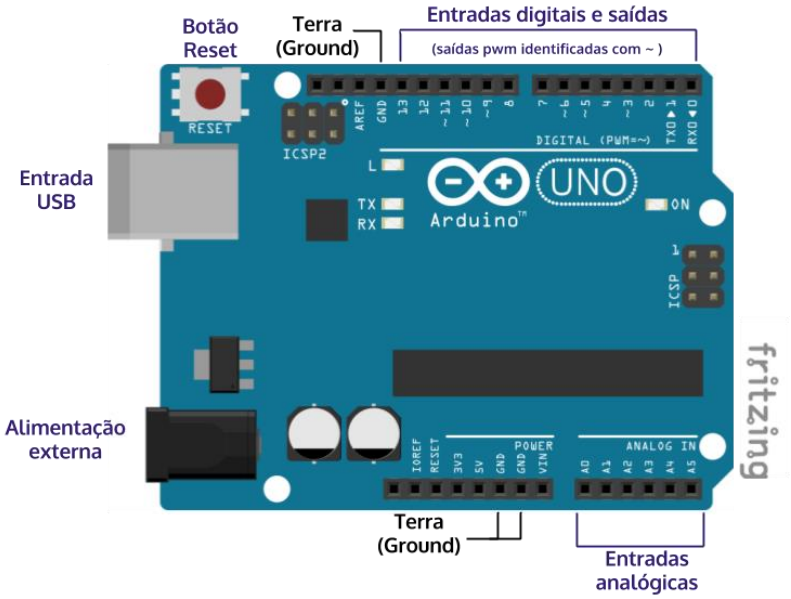
AULA 11 - IoTA

Programação física

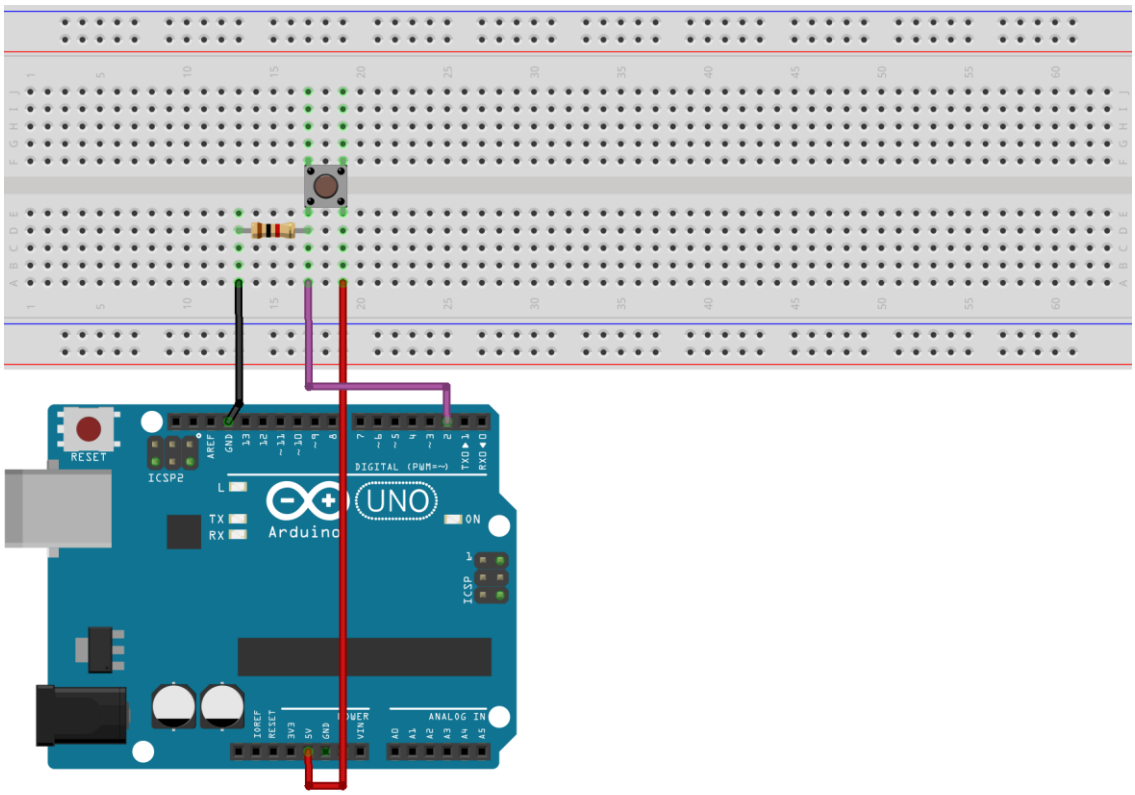
Entrada analógica x entrada digital
https://youtu.be/e-HiIS-WMrI

