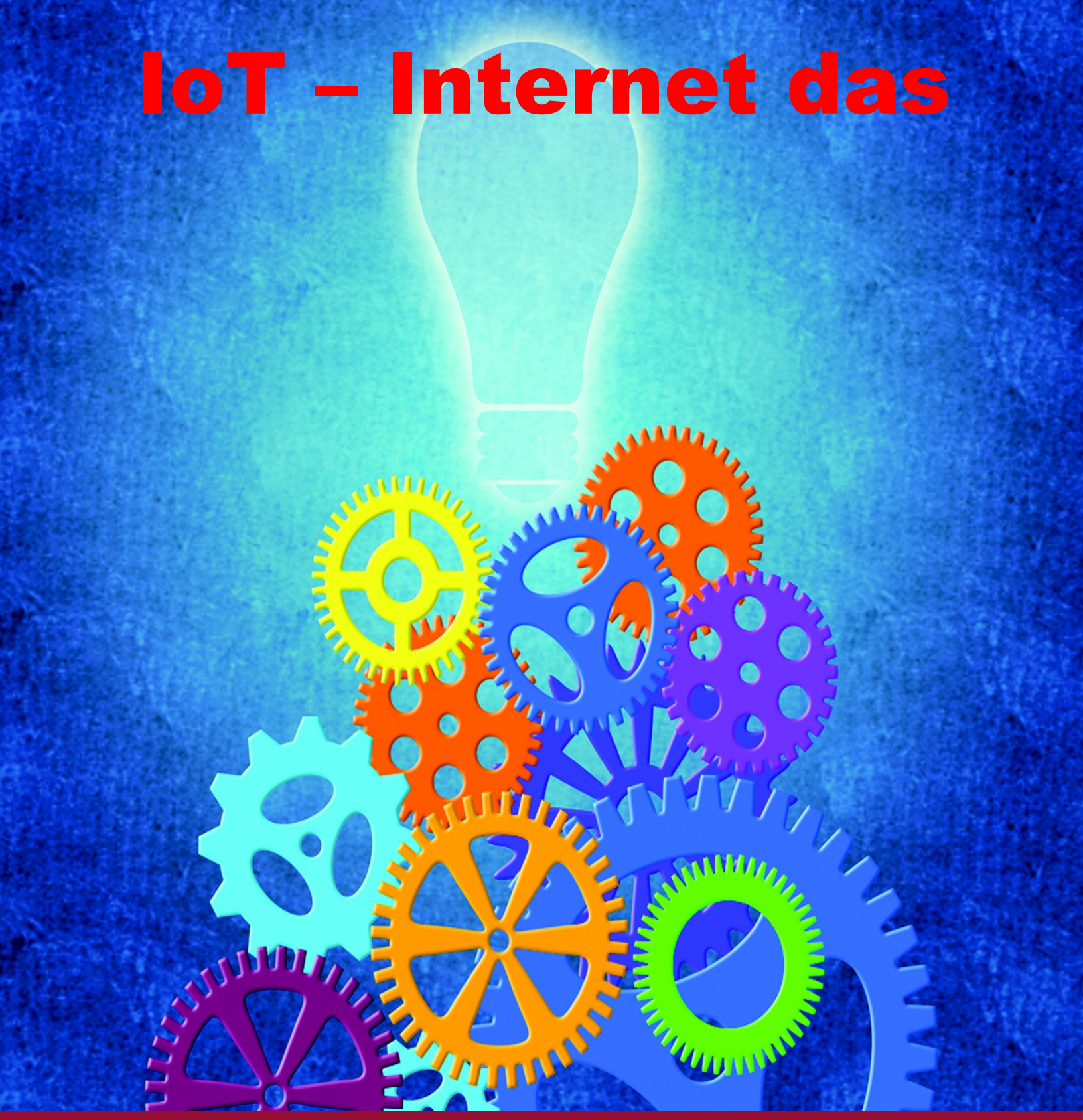
IoT – Internet das



Aula Prática 01

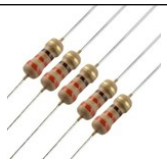
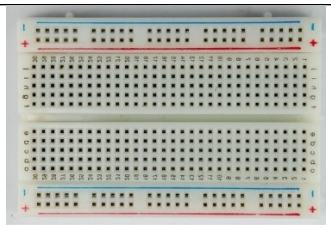
Fatec
Bebedouro

Objetivo da Aula:

Ensinar aos alunos a configuração básica do Arduino, a conexão de um LED, e a programação para fazer o LED piscar, simulando o "Hello World" das linguagens de programação tradicionais.

