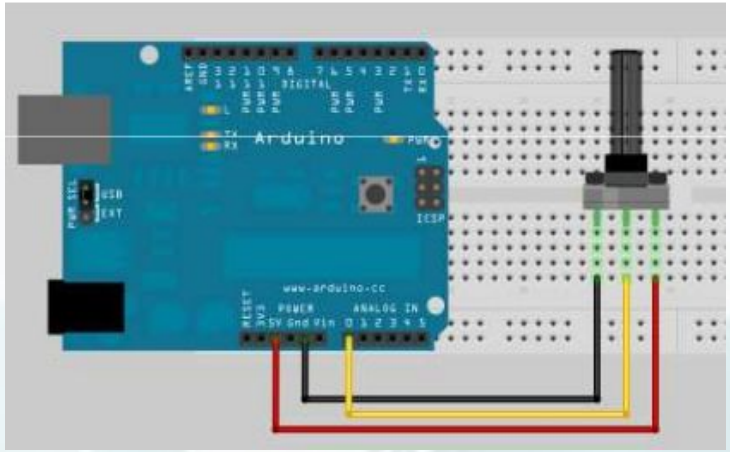
int val = map(x, 0, 0, 255)

Para realizarmos o experimento **11** serão utilizados os seguintes materiais: arduino, protoboard, fios de conexão, potenciômetro de 10k.
Sua principal função é , ler valor do potenciômetro e envia-lo para computador através da porta serial.



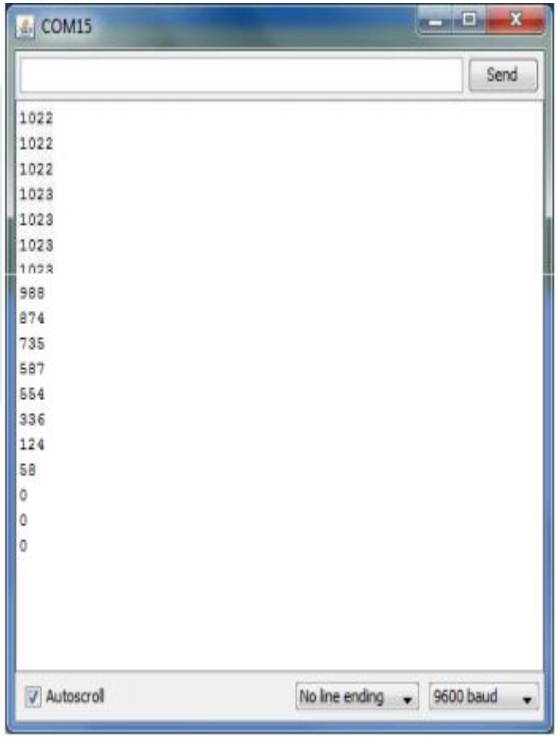
POTENCIOMETRO

Para realizarmos o experimento **11** serão utilizados os seguintes materiais: arduino, protoboard, fios de conexão, potenciômetro de 10k.
Sua principal função é , ler valor do potenciômetro e envia-lo para computador através da porta serial.
Favor executar o programa abaixo e, clique no botão Serial Monitor, gire o potenciômetro e observe os valores gerados



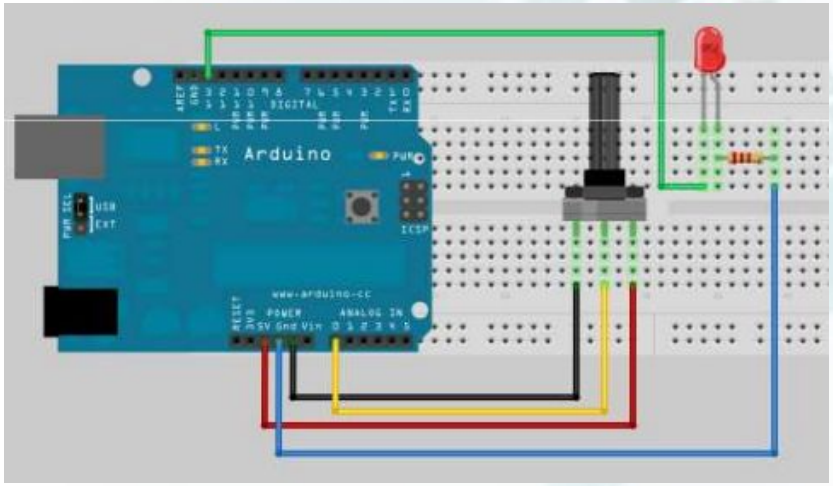
```
void setup()
{
 Serial.begin(9600);
}
void loop()
 {
  int val = analogRead(0);
  Serial.println(val);
  delay(500);
 }
```

Serial Monitor



Controle de LED com POTENCIOMETRO

Favor elabore um circuito para regular a intensidade do Led com o potenciômetro. Favor usar uma porta PWM como entrada do Led, assim como também, a função **"map"** para converter os valores de entrada do potenciômetro em valores de saída para o Led.



Para realizarmos o experimento **12** serão utilizados os seguintes materiais: arduino, protoboard, fios de conexão, potenciômetro de 10k, Led e resistor 220 ohm.

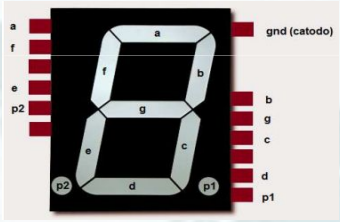
Sua principal função será, utilizar o potenciômetro para controle de velocidade do **"Pisca Led"**

```
void setup()
{
Serial.begin(9600);
pinMode(13,OUTPUT);
}
void loop()
{
int pot = analogRead(0);
Serial.println(pot);
digitalWrite(13,HIGH);
delay(pot);
digitalWrite(13,LOW);
delay(pot);
}
```

Display de 7 Segmentos

Display de M7 Segmentos

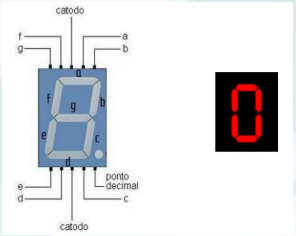
Este tipo de display de sete segmentos é composto de oito **LEDs** (Diodos emissores de luz) , sete para formar o número que deverá ser apresentado e, um para gerar o ponto decimal.
Em sendo assim, sua classificação pode ser um **anodo** (+), ou, um **catodo** (-).
Desta forma, os displays LED de 7 segmentos serão utilizados para inúmeras tarefas em automação.

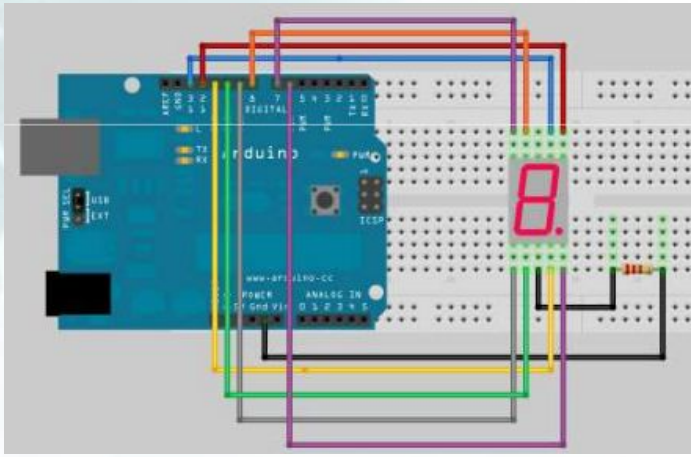


Conexões

- Pino 13 – Led A
- Pino 12 – Led B
- Pino 11 – Led C
- Pino 10 – Led D
- Pino 9 – Led E
- Pino 8 – Led F
- Pino 7 – Led G
- Pino 6 – Led do Ponto Decimal

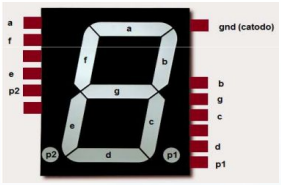
Para realizarmos o experimento **13** serão utilizados os seguintes materiais: arduino, protoboard, fios de conexão, display 7 segmentos e um resistor 220 ohm.
Sua principal função será, acender cada um dos 8 Led's do display

