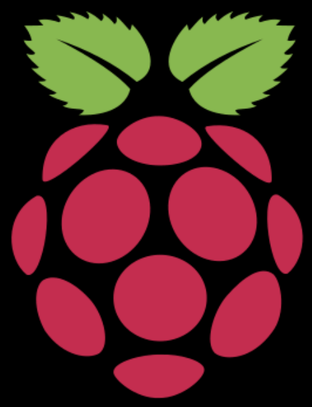
Projeto 06 + 07

Controle de Tempo – Teoria

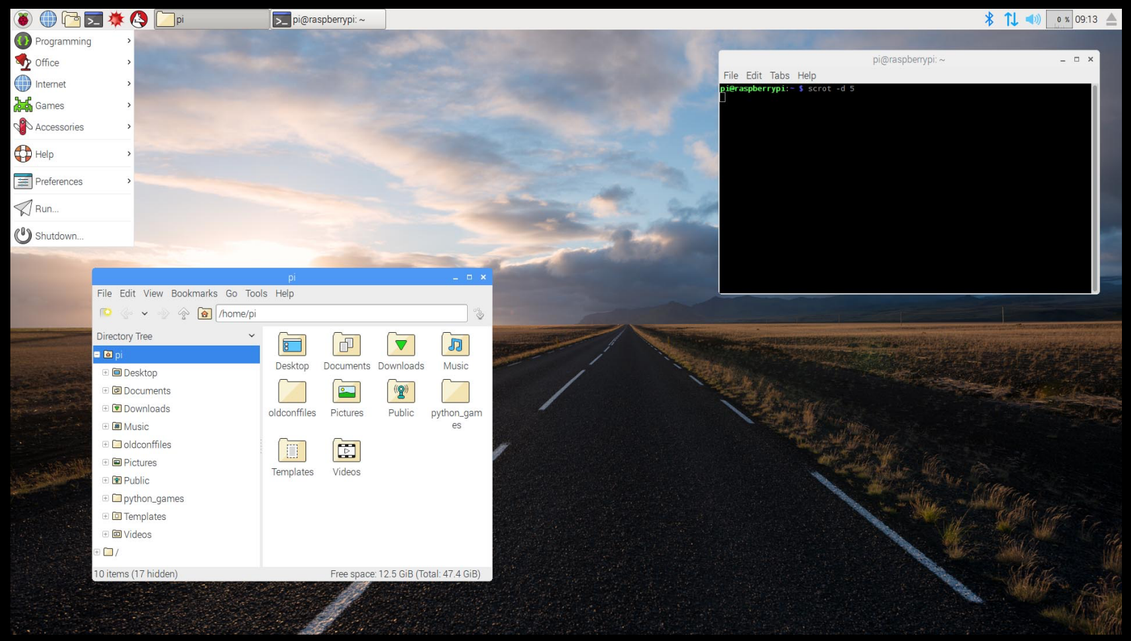
Jan K. S. – janks@puc-rio.br

ENG1419 – Programação de Microcontroladores



PC + microcontrolador

microcontrolador



Diferença Básica entre Raspberry Pi e Arduino

	Raspberry Pi 4B	Arduino Mega 2560
Preço (China)	R$ 300,00	R$ 55,00
Pinos Bem Identificados	não	sim
Pinos Digitais	26 (1 para PWM)	54 (15 para PWM)
Entradas Analógicas	0	16
Acessórios Shield	poucos	muitos
Potência em Repouso	1.5 W	0.425 W
Complexidade de Software	instalar e configurar Linux	nenhuma
Tempo de Inicialização	≈ 1 minuto	0s

Vantagens do Arduino

	Raspberry Pi 4B	Arduino Mega 2560
Processador	4 núcleos de 1.8GHz	1 núcleo de 16 MHz
Memória RAM	1 GB – 8 GB	8 kB
Armazenamento	cartão SD até 128 GB	256 kB + 4 kB
WiFi e Bluetooth	incluídos na placa	só com acessórios extras
Sistema Operacional	Linux	(nenhum)
Precisa de computador?	não	sim
Linguagem de Programação	Python e muitas outras	C / C++
Roda Crysis?	não	não

