Laboratório de Redes com Arduino
Camada Enlace

SSC0142 - Redes de Computadores

Prof. Kalinka Regina Lucas Jaquie Castelo Branco

Introdução

A plataforma *Open Source* Arduino

A plataforma eletrônica Arduino (`https://www.arduino.cc/`) começou com o objetivo de fornecer uma plataforma de desenvolvimento para estudantes. Todas as placas são *open source*, ou seja, qualquer pessoa pode produzir ou modificar uma placa Arduino.

O Arduino é hoje utilizado por profissionais, estudantes e entusiastas do mundo todo para realização de inúmeros projetos. Graças à sua facilidade de uso, é uma ótima porta de entrada para o 'mundo' dos sistemas embarcados.

Nesta atividade, simularemos a placa do Arduino UNO, uma placa com 14 pinos de I/O digital (utilizados para ler valores lógicos) e 6 entradas de sinal analógico (utilizadas para ler sinais contínuos de tensão).

O programa do Arduino é feito de forma diferente de programas para *desktop* e de muitos ambientes para microcontroladores. Existem dois métodos que devem ser programados:

1. o método `setup()`, executado uma vez quando o Arduino é alimentado;
2. o método `loop()`, executado repetidamente até que a energia seja cortada da placa ou ocorra um *reset*.

Assim, quando estamos planejando um programa no Arduino, devemos dividi-lo em duas etapas:

1. uma etapa de *configuração* (inicialização de portas, comunicação, variáveis, etc.) a ser executada no início do programa através do método `setup()` e
2. a sua aplicação, que permanecerá executando indefinidamente através do método `loop()`.

Ferramenta

Para que você se familiarize com o ambiente de desenvolvimento do Arduino, utilizaremos um ambiente simulado para a montagem de circuitos e programação, o laboratório de eletrônica da Autodesk Circuits (`circuits.io/lab`). Ele permite que você monte o circuito com o Arduino e teste seu software sem a necessidade de ter uma placa consigo ou de instalar qualquer software.

Atividade 1 — Hello, World!

Como não poderia deixar de ser, nosso primeiro programa será um *Hello, World!* Em sistemas embarcados isso é feito piscando um LED. ;) Siga as instruções abaixo:

- Acesse a página `https://www.tinkercad.com/circuits`. Você terá que criar uma conta no site. Acesse "Launch Tinkercad Circuits" na parte inferior da página. O ambiente tem vários componentes que podem ser utilizados. Monte o circuito mostrado pela Figura 1.

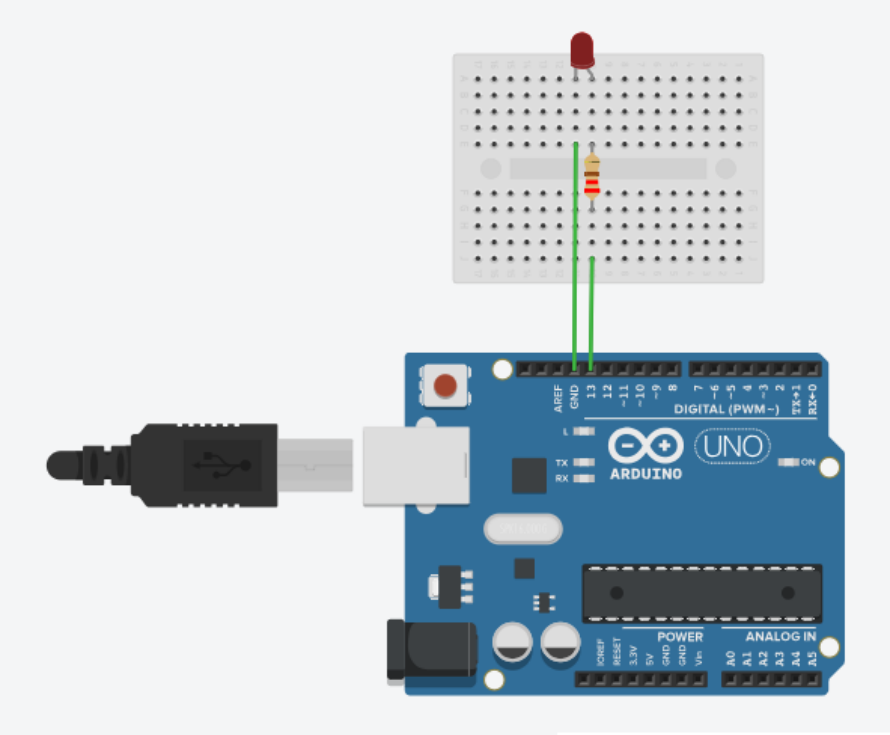

Figura 1: Circuito Hello, World!