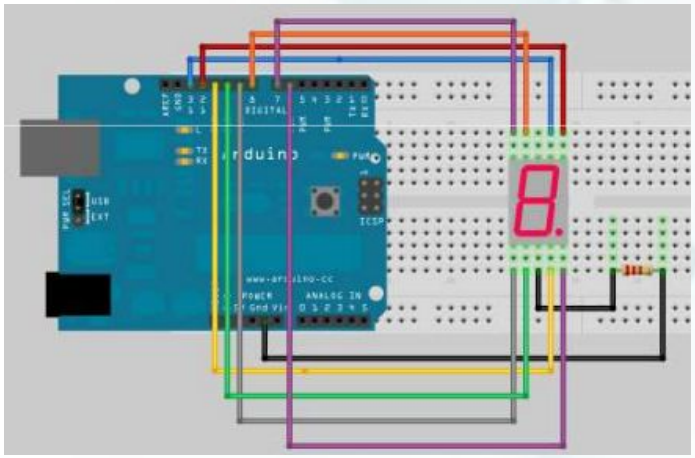




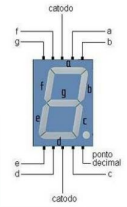
Display de 7 Segmentos

Para realizarmos o experimento **13** serão utilizados os seguintes materiais: arduino, protoboard, fios de conexão, display 7 segmentos e um resistor 220 ohm.
Sua principal função será, acender cada um dos 8 Led's do display





```
void setup()
{
Serial.begin(9600);
pinMode(13,OUTPUT);
pinMode(12,OUTPUT);
pinMode(11,OUTPUT);
pinMode(10,OUTPUT);
pinMode(9,OUTPUT);
pinMode(8,OUTPUT);
pinMode(7,OUTPUT);
pinMode(6,OUTPUT);
}
void loop()
{
digitalWrite(13,HIGH);
delay(300);
digitalWrite(13,LOW);
digitalWrite(12,HIGH);
delay(300);
digitalWrite(12,LOW);
digitalWrite(11,HIGH);
delay(300);
digitalWrite(11,LOW);
digitalWrite(10,HIGH);
delay(300);
```



cont. p13

```
digitalWrite(10,LOW);
digitalWrite(9,HIGH);
delay(300);
digitalWrite(9,LOW);
digitalWrite(8,HIGH);
delay(300);
digitalWrite(8,LOW);
digitalWrite(7,HIGH);
delay(300);
digitalWrite(7,LOW);
digitalWrite(6,HIGH);
delay(300);
digitalWrite(6,LOW);
}
```

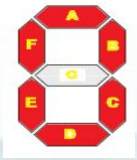
Contador

Para realizarmos o experimento **14** serão utilizados os seguintes materiais: arduino, protoboard, fios de conexão, display 7 segmentos e um resistor 220 ohm.
Sua principal função será, contar de 0 à 9.

Relação Led – número (de 0 à 9 – "E"=Erro)

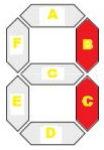
Led A – 0 – 2 – 3 – 5 – 6 – 7- 8 – 9 - E
Led B – 0 – 1 – 2 – 3 – 4 – 7 – 8 – 9
Led C – 0 – 1 – 3 – 4 – 5 – 6 – 7 – 8- 9
Led D - 0 – 2 – 3 – 5 – 6 – 8 – 9 - E
Led E – 0 – 2 – 6 – 8 - E
Led F – 0 – 4 – 5 – 6 – 8 – 9 - E
Led G – 2 – 3 – 4 – 5 – 6 – 8 – 9 - E
Led do Ponto Decimal

Número 0



Leds: A, B, C, D, E, F

Número 1

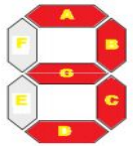


Leds: B, C

Número 2

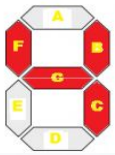
Leds: A, B, D, E, G

Número 3



Leds: A, B, C, D, G

Número 4



Leds: B, C, F, G

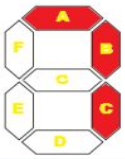
Número 5

Leds: A, C, D, F ,G

Número 6

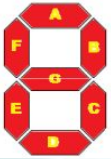
Leds: A, C, D, E, F ,G

Número 7



Leds: A, B, C

Número 8

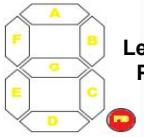


Leds: A, B, C, D, E, F, G

Número 9

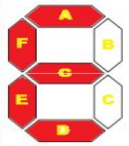
Leds: A, B, C, D, F, G

Número ponto Decimal



Leds: PD

Número ERRO



Leds: A, D, E, F, G

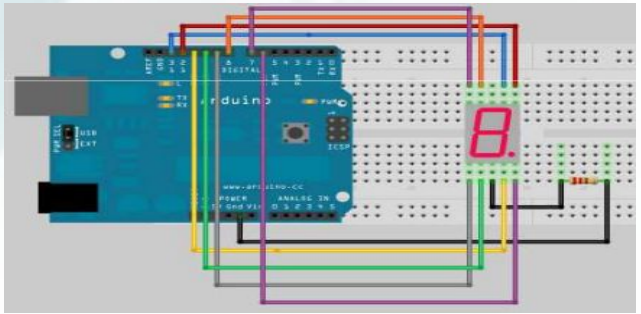
Contador

Para realizarmos o experimento **14** serão utilizados os seguintes materiais: arduino, protoboard, fios de conexão, display 7 segmentos e um resistor 220 ohm. Sua principal função será, contar de 0 à 9.

```
#define PINO_A 13
#define PINO_B 12
#define PINO_C 11
#define PINO_D 10
#define PINO_E 9
#define PINO_F 8
#define PINO_G 7
#define PINO_PD 6 // PONTO DECIMAL
void imprimeNumero(int n)
{
digitalWrite(13,LOW);
digitalWrite(12,LOW);
digitalWrite(11,LOW);
void setup()
{
Serial.begin(9600);
pinMode(13,OUTPUT);
}
void loop()
{
int pot = analogRead(0);
Serial.println(pot);
digitalWrite(13,HIGH);
delay(pot);
digitalWrite(13,LOW);
delay(pot);
}
```

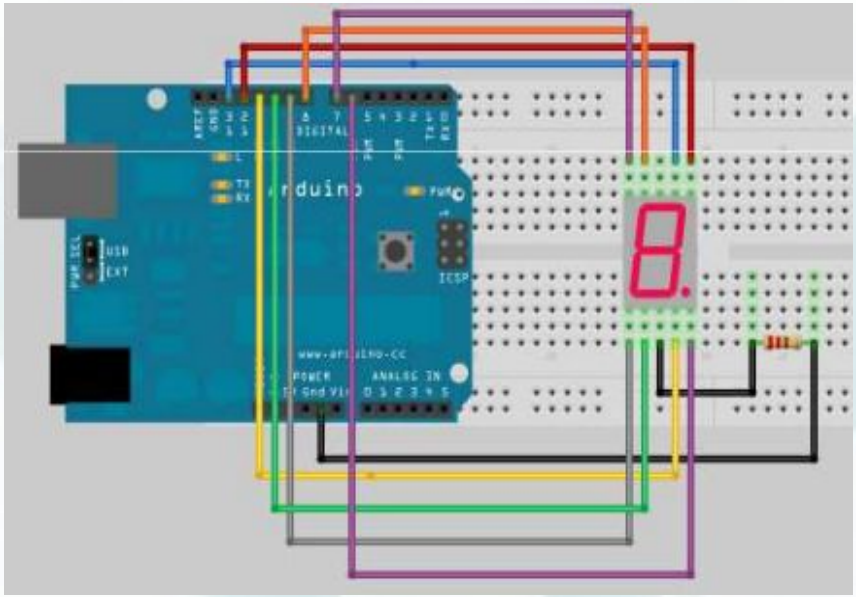
cont-1. p14

```
digitalWrite(10,LOW);
digitalWrite(9,LOW);
digitalWrite(8,LOW);
digitalWrite(7,LOW);
digitalWrite(6,LOW);
if (n==0 || n==2 || n==3 || n==5 || n==6 || n==7 || n==8 || n==9 || n<0 || n>9)
digitalWrite(PINO_A,HIGH);
if (n==0 || n==1 || n==2 || n==3 || n==4 || n==7 || n==8 || n==9)
digitalWrite(PINO_B,HIGH);
if (n==0 || n==1 || n==3 || n==4 || n==5 || n==6 || n==7 || n==8 || n==9)
digitalWrite(PINO_C,HIGH);
if (n==0 || n==2 || n==3 || n==5 || n==6 || n==8 || n==9 || n<0 || n>9)
digitalWrite(PINO_D,HIGH);
if (n==0 || n==2 || n==6 || n==8 || n<0 || n>9)
digitalWrite(PINO_E,HIGH);
if (n==0 || n==4 || n==5 || n==6 || n==8 || n==9 || n<0 || n>9)
digitalWrite(PINO_F,HIGH);
if (n==2 || n==3 || n==4 || n==5 || n==6 || n==8 || n==9 || n<0 || n>9)
digitalWrite(PINO_G,HIGH);
}
```



Contador

Para realizarmos o experimento **14** serão utilizados os seguintes materiais: arduino, protoboard, fios de conexão, display 7 segmentos e um resistor 220 ohm. Sua principal função será, contar de 0 à 9.

