

Programação em Linguagem C

Unidade I

Nesta unidade aprenderemos:

- ✓ escrever o código no computador e transferir para o Arduino.
- ✓ ligar e desligar um pino;
- ✓ utilizar o delay (tempo);
- ✓ ler um pino para saber seu estado (Se é "0" ou "1")
- ✓ estruturas de decisão if/else;
- ✓ operações lógicas.

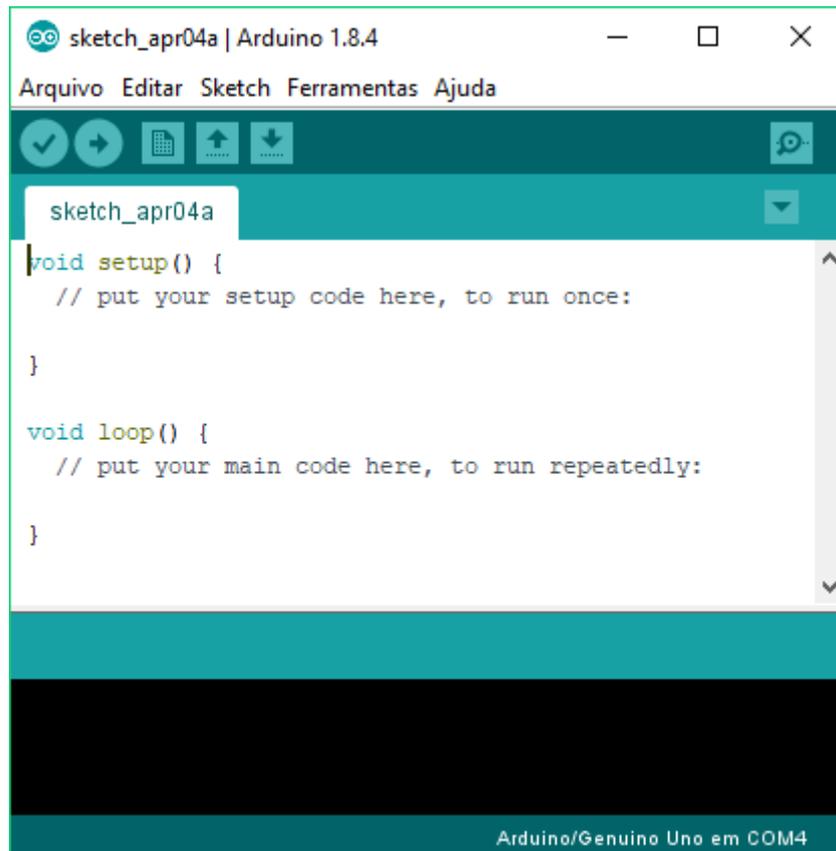
Operadores Relacionais

| Símbolo | Significado |
|---------|----------------|
| < | Menor |
| > | Maior |
| <= | Menor ou igual |
| >= | Maior ou igual |
| == | Igual |
| != | Diferente |

Operadores Lógicos

| Símbolo | Significado |
|---------|----------------|
| && | Conjunção (e) |
| | Disjunção (ou) |
| ! | Negação (não) |

Antes de tudo, precisamos saber que todo programa de Arduino é composto basicamente por duas partes:



```
sketch_apr04a
void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
}

Arduino/Genuino Uno em COM4
```

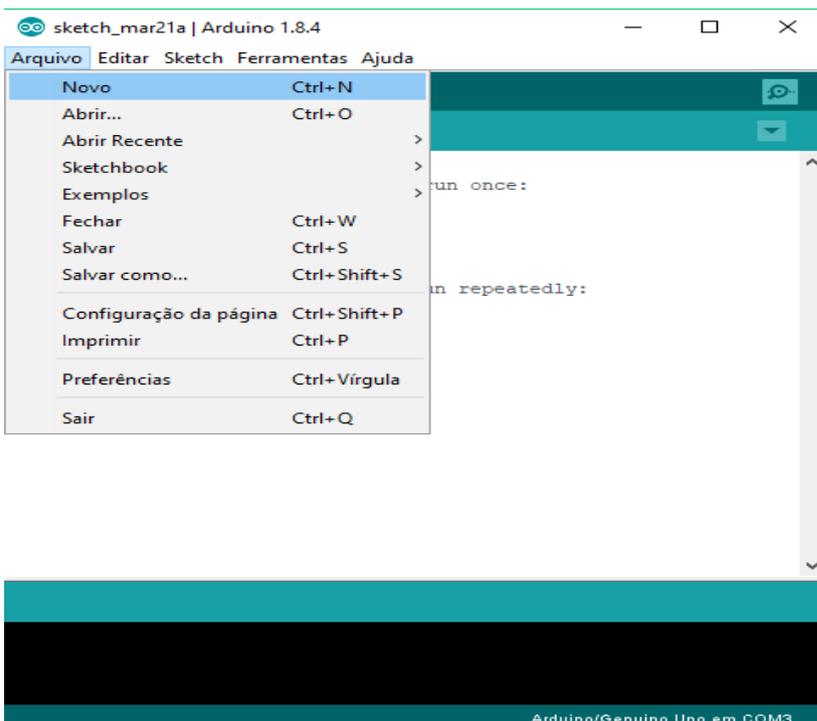
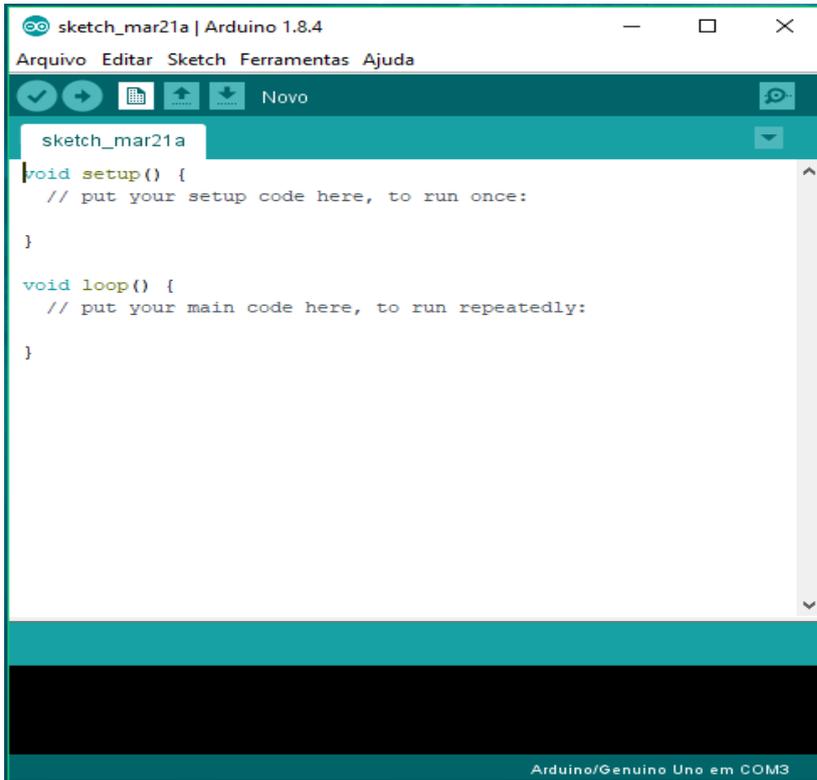
A primeira parte (**void setup**) é executada apenas uma vez, quando ligamos o Arduino, ou quando apertamos o botão reset. Já a segunda parte (**void loop**) será executada infinitas vezes, sequencialmente até que o Arduino seja desligado.

Dito isto, vamos aos exemplos.