Material para a Aula:

1. Componentes Necessários:

Quantidade	Componente	
1	Placa Arduino UNO	
1	LED (qualquer cor)	
1	Resistor de 220Ω	
1	Protoboard (placa de ensaio)	
2	Cabos jumper (fio de conexão)	
1	Cabo USB para conectar o Arduino ao computador	

2. Software Necessário:

- Arduino IDE (ambiente de desenvolvimento para programar o Arduino)

3. Conhecimentos: Conceitos Básicos sobre Circuitos Elétricos: Corrente, Resistência e Polaridade

1. Corrente Elétrica

Definição: A corrente elétrica é o fluxo de elétrons através de um condutor, como um fio. Em um circuito elétrico, a corrente é gerada pela diferença de potencial (tensão) entre dois pontos, como os polos de uma bateria ou fonte de energia.

Unidade de Medida:

- Medida em ampères (A).

Tipos de Corrente:

- Corrente Contínua (CC): Os elétrons fluem em uma direção única e constante, como ocorre em baterias.
- Corrente Alternada (CA): Os elétrons invertem sua direção periodicamente, como nas redes elétricas residenciais.

Fluxo de Corrente:

- Em um circuito, a corrente sempre flui do terminal positivo para o negativo em termos de convenção, embora os elétrons se movam na direção oposta, do negativo para o positivo (chamado de fluxo real de corrente).

Exemplo no Arduino: Quando ligamos um LED, a corrente flui do pino de saída do Arduino (positivo) através do LED e do resistor, retornando ao GND (terra, negativo), completando o circuito.

2. Resistência Elétrica

Definição: A resistência é uma medida da oposição que um material oferece ao fluxo de corrente elétrica. Todos os condutores possuem certa resistência, que limita a quantidade de corrente que pode passar por eles.

Unidade de Medida:

- Medida em ohms (Ω).

Fórmula:

- Regida pela Lei de Ohm, que é expressa como: V=I×RV = I \times RV=I×R Onde:
  - VVV é a tensão (em volts),
  - III é a corrente (em ampères),
  - RRR é a resistência (em ohms).

Função do Resistor: Um resistor é um componente passivo que limita a corrente em um circuito. No caso de um LED, sem um resistor, a corrente poderia ser alta demais, queimando o LED. O resistor controla a quantidade de corrente que flui através do LED, protegendo-o.

Exemplo no Circuito do LED: Em um circuito com Arduino, usamos um resistor de 220Ω em série com o LED para limitar a corrente que passa pelo LED, garantindo que ele não seja danificado.

3. Polaridade

Definição: A polaridade refere-se à direção da corrente elétrica em um circuito. Em circuitos de corrente contínua (CC), como os que usamos no Arduino, há dois polos:

- Polo Positivo (+): Onde a corrente "entra" no circuito.
- Polo Negativo (-): Onde a corrente "sai" do circuito.

Importância da Polaridade em Componentes: Alguns componentes, como LEDs e capacitores, são polarizados, o que significa que eles têm uma orientação correta para serem conectados ao circuito. Se conectados de forma incorreta, esses componentes podem não funcionar ou podem ser danificados.

- LED (Diodo Emissor de Luz):
    - O anodo (perna longa) deve ser conectado ao terminal positivo.
    - O catodo (perna curta) deve ser conectado ao terminal negativo (GND).

Exemplo de Polaridade no LED: Se o LED for conectado ao contrário no circuito com o Arduino, ele não acenderá porque o fluxo de corrente não pode passar pelo componente.

