



Programação e IoT - Aula 03

Microcontrolador

INTRODUÇÃO AO ARDUINO

- Arduino é uma plataforma de ***open-source: hardware e software***
- Criado em 2005 (Massimo Banzi e outros colaboradores) para auxiliar o **ensino de eletrônica** para estudantes de design e artistas
- **Objetivos:** desenvolver protótipos com o menor custo possível
- Grande comunidade de desenvolvedores



PLATAFORMA DE DESENVOLVIMENTO ARDUINO

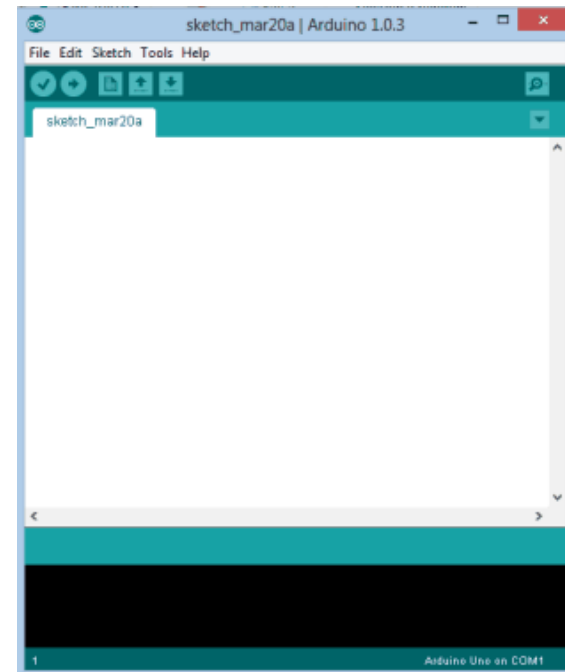
Hardware

Representado por uma placa de prototipagem



Software

Ambiente de desenvolvimento que possibilita desenvolver códigos e programar a placa



HARDWARE

Componente físico (placa).

- Existem diversas placas oficiais de Arduino
- De modo geral, tais placas possuem conectores que servem para interface com o mundo externo
- Os conectores estão ligados diretamente aos pinos de conexão do microcontroladores
- Já os pinos possuem funções específicas, como:
 - entrada e saída digital
 - entradas analógicas
 - saídas analógicas
 - alimentação
- Os pinos e suas respectivas funções dependem exclusivamente do microcontrolador presente na placa



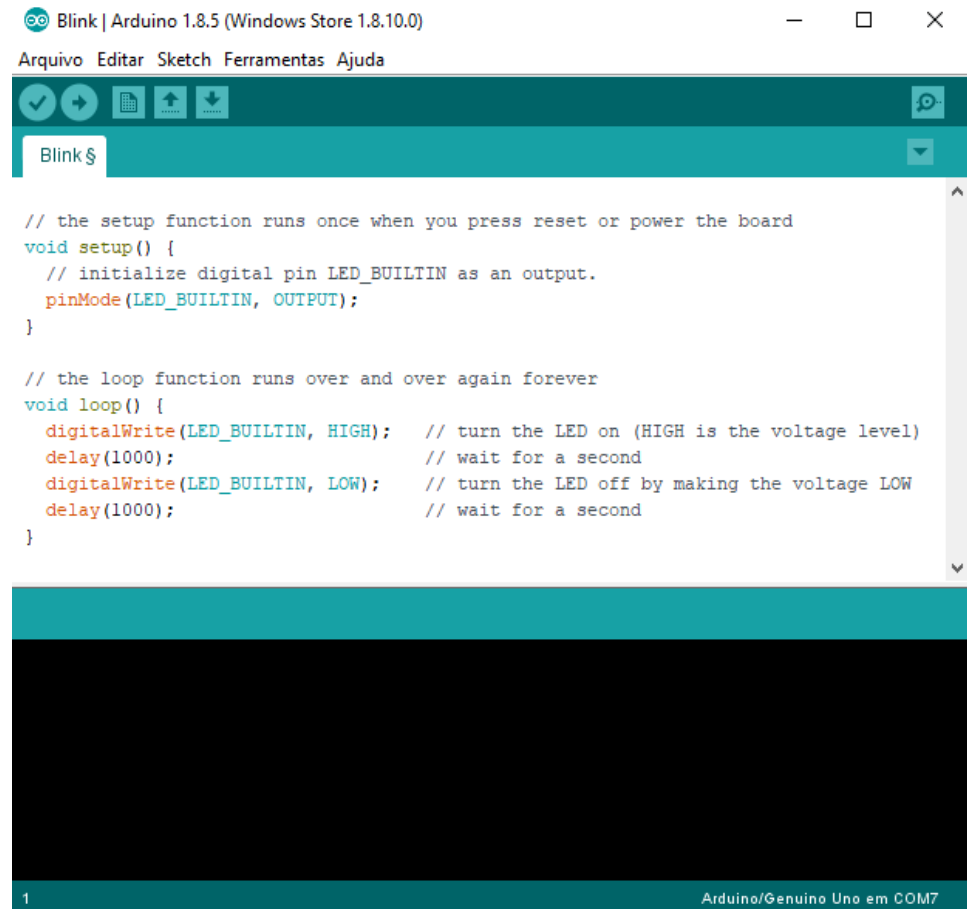
<https://www.arduino.cc/en/Main/Products>

SOFTWARE

Ambiente onde se realiza a programação.