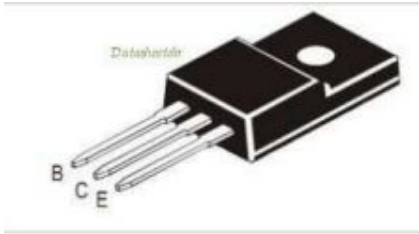


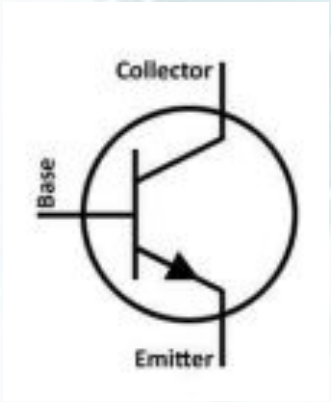
Cont-2. p14

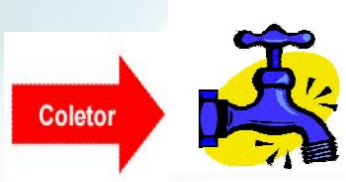
```
void setup()
{
Serial.begin(9600);
pinMode(13,OUTPUT);
pinMode(12,OUTPUT);
pinMode(11,OUTPUT);
pinMode(10,OUTPUT);
pinMode(9,OUTPUT);
pinMode(8,OUTPUT);
pinMode(7,OUTPUT);
pinMode(6,OUTPUT);
}

void loop()
{
for (int x=0; x<=10;x++)
{
imprimeNumero(x);
delay(1000);
}
}
```

Transistor

Os transistores funcionam como um controlador de fluxo de corrente.
São como uma torneiras, possuem 3 terminais denominados de **“base”, “emissor” e “coletor”.**
A “base” tem a função da manopla da torneira ou seja, controlar o fluxo de corrente que irá fluir entre o “coletor” e o emissor (o que entra na torneira pela caixa d´água e o que sai na boca da torneira).
O transistor também funciona com um amplificador.







Lampadas & Tomadas

Pelo motivo do arduino trabalhar com **5** volts corrente contínua, não é possível liga-lo a um fio 110 volts, simplesmente pelo motivo de que, a energia elétrica das residências cujas tomadas e lâmpadas trabalham de forma inversa, ou seja, com corrente alternada nos valores de **127** ou **220** volts.

Este é um dos principais motivos que torna-se inviável a transmissão de corrente contínua para longas distâncias. Em sendo assim, como fazer com que o arduino controle tomadas e lâmpadas?

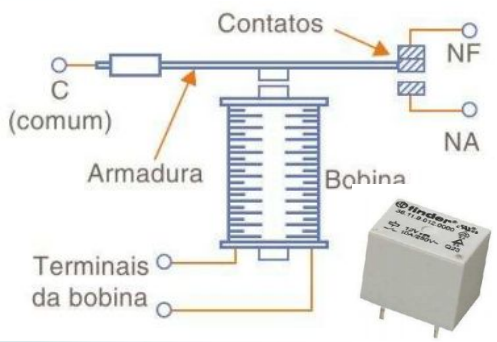
Existe um equipamento denominado relé, que nada mais é que um interruptor eletromagnético, que quando energizado faz com que o interruptor feche um determinado contato.

Por sua vez, esses contatos poderão estar normalmente abertos (desconectados) ou normalmente fechados (conectados), também conhecidos como as siglas **NO** e **NC** em inglês ou **NA** e **NF** em português.

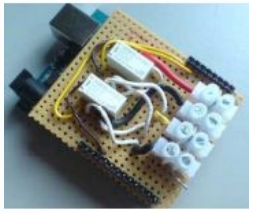
É de conhecimento que o arduino fornece no máximo **40mA** em suas portas.

No entanto , para acionar o Relé, é necessário de aproximadamente **70mA**. Em sendo assim, devemos ligar um transistor entre o Arduino e o Relé.

Rele



Rele & Arduino



Transistor

Lampadas & Tomadas

Tensão Reversa

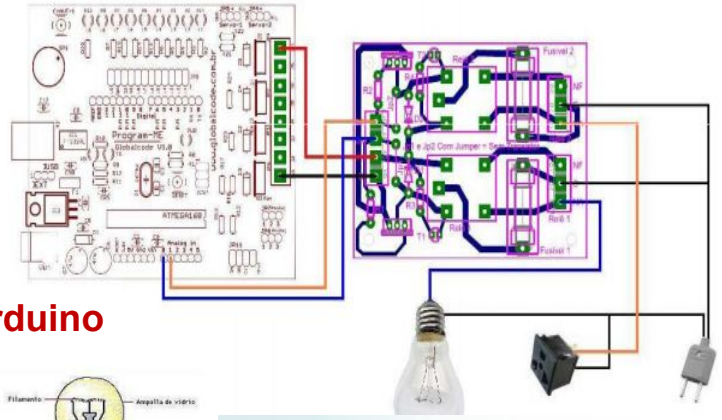
Um outro aspecto de suma importância, tem haver com a Tensão reversa, que é um fenômeno físico que acontece em todo mecanismo eletromagnético quando é desenergizado.

Assim que tiramos a energia de um dispositivo eletromagnético recebemos de volta um **"choque".**

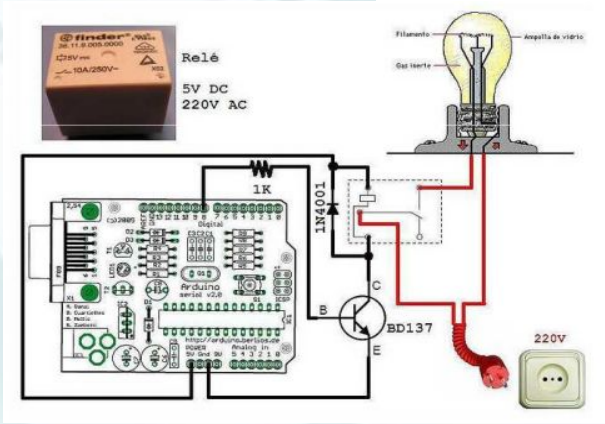
Isso poderia queimar o transistor ou o arduino, de tal modo, para evitar esse fenômeno, é utilizado um diodo, equipamento que permite a corrente transite em um único sentido.

O **diodo** pode ser ligado em paralelo com o mecanismo

Placa Tomada com Arduino

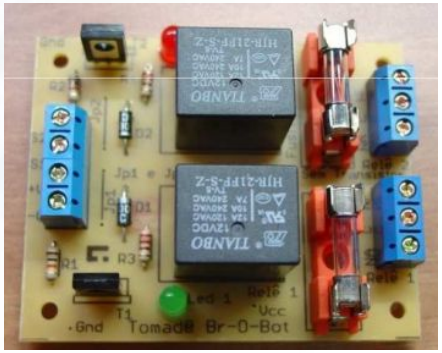


Placa Tomada com Arduino



Diodo

Placa Tomada



Controle de Lampadas & Tomadas

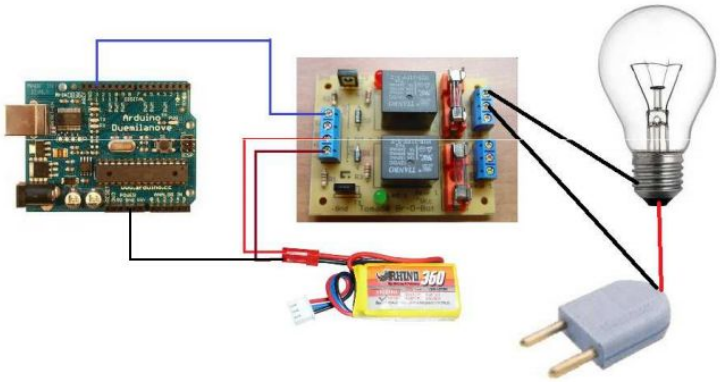
Lampadas & Tomadas

Para realizarmos o experimento **15** serão utilizados os seguintes materiais: arduino, protoboard, fios de conexão, placa Tomada (Relé, Transistor, Diodo), Bateria de 12v (ou fonte), Lâmpada de 127 volts e uma tomada de 127 volts.
Sua principal função será, piscar a lâmpada.