Vantagens do Raspberry Pi

Revisão de C/C++

- operações
- variáveis
- listas
- textos
- funções
- condicionais
- loops

- operações
- variáveis
- listas
- textos
- funções
- condicionais
- loops

Mapeamento de Conhecimentos

tipo de variável

ponto-e-vírgula

```python
inteiro = 15
decimal = 4.5
lista = [1, 2, 3]
texto = "Texto"
booleano = True
```

```cpp
int inteiro = 15;
float decimal = 4.5;
int lista[] = {1, 2, 3};
char texto[] = "Texto";
bool booleano = true;
```

Declaração de Variáveis

```python
lista = [1, 2, 3]
x = lista[0]
lista[1] = -99
```

```cpp
int lista[] = {1, 2, 3};
int x = lista[0];
lista[1] = -99;
```

Acesso a um Elemento da Lista

aspas DUPLAS
para texto

```python
texto = "Texto"
letra = texto[0]
x = (letra == "T")
```

```cpp
char texto[] = "Texto";
char letra = texto[0];
bool x = (letra == 'T');
```

aspas SIMPLES
para caracteres

Acesso a um Caractere da Lista

```python
# linha de comentário
from threading import Timer
```

```cpp
// linha de comentário
#include <TimerOne.h>
```

Comentários e Importação de Funções

```python
def soma2(x):
    return x + 2

y = soma2(4.5)

def imprime_oi():
    print("oi!")
```

tipo da variável de RETORNO

```cpp
float soma2 (float x) {
    return x + 2;
}

float y = soma2(4.5);

retorna "vazio"

void imprimeOi () {
    Serial.print("oi!");
}
```

Criação e Chamada de Funções

python

```python
if x > 0 and y > 0:
    z = 1
elif x < 0 or y < 0:
    z = 2
elif x != 0:
    z = 3
else:
    z = 4
```

```cpp
if (x > 0 && y > 0) {
    z = 1;
}
else if (x < 0 || y < 0) {
    z = 2;
}
else if (x != 0) {
    z = 3;
}
else {
    z = 4;
}
```

Condicionais

```python
x = 50
while x > 0:
    x = x / 5
```

```cpp
float x = 50;
while (x > 0)
{
    x = x / 5;
}
```

```python
for i in range(0, 5):
    print(i)
```

```cpp
for (int i = 0; i < 5; i++)
{
    Serial.println(i);
}
```

inicie i como 0;
a cada ciclo, enquanto i < 5,
faça i aumentar em 1 unidade

Loops

Hardware

pt.wikipedia.org/wiki/Arduino

Arduino

Origem: Wikipédia, a enciclopédia livre.

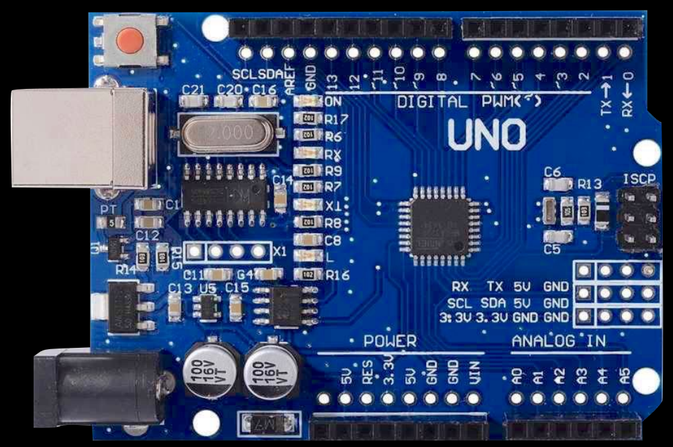
Nota: Para outros significados, veja Arduino (desambiguação).