- O ambiente de desenvolvimento integrado (IDE) do Arduino provê um framework para desenvolvimento de aplicações baseadas em microcontroladores
- Os programas criados no IDE são referenciados como **Sketch**
- A linguagem utilizada para programação é um subconjunto da linguagem C/C++
- A **biblioteca de funções** disponibilizada permite um alto grau de abstração do código em relação aos recursos utilizados no microcontrolador
- O IDE comunica-se com a placa do Arduino através de uma porta serial (COM virtual USB)

<https://www.arduino.cc/en/tutorial/sketch>

A screenshot of the Arduino IDE interface. The title bar reads "Blink | Arduino 1.8.5 (Windows Store 1.8.10.0)". The menu bar includes "Arquivo", "Editar", "Sketch", "Ferramentas", and "Ajuda". The toolbar shows icons for opening, saving, and running. The main text area displays the "Blink" sketch code, which includes comments and C++ code for setting up and looping the LED pin. The status bar at the bottom indicates "1" and "Arduino/Genuino Uno em COM7".

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

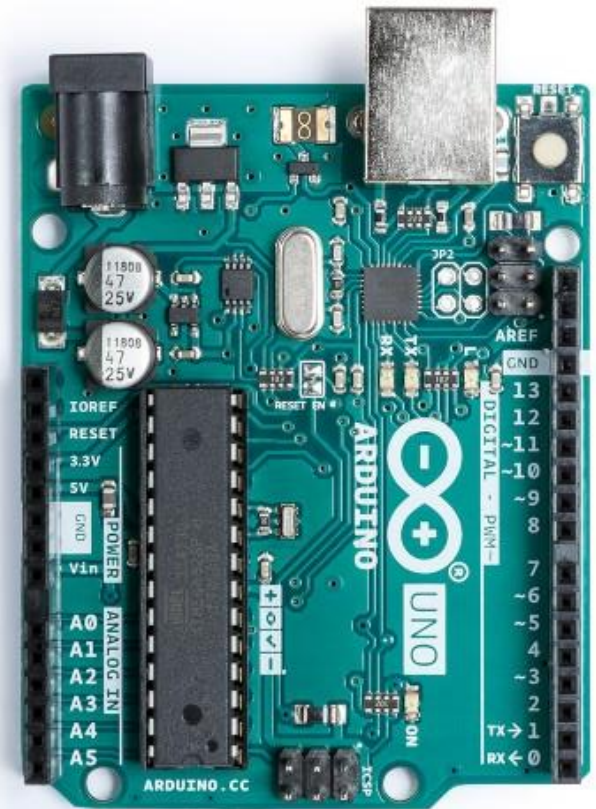
// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);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

Programação e IoT

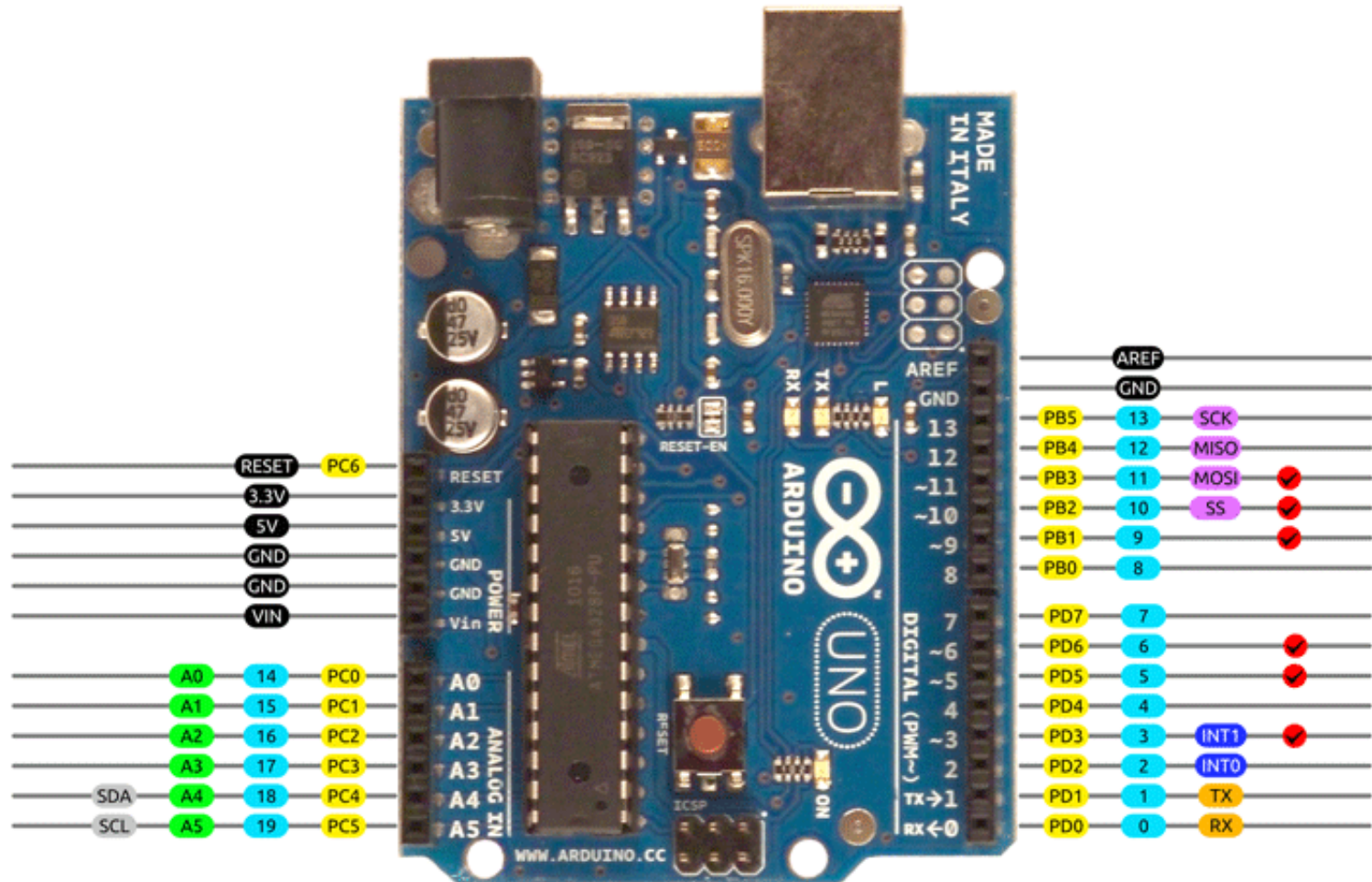
CARACTERÍSTICAS DO ARDUINO UNO REV3

ARDUINO UNO REV3

Dado	Valor
Microcontrolador	<u>ATmega328P</u>
Alimentação	7 ~12 V (recomendada) 6 < alimentação < 20 V
Pinos Digitais	14 pinos (0 ~ 13) (6 permitem saída PWM)
Pinos Analógicos	6 pinos (A0 ~A5)
Memória	FLASH: 32KB (0,5KB para boot) SRAM: 2KB EEPROM: 1KB
Clock	16 MHz