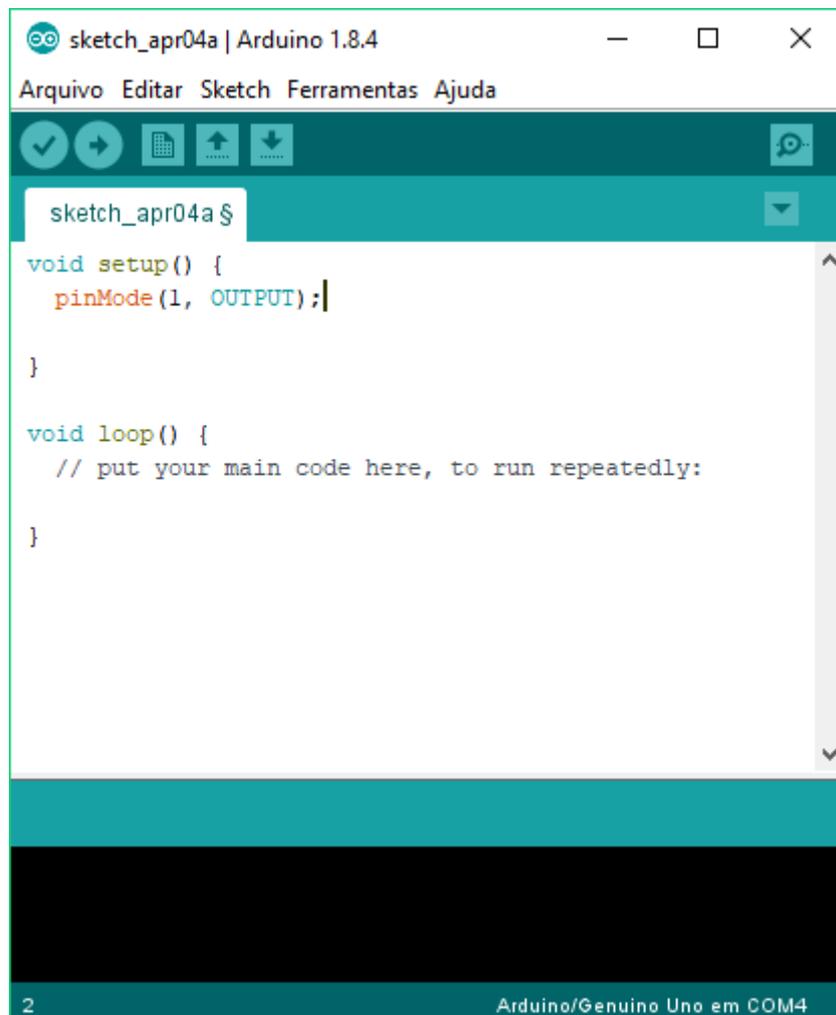
Exemplo 1: Como ligar (acender) e desligar (apagar) um LED que está conectado ao pino 1 do Arduino, com intervalos de 1 segundo.

Obs.: Neste exemplo estamos supondo que o terminal catodo do LED está conectado ao GND (“terra ou 0V”).

Passo 1: Iniciar um novo programa. Para isto, podemos utilizar o ícone “novo” na barra de ferramentas ou ir no menu Arquivo e selecionar novo.



Passo 2: Configurar os pinos do Arduino, se estes serão usados como entrada ou saída. Para isto, utilizamos a função **pinMode**. Observe a figura abaixo:



```
sketch_apr04a $
void setup() {
  pinMode(1, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

Onde: **pinMode** = modo do pino;

1 = pino físico que está sendo configurado;

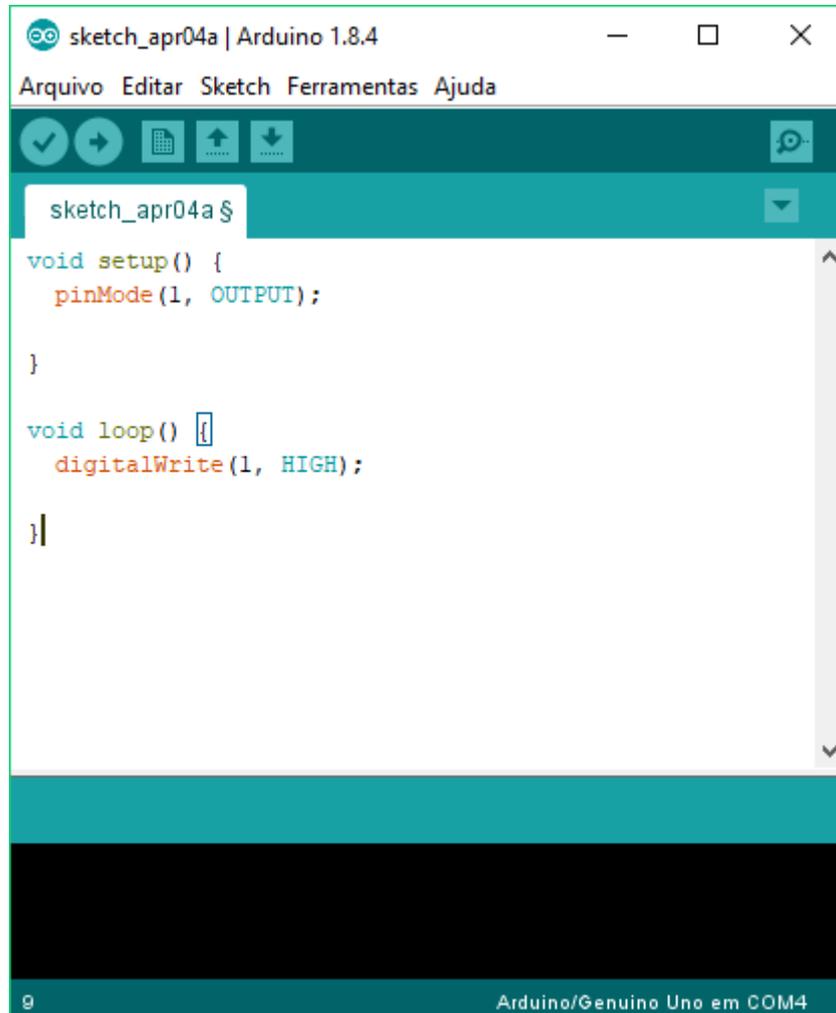
OUTPUT = saída.

Note que esta configuração está em **void setup**, pois, precisa ser executada uma única vez.

Nota.: um pino é definido como entrada quando este recebe um sinal externo e definido como saída quando envia um sinal. Exemplo:

- ✓ um pino conectado a um LED será configurado como **saída**, pois, este **pino enviará um sinal** para o LED acender ou apagar;
- ✓ um pino conectado a uma chave será configurado como **entrada**, pois, este pino **receberá um sinal** referente à chave aberta ou fechada.

Passo 3: como queremos que o pino apenas ligue e desligue, este será configurado como pino digital, pois, um pino digital somente pode assumir apenas um entre dois estados (ligado = 5V ou desligado = 0V). Para isto, utilizamos a função **digitalWrite**.

The image shows a screenshot of the Arduino IDE interface. The window title is "sketch_apr04a | Arduino 1.8.4". The menu bar includes "Arquivo", "Editar", "Sketch", "Ferramentas", and "Ajuda". The toolbar contains icons for a checkmark, a right arrow, a keyboard, an upload button, a download button, and a refresh button. The main editor area shows the following code:

```
sketch_apr04a $  
void setup() {  
  pinMode(1, OUTPUT);  
}  
  
void loop() {  
  digitalWrite(1, HIGH);  
}
```

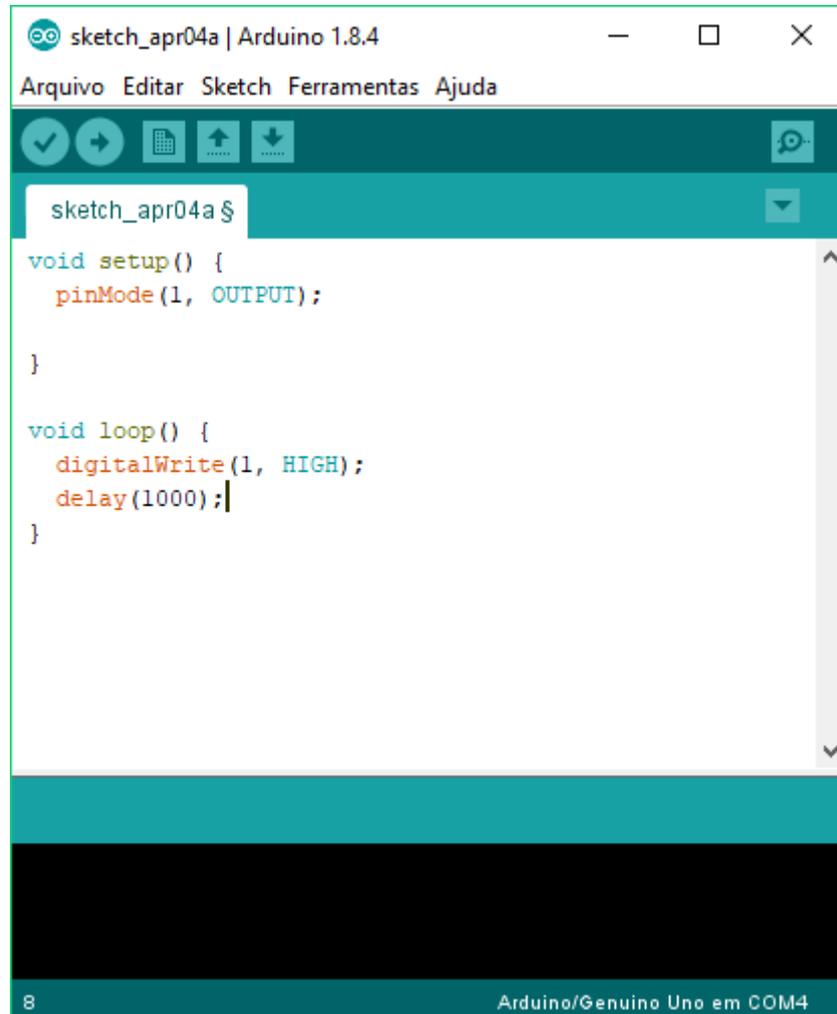
The status bar at the bottom indicates "9" on the left and "Arduino/Genuino Uno em COM4" on the right.

Onde: **digitalWrite** = escreva ("no pino digital");

1 = pino físico (conectado ao LED) que está recebendo o comando;

HIGH = alto (5V).

Passo 4: como o pino 1 foi colocado em nível alto durante o passo 3, este deve permanecer assim durante 1 segundo (conforme solicitado no exemplo). Para isto, utilizamos a função **delay**.

A screenshot of the Arduino IDE interface. The window title is "sketch_apr04a | Arduino 1.8.4". The menu bar includes "Arquivo", "Editar", "Sketch", "Ferramentas", and "Ajuda". The toolbar contains icons for check, run, upload, and download. The sketch editor shows the following code:

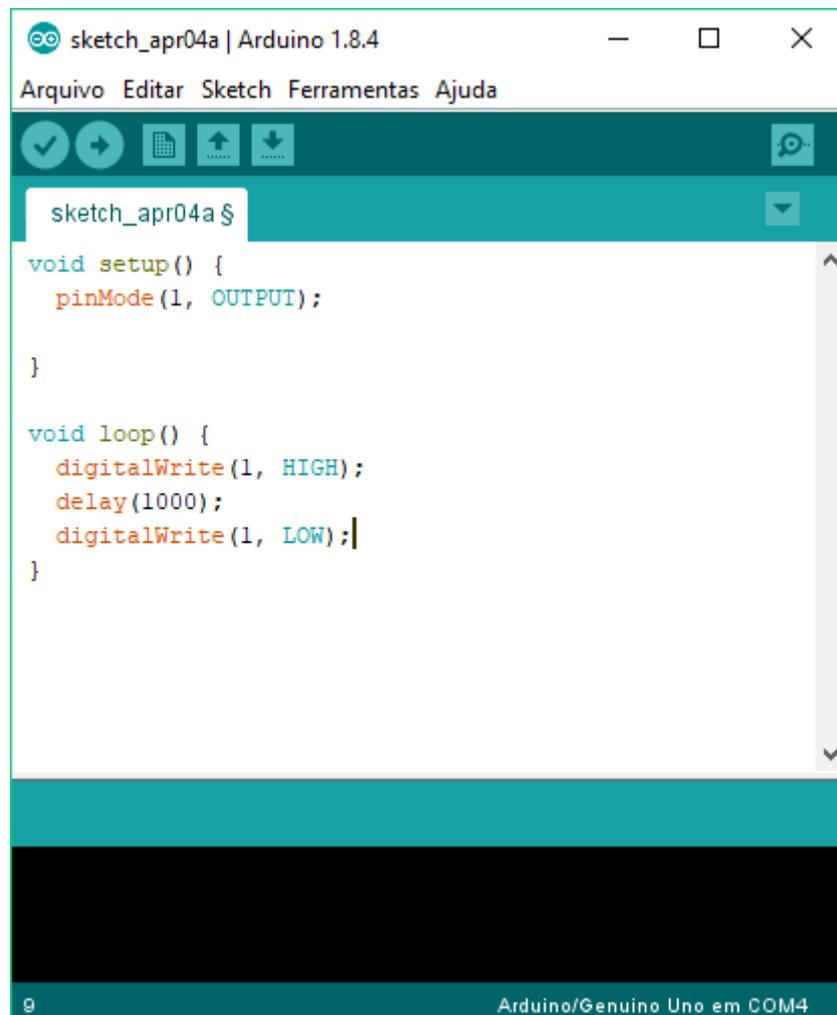
```
void setup() {  
  pinMode(1, OUTPUT);  
}  
  
void loop() {  
  digitalWrite(1, HIGH);  
  delay(1000);  
}
```

The status bar at the bottom indicates "8" and "Arduino/Genuino Uno em COM4".

Onde: **delay** = atraso;

1000 = 1s (o tempo é informado em ms, ou seja, 1000ms = 1s)

Passo 5: Durante os passos 3 e 4, o pino 1 foi colocado em nível alto e aguardado um tempo de 1 segundo. Agora vamos colocá-lo em nível baixo (0V).

A screenshot of the Arduino IDE interface. The window title is "sketch_apr04a | Arduino 1.8.4". The menu bar includes "Arquivo", "Editar", "Sketch", "Ferramentas", and "Ajuda". The toolbar shows icons for saving, running, uploading, and downloading. The sketch name "sketch_apr04a" is displayed in a dropdown menu. The main text area contains the following code:

```
void setup() {  
  pinMode(1, OUTPUT);  
}  
  
void loop() {  
  digitalWrite(1, HIGH);  
  delay(1000);  
  digitalWrite(1, LOW);  
}
```

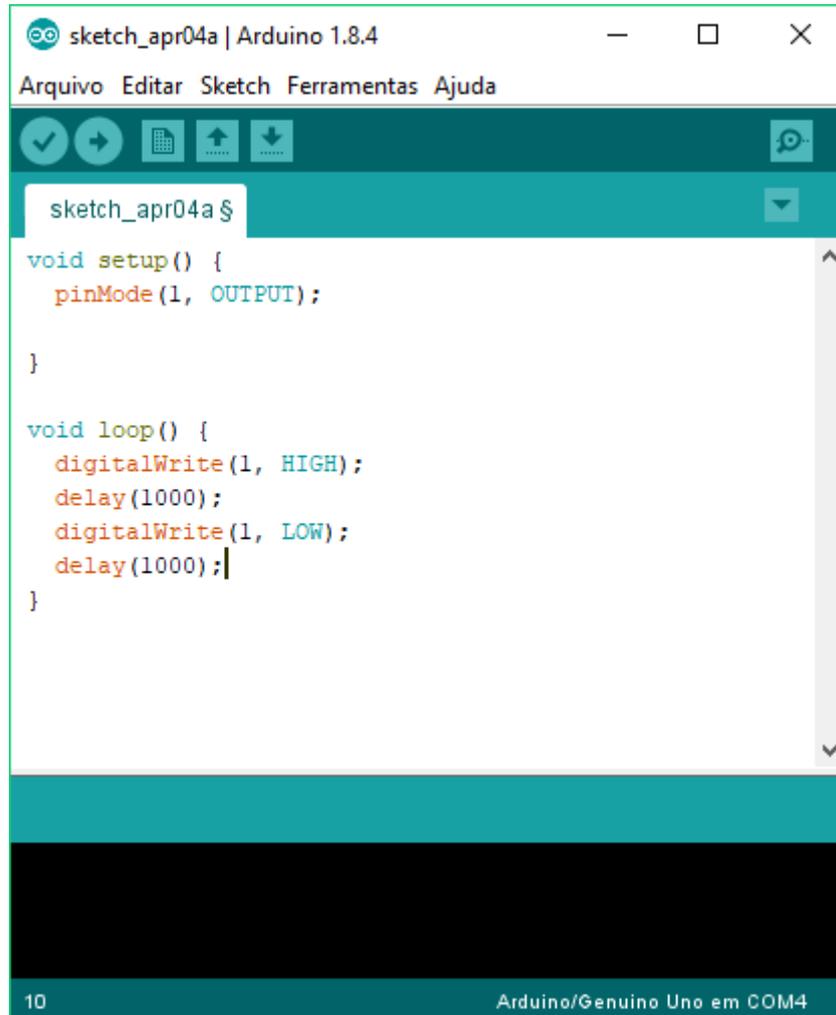
The status bar at the bottom indicates "9" and "Arduino/Genuino Uno em COM4".

Onde: **digitalWrite** = escreva ("no pino digital");

1 = pino físico (conectado ao LED) que está recebendo o comando;

LOW = baixo (0V).

Passo 6: conforme descrito no exemplo, devemos aguardar 1 segundo.

A screenshot of the Arduino IDE interface. The window title is "sketch_apr04a | Arduino 1.8.4". The menu bar includes "Arquivo", "Editar", "Sketch", "Ferramentas", and "Ajuda". The toolbar shows icons for check, run, upload, and download. The main editor area contains the following code:

```
sketch_apr04a $  
void setup() {  
  pinMode(1, OUTPUT);  
}  
  
void loop() {  
  digitalWrite(1, HIGH);  
  delay(1000);  
  digitalWrite(1, LOW);  
  delay(1000);  
}
```

The status bar at the bottom shows "10" on the left and "Arduino/Genuino Uno em COM4" on the right.

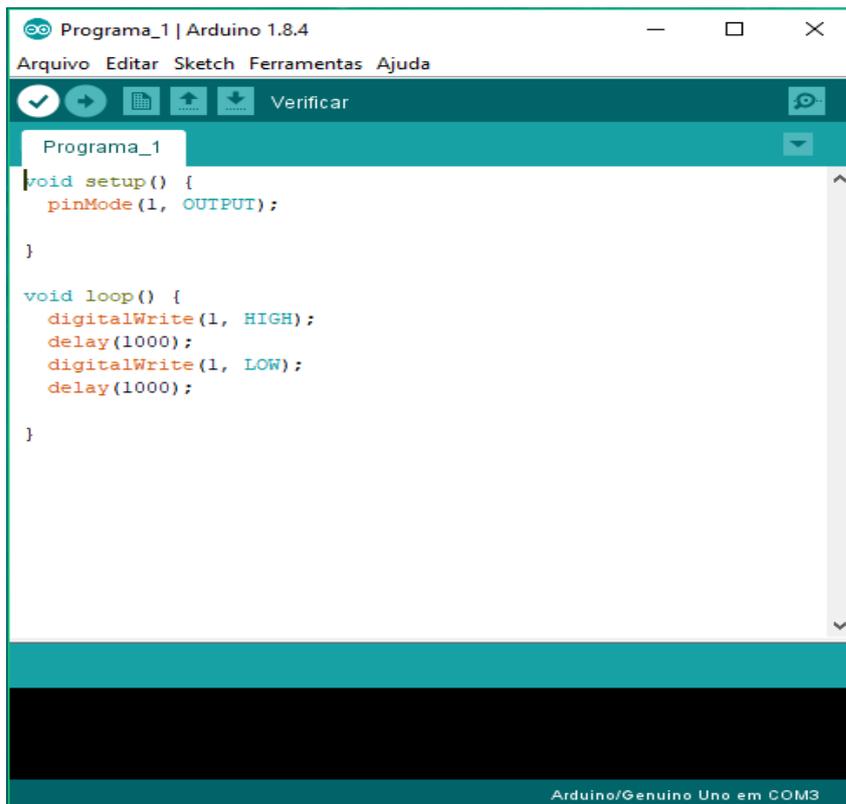
Onde: **delay** = atraso;

1000 = 1s (lembrando que o tempo é informado em ms, ou seja, 1000ms = 1s)

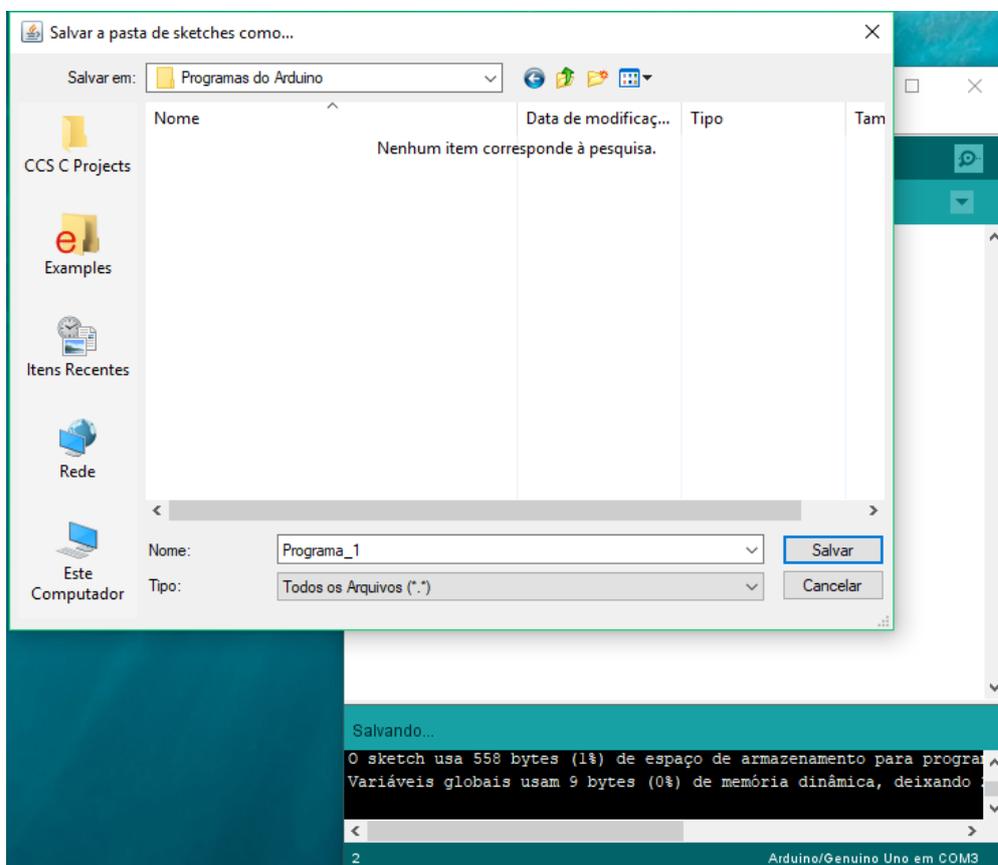
Com isto finalizamos o programa. Observe que o programa principal foi digitado em **void loop**, ou seja, será executado infinitas vezes, sequencialmente até que o Arduino seja desligado.

Resultado, o LED conectado ao pino 1 do Arduino, ficará oscilando entre ligado e desligado, durante intervalos de 1 segundo infinitas vezes até que o Arduino seja desligado.

