Aula 10: Oficina de Programação e Robótica

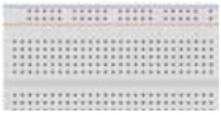
Nessa aula, vamos criar mais alguns projetos com o Arduíno, além de entender alguns conceitos de Eletrônica e da linguagem em que programamos na Placa.

*Material extraído do livro *Arduino Básico* (Michael McRoberts, Editora Novatec, 2011).

LED Piscante

Nosso primeiro projeto se assemelha muito ao que vimos na aula passada, mas vamos piscar um LED conectado a um dos pinos digitais do Arduíno em vez de utilizar o LED 13, soldado na placa. Utilizaremos os seguintes componentes, que serão vistos logo a seguir.

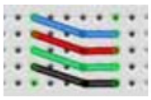
Protoboard



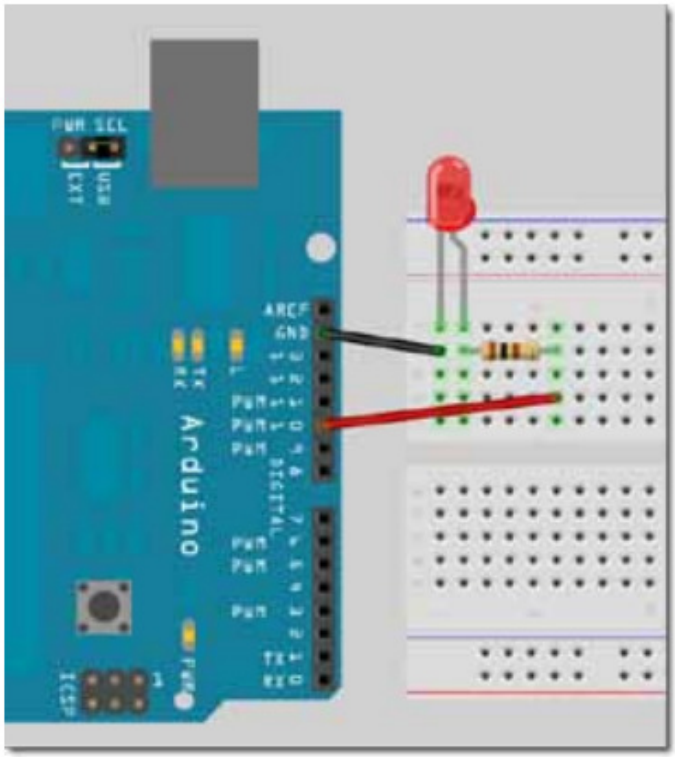
LED de 5 mm

Resistor de 100 ohms

Fios jumper



Os componentes acima precisam ser conectados da seguinte forma:



Não importa se você utiliza fios de cores diferentes ou furos diferentes na protoboard, desde que os componentes e os fios estejam conectados na mesma ordem da figura. Tenha cuidado ao inserir os componentes na protoboard. Caso sua protoboard seja nova, a superfície dos furos ainda estará rígida. A não inserção cuidadosa dos componentes pode resultar em danos.

Certifique-se de que seu LED esteja conectado corretamente, com o terminal (ou perna) mais longo conectado ao pino digital 10. O terminal longo é o ânodo do LED, e deve sempre ir para a alimentação de +5 V (nesse caso, saindo do pino digital 10); o terminal curto é o cátodo e deve ir para o terra (GND). Quando você estiver seguro de que tudo foi conectado corretamente, ligue seu Arduíno e conecte o cabo USB.

Abra a IDE do Arduíno e digite o código abaixo.

```
// Projeto 1 - LED piscante
int ledPin = 10;
void setup() {
	pinMode(ledPin, OUTPUT);
}
void loop() {
	digitalWrite(ledPin, HIGH);
	delay(1000);
	digitalWrite(ledPin, LOW);
	delay(1000);
}
```

Algoritmo 1 - Piscar o LED

Pressione o botão *Verify/Compile* no topo do IDE para certificar-se de que não há erros em seu código. Se não houver erros, clique no botão *Upload* para fazer o upload do código ao seu Arduíno. Caso tudo tenha sido feito corretamente, agora você deverá ver o LED vermelho, na protoboard, acendendo e apagando em intervalos de um segundo. Vamos analisar o código e o hardware para descobrir como ambos funcionam.