ARDUINO UNO REV3 (continuação)



AVR DIGITAL ANALOG POWER SERIAL SPI I2C PWM INTERRUPT

ARDUINO UNO REV3 (continuação)



Conexão dos pinos do microcontrolador e os terminais do Arduino.

ATmega328P

Arduino function

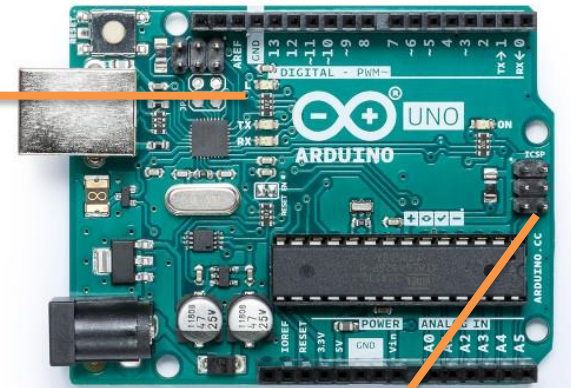
reset	(PCINT14/RESET) PC6	1
digital pin 0 (RX)	(PCINT16/RXD) PD0	2
digital pin 1 (TX)	(PCINT17/TXD) PD1	3
digital pin 2	(PCINT18/INT0) PD2	4
digital pin 3 (PWM)	(PCINT19/OC2B/INT1) PD3	5
digital pin 4	(PCINT20/XCK/T0) PD4	6
VCC	VCC	7
GND	GND	8
crystal	(PCINT6/XTAL1/TOSC1) PB6	9
crystal	(PCINT7/XTAL2/TOSC2) PB7	10
digital pin 5 (PWM)	(PCINT21/OC0B/T1) PD5	11
digital pin 6 (PWM)	(PCINT22/OC0A/AIN0) PD6	12
digital pin 7	(PCINT23/AIN1) PD7	13
digital pin 8	(PCINT0/CLKO/ICP1) PB0	14

28	PC5 (ADC5/SCL/PCINT13)
27	PC4 (ADC4/SDA/PCINT12)
26	PC3 (ADC3/PCINT11)
25	PC2 (ADC2/PCINT10)
24	PC1 (ADC1/PCINT9)
23	PC0 (ADC0/PCINT8)
22	GND
21	AREF
20	AVCC
19	PB5 (SCK/PCINT5)
18	PB4 (MISO/PCINT4)
17	PB3 (MOSI/OC2A/PCINT3)
16	PB2 (SS/OC1B/PCINT2)
15	PB1 (OC1A/PCINT1)

Arduino function

analog input 5
analog input 4
analog input 3
analog input 2
analog input 1
analog input 0
GND
analog reference
VCC
digital pin 13(LED)
digital pin 12
digital pin 11(PWM)
digital pin 10 (PWM)
digital pin 9 (PWM)

Os pinos digitais 11, 12 & 31 são utilizados no conector ICSP para MOSI, MISO & SCK (conectados aos pinos 17, 18 & 19 do ATmega328P). Evite cargas de baixa impedância nestes pinos, quando utilizar o conector ICSP.



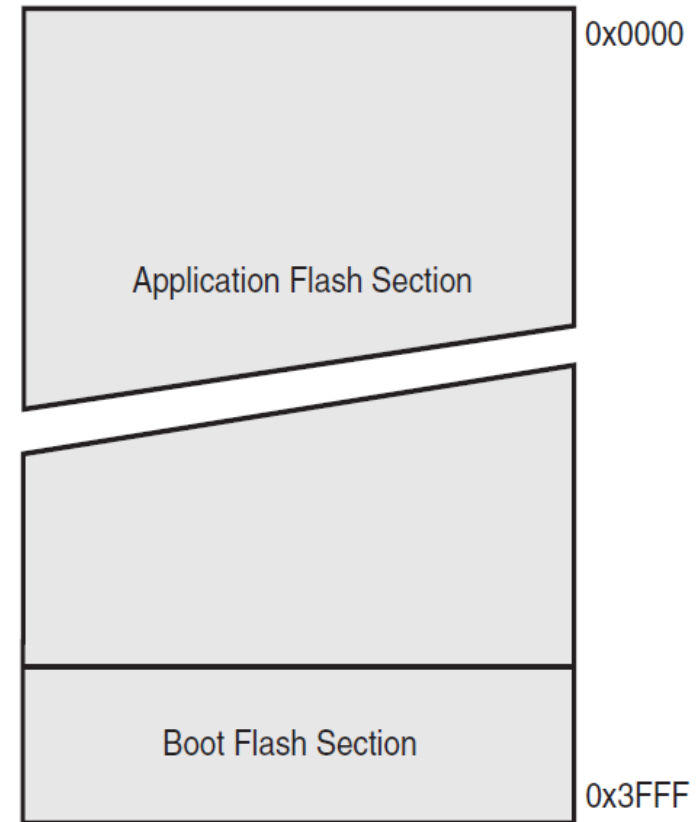
Forma de programação

MICROCONTROLADOR ATmega328P

MEMÓRIA FLASH

- Possui 32K words (16-bits) de memória NOR FLASH
- Memória não volátil
- Permite 10.000 ciclos de apagar/escrever
 - Apaga um bloco de bytes
 - Permite atualização
- A seção de BOOT permite que a seção de aplicação seja alterada por software, sem utilizar o JTAG
 - Pode ser configurado para 128, 256, 512 ou 1024 words

Área da memória onde o programa é armazenado, mesmo sem energia os dados ficam armazenados e possuem um tempo de acesso rápido.