Arduíno[2][4][5] é uma plataforma de prototipagem eletrônica de hardware livre e de placa única,[6] projetada com um microcontrolador Atmel AVR com suporte de entrada/saída embutido, uma linguagem de programação padrão,[7] a qual tem origem em Wiring, e é essencialmente C/C++.[8] O objetivo do projeto é criar ferramentas que são acessíveis, com baixo custo, flexíveis e fáceis de se usar por novatos e profissionais. Principalmente para aqueles que não teriam alcance aos controladores mais sofisticados e de ferramentas mais complicadas.[9]

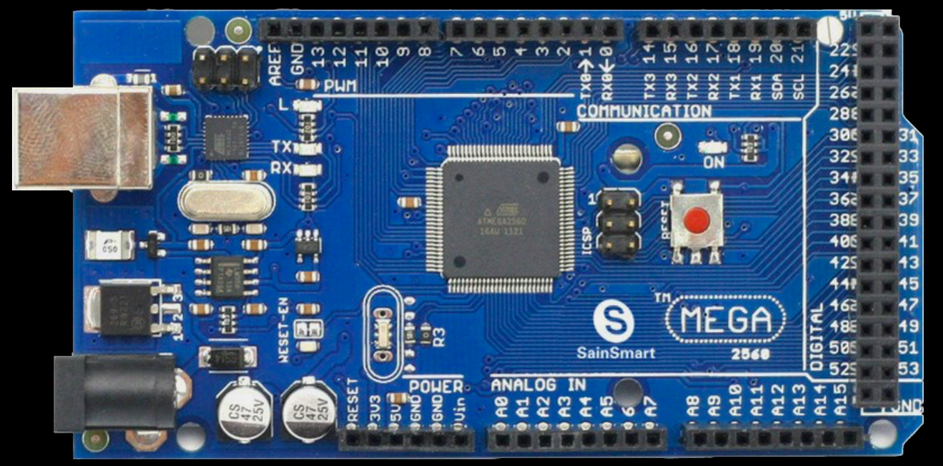
Pode ser usado para o desenvolvimento de objetos interativos independentes, ou ainda para ser conectado a um computador hospedeiro. Uma típica placa Arduino é composta por um controlador, algumas linhas de E/S digital e analógica, além de uma interface serial ou USB, para interligar-se ao hospedeiro, que é usado para programá-la e interagi-la em tempo real. Ela em si não possui qualquer recurso de rede, porém é comum combinar um ou mais Arduinos deste modo, usando extensões apropriadas chamadas de shields[10]. A interface do hospedeiro é simples, podendo ser escrita em várias linguagens. A mais popular é a Processing, mas outras que podem comunicar-se com a conexão serial são: Max/MSP,[11] Pure Data,[12] SuperCollider,[13] ActionScript[14] e Java.[15] Em 2010 foi realizado um documentário sobre a plataforma chamado *Arduino: The Documentary*.