Entradas digitais e analógicas
Conforme apresentado no vídeo, um botão possui apenas dois estados: pressionado ou solto. Já o potenciômetro, pode estar em várias posições - ou seja, possui vários estados possíveis.
Sensores como o botão, com dois estados, são chamados digitais, enquanto sensores com vários estados, como o potenciômetro ou LDR, são chamados analógicos.
Para conectar os sensores à sua placa Arduino, você deve identificá-los como analógicos ou digitais.
Nas entradas digitais (pinos 0 a 13), conectamos sensores digitais, como o botão. Essas entradas leem apenas dois estados: 0 e 1.
E nas entradas analógicas (pinos A0, A1, A2, A3, A4 e A5), conectamos os sensores analógicos, como o potenciômetro. Essas entradas podem ler 1024 estados diferentes: de 0 a 1023.

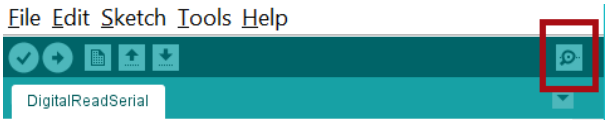


Como ler entradas digitais?
https://youtu.be/fi3XpdQ8yiQ

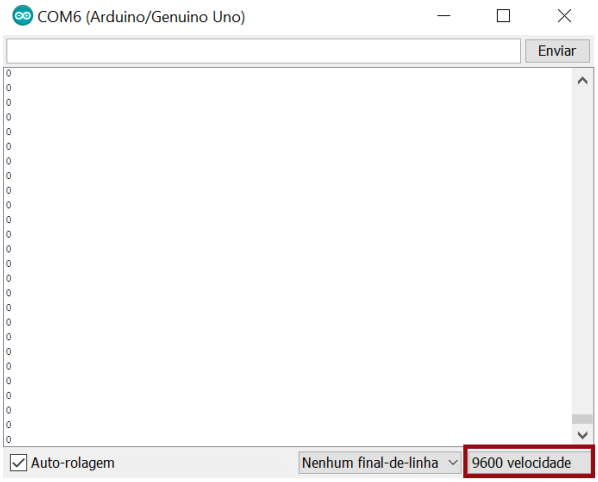
Lendo uma entrada digital
Para conectar um botão à placa, precisaremos utilizar uma configuração de resistores do tipo pull down. Você pode revisar este conceito na aula 6 do curso Eletrônica: conceitos e componentes básicos.
O circuito na protoboard deve ser similar a este:



Utilizando o código abaixo, a placa identificará quando apertamos ou não o botão e mostrará um valor na tela para cada uma das posições: pressionado (1) ou não pressionado (0). Copie e cole o código para o IDE Arduino no seu computador.
Após fazer o upload do código para sua placa, abra o Monitor Serial (ícone de lupa, no canto superior direito) e observe o que ocorre quando você pressiona o botão.



Caso o monitor não esteja imprimindo nenhum valor, verifique se a baudrate (ou velocidade) selecionada é a mesma que foi definida no seu programa (no caso, 9600).