Placa Tomada


```
void setup()
{
pinMode(13,OUTPUT);
}
void loop()
{
digitalWrite(13,HIGH);
delay(1000);
digitalWrite(13,LOW);
delay(1000);
}
```

Placa Tomada com Arduino

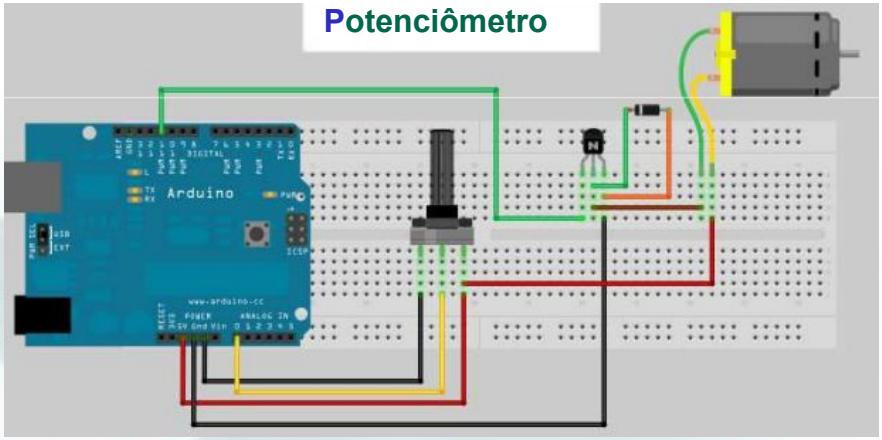


Controle de Motor & Potenciômetro

Controle de Motor & Potenciômetro

Para realizarmos o experimento **17** serão utilizados os seguintes materiais: arduino, protoboard, fios de conexão, potenciômetro10k, transistor Tip 122, diodo e motor DC
Sua principal função será, controlar a velocidade do motor com Potenciômetro

Motor & Potenciômetro




```
void setup()
{

}
void loop()
{

}
```

Diodo

Motores & Arduino

Motores & Arduino

Os motores tem como função movimentar peças (braços, superfícies, etc), assim como locomover (rodas, esteiras, pernas, etc). Suas principais características correspondem à velocidade de rotação / rpm, força, precisão.

Redutor

Os redutores têm por função, transformar a rotação em força. Os principais tipos são, o motor DC/CC (velocidade), o servo motor (precisão e/ou força) e o motor de passo (precisão e ou força).

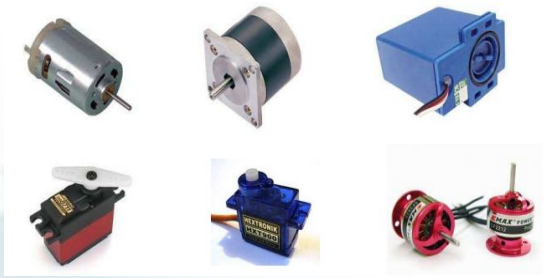
Motor DC

São motores poucos complexos detentores das seguintes características:
Possuidores alta velocidade (conforme redução), não possuem precisão angular, a potencia é controlado com PWM. Assim como o relé, precisamos ligar em um transistor com diodo de proteção e, para inverter-se a direção da rotação tem-se que inverter a polaridade.

Servo & Motor

Ao contrário dos motores **DC** os **Servo Motores** são detentores das seguintes características :
São motores com precisão angular, de fácil comando. São motores que possuem mais redução e mais dirigibilidade (driver).
É interessante notar estes motores, tem por padrão virar apenas de 0 a 180 graus.
Existe a possibilidade de podermos **hackear** ou comprar **servos full-rotation**.
Já com relação às conexões, estas se apresentam de forma super simples, tais como, GND, 5v e Sinal Digital

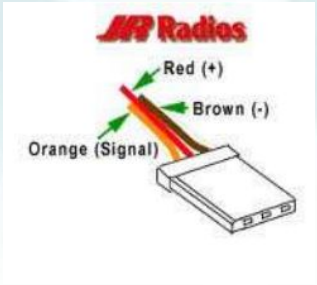
Motores com Arduino

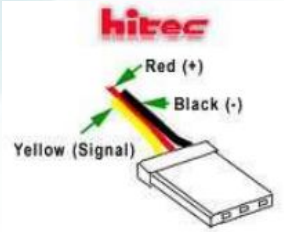


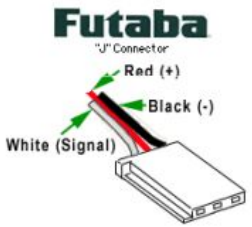
Motor DC

Servo Motor

Servo Motor - Conexões







Controlando o Servo com Potenciômetro

Classe Servo

Este tipo de classe, é utilizada para controlar **Servos Motores,** e estão presente na biblioteca: **Servo.h**.

Os métodos utilizados são os seguintes:

Servo.attach(porta) (porta) □ conecta servo à uma conecta servo à uma porta digital

Servo.write(val) □ define ângulo do braço do servo (0 – 180 graus)

Servo.detach() □ desconecta servo de uma porta digital

Para realizarmos o experimento **18** serão utilizados os seguintes materiais: arduino, protoboard, fios de conexão, potenciômetro 10k e o s servo motor.
Sua principal função será, Controlar o servo motor com potenciômetro.
Favor executar o programa, gire o potenciômetro e observe o servo

Servo & Potenciômetro

```
#include <Servo.h>
Servo meuServo;
void setup()
{
Serial.begin(9600);
meuServo.attach(4);
}
```

```
void loop()
{
int val = analogRead(0);
int x = map(val,0,1023,0,179);
Serial.print(val);
Serial.print(" - ");
Serial.println(x);
meuServo.write(x);
delay(10);
}
```

Sensor de Distância

Os Sensores de Distância, medem a distância do sensor até um objeto ou obstáculo. São utilizados em robôs para detectar obstáculos. Existem dois tipos, os: Ultrassônicos e os Infravermelhos.

O Sensor **Ultrassônico HC-SR04**, emite um sinal ultrassônico para detectar obstáculos, onde, a faixa de medição estende-se entre 2 centímetros à 4 metros. Também é possuidor de um emissor e um receptor ultrassônico com 4 conexões: **5V Supply (VCC)**, **Tigger Pulse Input (Trig)**, **Echo Pulse Output (Echo)** e o **0V Ground (GND).**

Classe Ultrasonic

Classe que utilizada dois métodos com o Sensores Ultrassônicos padrão **SR04** :
Método construtor: Ultrasonic Ultrasonic(trigger_pin trigger_pin, echo_pin echo_pin)

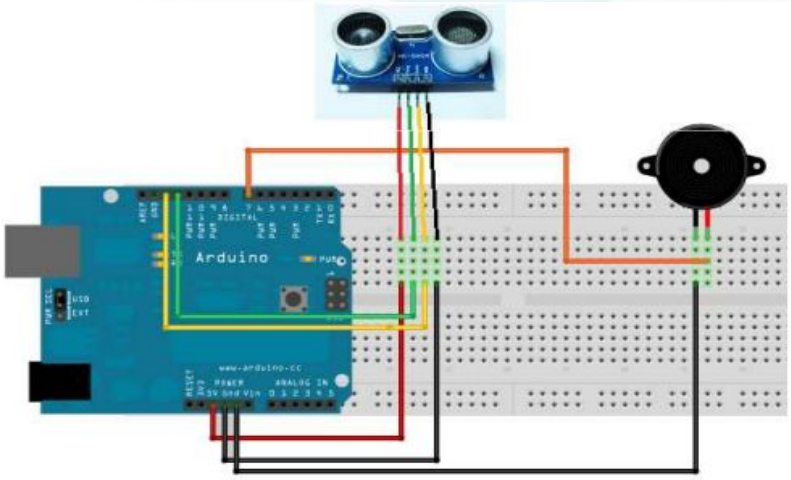
Método de medição: Ultrasonic.Ranging(sist_med)

Serão apresentados dois tipos de sistemas de medição, o sistema que realiza medição em **centímetros (CM)** e o que realiza medição em **polegadas (INC)**.