Dados sobre o Arduino

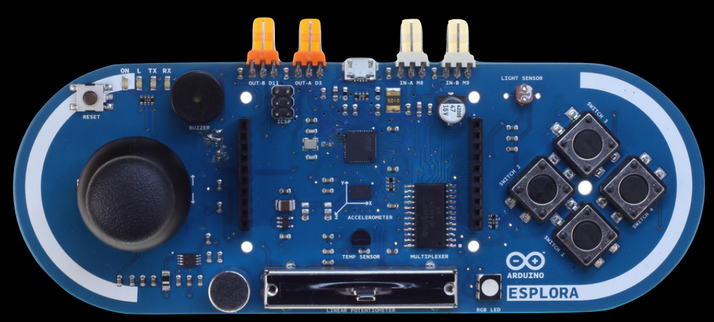
Arduino Uno



Arduino Mega 2560



Arduino Esplora



Arduino Nano

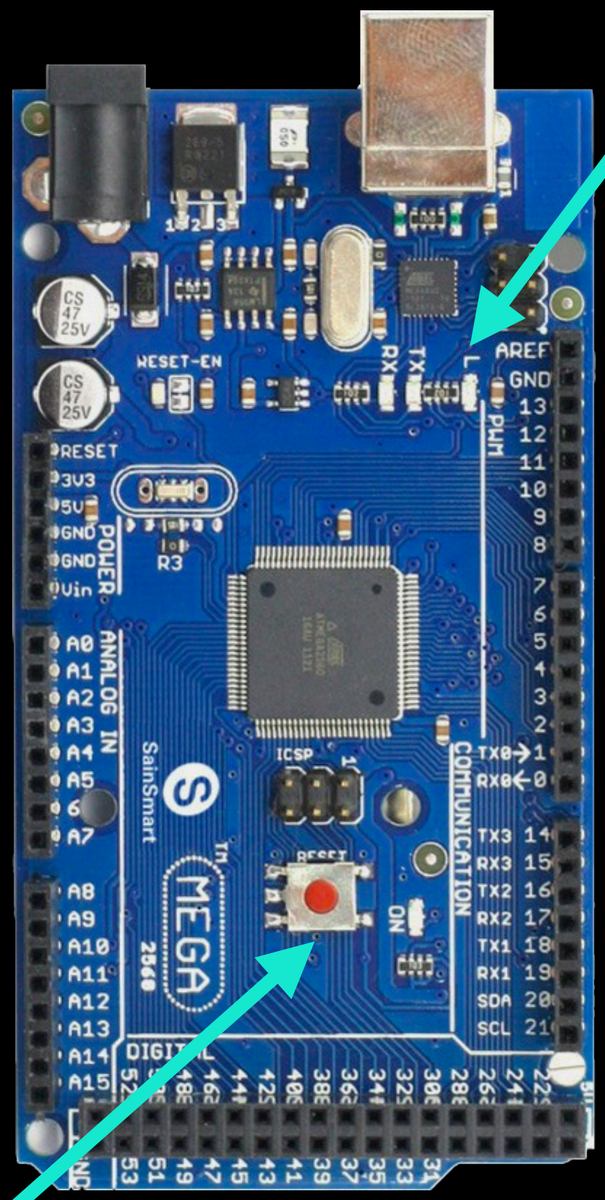
Exemplos de Modelos de Arduino

alimentação

LED interno

pinos de
terra e VCC

pinos digitais