Análise do Código

Agora vamos analisar o código que fizemos acima. A primeira linha de código (em verde) é apenas um **comentário**. No Reeborg, um comentário era feito utilizando o caractere #, aqui, utilizaremos duas barras //.

Há outro formato para comentários: um bloco de instrução dentro dos sinais /* e */. Dessa forma, tudo que estiver entre os símbolos /* e */ será ignorado e não será executado, veja:

```
/*
       Projeto 1 - LED piscante

       Projeto desenvolvido na Oficina de Programação e Robótica do
       Programa Estrela Dalva.

       Data: 23/08/2013

*/
int ledPin = 10;
void setup() {
  pinMode(ledPin, OUTPUT);
}
void loop() {
  digitalWrite(ledPin, HIGH);
  delay(1000);
  digitalWrite(ledPin, LOW);
  delay(1000);
}
```

Na segunda linha do programa, temos:

```
int ledPin = 10;
```

Vocês lembram o que é uma variável? Uma variável é um local em que podemos armazenar dados. Diferente do que fazíamos no Reeborg, agora precisamos especificar qual o tipo de dado que a variável vai armazenar (se um número inteiro, real, valor lógico etc.). Como desejamos armazenar um número inteiro na variável, utilizamos o tipo de dado ***int***. Um inteiro é um número dentro do intervalo de -32.768 e 32.767. Em seguida, você atribui a esse inteiro o nome *ledPin* e dá a ele um valor de 10, que simboliza que você vai utilizar o **pino digital** 10. Ao final da instrução há um **ponto e vírgula**. Esse símbolo significa que a instrução, agora, está completa. **Nunca podemos deixar de colocar ponto e vírgula no final das nossas instruções.**

Em seguida, vemos a função ***setup()***:

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

Um *sketch* do Arduíno **precisa** ter uma função ***setup()*** e uma função ***loop()***, do contrário, não funcionará. A função ***setup()*** é executada somente uma vez no início do programa, e é nela que você emitirá instruções gerais para preparar o programa antes que o loop principal seja executado. Vamos ver funções em maiores detalhes no futuro, quando começarmos a criar as nossas.

Por enquanto, precisamos saber apenas que o nome da função é *setup* e que o código (ou bloco de código) pertencente a ela está **entre chaves** – { e }. **As funções sempre são**

delimitadas pelos caracteres { e }. No nosso caso, existe apenas um comando dentro da função *setup*, o ***pinMode***, que diz ao Arduíno que você deseja definir o modo de um de seus pinos como ***Saída* (Output)**, e não ***Entrada* (Input)**. Dentro dos parênteses, você coloca o **número do pino** e o **modo (OUTPUT ou INPUT)**. O número de seu pino é *ledPin*, previamente definido com o valor 10. Dessa forma, essa instrução está simplesmente dizendo ao Arduíno que o pino digital 10 deve ser definido como modo OUTPUT. Como a função *setup()* executa apenas uma vez, agora você avança para o loop principal do sketch.

```
void loop() {
  digitalWrite(ledPin, HIGH);
  delay(1000);
  digitalWrite(ledPin, LOW);
  delay(1000);
}
```

A função ***loop()*** é a função principal do programa e executa continuamente enquanto o Arduíno estiver ligado. Todas as declarações dentro da função *loop()* (dentro das chaves) são executadas uma de cada vez, passo a passo, até que se alcance o fim da função. Nesse ponto, o loop reinicia desde o princípio e assim infinitamente, ou até que o Arduíno seja desligado.

Neste projeto, você deseja que o LED acenda, fique aceso por um segundo, apague, permaneça apagado por um segundo e então repita o processo. Os comandos para dizer ao Arduíno como fazer essas operações estão dentro da função *loop()*, pois você deseja que sejam repetidos seguidas vezes. A primeira instrução é:

```
digitalWrite(ledPin, HIGH);
```

Ela escreve um valor ***HIGH*** ou ***LOW*** para o pino dentro da instrução (nesse caso, *ledPin*, que é o pino digital 10). Quando você define um pino como *HIGH*, está enviando **5 volts** para ele. Quando define como *LOW*, o pino se torna **zero volt, ou terra**. Essa instrução, portanto, envia 5V para o pino 10 e acende o LED. Depois dela, temos:

```
delay(1000);
```