Sensor de Distância

Para realizarmos o experimento **19** serão utilizados os seguintes materiais: arduino, protoboard, fios de conexão, Sensor HC-SR-04 e Buzzer.
Sua principal função será, acionar a buzina quando estiver próximo de um obstáculo.
Favor executar o programa, posicione a palma da mão em frente de sensor, fazendo variar a distância

Detector & Obstáculo




```
#include <Ultrasonic.h>
Ultrasonic ultrasonic(12,13);
void setup()
 {
  pinMode(7,OUTPUT);
 }

void loop()
{
int dist = ultrasonic.Ranging(CM);
Serial.print(dist);
Serial.println(" cm");

 if (dist < 30)
   {
    digitalWrite(7,HIGH);
    delay(100);
    digitalWrite(7,LOW);
    delay(dist*15);
   }
delay(100);
}
```

Para os humanos: sentidos, pensamentos, e ações
Para os robôs: os sentidos serão os sensores, os pensamentos serão os processamentos (programas) e as ações os atuadores.

Dessa forma, um robô aciona seus atuadores, baseado em seu processamento que teve como entrada os dados vindos de seus sensores.

Todos os robôs têm em comum a realização de algum tipo de movimento sendo que também podemos distinguir os robôs pela sua capacidade de processamento, sendo assim poderíamos classificar os robôs como:
assim poderíamos classificar os robôs como:

Robô "inteligente": pode se mover de forma autônoma e segura por um ambiente não preparado e atingir um objetivo ou efetivar uma tarefa.

Robô não "inteligente": deve repetir de forma confiável a mesma tarefa para que foi programado.

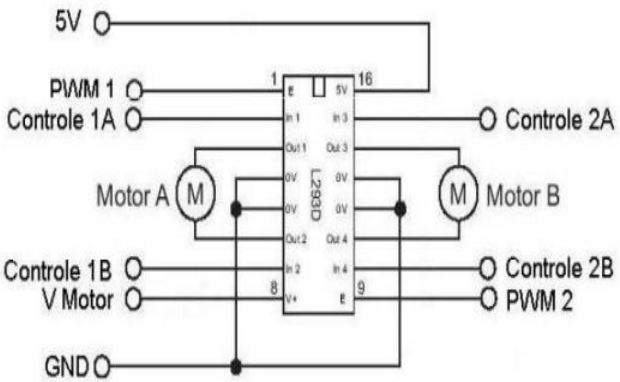
Tipos de robôs: Manipuladores ou braços robóticos, Robôs móveis com rodas; Robôs móveis com pernas e os Humanóides.

Controle de dois motores

A maneira que invertemos a polaridade de um motor **DC**, ele irá girar no sentido inverso, utilizando um **CI** auxiliar, poderemos então controlar 2 motores.
Por sua vez o **CI** auxiliar permite inverter a polaridade e, a solução é utilizar transistores em ponte H.
Em sendo assim, podemos usar um circuito de ponte H pronto **(Motor Control Shield)**

Robótica & Arduíno

Circuito L293D



Shield: Motor Control ou Motor Driver

Adaptação de um Servo motor para rotação contínua

Remoção do limitador

Troca do potenciômetro por resistores fixos 2.2k

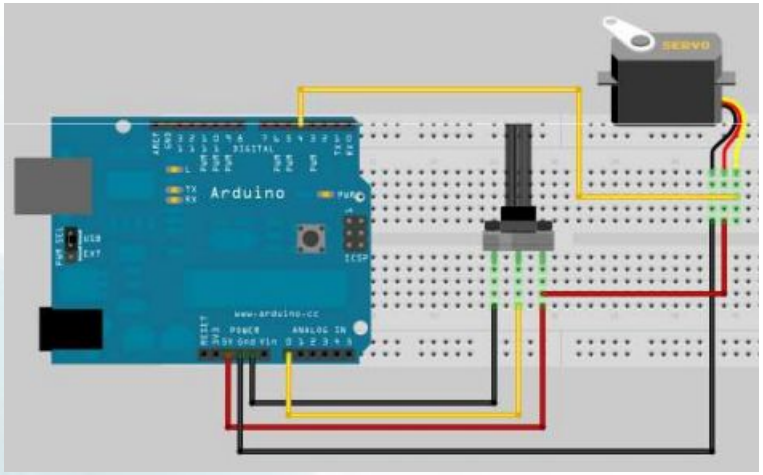
Para realizarmos o experimento **20** serão utilizados os seguintes materiais para Hackear o Servo: servo 9g, chave philips pequena, alicate de corte, ferro de solda, estanho e 2 resistores de 2.2k

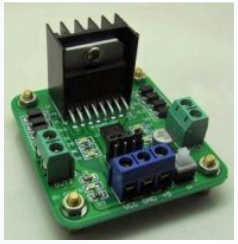
Para realizarmos o experimento **20** serão utilizados os seguintes materiais para Testar o Servo: arduino, protoboard, fios de conexão, sensor HC-SR-04, e buzzer
Sua principal função será, Hackear o servo para ficar com rotação contínua

Etapas a serem realizadas:
- 1 – Abrir o servo
- 2 – Desmontar o servo
- 3 – Cortar o limitador mecânico
- 4 – Cortar os fios do potenciômetro
- 5 – Soldar os resistores
- 6 – Montar o servo

Favor execute o programa meuServo.attach(4);

```
void setup()
{
#include <Servo.h>
Servo meuServo;
void setup()
{
meuServo.attach(4);
}
void loop()
{

meuServo.write(180);
delay(3000);
}
meuServo.write(90);
delay(3000);
}
```

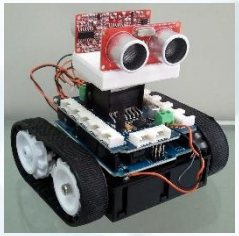




Robô que Anda

Robô que Anda

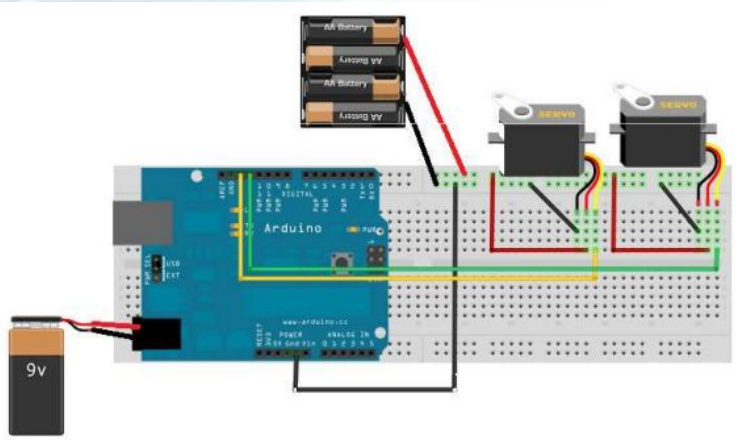
Para realizarmos o experimento **21** serão utilizados os seguintes materiais: arduino, protoboard, fios de conexão, 2 Servos hackeados, 2 Rodas, caixa de papelão e fita adesiva / elástico.
Sua função será, construir um robô com duas rodas e fazê-lo andar
Favor conectar a bateria 4,8v e execute o programa.
Posteriormente teste o robô com a bateria de 9v ligada ao Arduino



```
#include <Servo.h>
Servo servo1;
Servo servo2;
void setup()
{
servo1.attach(12);
servo2.attach(13);
}
void loop()
{
// andar para frente (ou para trás)
servo1.write(0);
servo2.write(180);
delay(4000);

// parar
servo1.write(90);
servo2.write(180);
delay(2000);
```

cont-1. p21

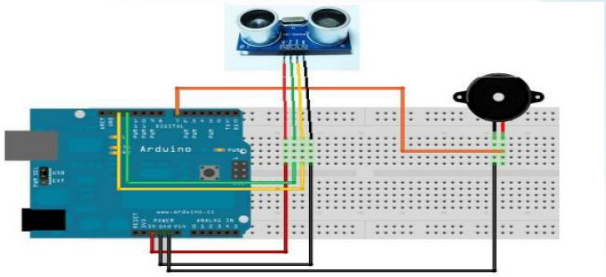
```
// andar para trás (ou para frente)
servo1.write(180);
servo2.write(0);
delay(4000);
// parar
servo1.write(90);
servo2.write(180);
delay(2000);
}
```