com PWM



entradas
analógicas

pinos digitais
para comunicação

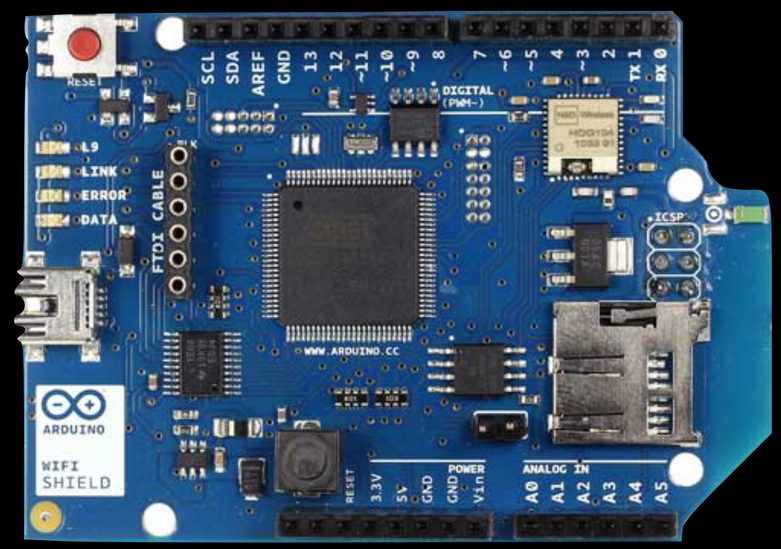
botão reset

pinos digitais

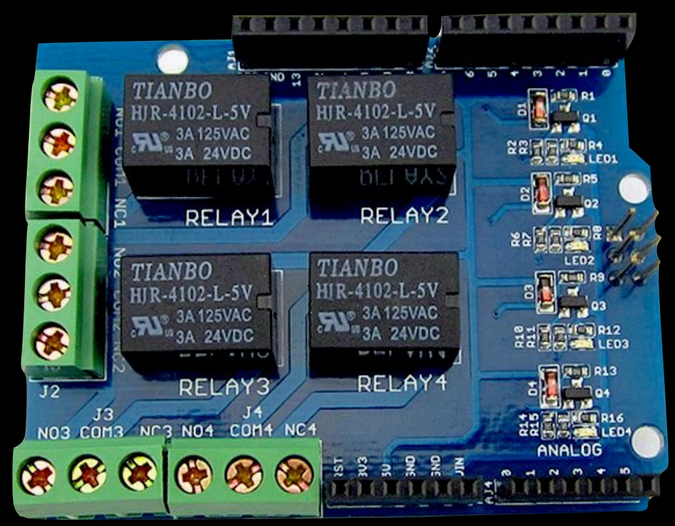
Arduino Mega 2560

Shield Multifunção

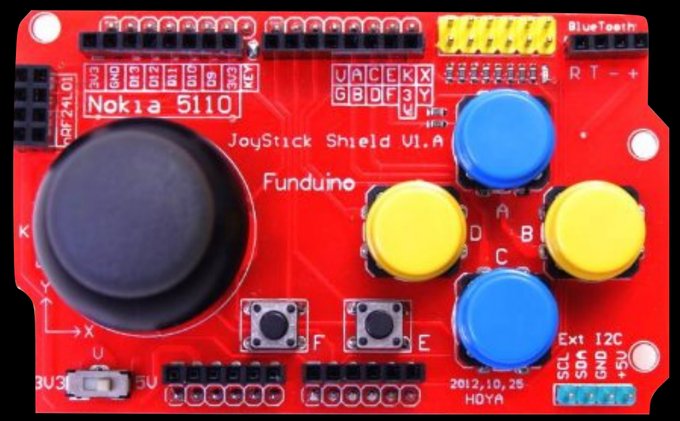
