

Prática Arduino

Roteiro de Aula Prática

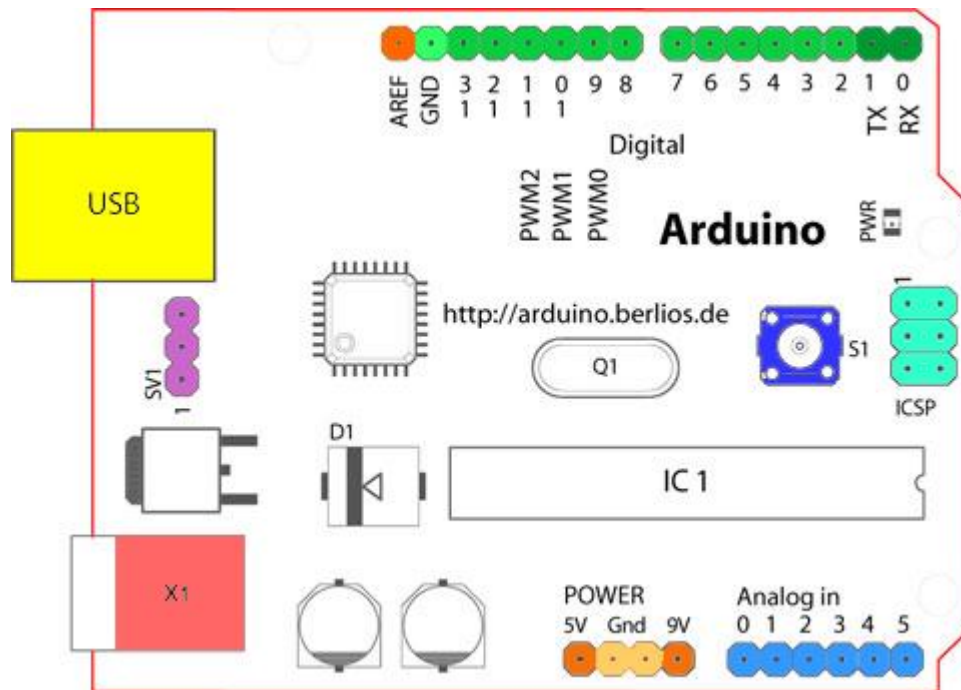
Nome: _____ Matric.: _____

O que é Arduino?

Segundo o próprio site, o Arduino é uma plataforma eletrônica *open-source* baseada no fácil uso do hardware e do software. O Arduino pode ser usado por **qualquer um** que queira montar projetos interativos.

Hardware

Existem diferentes placas Arduino. Os aspectos gerais das placas são semelhantes, diferenciando-se entre si em poucos aspectos, como por exemplo, o micro controlador.