Passo 7: Copilar e salvar o programa. Clique no ícone verificar, conforme imagem;

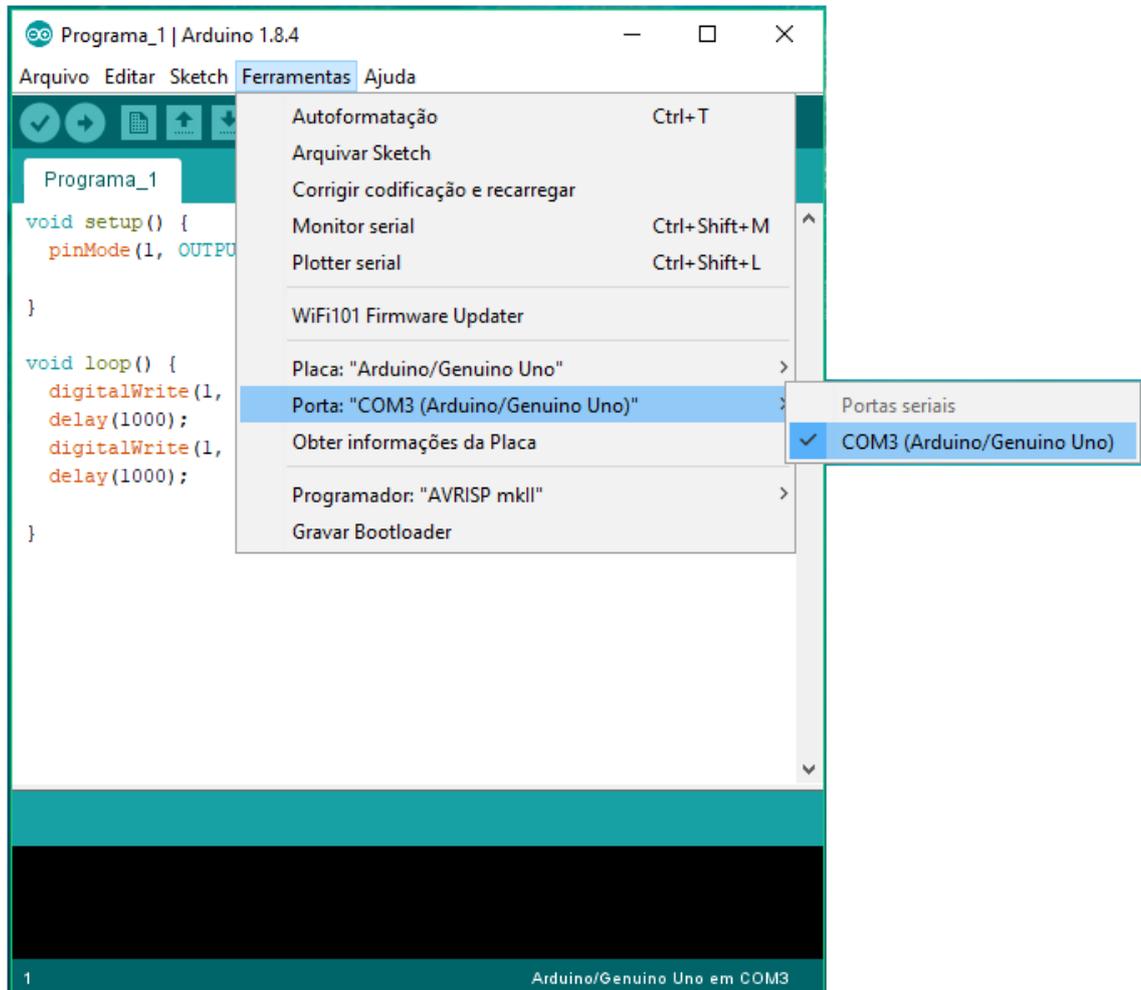


Com isto, abrirá uma nova janela onde podemos inserir um nome e selecionar o local para salvar o projeto. Observe a imagem abaixo:



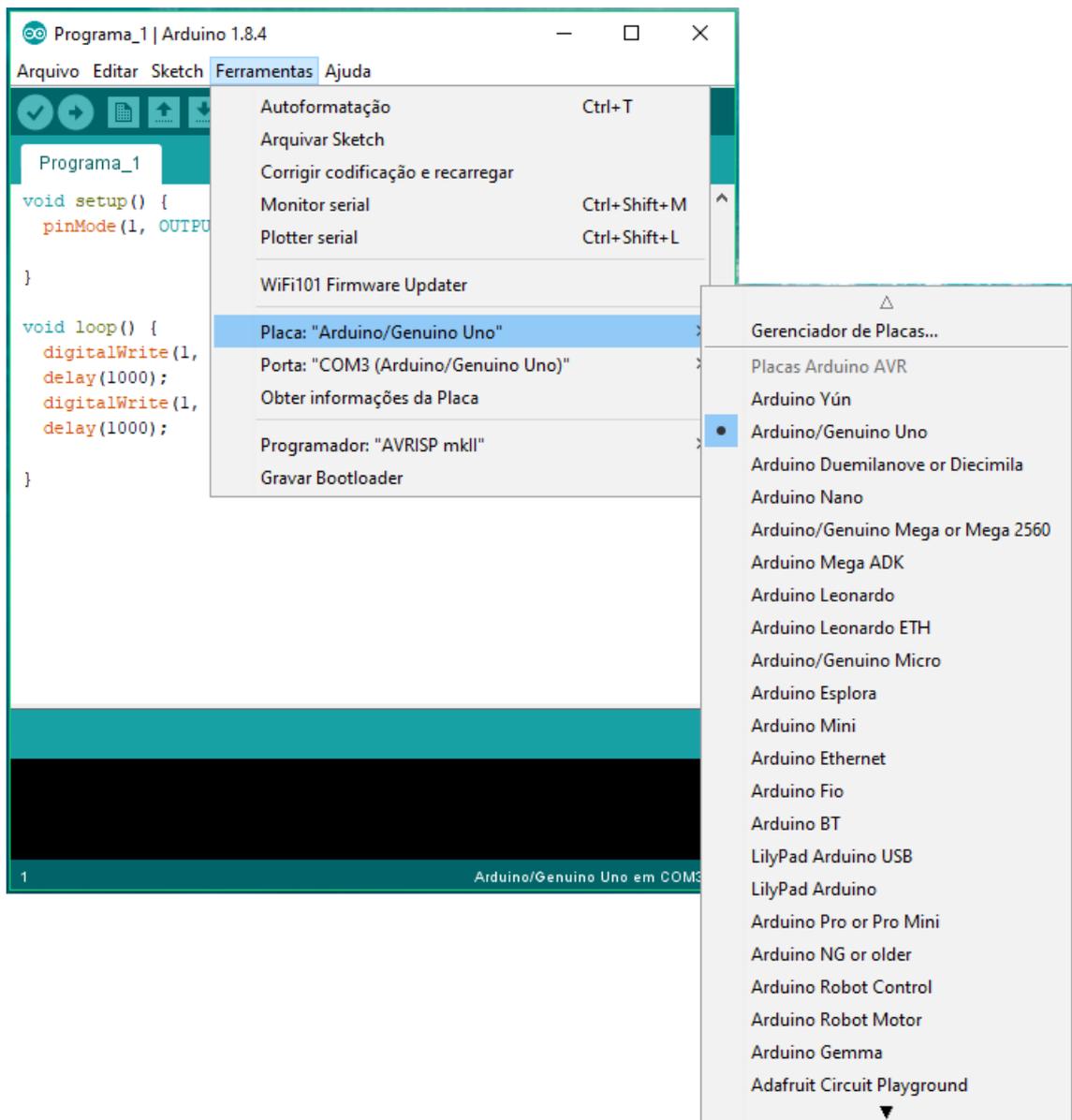
Passo 8: Transferir o programa do computador para o Arduino.

Passo 8.1.: Após conectar o Arduino ao computador por meio do cabo USB, vá no **menu Ferramentas, Porta:** e selecione a COM onde aparecer o Arduino.



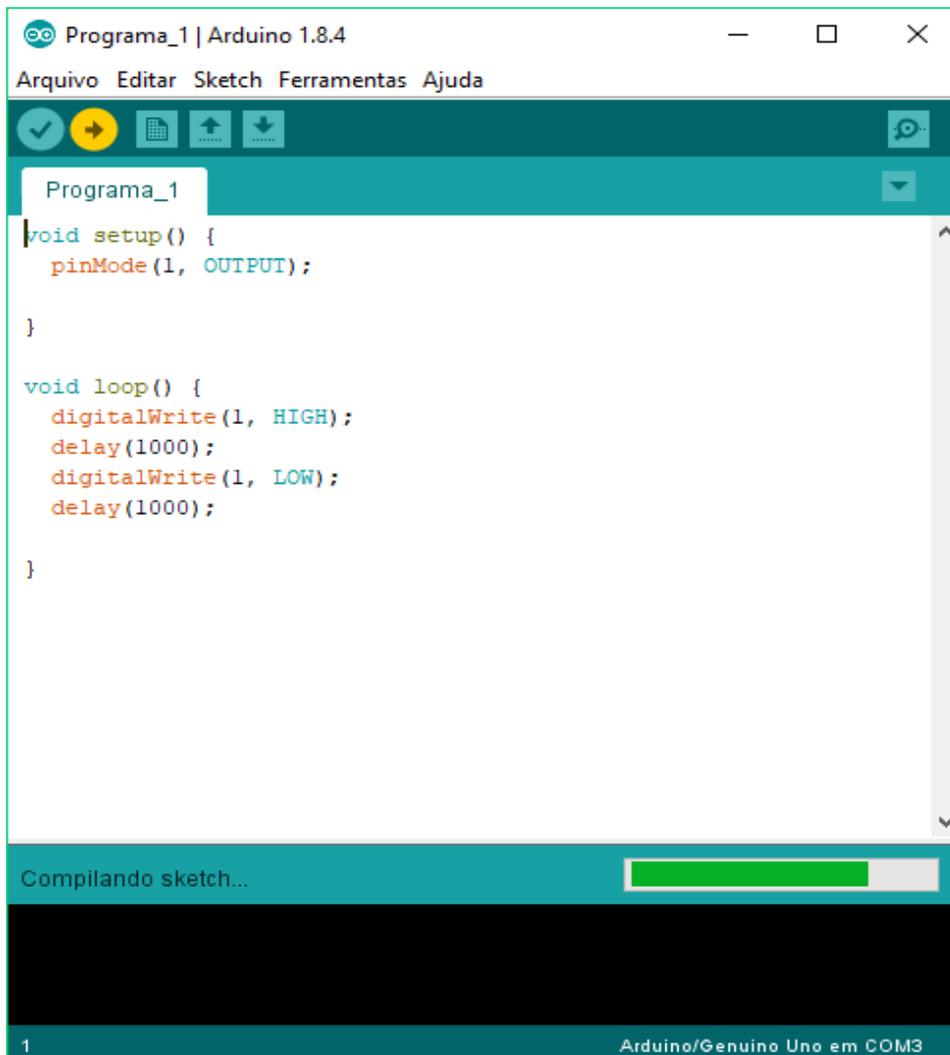
Note que neste exemplo o computador reconheceu o Arduino na porta de comunicação **COM3**.

Passo 8.2.: Como existem muitos modelos de Arduino, ainda no menu **Ferramentas**, opção **Placa**: Selecione o modelo correto do seu Arduino (caso necessário, já que o próprio computador faz o reconhecimento do dispositivo).