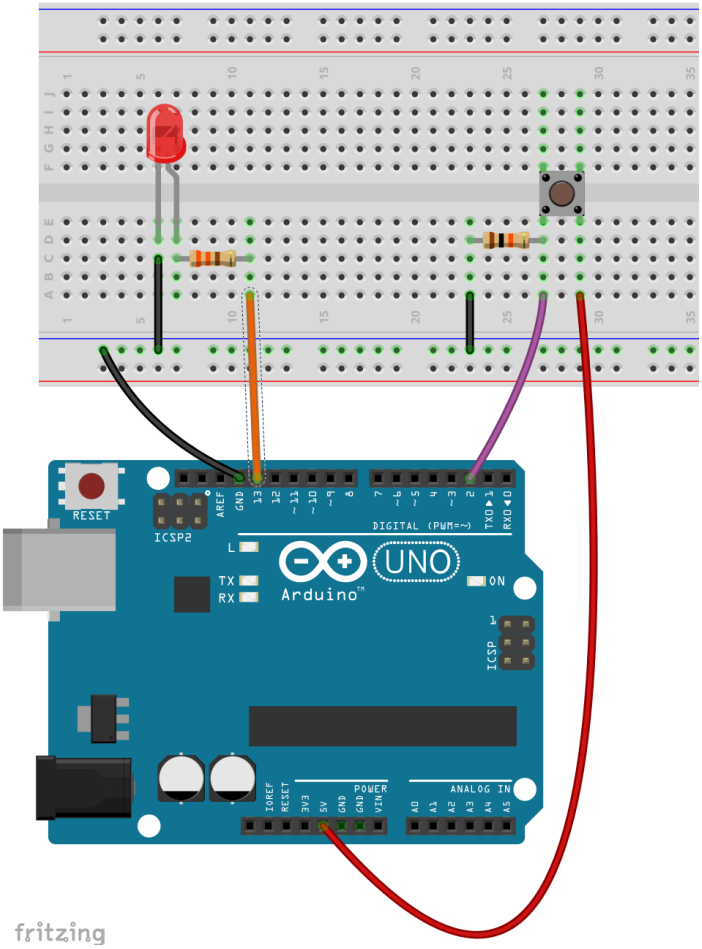
```
/* Leitura de entrada digital com botão
Se o botão estiver apertado, o monitor serial imprimirá 1.
Senão, ele imprimirá 0.
*/

const int botao = 2;          // pino no qual o botao está conectado
int estadoBotao = 0;          // variável para guardar o valor do estado do botão (pressionado ou não)

void setup() {                // configuração inicial
  Serial.begin(9600);         // inicialização do monitor serial na baudrates de 9600
  pinMode(botao, INPUT);      // inicialização do pino do botão como entrada (INPUT)
}

void loop() {                           //loop: tudo que estiver dentro se repetirá indefinidamente
  estadoBotao = digitalRead(botao);     // Lê o estado do botão (que pode ser 0 ou 1)
  Serial.println(estadoBotao);          // imprime o estado do botão no monitor serial
}
```

Como escrever nas saídas digitais?

https://youtu.be/busiPvOFYJg

Controlando o LED com o botão

Para acender um LED com o botão, você pode utilizar o seguinte circuito, apresentado no vídeo:



O código para Arduino encontra-se abaixo. Copie e cole o código para seu IDE Arduino, faça o upload para a sua placa e observe o que acontece quando você aperta o botão.
Tente mudar o código para fazer o contrário: que o LED se apague quando pressionamos o botão e se acenda quando o soltamos.