MICROCONTROLADOR ATmega328P

MEMÓRIA EEPROM

- Possui 1K bytes de memória EEPROM
- Memória não volátil
- Endereçamento especial: 0 ~ 0x03FF
- Permite 100.000 ciclos de apagar/escrever
 - Permite escrita em um único byte
- A operação de gravação é lenta: 3,3ms
 - 26.368 ciclos de clock

Versão atualizada da EPROM (escreve e apaga por luz ultravioleta). A EEPROM utiliza a eletricidade para escrever e apagar.

MICROCONTROLADOR ATmega328P

MEMÓRIA SRAM

32 Registers	0x0000 - 0x001F
64 I/O Registers	0x0020 - 0x005F
160 Ext I/O Reg.	0x0060 - 0x00FF
Internal SRAM (2048 x 8)	0x0100 0x08FF

- Possui 2K bytes de memória SRAM
 - Endereços adicionais para os registradores de I/O
- Memória volátil e rápida
 - Uso temporário

Onde o programa cria e manipula as variáveis.

Programação e IoT

PROGRAMANDO O ARDUINO UNO REV3

ESTRUTURA DO CÓDIGO

- A função **setup** é executada somente na inicialização do programa. É utilizada para realizar as configurações iniciais do microcontrolador:
 - Definição dos pinos de I/O
 - Parametrização da comunicação serial
 - Entre outras
- A função **loop** corresponde ao executado continuamente pelo microcontrolador
 - Isto é, ao encerrar a execução, a função será chamada novamente

Blink §

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

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


VARIÁVEIS

Tipo	Número de bits	Faixa de valores	Exemplo
<u>bool</u>	1	0 ou 1 LOW ou HIGH false ou true	<code>bool meuFlag = false;</code>
<u>char</u>	8	-128 a 127	<code>char contador = -10;</code> <code>char lf = '\n';</code>
<u>unsigned char</u> ou <u>byte</u>	8	0 a 255	<code>unsigned char contador = 210;</code> <code>byte mascara1 = 0x6A;</code> <code>byte mascara2 = 0xb01101010;</code>
<u>int</u>	16	-32,768 a 32,767	<code>int contador = -3456;</code>
<u>unsigned int</u>	16	0 a 65,535	<code>int contador = 34567;</code>
<u>long</u>	32	-2,147,483,648 a 2,147,483,647	<code>long speedOfLight = 186000L;</code>
<u>unsigned long</u>	32	0 a 4,294,967,295	<code>unsigned long time = millis();</code>

FUNÇÕES BÁSICAS - delay()

Pausa o programa por uma quantidade especificada de tempo (em milissegundos).

- Um segundo = 1000 milissegundos
- Sintaxe
 - delay(ms)
- Parâmetros
 - ms: o número de milissegundos para pausar o programa (unsigned long)
- Função sem retorno

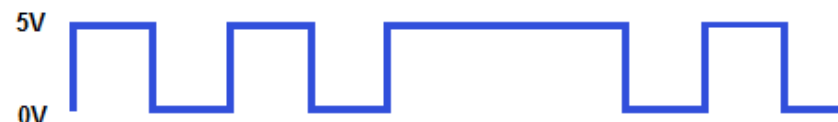
```
void setup() {  
    pinMode(LED_BUILTIN, OUTPUT);  
}  
  
void loop() {  
    digitalWrite(LED_BUILTIN, HIGH);  
    delay(1000);  
    digitalWrite(LED_BUILTIN, LOW);  
    delay(1000);  
}
```

Programação e IoT

USANDO OS PINOS DIGITAIS DO ARDUINO UNO REV3

CONCEITOS BÁSICOS - PINOS: ENTRADA E SAÍDA DIGITAL

- A [placa Arduino UNO](#) possui 14 pinos que podem ser configurados como entrada ou saídas digitais. Estes pinos são numerados de 0 a 13
- Tais pinos operam com **dois níveis de tensão definidos**:
 - Nível lógico **alto**, correspondente a **5V**
 - Nível lógico **baixo**, correspondente a **0V**.
- Assim, uma saída digital pode ser configurada para definir um nível lógico alto ou baixo. O efeito resultante é determinado pelo elemento conectado ao pino
- Da mesma maneira, uma entrada digital pode estar em nível lógico alto ou baixo, sendo este nível determinado pelo elemento conectado ao pino



FUNÇÕES BÁSICAS - pinMode()

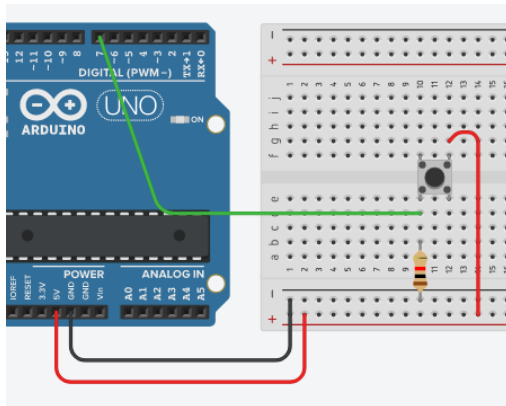
Define o modo de operação de um determinado pino: entrada ou saída digital (com ou sem pullup)

- Sintaxe
 - pinMode(**numeroPino**, **modo**)
- Parâmetros
 - **numeroPino**: número do pino
 - **modo**: pode ser
 - INPUT
 - OUTPUT
 - INPUT_PULLUP
- Função sem retorno