Neste exemplo, foi utilizado o Arduino Uno.

Passo 8.3.: Após realizar os passos 8.1 e 8.2, clique na opção **carregar** para transferir o programa para o Arduino.

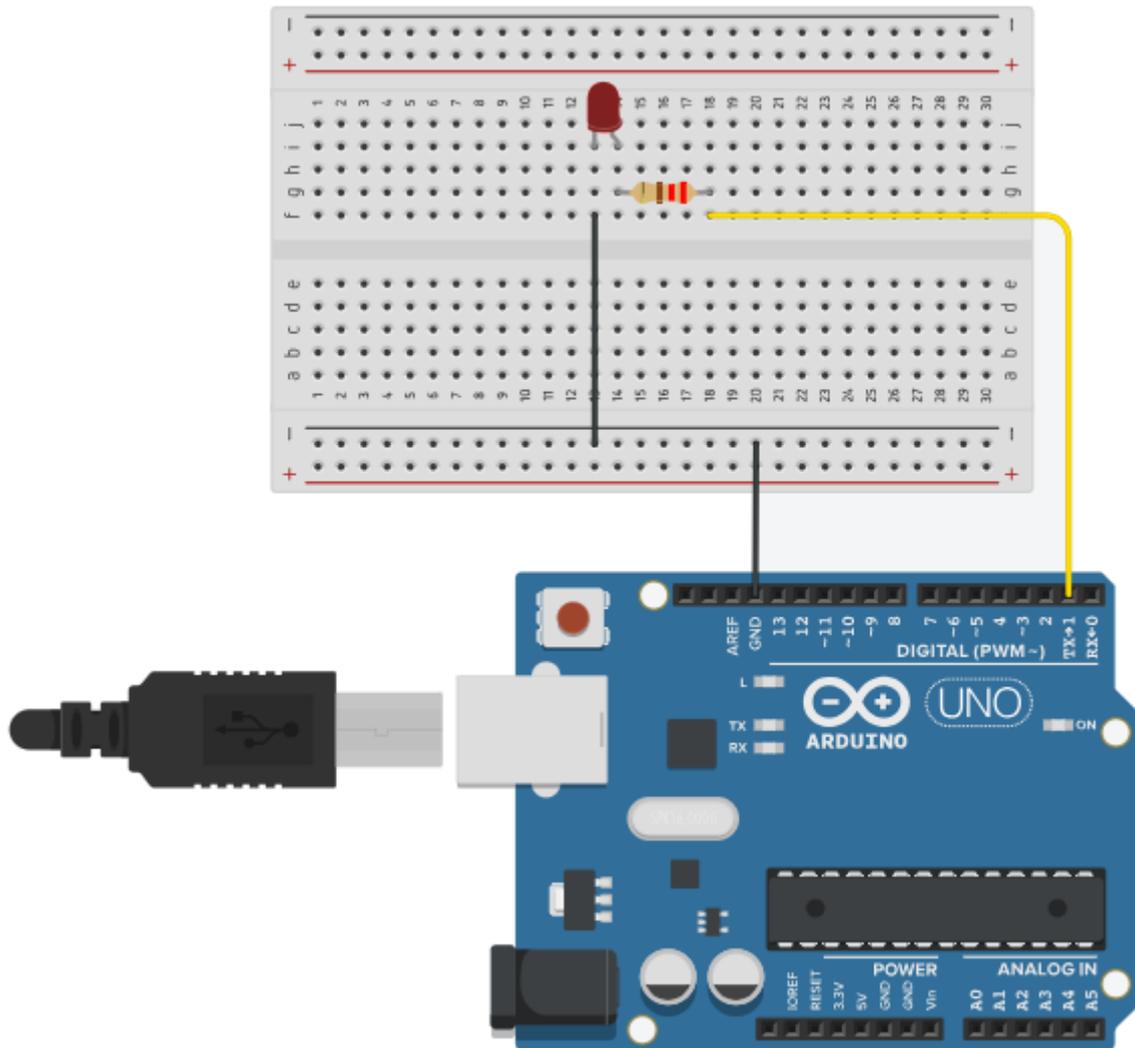


Observe que imediatamente aparecerá uma barrinha verde no canto inferior direito da janela, o processo de transferência estará concluído quando esta desaparecer.

Caso haja algum erro de digitação do programa ou comunicação, o mesmo será indicado na parte inferior da janela. Conforme mostrado abaixo:



Montagem



Exercício 1: Faça um semáforo (somente veículo).

Funcionamento.:

- O vermelho deve ser a primeira cor a ser ligada;
- 6 segundos de verde, 2 segundos de amarelo e 4 segundos de vermelho.

Resolução:

```

Exercicio_1 | Arduino 1.8.4
Arquivo Editar Sketch Ferramentas Ajuda

Exercicio_1 §

/*****
 * Este programa refere-se à um semáforo simples.
 * Onde o semáforo deve ser iniciado pela cor Vermelha.
 * Sequência: vermelho, verde, amarelo
 * Tempos: 6s de verde, 2s de amarelo e 4s de vermelho.
 *****/

void setup() { //rotina executada apenas uma vez
  pinMode(0, OUTPUT); //pino 0, saída, LED verde
  pinMode(1, OUTPUT); //pino 1, saída, LED amarelo
  pinMode(2, OUTPUT); //pino 2, saída, LED vermelho
}

void loop() { //repetição infinita do programa
  digitalWrite(2, HIGH); //liga vermelho
  delay(4000); //aguarda 4 segundos

  digitalWrite(2, LOW); //desliga vermelho
  digitalWrite(0, HIGH); //liga verde
  delay(6000); //aguarda 6 segundos

  digitalWrite(0, LOW); //desliga verde
  digitalWrite(1, HIGH); //liga amarelo
  delay(2000); //aguarda 2 segundo

  digitalWrite(1, LOW); //desliga amarelo
} //repete tudo a partir de void loop!

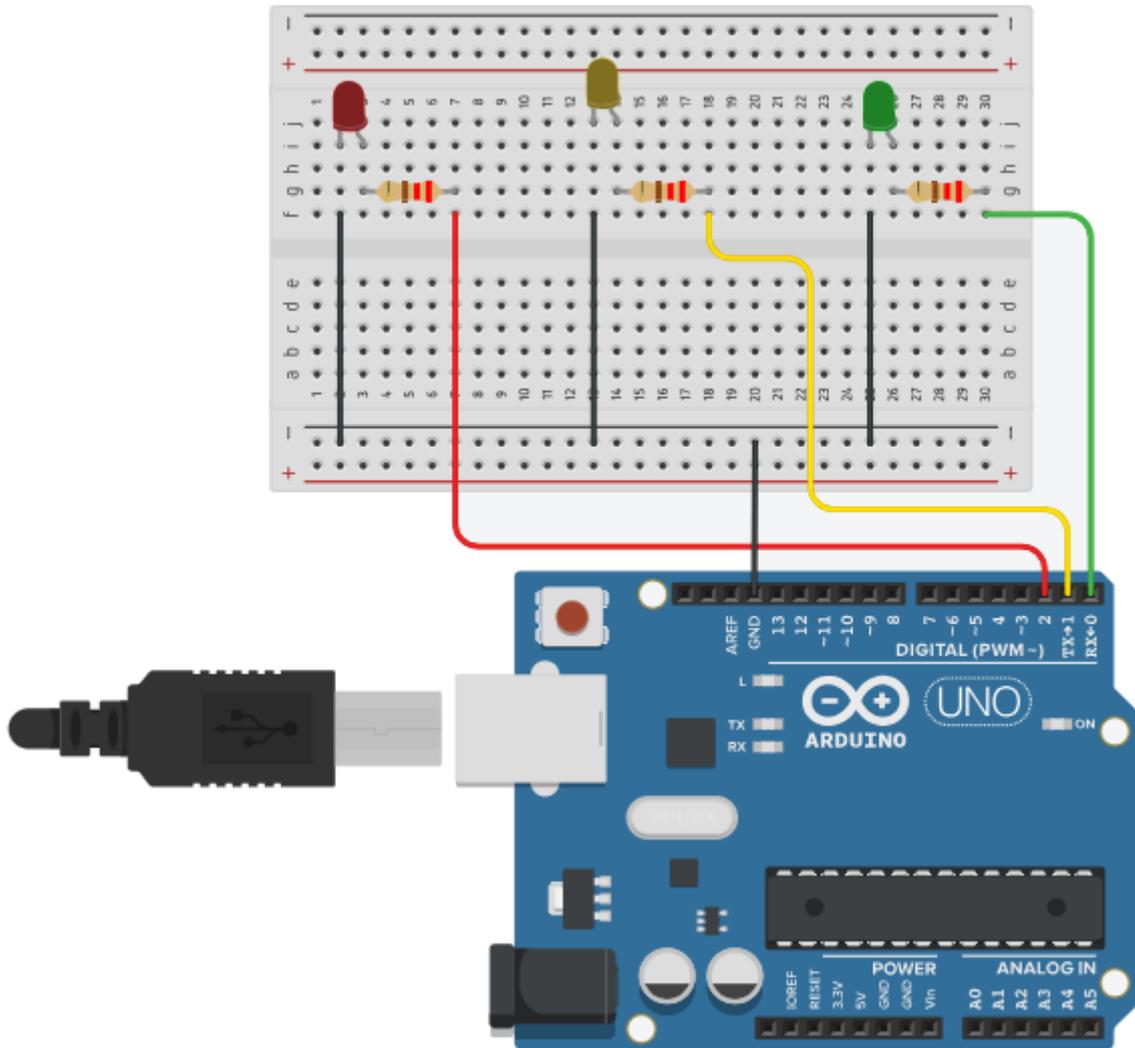
Compilação terminada.

O sketch usa 1030 bytes (3%) de espaço de armazenamento para programas. O máximo
Variáveis globais usam 9 bytes (0%) de memória dinâmica, deixando 2039 bytes

26 Arduino/Genuino Uno em COM4
  
```

Nota.: Foi utilizado `/*` para fazer o comentário inicial do programa e finalizado com `*/`. E utilizado `//` para fazer o comentário individual das linhas.

Montagem

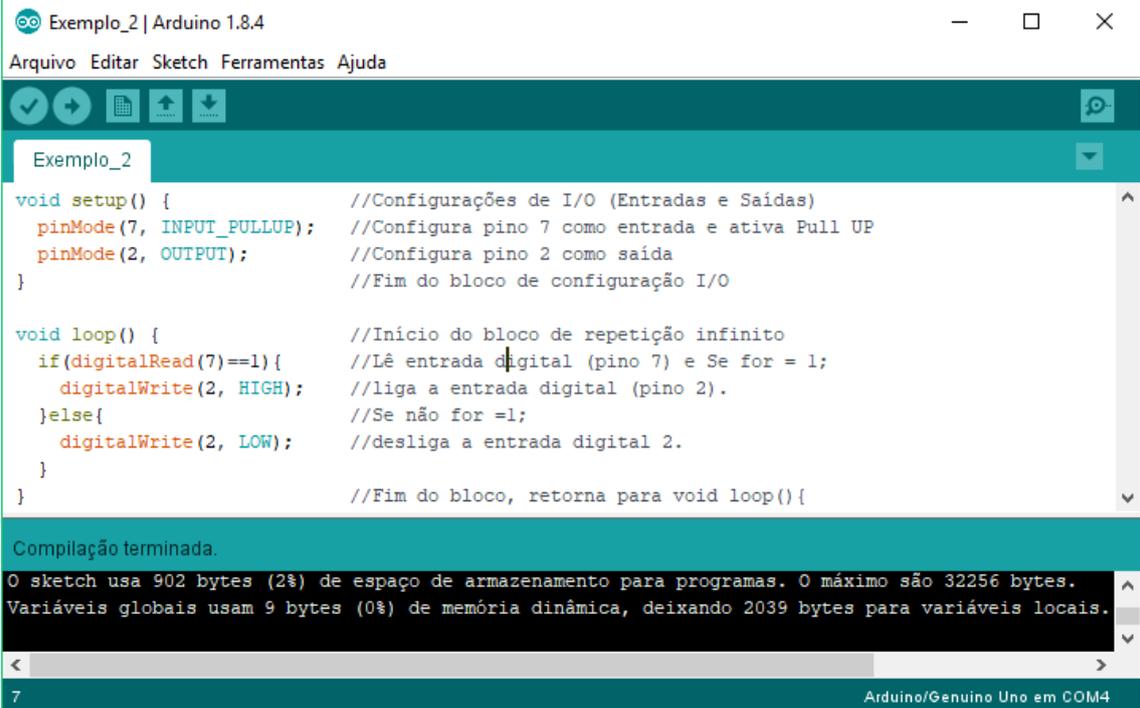


Exemplo 2: Controlar o acionamento de um LED através de um botão liga/desliga.

Dados.: Botão liga/desliga conectado ao pino 7 do Arduino, envia nível lógico 0 quando é pressionado. LED conectado ao pino 2, acende com nível lógico 1.

Funcionamento.:

- O LED deverá acender quando o botão estiver “fechado”;
- O LED deverá apagar quando o botão estiver “aberto”.



```

Exemplo_2 | Arduino 1.8.4
Arquivo Editar Sketch Ferramentas Ajuda

Exemplo_2
void setup() { //Configurações de I/O (Entradas e Saídas)
  pinMode(7, INPUT_PULLUP); //Configura pino 7 como entrada e ativa Pull UP
  pinMode(2, OUTPUT); //Configura pino 2 como saída
} //Fim do bloco de configuração I/O

void loop() { //Início do bloco de repetição infinito
  if(digitalRead(7)==1){ //Lê entrada digital (pino 7) e Se for = 1;
    digitalWrite(2, HIGH); //liga a entrada digital (pino 2).
  }else{ //Se não for =1;
    digitalWrite(2, LOW); //desliga a entrada digital 2.
  }
} //Fim do bloco, retorna para void loop(){