Essa instrução simplesmente diz ao Arduíno para esperar **1.000 milissegundos, ou seja, 1 segundo** (porque há 1.000 milissegundos em um segundo) antes de executar a instrução seguinte. Você consegue dizer o que a instrução abaixo faz?

```
digitalWrite(ledPin, LOW);
```

Perguntas:

1. O que precisaríamos fazer para acender cada LED por 5 segundos?
2. Quanto tempo o LED representado pelo código abaixo fica aceso? E apagado?

```
int ledPin = 10;
int tempo = 20;
void setup() {
       pinMode(ledPin, OUTPUT);
}
void loop() {
       digitalWrite(ledPin, LOW);
       delay(tempo);
       digitalWrite(ledPin, HIGH);
       delay(tempo / 2);
}
```

3. Quantos erros existem no código abaixo? Quais são?

```
int ledPin = 10;
int tempo = 20;
void setup()
       pinMode(ledPin, OUTPUT);
}
void loop() {
       digitalWrite(ledPin, HIGH)
delay(tempo);
        digitalWrite(ledPin, LOW);
       delay(tempo / 2);
}
```

Análise do hardware

Protoboard

A protoboard é um dispositivo reutilizável, sem solda, usado para criar protótipos de um circuito eletrônico ou para experimentar projetos de circuitos. A placa consiste em uma série de furos em uma grade; sob a placa, esses furos são conectados por uma tira de metal condutivo.

As tiras ao longo do topo e da base correm em paralelo à placa, e são projetadas para carregar o **barramento de alimentação** e o **barramento do fio terra**. Os componentes no meio da placa convenientemente conectam com os 5V (ou a voltagem que você estiver utilizando) ou com o fio terra.

As tiras no centro correm a 90 graus dos barramentos de alimentação e do fio terra, e há um espaço vazio no meio para que você possa colocar **Circuitos Integrados**, de modo que cada pino do chip vá para um conjunto diferente de furos e, portanto, para um barramento

diferente. Geralmente, as placas têm pequenos encaixes nas laterais que permitem conectar diversas placas, umas às outras, para criar protoboards maiores; isso é útil para projetos mais complexos.

Resistor

O resistor (muitas vezes chamado incorretamente de *resistência*) é um dispositivo projetado para provocar resistência a uma corrente elétrica, causando uma queda na **tensão** (também chamada de **voltagem**) em seus terminais. Eventualmente, ele também é utilizado para aquecimento, como no caso de chuveiros elétricos.

Você pode pensar em um resistor como um cano de água muito mais fino do que o cano conectado a ele. Conforme a água (ou a corrente elétrica) entra no resistor, o cano se torna mais fino e o volume da água (corrente) saindo na outra ponta é, dessa forma, reduzido. No nosso caso, vamos utilizar resistores para diminuir a voltagem ou a corrente para outros dispositivos.

O valor de **resistência** é conhecido como **ohm**, e seu **símbolo é o ômega grego, Ω**. Nesse caso, o pino digital 10 está emitindo 5V de **corrente** contínua a 40 mA (**miliampères**) e seu LED requer uma voltagem de 2V e uma corrente de 35 mA. Portanto, você necessita de um resistor que reduza os 5V para 2V, e a corrente de 40 mA para 35 mA, caso queira exibir o LED com brilho máximo. Se você deseja um LED de menor luminosidade, pode utilizar um valor mais alto de resistência.

ATENÇÃO: *NUNCA* utilize um valor de resistor **MAIS BAIXO** que o necessário. Você colocará corrente demais no LED, danificando-o permanentemente. Você também poderia danificar outros componentes de seu circuito.

A fórmula para calcular o resistor necessário é:

$$R = \frac{V_S - V_L}{I}$$

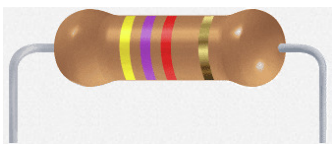
Onde V_S é a voltagem fornecida, V_L é a voltagem do LED e I é a corrente do LED. Nosso LED de exemplo tem uma tensão de 2 V e uma corrente de 35 mA, conectado a um pino digital do Arduíno, de 5 V. Assim, o valor necessário para o resistor seria de $R = (5 - 2)/0{,}035$, o que resulta em $R = 85{,}71\Omega$.