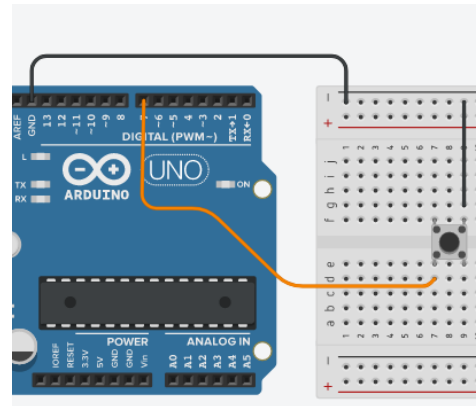
```
void setup() {  
    pinMode(2, OUTPUT);  
    pinMode(3, INPUT);  
}  
  
void loop() {  
  
    bool chave = digitalRead(3);  
  
    if(chave == false)  
    {  
        digitalWrite(2, HIGH);  
    }  
    else  
    {  
        digitalWrite(2, LOW);  
    }  
}
```

FUNÇÕES BÁSICAS - pinMode()

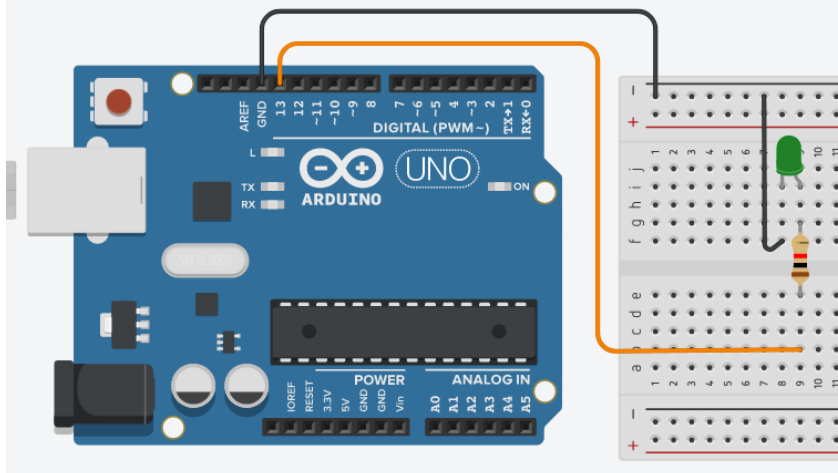
```
void setup(){  
  pinMode(7, INPUT);  
}
```



```
void setup(){  
  pinMode(7, INPUT_PULLUP);  
}
```



Resistores internos
conectados a alimentação.



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

FUNÇÕES BÁSICAS - digitalRead()

Lê o valor de um pino digital específico

- Sintaxe
 - digitalRead(**numeroPino**)
- Parâmetro
 - **numeroPino**: número do pino de entrada
- Retorna a condição atual do pino:
 - HIGH ou LOW

```
void setup() {  
    pinMode(2, OUTPUT);  
    pinMode(3, INPUT);  
}  
  
void loop() {  
  
    bool chave = digitalRead(3);  
  
    if(chave == false)  
    {  
        digitalWrite(2, HIGH);  
    }  
    else  
    {  
        digitalWrite(2, LOW);  
    }  
}
```


FUNÇÕES BÁSICAS - digitalWrite()

Define a condição de um pino de saída digital específico: HIGH ou LOW

- Sintaxe
 - digitalWrite(**numeroPino**, **valor**)
- Parâmetros
 - **numeroPino**: número do pino de saída
 - **valor**: HIGH ou LOW
- Função sem retorno

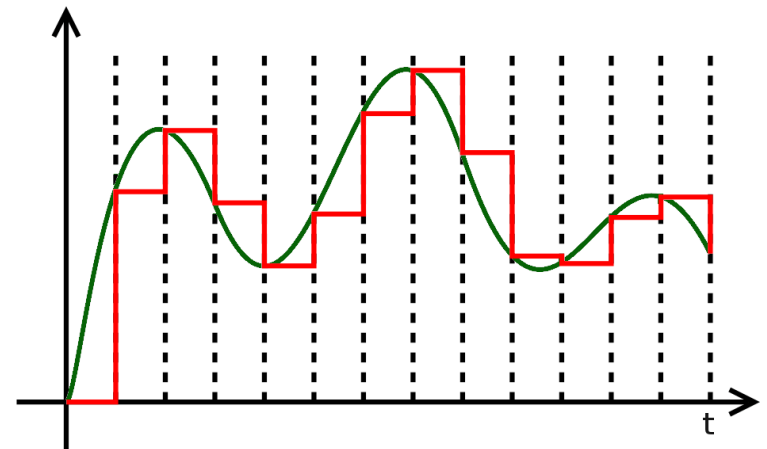
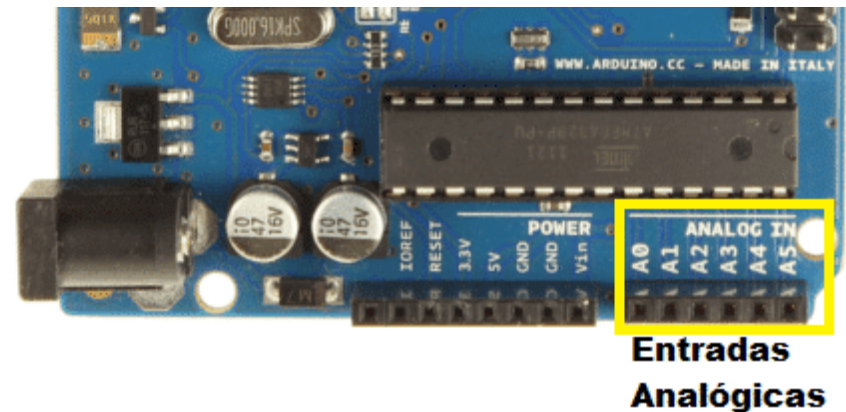
```
void setup() {  
    pinMode(2, OUTPUT);  
    pinMode(3, INPUT);  
}  
  
void loop() {  
  
    bool chave = digitalRead(3);  
  
    if(chave == false)  
    {  
        digitalWrite(2, HIGH);  
    }  
    else  
    {  
        digitalWrite(2, LOW);  
    }  
}
```