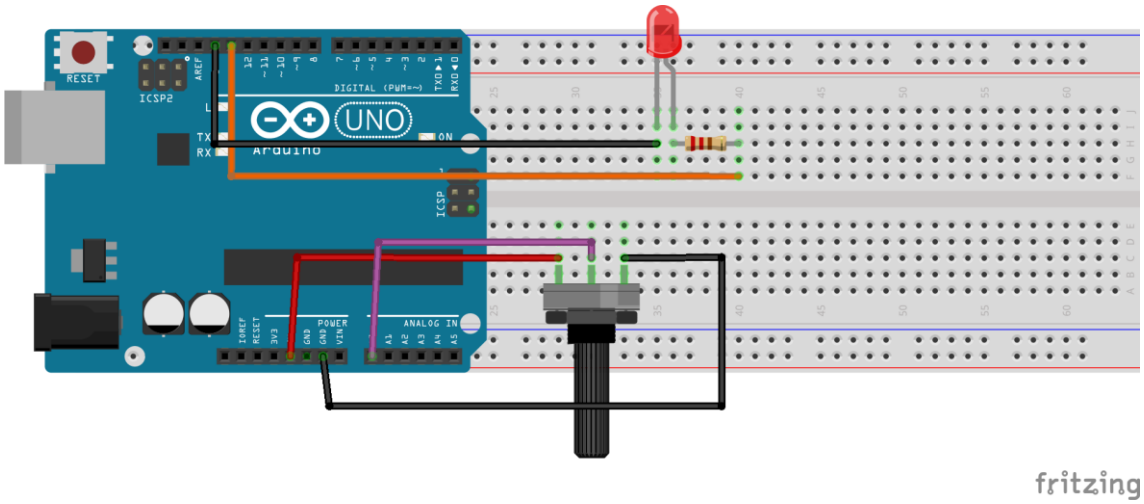
```
/*
Código de escrita em pino digital: botão e LED
Se o botão estiver apertado, o LED acende.
*/

const int botao = 2;                // pino no qual o botão está conectado
const int led = 13;                 // pino no qual o LED está conectado
int estadoBotao = 0;                // variável para guardar o valor do estado do botão (pressionado ou
não)

void setup() {
  Serial.begin(9600);               // inicialização do monitor serial
  pinMode(led, OUTPUT);             // inicialização do pino do led como saída (OUTPUT)
  pinMode(botao, INPUT);            // inicialização do pino do botão como entrada
}

void loop() {
  estadoBotao = digitalRead(botao);       // Lê o estado do botão (que pode ser 0 ou 1)
  if (estadoBotao == 1) {                 // Checa o estado do botão. Se estiver apertado (1):
    digitalWrite(led, 1);                 // acende o LED
    Serial.println("1");                  // imprime "1" no monitor serial
  }
  else {                                  // Senão:
    digitalWrite(led, 0);                 // apaga o LED
    Serial.println("0");                  // imprime "0" no monitor serial
  }
}
```

Lendo Entradas Analógicas

https://youtu.be/RXnsjLlZBu8

Usando entradas analógicas: potenciômetro

Conforme apresentado no vídeo, a tensão lida do potenciômetro é um valor contínuo - ou seja, seu sinal é analógico. Para ler esse tipo de sinal, utilizamos as portas de entrada analógicas da Arduino (A0-A5).
Abaixo, apresentamos o circuito utilizado no vídeo, com um potenciômetro e um LED:

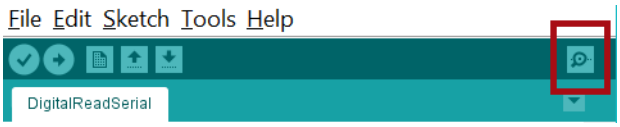


Para testar o programa apresentado com este circuito, você pode utilizar o código abaixo. Faça o upload para a sua placa e observe o que ocorre. Em seguida, faça modificações no programa para entender como as variáveis o afetam.

```
/*
Código de leitura em pino analógico: potenciômetro e LED
A frequência na qual o LED pisca depende da leitura de A0.
*/

int pinoPot = A0;   // pino de entrada do potenciômetro
int pinoLED = 13;   // pino de saída no LED
int valorPot = 0;   // variável c/ valor lido do potenciômetro

void setup() {

  pinMode(pinoLED, OUTPUT);   // declarando o pino do LED como saída (OUTPUT)
}

void loop() {

  valorPot = analogRead(pinoPot);     // lendo o valor do sensor

  digitalWrite(pinoLED, HIGH);       // Liga o LED
  delay(valorPot);                   // para o programa por <valorPot> milissegundos
  digitalWrite(pinoLED, LOW);        // desliga o LED
  delay(valorPot);                   // para o programa por <valorPot> milissegundos:
}
```

Monitor serial

O monitor serial é um recurso do ambiente Arduino que permite a comunicação entre seu computador e a placa, recebendo e enviando dados seriais.
Normalmente usamos este recurso para o controle da placa a partir do teclado, e para leituras que auxiliam na depuração de códigos.
Você já usou o monitor serial na aula inicial desta semana. Agora vamos entender alguns detalhes sobre o seu funcionamento.
Para que ele funcione, sua placa deve estar conectada ao computador via cabo USB.
Mantenha o circuito com o potenciômetro conectado ao pino A0 e utilize o código que encontra-se abaixo.
Vamos começar lendo os valores do potenciômetro no monitor serial. Para isso, após fazer o upload do código para sua placa, abra o monitor serial clicando no ícone do canto superior direito.