Sempre escolha o resistor com valor mais próximo MAIOR do que o valor necessário. Se você escolher um valor menor, muita corrente atravessará o resistor, danificando-o. Mas como encontrar um resistor de 100 Ω? Um resistor é pequeno demais para conter informações de fácil leitura; por isso, **resistores utilizam um código de cores**. Ao redor do resistor você tipicamente encontrará quatro faixas de cores. Utilizando o código da tabela abaixo você pode descobrir o valor de um resistor. Da mesma forma, você pode descobrir o código de cores de uma determinada resistência.

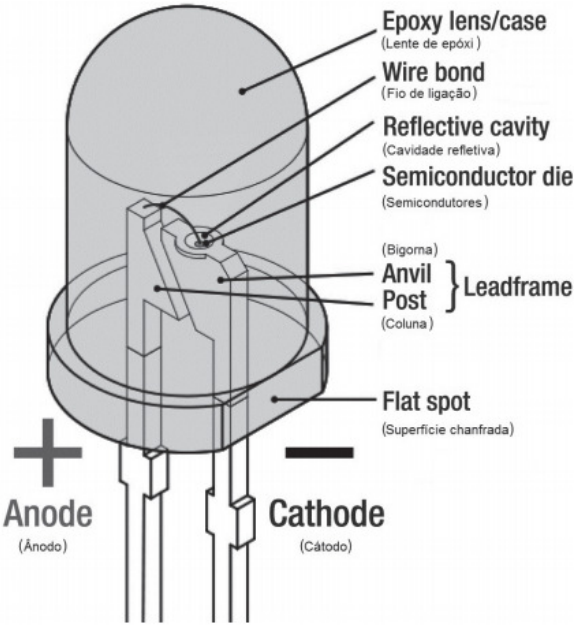
Cor	1°. Algarismo Significativo	2°. Algarismo Significativo	3°. Algarismo Significativo	Múltiplo	Tolerância
Preto		0	0	x 1	
Marrom	1	1	1	x 10	± 1%
Vermelho	2	2	2	x 10^2	± 2%
Laranja	3	3	3	x 10^3	
Amarelo	4	4	4	x 10^4	
Verde	5	5	5	x 10^5	
Azul	6	6	6	x 10^6	
Violeta	7	7	7		
Cinza	8	8	8		
Branco	9	9	9		
Ouro				x 10^{-1}	± 5%
Prata				x 10^{-2}	± 10%
Ausência					± 20%

De acordo com a tabela, para um resistor de 100Ω, você necessita de 1 na primeira faixa, que é marrom, seguido por um 0 na faixa seguinte, que é preta. Então, deve multiplicar isso por 10^1 (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 efetivo do resistor varia entre 95 Ω e 105 Ω. Dessa forma, se você tem um LED que requer 2 V e 35 mA, necessitará de um resistor com uma combinação de faixas Marrom, Preto, Marrom. O site www.areaseg.com/sinais/resistores.html pode ajudá-lo bastante.

Da mesma forma, se você encontrasse um resistor e quisesse saber seu valor, poderia fazer o mesmo processo em ordem inversa. Assim, se encontrasse o resistor da figura abaixo e quisesse descobrir seu valor para que pudesse guardá-lo em uma caixa devidamente marcada, poderia consultar a tabela e ver que ele tem um valor de...?



LEDs



A sigla LED significa *Light Emitting Diode* (Diodo Emissor de Luz). Um diodo é um dispositivo que permite o fluxo de corrente em apenas uma direção; é como uma válvula em um sistema de distribuição de água, mas nesse caso ele permite o fluxo da corrente elétrica em uma direção. Caso a corrente tente reverter e retornar na direção oposta, o diodo impede que ela o faça. Diodos podem ser úteis para prevenir que alguém conecte acidentalmente a alimentação e o fio terra aos terminais errados em um circuito, danificando os componentes.

Caso examine cuidadosamente seu LED, você perceberá dois detalhes: os terminais têm comprimentos diferentes, e um lado do LED é chanfrado, em vez de cilíndrico. Essas são pistas que indicam qual terminal é o **anodo (positivo)** e qual é o **catodo (negativo)**: **o terminal mais comprido (anodo) é conectado à alimentação de energia positiva e o terminal com o lado chanfrado (catodo) vai para o terra**.

Se você conectar seu LED da forma errada, isso não o danificará (a menos que você coloque correntes muito elevadas nele). Entretanto, **é essencial que você sempre coloque um resistor em série com o LED, para garantir que a corrente certa chegue ao LED**. Você pode danificar permanentemente o LED se não o fizer.