Programação e IoT

USANDO OS PINOS ANALÓGICOS DO ARDUINO UNO REV3

CONCEITOS BÁSICOS - PINOS: ENTRADA ANALÓGICA

- A [placa Arduino UNO](#) possui 6 pinos que podem ser configurados como entrada analógicas
 - Possuem um conversor analógico/digital
 - Estes pinos são numerados de A0 a A5
- Internamente o Arduino possui um conversor A/D com resolução de 10 bits
 - Dessa forma, uma entrada que varia entre 0V e 5V terá um valor representado entre 0 e 1023 ($2^{10} - 1$)
 - Ou ainda, cada incremento corresponde a 4,88mV



FUNÇÕES BÁSICAS – analogRead()

Lê o valor de um pino analógico específico e converte para um valor digital correspondente

- Isso significa que este irá mapear tensões entre 0 e a tensão operacional (5V ou 3.3V) para valores inteiros entre 0 e 1023
- O intervalo de entrada pode ser mudado através da função analogReference()
- Sintaxe
 - analogRead(**nomePino**);
- Parâmetros
 - **nomePino**: nome do pino
 - A0 a A5
- Retorno
 - Leitura analógica do pino (int)
 - Valor entre 0 e 1023

```
void setup() {  
    Serial.begin(9600);  
}  
  
void loop() {  
    int ldr = analogRead(A0);  
  
    if(ldr > 450)  
    {  
        Serial.println("Claro");  
    }  
    else  
    {  
        Serial.println("Escuro");  
    }  
}
```

FUNÇÕES BÁSICAS – analogReference()

Configura a tensão de referência para a entrada analógica (o valor máximo do intervalo de entrada)

- Sintaxe
 - analogReference(**tipo**);
- Parâmetros
 - **tipo**: qual tipo de referência usar:
 - DEFAULT (5 V no arduino UNO)
 - INTERNAL (1,1 V no arduino UNO)
 - EXTERNAL (0 a 5 V aplicado no pino)
- Função sem retorno

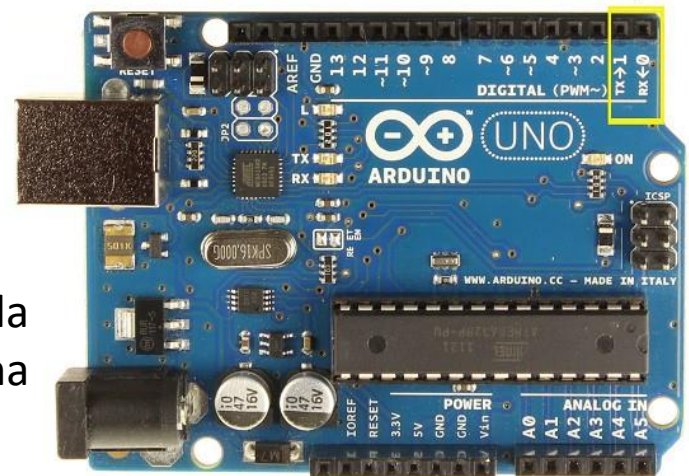
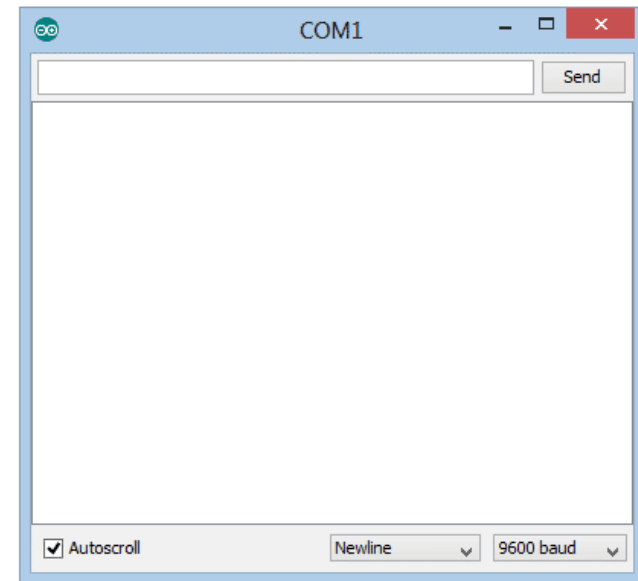
```
void setup() {  
    Serial.begin(9600);  
    analogReference(DEFAULT);  
}  
  
void loop() {  
    int ldr = analogRead(A0);  
  
    if(ldr > 450)  
    {  
        Serial.println("Claro");  
    }  
    else  
    {  
        Serial.println("Escuro");  
    }  
}
```

Programação e IoT

COMUNICAÇÃO SERIAL NO ARDUINO UNO REV3

CONCEITOS BÁSICOS - COMUNICAÇÃO SERIAL

- A placa Arduino UNO possui apenas um canal de comunicação por hardware.
- Com este recurso é possível receber e enviar informações de e para outros dispositivos.
 - Por exemplo, o computador ou um módulo GPS
- A comunicação é dita serial pois cada byte de uma transação é transferido bit a bit
- As principais características de configuração de um canal serial são as seguintes:
 - taxa de comunicação (baudrate)
 - bits de dados (data bits)
 - bits de parada (stop bit)
 - paridade (parity bit)
- Cabe ressaltar que os elementos envolvidos da comunicação devem estar configurados da mesma maneira.