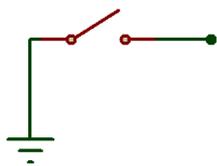
Compilação terminada.
O sketch usa 902 bytes (2%) de espaço de armazenamento para programas. O máximo são 32256 bytes.
Variáveis globais usam 9 bytes (0%) de memória dinâmica, deixando 2039 bytes para variáveis locais.
7 Arduino/Genuino Uno em COM4
  
```

Comentários.: A instrução `pinMode(7, INPUT_PULLUP)` utilizada neste exemplo, tem o seguinte significado:

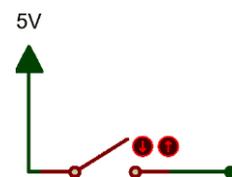
`pinMode(7, INPUT` pino 7 em modo entrada;

`_PULLUP)`. Para cima. Força o pino a permanecer em 5V quando não houver sinal no mesmo.

Observe as imagens:

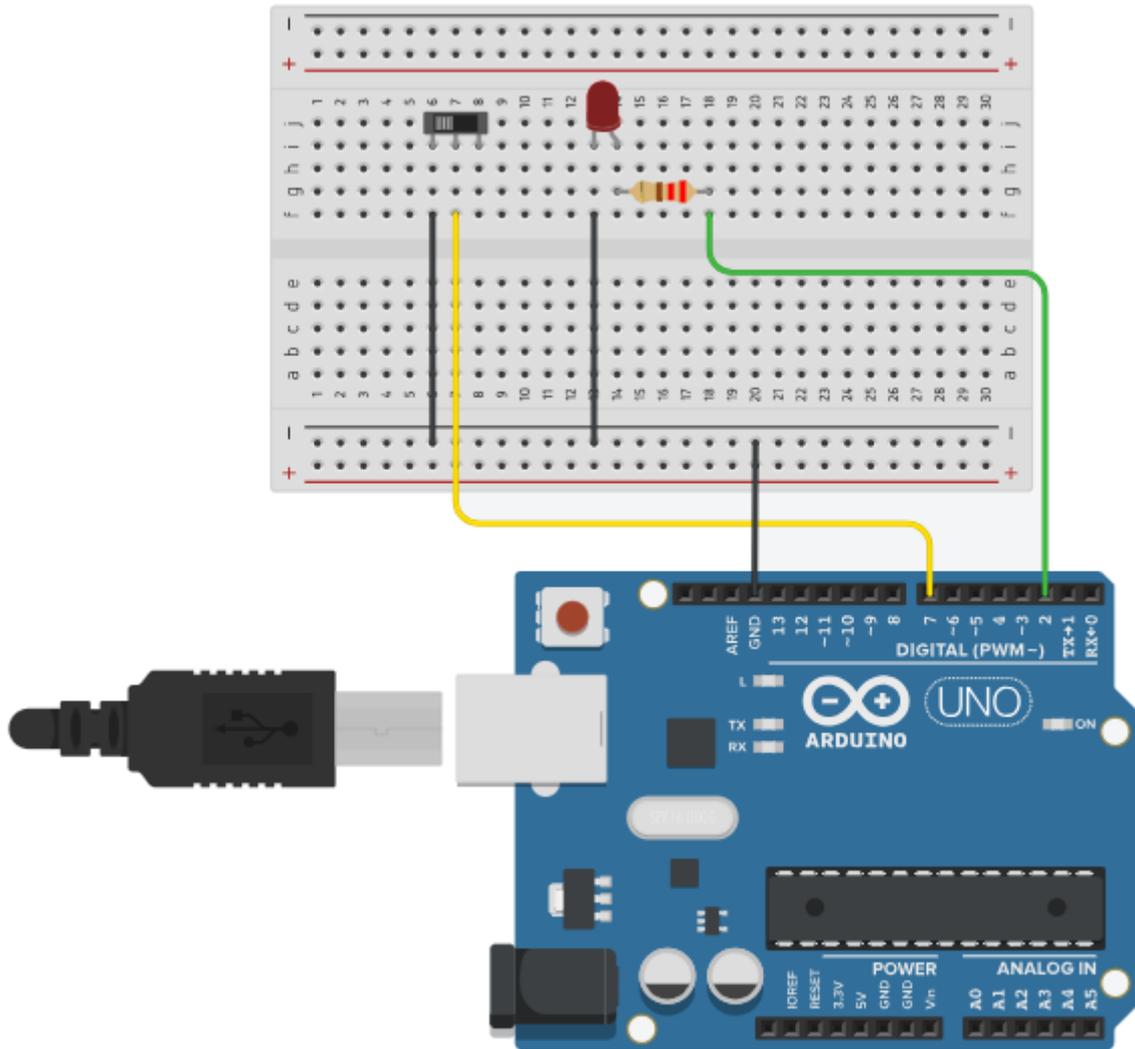


Uma chave envia sinal somente quando estiver fechada, logo, precisamos de resistores de **PULL UP** ou **PULL DOWN**, pois, para que o programa tome uma decisão, é preciso ter sinal 1 ou 0. **PULL UP** força nível 1 e **DOWN** nível 0.



Entretanto, não necessitamos inserir resistores de PULL UP no circuito, pois, o Arduino já os possui internamente e são ativados pela instrução `_PULLUP`.

Montagem

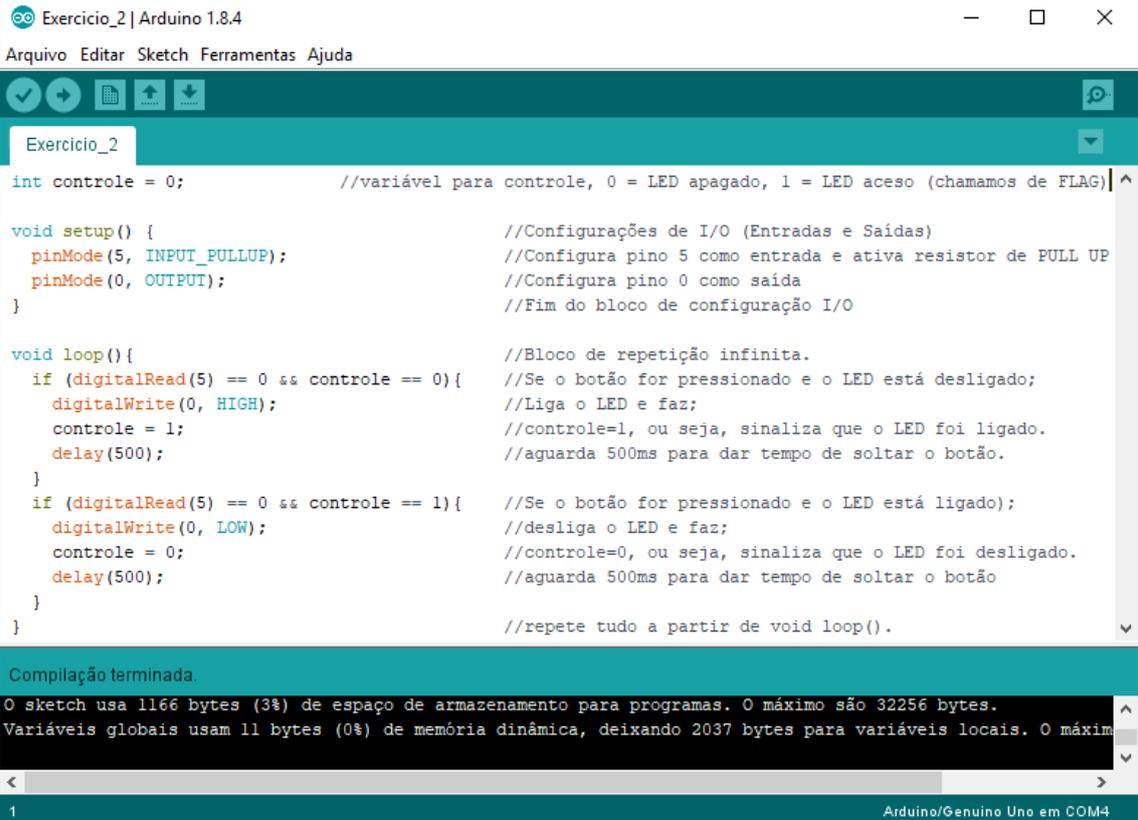


Exercício 2: Controlar o acionamento de um LED através de um botão de pulso.

Dados.: Botão de pulso conectado ao pino 5 do Arduino, envia nível lógico 0 quando pressionado. LED conectado ao pino 0 do Arduino, acende com nível lógico 1.

Funcionamento.:

- Inicialmente o LED deverá estar apagado;
- O LED deverá alterar seu estado (de ligado para desligado e vice-versa) toda vez que o botão for pressionado.



```
Exercicio_2 | Arduino 1.8.4
Arquivo Editar Sketch Ferramentas Ajuda

Exercicio_2

int controle = 0;           //variável para controle, 0 = LED apagado, 1 = LED aceso (chamamos de FLAG)

void setup() {
  pinMode(5, INPUT_PULLUP); //Configurações de I/O (Entradas e Saídas)
  pinMode(0, OUTPUT);       //Configura pino 5 como entrada e ativa resistor de PULL UP
                             //Configura pino 0 como saída
                             //Fim do bloco de configuração I/O
}

void loop(){
  //Bloco de repetição infinita.
  if (digitalRead(5) == 0 && controle == 0){ //Se o botão for pressionado e o LED está desligado;
    digitalWrite(0, HIGH);                 //Liga o LED e faz;
    controle = 1;                          //controle=1, ou seja, sinaliza que o LED foi ligado.
    delay(500);                             //aguarda 500ms para dar tempo de soltar o botão.
  }
  if (digitalRead(5) == 0 && controle == 1){ //Se o botão for pressionado e o LED está ligado;
    digitalWrite(0, LOW);                  //desliga o LED e faz;
    controle = 0;                         //controle=0, ou seja, sinaliza que o LED foi desligado.
    delay(500);                           //aguarda 500ms para dar tempo de soltar o botão
  }
}                                           //repete tudo a partir de void loop().

Compilação terminada.
O sketch usa 1166 bytes (3%) de espaço de armazenamento para programas. O máximo são 32256 bytes.
Variáveis globais usam 11 bytes (0%) de memória dinâmica, deixando 2037 bytes para variáveis locais. O máximo

1 Arduino/Genuino Uno em COM4
```

Montagem