Conceitos em Prática: Montagem do Circuito do LED

Circuito Simples: Ao montar um circuito com um LED e um resistor, o conhecimento de corrente, resistência e polaridade é fundamental para garantir o correto funcionamento:

1. Corrente: A corrente elétrica deve fluir do pino de saída do Arduino, atravessar o resistor e o LED, e voltar ao GND para completar o circuito.
2. Resistência: O resistor limita a corrente que passa pelo LED, garantindo que a corrente não exceda o limite que o LED pode suportar (geralmente cerca de 20mA).
3. Polaridade: O LED deve ser conectado corretamente, com o anodo no pino de saída do Arduino e o catodo ao GND.

Resumo

- Corrente Elétrica: O fluxo de elétrons através de um circuito, medido em ampères.
- Resistência Elétrica: A oposição ao fluxo de corrente, medida em ohms, que protege os componentes de sobrecarga.
- Polaridade: A direção correta da corrente, especialmente importante para componentes polarizados como LEDs.

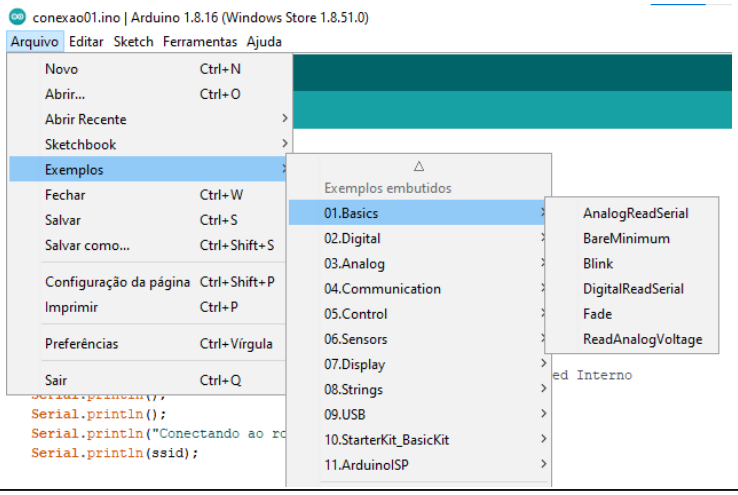
Esses conceitos são essenciais para o entendimento de circuitos básicos e ajudam a construir uma base sólida para projetos mais avançados com o Arduino.

- Introdução ao ambiente Arduino IDE

Tutorial – Fazendo o Arduino piscar led

Primeiro, certifique-se de que seu Arduino esteja desligado, desconectando-o do cabo USB. Agora, pegue sua protoboard, o LED, o resistor e os fios, e conecte tudo como mostra a figura: Este código já vem junto com a IDE do Arduino. Você pode acessar em:

Arquivo > Exemplos > 01.Basics > Blink



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 e o anodo do LED, e deve sempre ir para a alimentação de +5 V (nesse caso, saindo do pino digital 13); o terminal curto e o cátodo e deve ir para o terra (GND). Quando você estiver seguro de que tudo foi conectado corretamente, ligue seu Arduino e conecte o cabo USB.

Descrição Geral:

- O código acende e apaga o LED de maneira intermitente, simulando o efeito de piscar.
- No Arduino UNO, LED integrado à placa está conectado ao pino digital 13.
- A constante LED_BUILTIN já está definida no Arduino como o pino onde o LED integrado está conectado, variando conforme o modelo da placa. Isso garante que o mesmo código funcione em diferentes modelos de Arduino sem precisar alterar o pino manualmente.

Função setup()

A função setup() é executada uma vez quando a placa é ligada ou resetada. Nesta função, configuramos os parâmetros iniciais da placa.

```
void setup() {
  // Inicializa o pino digital LED_BUILTIN como uma saída.
  pinMode(LED_BUILTIN, OUTPUT);
}
```

pinMode(LED_BUILTIN, OUTPUT);: Este comando define o pino do LED integrado como saída. O Arduino precisa saber se os pinos estão sendo usados para receber dados (entrada) ou enviar sinais saída). No caso do LED, estamos enviando um sinal para ligá-lo ou desligá-lo, por isso ele é configurado como saída.