FUNÇÕES BÁSICAS - Serial.begin()

Configura a taxa de transferência em bits por segundo (baud rate) para transmissão serial

- Sintaxe
 - Serial.begin(speed)
 - Serial.begin(speed, config)
- Parâmetros
 - speed: a taxa de transmissão em bits per second (baud rate) - (long)
 - config: parâmetro opcional com o número de bits, paridade, e stop bits:
 - SERIAL_8N1 (o padrão)
- Função sem retorno

```
void setup() {  
    pinMode(3, INPUT);  
    Serial.begin(9600);  
}  
  
void loop() {  
  
    bool chave = digitalRead(3);  
  
    if(!chave)  
    {  
        Serial.println("A chave foi pressionada");  
    }  
}
```

FUNÇÕES BÁSICAS - Serial.println()

Imprime dados na porta serial como texto ASCII seguido pelo caractere de retorno de carruagem (ASCII 13 ou '\r') e um caractere de nova linha (ASCII 10 ou '\n').

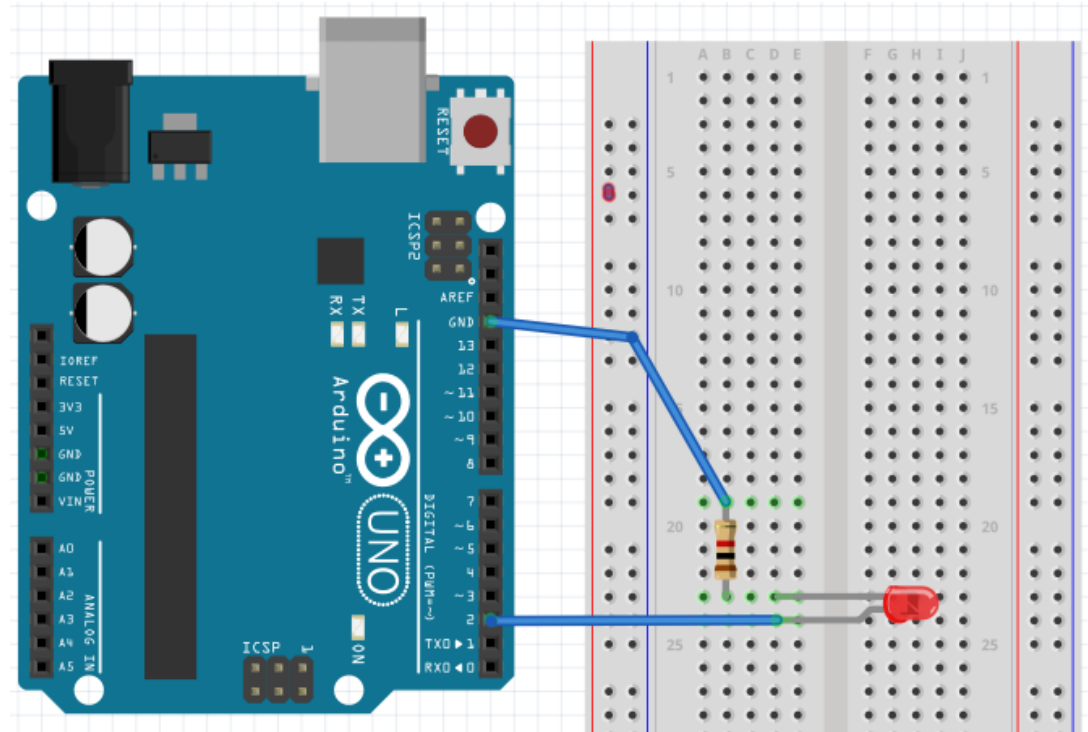
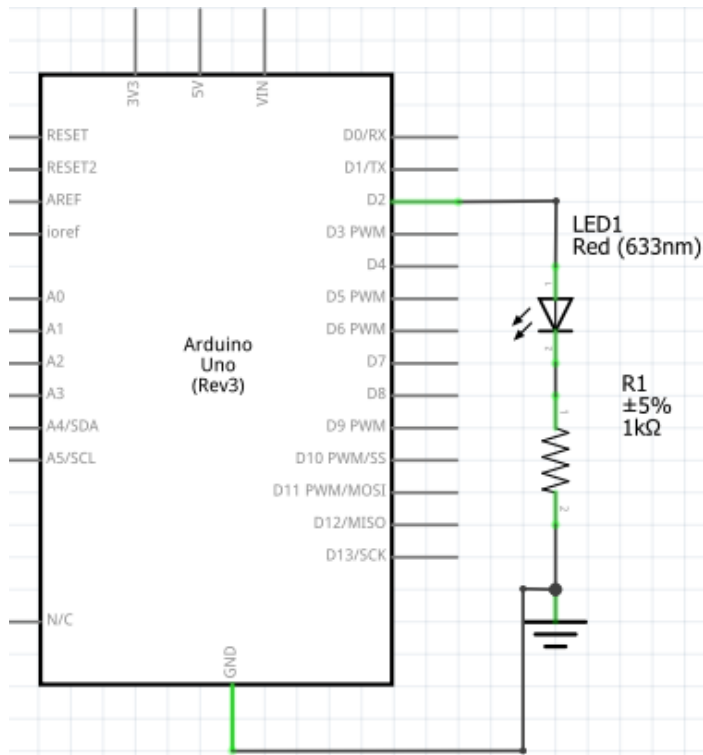
- Essa função assume as mesmas formas que [Serial.print\(\)](#)
- Sintaxe
 - Serial.println(val)
 - Serial.println(val, formato)
- Parâmetros
 - **val**: o valor a ser impresso - qualquer tipo de dados
 - **formato**: parâmetro opcional para especificar a base do numeral (se **val** for int) ou número de casas decimais (se **val** for float)
 - DEC, HEX, OCT ou BIN
- Retorno
 - Número de bytes escritos, porém a leitura desse número é opcional – (size_t)

```
void setup() {  
    pinMode(3, INPUT);  
    Serial.begin(9600);  
}  
  
void loop() {  
  
    bool chave = digitalRead(3);  
  
    if(!chave)  
    {  
        Serial.println("A chave foi pressionada");  
    }  
}
```