Robô que Anda , Para e Vira

Robô que Anda , Para e Vira

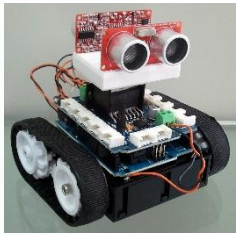
Para realizarmos o experimento **22** serão utilizados os seguintes materiais: arduino, protoboard, fios de conexão, 2 Servos hackeados, 2 Rodas, caixa de papelão e fita adesiva / elástico.
Sua função será, construir um robô com duas rodas e fazê-lo andar, parar e virar.
Favor conectar a bateria 4,8v e execute o programa.
Posteriormente teste o robô com a bateria de 9v ligada ao Arduino



```
#include <Servo.h>
Servo servo1;
Servo servo2;
void setup()
{
}
void andarFrente(int tempo)
{
servo1.attach(13);
servo2.attach(12);
 servo1.write(180);
servo2.write(0);
delay(tempo);
}
void andarTras(int tempo)
{
servo1.attach(13);
servo2.attach(12);
 servo1.write(0);
servo2.write(180);
delay(tempo);
}
void parar(int tempo)
{
servo1.detach();
servo2.detach();
delay(tempo);
}
```

cont-1. p22

```
void virarEsq(int tempo)
{
servo1.attach(13);
servo2.attach(12);
 servo1.write(90);
servo2.write(0);
delay(tempo);
}
void virarDir(int tempo)
{
servo1.attach(13);
servo2.attach(12);
 servo1.write(180);
servo2.write(90);
delay(tempo);
}
void loop()
{
andarFrente(2000);
virarEsq(1000);
andarFrente(1000);
parar(1000);
pararTras(2000);
parar(1000);
}
```



Robô que Anda , Para , Vira e Desvia de Obstáculos

```
#include <Servo.h>
#include <Ultrasonic.h>
Ultrasonic ultrasonic(10,9);
Servo servo1;
Servo servo2;
void setup()
{
pinMode(8,OUTPUT);
Serial.begin(9600);
}
void andarFrente(int tempo)
{
servo1.attach(13);
servo2.attach(12);
 servo1.write(180);
servo2.write(0);
delay(tempo);
}
void andarTras(int tempo)
{
servo1.attach(13);
servo2.attach(12);
 servo1.write(0);
servo2.write(180);
delay(tempo);
}
```

cont-1. p23

```
void parar(int tempo)
{
servo1.detach();
servo2.detach();
delay(tempo);
}
void virarEsq(int tempo)
{
servo1.attach(13);
servo2.attach(12);
 servo1.write(90);
servo2.write(0);
delay(tempo);
}
void virarDir(int tempo)
{
servo1.attach(13);
servo2.attach(12);
 servo1.write(180);
servo2.write(90);
delay(tempo);
}
```

cont-2. p23

```
void buzina() {
digitalWrite(8,HIGH);
delay(50);
digitalWrite(8,LOW);
delay(50);
digitalWrite(8,HIGH);
delay(50);
digitalWrite(8,LOW);
delay(50);
}
int lerDistancia() {
int dist = ultrasonic.Ranging(CM);
return dist;
}
void loop() {
andarFrente(200);
Int dist=lerDistancia();
Serial.println(dist);
while (dist < 30)
{
parar(0);
buzina();
virarEsq(500);
dist = lerDistancia();
Serial.println(dist);
} }
```

Robô que Anda , Para , Vira e Desvia de Obstáculos

Robô Lógica Apurada

```
#include <Servo.h>
#include <Ultrasonic.h>
Ultrasonic ultrasonic(10,9);
Servo servo1;
Servo servo2;
void setup()
{
pinMode(8,OUTPUT);
Serial.begin(9600);
}
void andarFrente(int tempo)
{
servo1.attach(13);
servo2.attach(12);
 servo1.write(180);
servo2.write(0);
delay(tempo);
}
void andarTras(int tempo)
{
servo1.attach(13);
servo2.attach(12);
 servo1.write(0);
servo2.write(180);
delay(tempo);
}
```

cont-1. p24

```
void parar(int tempo)
{
servo1.detach();
servo2.detach();
delay(tempo);
}
void virarEsq(int tempo)
{
servo1.attach(13);
servo2.attach(12);
 servo1.write(90);
servo2.write(0);
delay(tempo);
}
void virarDir(int tempo)
{
servo1.attach(13);
servo2.attach(12);
 servo1.write(180);
servo2.write(90);
delay(tempo);
}
```

cont-2. p24

```
void buzina() {
digitalWrite(8,HIGH);
delay(50);
digitalWrite(8,LOW);
delay(50);
digitalWrite(8,HIGH);
delay(50);
digitalWrite(8,LOW);
delay(50);
}
int lerDistancia() {
int dist = ultrasonic.Ranging(CM);
return dist;
}
void loop() {
andarFrente(200);
Int dist=lerDistancia();
Serial.println(dist);
if (dist<20)
andarTras(1000);
else
virarEsq(1000);
dist = lerDistancia();
Serial.println(dist);
} }
```

Robô que Anda , Para , Vira e Segue a Luz

Robô que Anda

Para realizarmos o experimento **25** serão utilizados os seguintes materiais: Robôs, Fios, 2 LDR (sensores de luz) e 2 Resistores de 10k.
Sua função será, fazer com que o robô deve andar em direção à luz

cont- p25

```
.
.
.
.
int leLuzEsq()
{
int luz = analogRead(0);
return luz;
}
int leLuzDir()
{
int luz = analogRead(1);
return luz;
}

void loop()
{
andarFrente(200);
int dist = lerDistancia();
Serial.println(dist);
while (dist < 30)
{
parar(0);
buzina();
if (dist < 20)
andarTras(1000);
else
virarEsq(1000);
dist = lerDistancia();

Serial.println(dist);
}
int luzEsq = leLuzEsq();
int luzDir = leLuzDir();
Serial.print(luzEsq);
Serial.println(luzDir);
}
while (luzEsq < (luzDir-50))
{
virarEsq(400);
int luzEsq = leLuzEsq();
int luzDir = leLuzDir();
Serial.print(luzEsq);
Serial.print(" - ");
Serial.println(luzDir); }
}
```

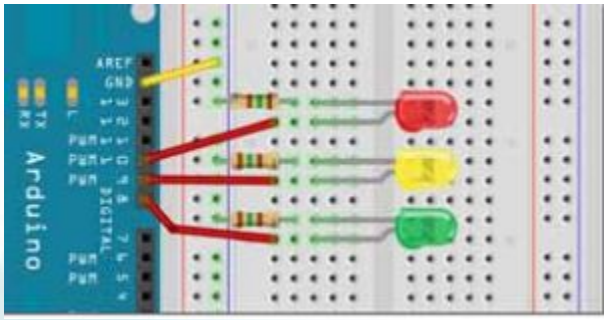
Favor transferia o programa para o robô e verifique se ele segue a luz

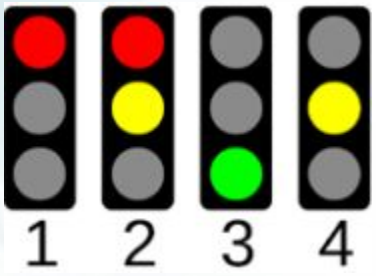
Semáforo

Semáforo

Para realizarmos o experimento serão utilizados os seguintes materiais: três LEDs, com o ânodo de cada um indo para os pinos digitais 8, 9 e 10, por meio de um resistor de 150 Ω cada e fios de jumper's.

```
int ledDelay = 10000; // espera entre as alterações
int redPin = 10;
int yellowPin = 9;
int greenPin = 8;
void setup() {
pinMode(redPin, OUTPUT);
pinMode(yellowPin, OUTPUT);
pinMode(greenPin, OUTPUT);
}
void loop() {
digitalWrite(redPin, HIGH); // acende a luz vermelha
delay(ledDelay); // espera 5 segundos
digitalWrite(yellowPin, HIGH); // acende a luz amarela
delay(2000); // espera 2 segundos
digitalWrite(greenPin, HIGH); // acende a luz verde
digitalWrite(redPin, LOW); // apaga a luz vermelha
digitalWrite(yellowPin, LOW); // apaga a luz amarela
delay(ledDelay); // espera ledDelay milissegundos
digitalWrite(yellowPin, HIGH); // acende a luz amarela
digitalWrite(greenPin, LOW); // apaga a luz verde
delay(2000); // espera 2 segundos
digitalWrite(yellowPin, LOW); // apaga a luz amarela
        // agora nosso loop se repete
}
```