Função loop()

A função loop() é repetida infinitamente. É aqui que o código do Arduino vai executar as instruções continuamente.

```
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);  // Liga o LED (HIGH define o nível de tensão como alto
  delay(1000);                      // Aguarda por 1 segundo
  digitalWrite(LED_BUILTIN, LOW);   // Desliga o LED (LOW define o nível de tensão como ba
  delay(1000);                      // Aguarda por 1 segundo
}
```

1. digitalWrite(LED_BUILTIN, HIGH);: Este comando envia um sinal de 5V para o pino do LED, ligando o LED. O parâmetro HIGH significa que o pino está recebendo um nível de tensão alto (5V).
2. delay(1000);: Este comando faz o Arduino pausar a execução do código por 1000 milissegundos, ou seja, 1 segundo.
3. digitalWrite(LED_BUILTIN, LOW);: Este comando envia um sinal de 0V (nível baixo) para o pino do LED, desligando o LED. O parâmetro LOW significa que o pino está sendo desligado.
4. delay(1000);: Novamente, o Arduino aguarda 1 segundo antes de repetir o ciclo.

Resumo do Ciclo de Execução

1. O pino do LED integrado é configurado como saída.
2. O LED é ligado enviando um sinal de 5V para o pino.
3. O código aguarda 1 segundo com o LED aceso.
4. O LED é desligado enviando um sinal de 0V para o pino.
5. O código aguarda 1 segundo com o LED apagado.
6. O processo é repetido indefinidamente, fazendo o LED piscar.

Modificações e Experimentações

Mudar o tempo de atraso: Você pode alterar os valores no comando delay() para mudar o intervalo de tempo que o LED fica ligado e desligado.

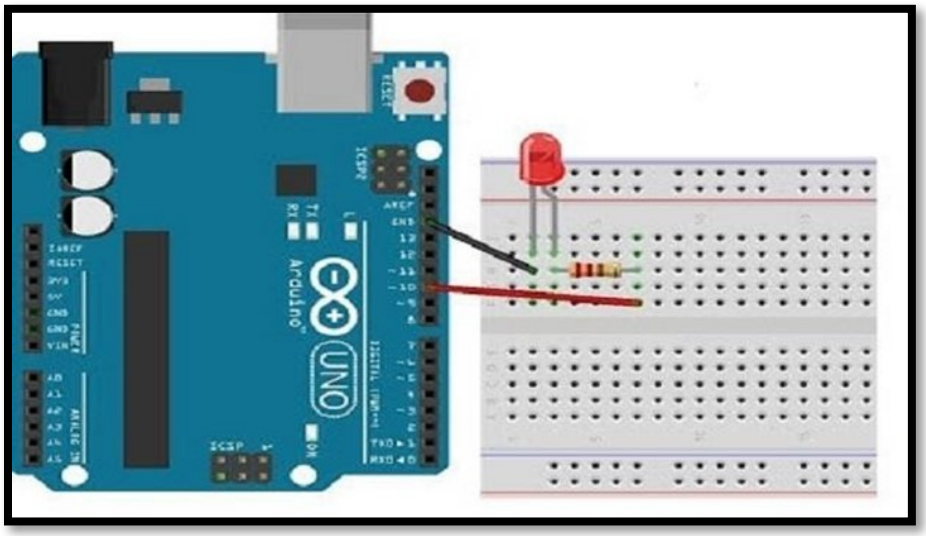
Exemplo:

1. A partir do código fonte apresentado neste tutorial, faça as modificações necessárias para que o LED fique:
   - 3 segundos aceso e 3 segundos apagado

- 200 milissegundos aceso e 500 milissegundos apagado

2. Alterando delay(500); para um piscar mais rápido (meio segundo).
3. Usar outro pino: Se você quiser usar um LED externo em vez do LED integrado, pode alterar LED_BUILTIN pelo número do pino (13) onde o LED externo está conectado.

Este código é o equivalente ao "Hello World" em programação Arduino, pois é simples e fácil de entender, além de ser uma excelente introdução ao uso de pinos digitais e controle de saída.