  Você precisará adicionar os seguintes componentes (procure-os em "Componentes" — selecione "Todos" para encontrar):

  - Placa de ensaio mini;
  - LED;
  - Resistor (configure a resistência para 220Ω);
  - Arduino UNO R3.

- Para ver o LED piscar, selecione "Start Simulation".

Porque o LED pisca? Vamos agora analisar o código que faz o LED piscar. Para visualizar o código, pare a smulação e selecione "Código" no modo texto (conforme a

Figura 2.).

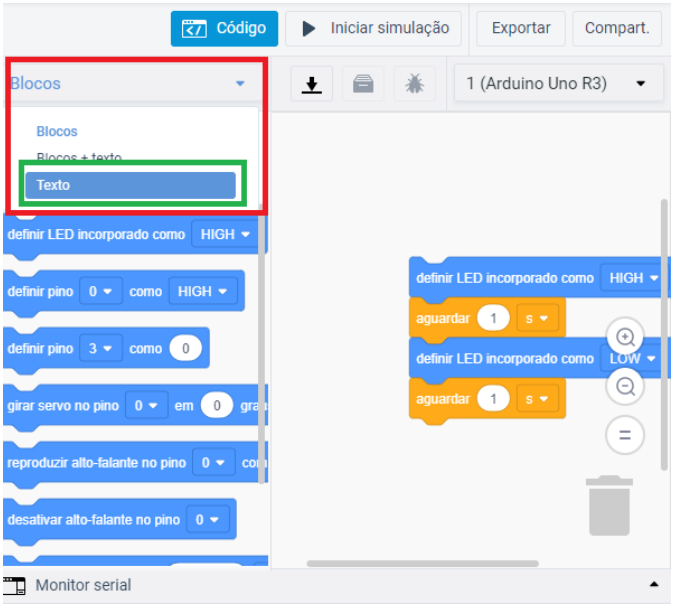

Figura 2: Selecionar o código fonte em modo texto

Programa 1: "Hello, World!"

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

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

Analisando o código Conforme vimos antes, o código do Arduino é dividido em duas etapas. Na configuração, realizada pelo método `setup()`, o pino é inicializado pelo método `pinMode()`, que possui dois argumentos: 1) O número do pino (no caso, 13) e 2) como ele é configurado (no caso, como saída).
No método `loop()`, o estado do LED (ligado ou desligado, HIGH ou LOW) é definido através do método `digitalWrite()`. Para que as mudanças fiquem visíveis para o olho humano, é inserido um *delay* de 1 segungo entre elas (o método `delay()` espera uma quantidade de milisegundos inserida).

Agora é sua vez! Modifique o exemplo padrão para que você faça um semáforo, conforme ilustrado pela Figura 3 (Você pode ligar os LEDs em quaisquer pinos digitais, de 0 a 13). Lembre-se de configurar os pinos como saída e colocar *delays* de forma que:

- O LED vermelho fique aceso por 5 segundos;
- O LED amarelo fique aceso por 1 segundo;
- O LED verde fique aceso por 3 segundos.