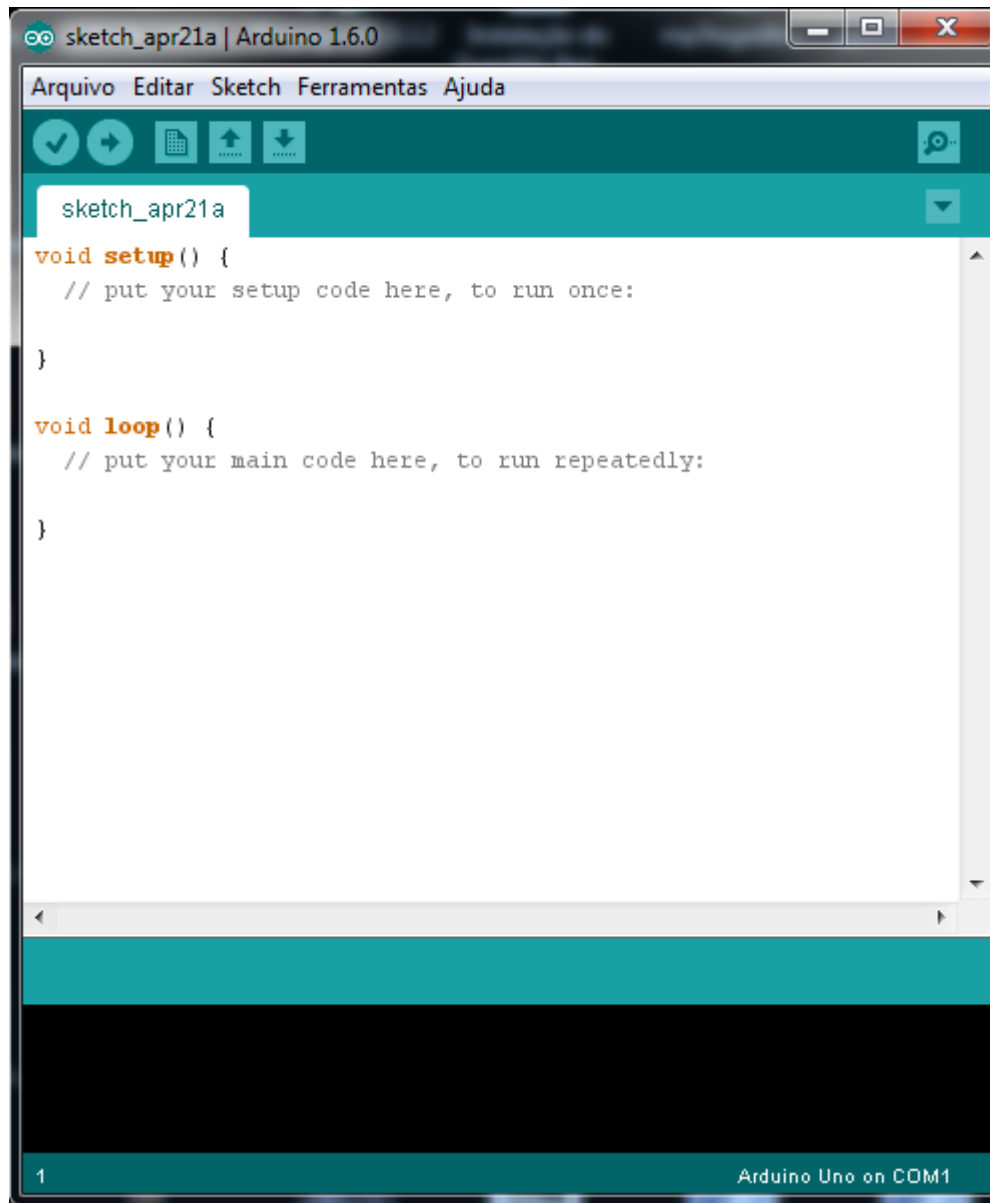


Começando no sentido horário, pela parte de cima e no centro:






- AREF (Laranja) - Analog Reference pin
- GND (Verde Claro) – Ground Pin – Pino Terra
- Pinos digitais (Verde) – 2 ao 13
- Pinos digitais Entrada/Saída Serial (Verde Escuro) – 0 e 1.
- Botão Reset (Azul Escuro) – S1
- In-Circuit Serial Programming (Azul Esverdeado) – ICSP
- Pinos Analógicos (Azul Claro) – 0 ao 5
- Pinos de Alimentação e Aterramento (Alimentação - Laranja; Aterramento – Laranja Claro)
- Entrada Externa de Alimentação (Rosa) – X1
- Jumpers de configuração de alimentação externa e Alimentação USB (Roxo) – SV1
- USB (Amarelo) – Usado para carregar os programas na placa e como fonte de alimentação.

Software

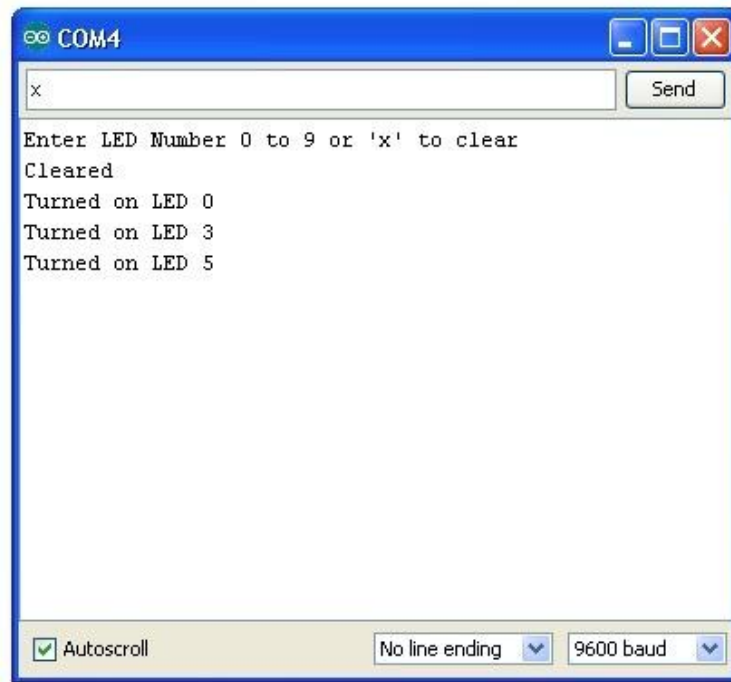
O Arduino conta com um Ambiente Integrado de Desenvolvimento (IDE), que pode ser baixado no próprio site:



É nesse ambiente que desenvolveremos os programas em nossas atividades. Abaixo, os principais botões do IDE:

-  Verificar – Checa se há erros no código
-  Upload – Compila o código e carrega o programa na placa Arduino
-  Novo – Cria um novo projeto
-  Abrir – Abre um novo projeto
-  Serial Monitor – Abre o Monitor Serial

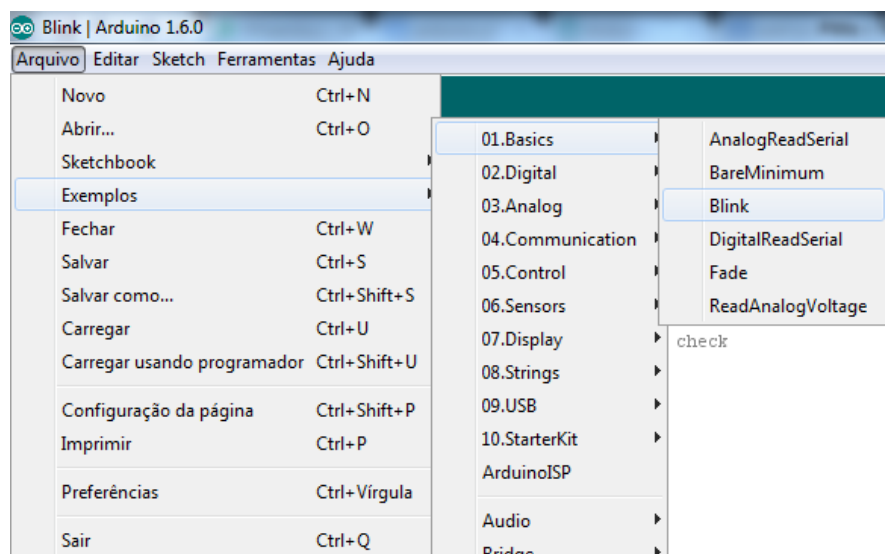
Serial Monitor



O Serial Monitor mostra os dados que são mandados da placa Arduino (USB ou placa serial). Para mandar dados para a placa, digite o texto e clique em “Enviar” ou pressione enter.

Começando



Agora que já conhecemos um pouco sobre o Arduino, vamos rodar nosso primeiro programa. Primeiramente, conecte a placa Arduino no Computador e em seguida clique no menu Arquivo – Exemplos – 01.Basics – Blink.



Após abrir o arquivo, o seguinte código deverá aparecer na tela:

```
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH); // turn the LED on (HIGH is
the voltage level)
  delay(1000);           // wait for a second
  digitalWrite(13, LOW); // turn the LED off by making
the voltage LOW
  delay(1000);           // wait for a second
}
```

Clique no botão  para verificar se não existem erros no código. Em seguida, clique em  para mandar o programa para a placa Arduino. Se não houver erros, você irá ver o LED do Arduino piscar em intervalos de 1 segundo.

Entendendo o código

Função setup()

Esta função é chamada quando o programa inicia. Ela roda somente uma vez, após a placa Arduino ligar ou após um Reset. Nessa função são iniciadas as variáveis, os pinos etc.

Função loop()

Depois da função setup(), a qual inicializa e define os valores iniciais, a função loop é chamada. Ela roda consecutivamente, isto é, fica em *loop* durante todo o tempo que a placa está ligada. Com isso, ela permite que o programa carregado mude e responda. Assim, use essa função para controlar a placa Arduino.