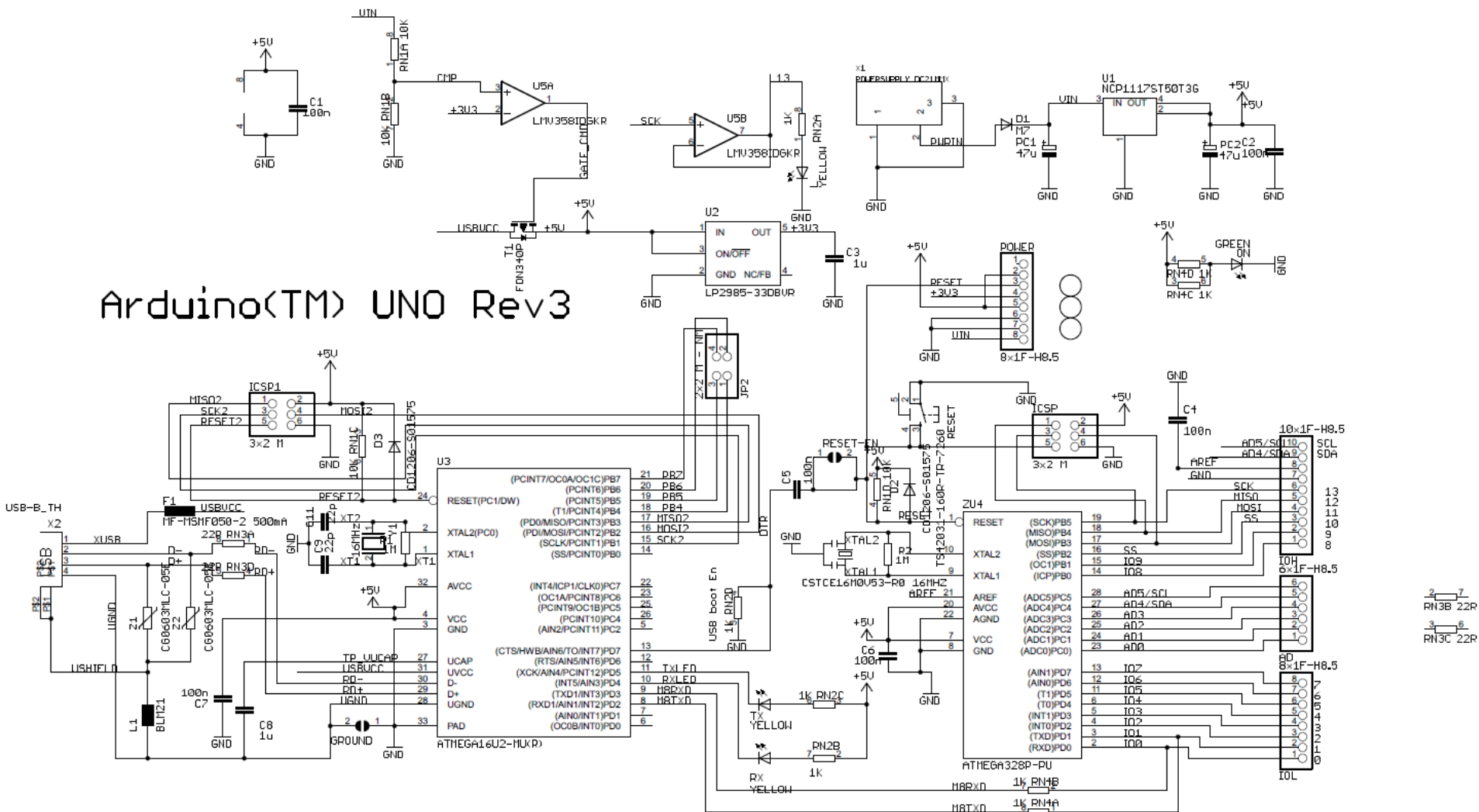
Programação e IoT

O ARDUINO UNO REV3

DIAGRAMA UNIFILAR X DIAGRAMA DE MONTAGEM



ESQUEMÁTICO DO ARDUINO UNO REV3



Programação e IoT

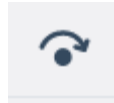
TINKERCAD DEBUGGER

TINKERCAD DEBUGGER

- O Tinkercad possui uma funcionalidade de depuração muito útil na verificação do código criado
- Após iniciar a simulação de um código pressione o botão DEPARADOR



- Clique na linha do código para definir pontos de parada para o código
- Passe o mouse sobre as variáveis
 - Mostra seu valor corrente
- Pressione o botão NEXT



Avança para a próxima linha

- Pressione o botão CONTINUAR



- Avança para a próxima parada

```
9  int contador = 0;
10
11  void setup()
12  {
13      pinMode(4, INPUT);
14      pinMode(8, OUTPUT);
15  }
16
17  void loop()
18  {
19      botao = digitalRead(4);
20      // Alterna o estado do pisca do LED ao pressionar
21      // botão
22      if (botao != botaoAnterior) {
23          botaoAnterior = botao;
24          if (botao == 1) {
25              contador += 1;
26          }
27      }
28      // Acende o LED se contador for par
29      if (contador % 2 == 0) {
30          // turn the LED on (HIGH is the voltage level)
31          digitalWrite(8, HIGH);
32          delay(200); // Wait for 200 millisecond(s)
33          // turn the LED off by making the voltage LOW
```



REFERÊNCIAS

- <https://www.embarcados.com.br/arduino-primeiros-passos/>
- <https://www.embarcados.com.br/pinos-digitais-do-arduino/>
- <https://www.embarcados.com.br/arduino-entradas-analogicas/>
- <https://www.embarcados.com.br/arduino-comunicacao-serial/>
- <https://www.arduino.cc/reference/en/#functions>
- <https://www.robocore.net/tutoriais/como-utilizar-uma-protoboard.html>