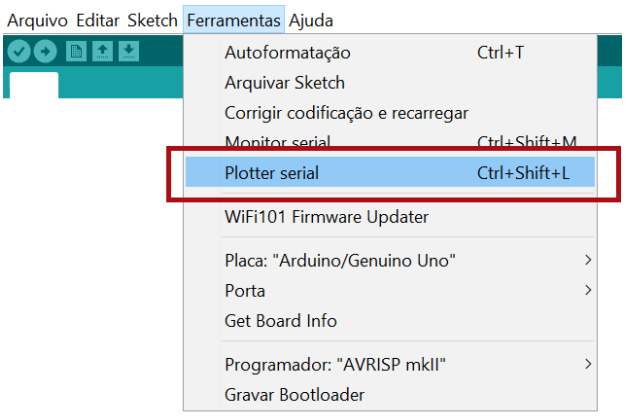


Observe o que ocorre quando você gira o eixo do potenciômetro.

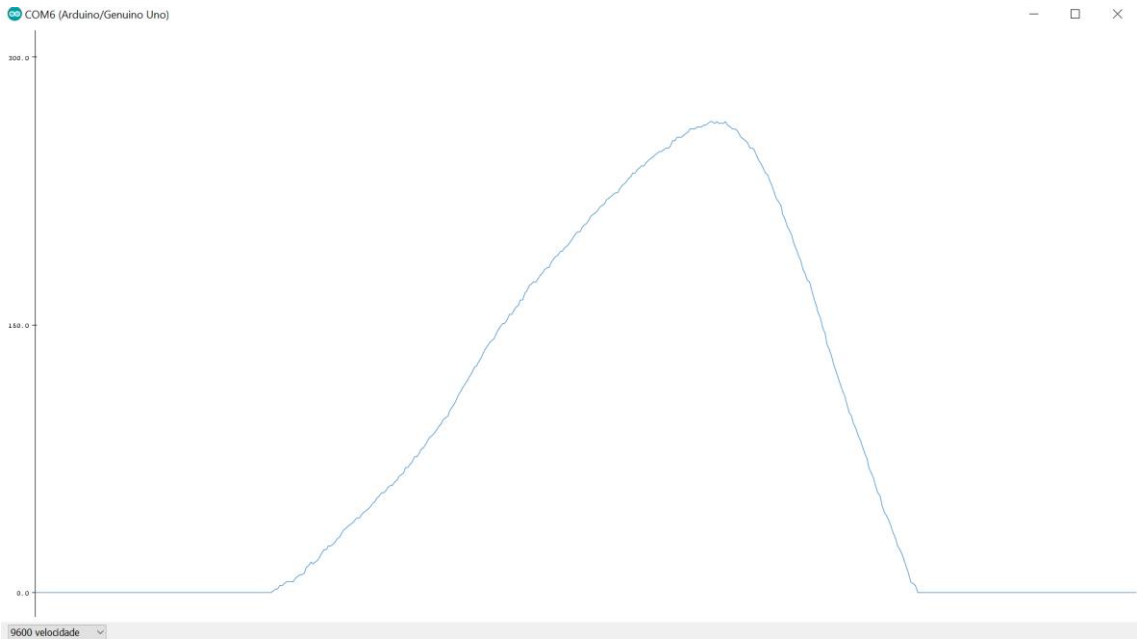
```
void setup() {
  Serial.begin(9600);
}

void loop() {
  int sensorValue = analogRead(A0);   // lê o valor de entrada no pino A0:
  Serial.println(sensorValue);         // imprime o valor lido:
}
```

Plotter serial

Outra forma de visualizar os valores de leitura de sensores é utilizando o plotter serial.
Esta ferramenta, disponível a partir da versão 1.6.6. da IDE Arduino, cria gráficos em tempo real dos valores de leitura das entradas, e pode ser bastante útil para visualizar o comportamento de sensores.
Para acessá-la, após fazer o upload do seu código, vá até Ferramentas > Plotter Serial:



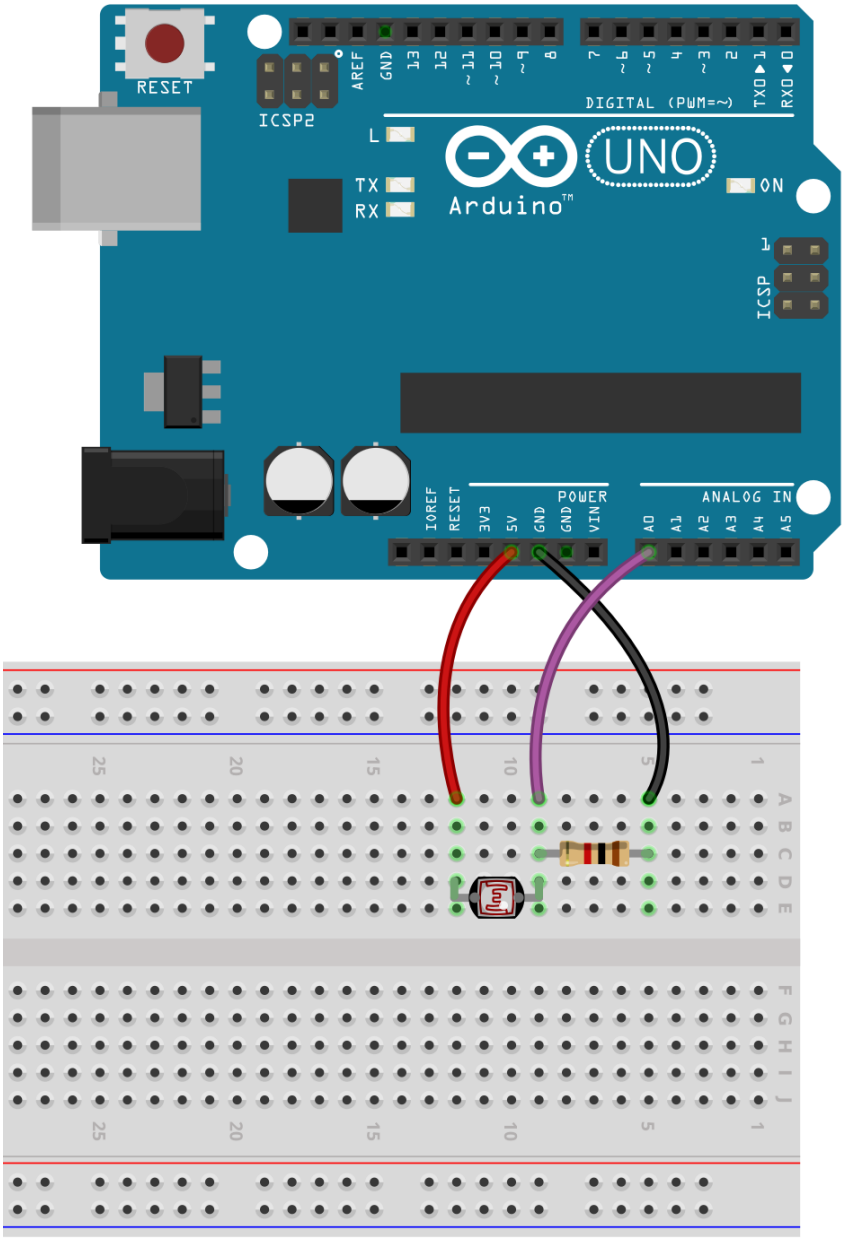
Se sua placa estiver conectada ao computador e seu programa tiver sido carregado corretamente, os valores de leitura começarão a ser plotados em um gráfico que é atualizado em tempo real:



Para experimentar e visualizar a saída do plotter serial você pode utilizar o mesmo circuito e código do exemplo anterior (com o potenciômetro).

Usando um LDR

Da mesma forma que você utilizou o potenciômetro no exemplo anterior, poderia ter utilizado qualquer sensor analógico no seu circuito.
No circuito abaixo, usamos um LDR para medir a intensidade luminosa. Observe o circuito abaixo: ele é muito similar ao do botão, com um resistor pull down.