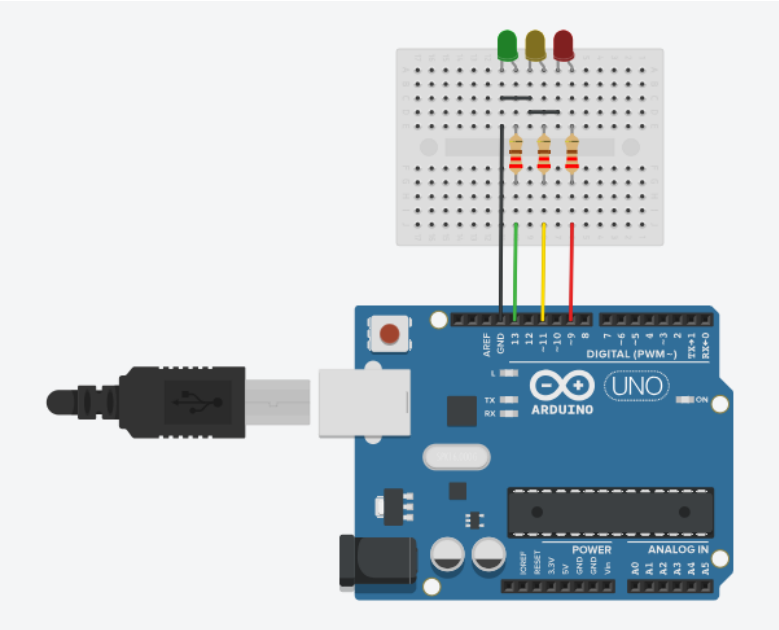

Figura 3: Semáforo

Quando terminar o programa, selecione "Iniciar Simulação" para ver seu código rodando!

Atividade 2 — Hello Back!

Agora que já sabemos como ligar um LED, faremos um programa que leia entradas do usuário, utilizando botões. Para isso, monte o circuito da Figura 4

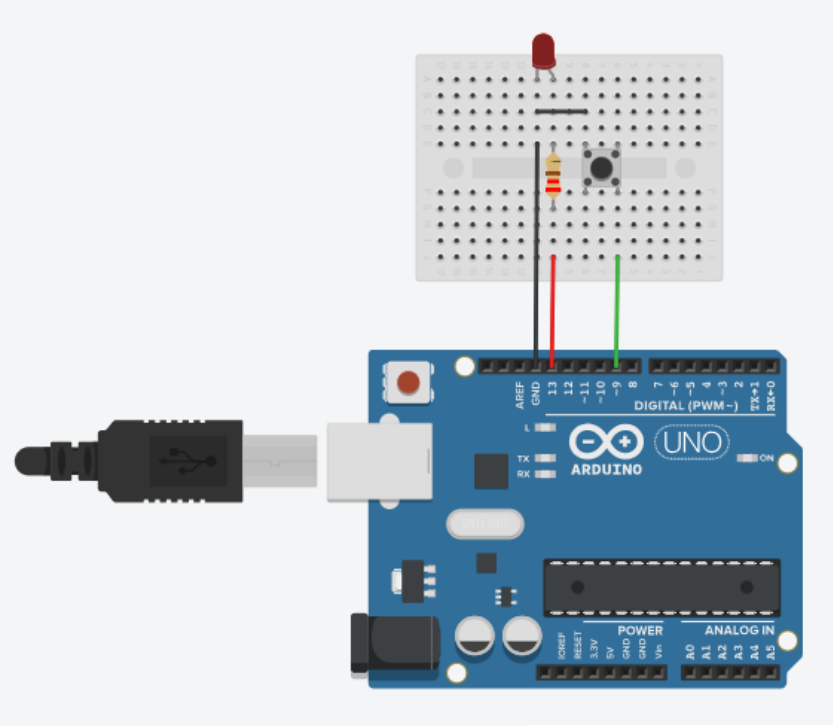

Figura 4: Leitura de Botão

Código. Substitua o código do Arduino pelo código abaixo. O que acontece?

```
#define LED_VERMELHO 13 // Pino do LED
#define BOTAO_1 9       // Pino do botao

void setup() {
  pinMode(LED_VERMELHO, OUTPUT);  //LED_VERMELHO como saÃ da
  pinMode(BOTAO_1, INPUT_PULLUP); //BOTAO_1 como entrada
  digitalWrite(LED_VERMELHO,LOW);
}

// the loop routine runs over and over again forever:
void loop() {
  while(digitalRead(BOTAO_1) == HIGH);
  digitalWrite(LED_VERMELHO, (digitalRead(LED_VERMELHO) + 1) %2);
  delay(100);
}
```

O programa anterior espera que o botão seja pressionado para alternar o estado do LED. O botão tem seu pino configurado como entrada (`INPUT_PULLUP`) no método `pinMode()`; a leitura do valor é feita através do método `digitalRead()` (quando o botão está pressionado, é lido LOW; quando não está, é lido HIGH). Se você mantiver o botão pressionado, o LED mudará de estado a cada *loop*, piscando rapidamente!

Agora é sua vez! Modifique o exemplo acima (circuito e programa) de forma que exitam dois botões: um para ligar o LED e outro para desligar o LED (para salvar a leitura do botão, guarde o retorno do método `digitalRead()` em uma variável do tipo `int`).

Quando terminar o programa, selecione "Iniciar Simulação" para ver seu código rodando!