Constantes

As constantes são expressões pré-definidas na linguagem do Arduino. Elas são usadas para tornar os programas mais fáceis de ler. Elas são divididas em grupos:

Constantes Lógicas

false – essa constante é definida como 0 (zero).

true – geralmente essa constante é definida como 1, mas mais precisamente falando, qualquer número diferente de zero é tido como true.

Definição de modos dos pinos digitais

INPUT – Representa um pino de **entrada**. Essa entrada pode ser, por exemplo, a leitura de um sensor de temperatura.

OUTPUT – Representa um pino de **saída**. É usado para mandar certa quantidade de corrente para outros circuitos ligados à placa Arduino.

Definição dos níveis dos pinos

HIGH – Quando um pino está configurado como **INPUT**, o micro controlador reportará **HIGH** se:

- Uma voltagem maior que 3 volts está presente no pino (placas de 5V)
- Uma voltagem maior que 2 volts está presente no pino (placas de 3.3V)

Quando o pino está configurado como **OUTPUT**, o pino está:

- a 5 volts (Placas de 5V)
- a 3.3 volts (placas de 3.3V)

Nesse estado (**OUTPUT + HIGH**) o pino pode fornecer corrente. Ex: Acender um LED que está conectado em a um resistor em série ao pino GND (terra).

LOW - Quando um pino está configurado como **INPUT**, o micro controlador reportará **LOW** se:

- Uma voltagem menor que 3 volts está presente no pino (placas de 5V)
- Uma voltagem menor que 2 volts está presente no pino (placas de 3.3V)

Quando o pino está configurado como **OUTPUT**, o pino está a 0 volts (em ambas placas de 5V e 3.3V).

Nesse estado (**OUTPUT + LOW**) o pino pode consumir corrente. Ex: Acender um LED que está conectado em a um série em +5 volts (ou +3.3 volts).

Função pinMode()

Configura o pino digital especificado para se comportar como entrada (input) ou saída (output).

Sintaxe:

`pinMode(pino,modo)`

Parâmetros

Pino – o número do pino o qual você deseja configurar (de acordo com os números apresentados no esquema da placa do Arduino, a primeira imagem dessa prática).

Modo: INPUT (entrada), OUTPUT (saída).

Retorno: Nenhum

Função digitalWrite()

Escreve o valor HIGH ou LOW no pino digital

Sintaxe:

`digitalWrite(pino,valor)`

Parâmetros

Pino – o número do pino o qual você deseja configurar (de acordo com os números apresentados no esquema da placa do Arduino, a primeira imagem dessa prática).

Valor: HIGH ou LOW

Retorno: Nenhum

Função digitalRead()

Lê o valor HIGH ou LOW do pino digital

Sintaxe:

`digitalRead(pino)`

Parâmetros

Pino – o número do pino o qual você deseja configurar (de acordo com os números apresentados no esquema da placa do Arduino, a primeira imagem dessa prática).

Retorno: HIGH ou LOW

Função delay()

Pausa o programa por certo período de tempo (em milissegundos) que foi especificado como parâmetro. Note que em 1 segundo há 1000 milissegundos.

Sintaxe:

`delay(ms)`

Parâmetros

ms – o número de milissegundos que o programa vai ficar pausado.

Retorno: Nenhum

Faça você mesmo

Agora que já sabemos como o programa exemplo funciona, mude o código para realizar as seguintes tarefas:

1. A cada vez que a função loop seja executada, o LED demore 1 segundo a mais para piscar, ou seja, fica um segundo a mais aceso e um segundo a mais apagado.
2. A cada vez que a função loop seja executada, o LED demore metade do tempo inicial para piscar. Por exemplo: Se o tempo inicial é de 10 segundos, na segunda vez que a função loop for executada, o LED irá demorar apenas 5 segundos para piscar, depois 2,5 segundos e sucessivamente, até que o tempo seja tempo seja tão pequeno que o LED permanecerá somente apagado.
3. Fazer o LED piscar 3 vezes com intervalo de 3 segundos e depois 5 vezes com intervalo de 1 segundo.
4. Fazer o LED piscar aumentando o intervalo em 500 milissegundos, começando do zero, até que seja atingido o tempo de 3 segundos. Após isso, fazer o LED piscar diminuindo o intervalo em 500 milissegundos, até que o intervalo seja zero.