Código

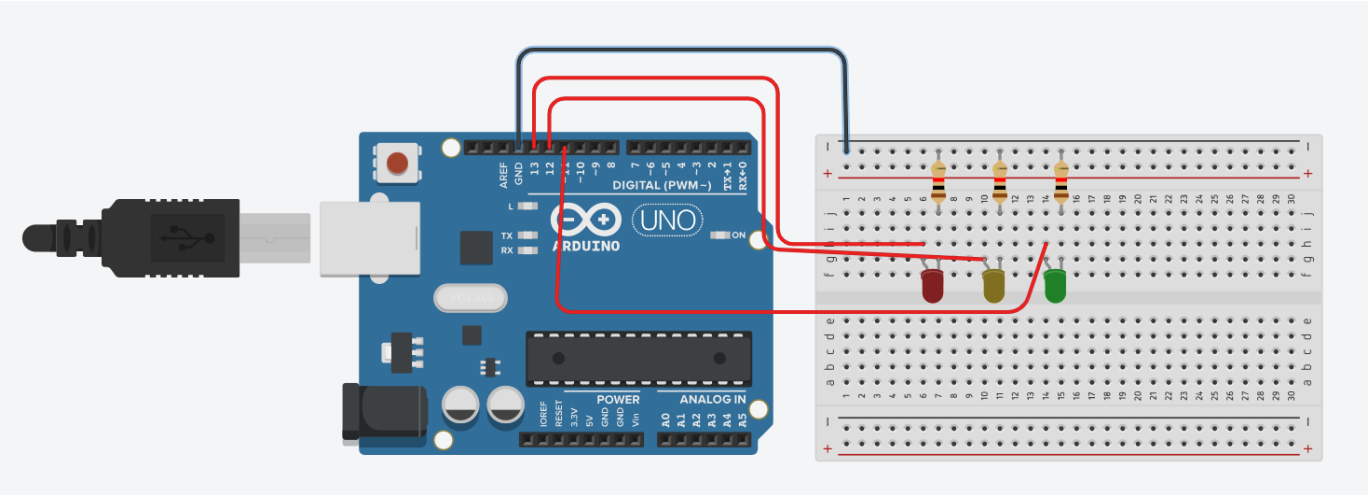
```
/*  Piscar Acende um LED por um segundo, e depois apaga pelo mesmo tempo,
repetidamente.  */

// Estabeleca um nome para o pino 13:
int led = 13;

// Se executa cada vez que o Arduino inicia:
void setup() {
  // Inicializa o pino digital como saída.
  pinMode(led, OUTPUT);
}

// A funcao loop() continua executando enquanto o Arduino estiver
alimentado,
// ou ate que o botao reset seja acionado.

void loop() {
  digitalWrite(led, HIGH);   // Acende o LED
  delay(1000);               // Aguarda um segundo (1s = 1000ms)
  digitalWrite(led, LOW);    // Apaga o LED
  delay(1000);               // Aguarda um segundo (1s = 1000ms)
}
```




```
// C++ code
//
void setup()
{
  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(11, OUTPUT);
}

void loop()
{
  //led vermelho
  digitalWrite(13, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(13, LOW);
  delay(1000); // Wait for 1000 millisecond(s)

  //led amarelo
  digitalWrite(12, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(12, LOW);
  delay(1000); // Wait for 1000 millisecond(s)

  //led verde
   digitalWrite(11, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(11, LOW);
  delay(1000); // Wait for 1000 millisecond(s)
}// C++ code
//
void setup()
{
  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(11, OUTPUT);
}

void loop()
{
  //led vermelho
  digitalWrite(13, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(13, LOW);
  delay(1000); // Wait for 1000 millisecond(s)
```

```
//led amarelo
digitalWrite(12, HIGH);
delay(1000); // Wait for 1000 millisecond(s)
digitalWrite(12, LOW);
delay(1000); // Wait for 1000 millisecond(s)

//led verde
 digitalWrite(11, HIGH);
delay(1000); // Wait for 1000 millisecond(s)
digitalWrite(11, LOW);
delay(1000); // Wait for 1000 millisecond(s)
                                          }
```