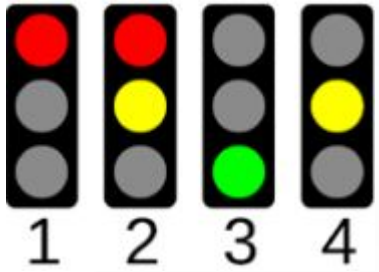


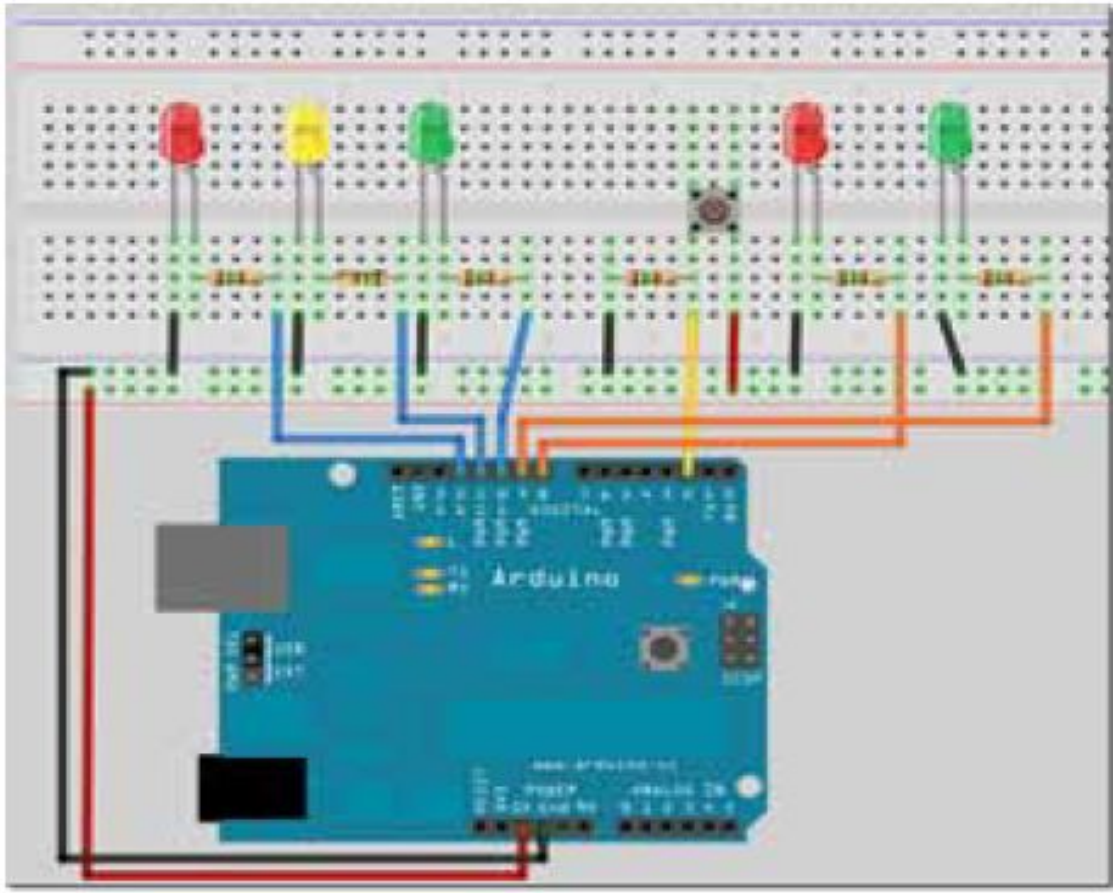


Semáforo Interativo

Semáforo Interativo

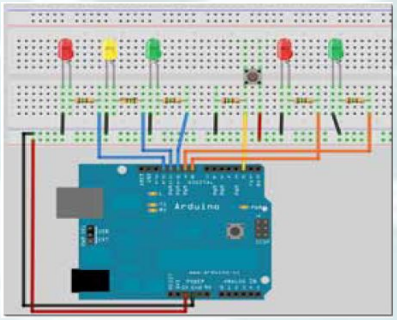
Para realizarmos o experimento serão utilizados os seguintes materiais: 2 LEDs, vermelhos difusos, LED amarelo difuso, 2 LEDs verdes difusos, Resistor de 150 ohms, 4 resistores e Botão.
Sua função será, incluir um farol de pedestre e um botao, que será pressionado pelos pedestres para solicitar a travessia da rua.





Semáforo Interativo

```
int carRed = 12; // estabelece o semáforo para carros
int carYellow = 11;
int carGreen = 10;
int pedRed = 9; // estabelece o semáforo para pedestres
int pedGreen = 8;
int button = 2; // pino do botão
int crossTime = 5000; // tempo para que os pedestres atravessem
unsigned long changeTime; // tempo desde que o botão foi pressionado
void setup() {
pinMode(carRed, OUTPUT);
pinMode(carYellow, OUTPUT);
pinMode(carGreen, OUTPUT);
pinMode(pedRed, OUTPUT);
pinMode(pedGreen, OUTPUT);
pinMode(button, INPUT); // botão no pino 2
// acende a luz verde
digitalWrite(carGreen, HIGH);
digitalWrite(pedRed, HIGH);
}
void loop() {
int state = digitalRead(button);
/* verifica se o botão foi pressionado e se transcorreram 5 segundos desde a última
vez que isso ocorreu */
if (state == HIGH && (millis() - changeTime) > 5000) {
changeLights(); // Chama a função para alterar as luzes
}
}
```



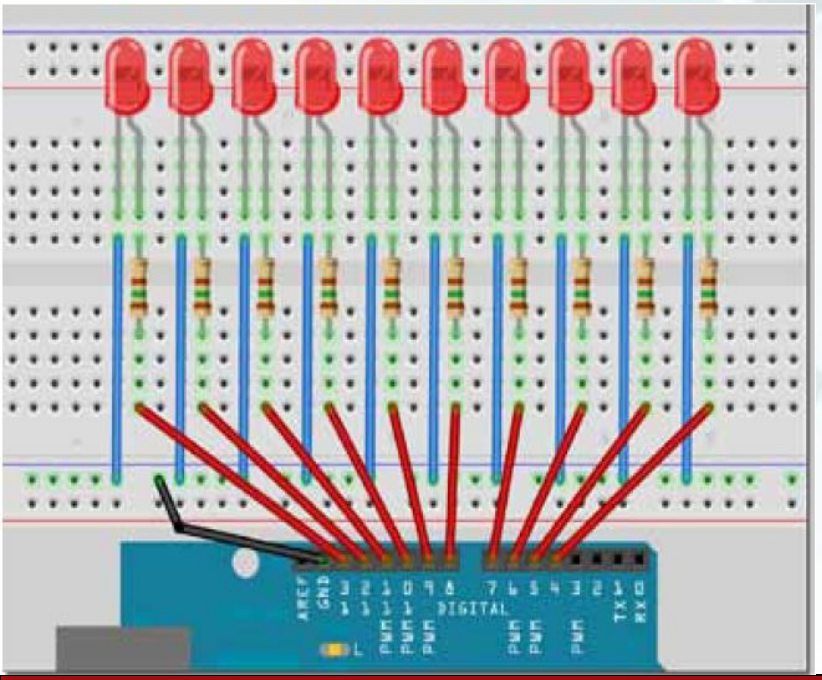
Continuação

```
void changeLights() {
digitalWrite(carGreen, LOW); // apaga o verde
digitalWrite(carYellow, HIGH); // acende o amarelo
delay(2000); // espera 2 segundos
digitalWrite(carYellow, LOW); // apaga o amarelo
digitalWrite(carRed, HIGH); // acende o vermelho
delay(1000); // espera 1 segundo, por segurança
digitalWrite(pedRed, LOW); // apaga o vermelho dos pedestres
digitalWrite(pedGreen, HIGH); // acende o verde dos pedestres
delay(crossTime); // espera por um intervalo de tempo predefinido
// pisca o verde dos pedestres
for (int x=0; x<10; x++) {
digitalWrite(pedGreen, HIGH);
delay(250);
digitalWrite(pedGreen, LOW);
delay(250);
}
// acende o vermelho dos pedestres
digitalWrite(pedRed, HIGH);
delay(500);
digitalWrite(carYellow, HIGH); // acende o amarelo
digitalWrite(carRed, LOW); // apaga o vermelho
delay(1000);
digitalWrite(carGreen, HIGH); // acende o verde
digitalWrite(carYellow, LOW); // apaga o amarelo
// registra o tempo desde a última alteração no semáforo
changeTime = millis();
// depois retorna para o loop principal do programa
}
```

Efeito Interativo de Iluminação Sequencial Com LED's

Iluminação Sequencial

Para realizarmos o experimento serão utilizados os seguintes materiais: 10 LEDs de 5 mm ,10 resistores limitadores de corrente.
Sua função será, acender as luzes sequencialmente.