Mantenha o mesmo código do exemplo do potenciômetro, e abra o plotter serial. Observe como os valores mudam à medida que a luminosidade que incide no LDR varia (tampe o LDR com a palma de sua mão, aguarde um pouco e observe o resultado no plotter serial).
Se você tiver outros sensores, faça testes também com eles. E, se tiver dúvidas sobre o circuito mais adequado para eles, busque informações na Internet e compartilhe o que aprendeu no fórum.
Além disso, você também pode conectar um botão ao seu circuito e observar no plotter o que ocorre quando o pressionamos.

Usando variáveis

Pense o que ocorreria no exemplo anterior caso quiséssemos mudar o pino no qual o LED está conectado. Teríamos de alterar nosso código em todos os lugares que o número "13" está escrito.

Agora considere o exemplo abaixo:

```
int LED = 13;
int intervalo = 1000;

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

void loop() {
  digitalWrite(LED, HIGH);
  delay(intervalo);
  digitalWrite(LED, LOW);
  delay(intervalo);
}
```

Nesse trecho, LED e intervalo são duas variáveis que armazenam respectivamente os valores 13 e 1000.
Variáveis são áreas de memória da Arduino na qual armazenamos dados e as quais atribuímos nomes.
No nosso exemplo, o valor 13 está armazenado em um espaço da memória ao qual demos o nome "LED", e o valor 1000 está armazenado em um espaço de memória a qual demos o nome "intervalo".
Há diversos motivos pelos quais usamos variáveis em nosso código. Os principais são:

1) o uso de variáveis torna mais fácil alterar e fazer ajustes no nosso código. Se tivéssemos mudado o LED para o pino 9, bastaria alterar a linha que atribui o valor à variável:

```
int LED = 9;
```

2) com variáveis, nosso código fica mais fácil de ler. Ao atribuir nomes significativos à valores em nossos programas nós facilitamos sua compreensão à quem está tentando entender o que ele faz.
O comando digitalWrite(LED, HIGH) tem o significado mais claro ("escrever HIGH em LED") do que digitalWrite(13, HIGH) ("escrever HIGH em 13").
Claro que a melhora na legibilidade depende dos nomes que escolhemos para as variáveis. Se tivéssemos chamado a variável de X (para matar a saudade das nossas aulas de álgebra (^‿^)), não estaríamos acrescentando significado ao nosso programa. Por isso, escolha seus nomes com cuidado! No nosso exemplo, dêmos o nome "LED" porque planejamos colocar nesta variável o número do pino da Arduino onde conectamos o LED.

3) Usamos variáveis para guardar valores que obtivemos agora, mas que vamos usar mais tarde em nosso código. É o que fizemos ao ler o valor de um botão para usar mais tarde:

```
int estadoBotao = 0;

[...]

void loop() {
  estadoBotao = digitalRead(botao);
  if (estadoBotao == 1) {
    digitalWrite(led, 1);
```

Mas como criar uma variável?

A sintaxe do comando para criar ("declarar" é o nome que usamos normalmente) e atribuir o valor a uma variável tem três partes:

```
<tipo da variável> <nome da variável> = <valor inicial da variável>;
```

Por exemplo:

```
int estadoBotao = 0;
```

Usamos int pois nossa variável será um número inteiro. Variáveis do tipo int são usadas para armazenar valores inteiros entre -32768 e 32767.

Referências:

- Material integralmente extraído e adaptado do curso **Programação Física**, da plataforma 'Code IoT', criado em parceria com a Samsung e LSI-TEC Escola Politécnica da USP - https://codeiot.org.br/courses/course-v1:LSI-TEC+IOT104+2020_O2/about - Acessado em 10/10/2020.
    - Licença disponível em https://creativecommons.org/licenses/by-nc-nd/4.0/deed.pt_BR - Acessado em 10/10/2020.
    - Plano de aula disponível em https://codeiot.org.br/assets/courseware/v1/33ea0953f26682e20a912ac7cb685884/asset-v1:LSI-TEC+IOT104+2018_S2+type@asset+block/Plano_de_Aula_Progamacao_Fisica_Semana_2.pdf - Acessado em 10/10/2020.