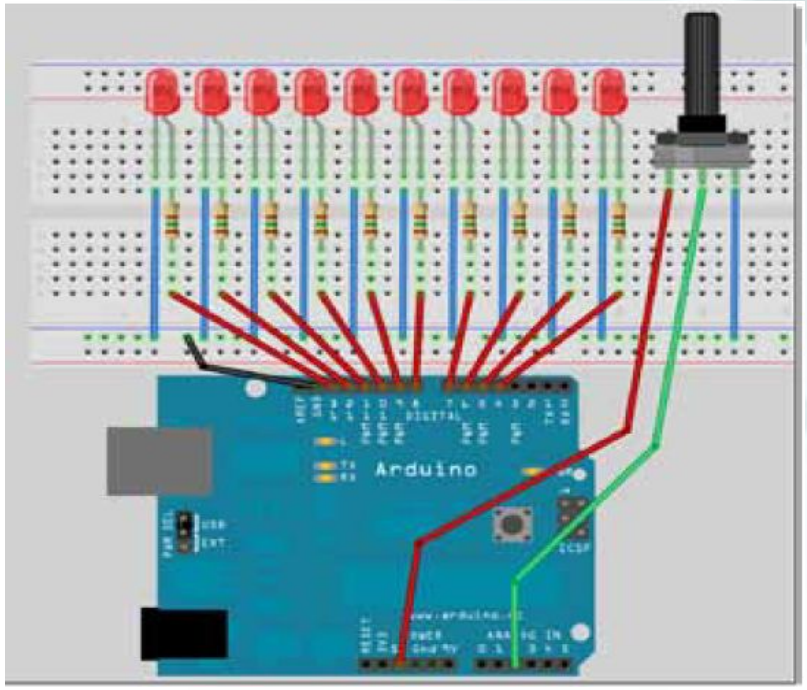


```
byte ledPin[] = {4, 5, 6, 7, 8, 9, 10, 11, 12, 13}; // cria um array para os pinos dos LEDs
int ledDelay(65); // intervalo entre as alterações
int direction = 1;
int currentLED = 0;
unsigned long changeTime;
void setup() {
for (int x=0; x<10; x++) { // define todos os pinos como saída
pinMode(ledPin[x], OUTPUT); }
changeTime = millis();
}
void loop() {
if ((millis() - changeTime) > ledDelay) { // verifica se já transcorreram ledDelay ms desdea última alteração
changeLED();
changeTime = millis();
  }
}
void changeLED() {
for (int x=0; x<10; x++) { // apaga todos os LEDs
digitalWrite(ledPin[x], LOW);
}
digitalWrite(ledPin[currentLED], HIGH); // acende o LED atual
currentLED += direction; // incrementa de acordo com o valor de direction altera a direção se tivermos atingido o fim
if (currentLED == 9) {direction = -1;}
if (currentLED == 0) {direction = 1;}
}
```

Efeito Interativo de Iluminação Sequencial Com LED's

Iluminação Sequencial

Para realizarmos o experimento serão utilizados os seguintes materiais: 10 LEDs de 5 mm ,10 resistores limitadores de corrente.
Sua função será, acender as luzes sequencialmente.

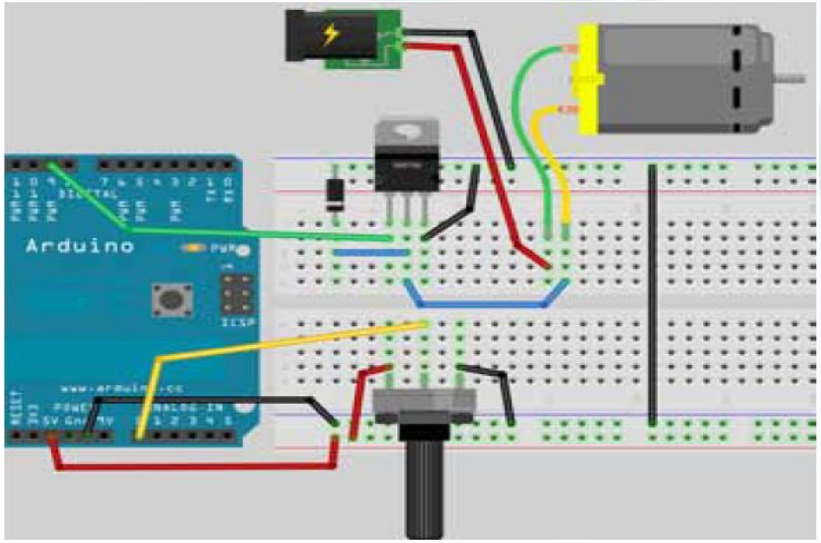



```
byte ledPin[] = {4, 5, 6, 7, 8, 9, 10, 11, 12, 13}; // Cria um array para os pinos dos LEDs
int ledDelay; // intervalo entre as alterações
int direction = 1;
int currentLED = 0;
unsigned long changeTime;
int potPin = 2; // seleciona o pino de entrada para o potenciômetro
void setup() {
for (int x=0; x<10; x++) { // define todos os pinos como saída
pinMode(ledPin[x], OUTPUT);
  }
changeTime = millis();
}
void loop() {
ledDelay = analogRead(potPin); // lê o valor do potenciômetro
if ((millis() - changeTime) > ledDelay) { // verifica se transcorreram ledDelay ms desde a  última alteração
changeLED();
changeTime = millis();
  }
}
void changeLED() {
for (int x=0; x<10; x++) { // apaga todos os LEDs
digitalWrite(ledPin[x], LOW);
}
digitalWrite(ledPin[currentLED], HIGH); // acende o LED atual
currentLED += direction; // incrementa de acordo com o valor de direction
// altera a direção se tivermos atingido o fim
if (currentLED == 9) {direction = -1;}
if (currentLED == 0) {direction = 1;}
}
```

Efeito Interativo de Iluminação Sequencial Com LED's
Mood lamp com controle serial

Efeito Iluminação Sequencial

Para realizarmos o experimento serão utilizados os seguintes materiais: Motor CC, Potenciômetro de 10 Kω, Transistor TIP120, Diodo 1N4001, Plugue femea, Fonte de alimentação externa ou um equivalente adequado.
Sua função será, controlar sua lâmpada enviando comandos do PC para o arduino utilizando o Serial Monitor, do IDE do arduino. A comunicação serial e o processo de envio de dados, um bit de cada vez, por um link de comunicação.




```
char buffer[18];
int red, green, blue;
int RedPin = 11;
int GreenPin = 10;
int BluePin = 9;
void setup() {
Serial.begin(9600);
Serial.flush();
pinMode(RedPin, OUTPUT);
pinMode(GreenPin, OUTPUT);
pinMode(BluePin, OUTPUT);
}
void loop() {
if (Serial.available() > 0) {
int index=0;
delay(100); // deixe o buffer encher
int numChar = Serial.available();
if (numChar>15) {
numChar=15; }
while (numChar--) {
buffer[index++] = Serial.read(); }
splitString(buffer);  }
}
void splitString(char* data) {
Serial.print("Data entered: ");
Serial.println(data);
char* parameter;
parameter = strtok (data, " ,");
```

continuação

```
while (parameter != NULL) {
setLED(parameter);
parameter = strtok (NULL, " ,");
} // Limpa o texto e os buffers seriais
for (int x=0; x<16; x++) {
buffer[x]='\0'; }
Serial.flush();
}
void setLED(char* data) {
if ((data[0] == 'r') || (data[0] == 'R')) {
int Ans = strtol(data+1, NULL, 10);
Ans = constrain(Ans,0,255);
analogWrite(RedPin, Ans);
Serial.print("Red is set to: ");
Serial.println(Ans); }
if ((data[0] == 'g') || (data[0] == 'G')) {
int Ans = strtol(data+1, NULL, 10);
Ans = constrain(Ans,0,255);
analogWrite(GreenPin, Ans);
Serial.print("Green is set to: ");
Serial.println(Ans); }
if ((data[0] == 'b') || (data[0] == 'B')) {
int Ans = strtol(data+1, NULL, 10);
Ans = constrain(Ans,0,255);
analogWrite(BluePin, Ans);
Serial.print("Blue is set to: ");
Serial.println(Ans);
} }
```