Shield com WiFi



Shield com Relés



Shield com Joystick

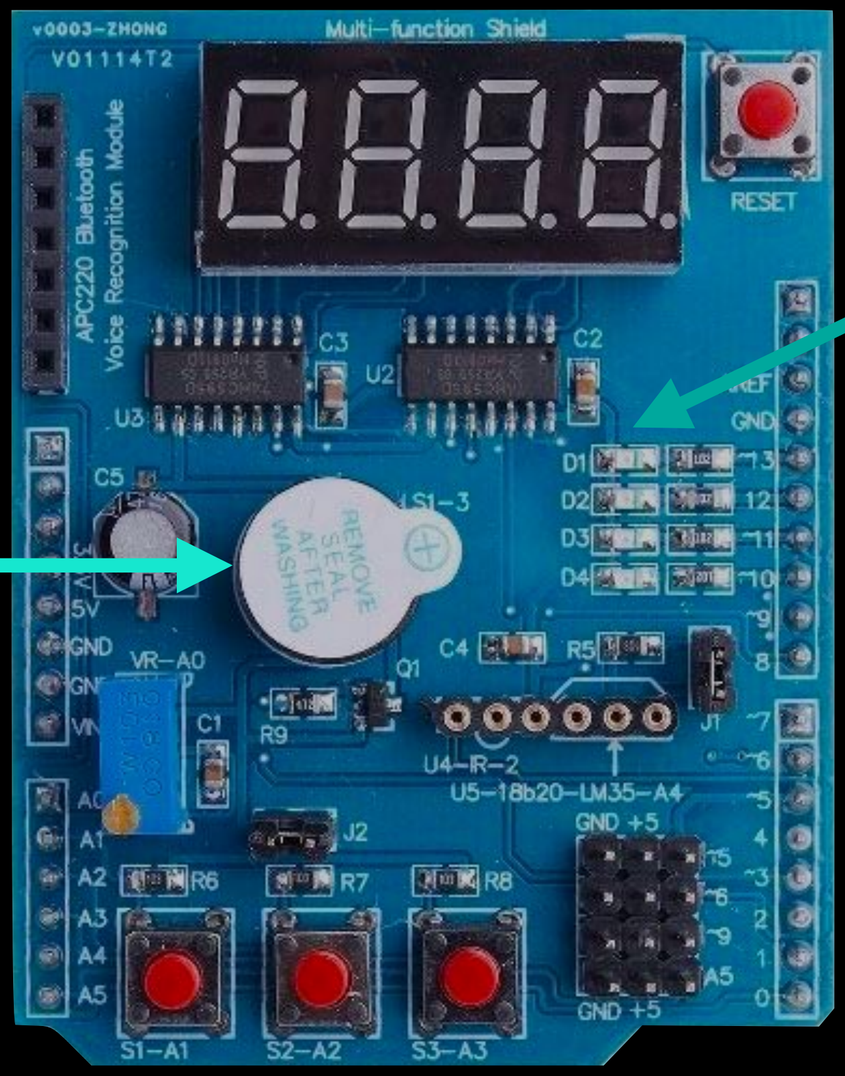


Exemplos de Shields para Arduino

displays de 7 segmentos

LEDs

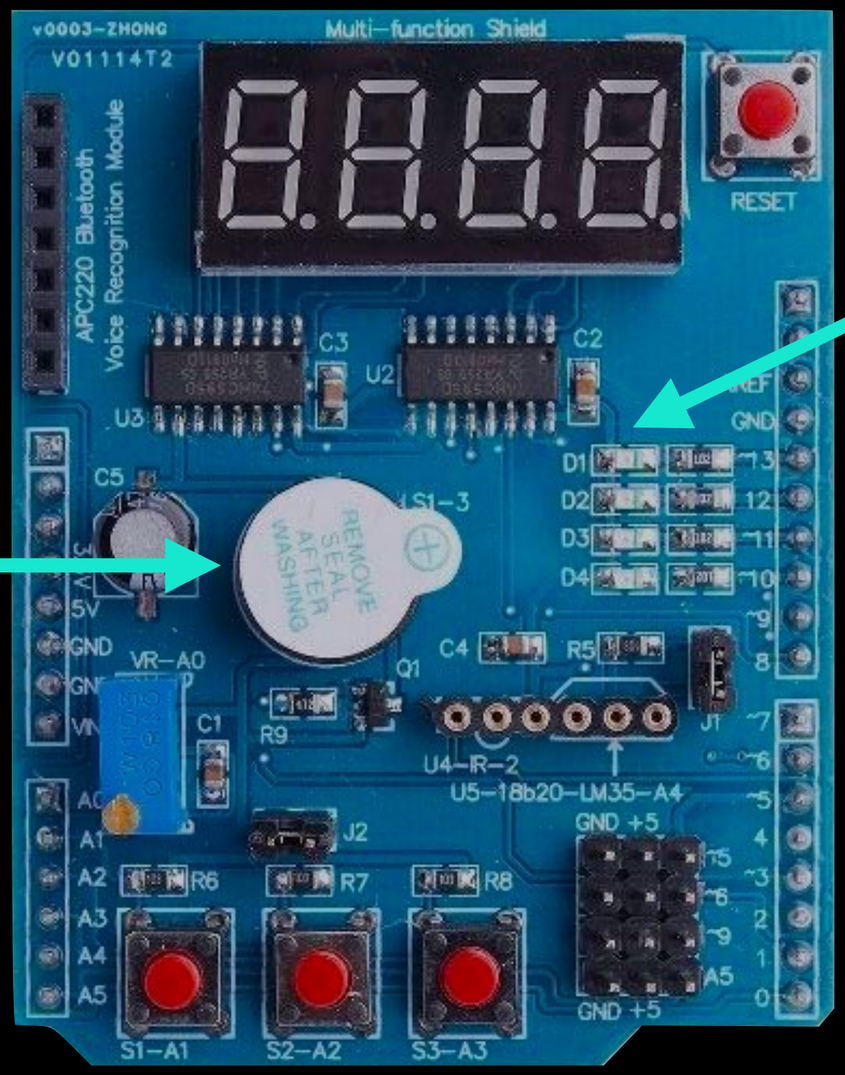
campainha



botões digitais

Shield Multifunção

pinos 4, 7 e 8

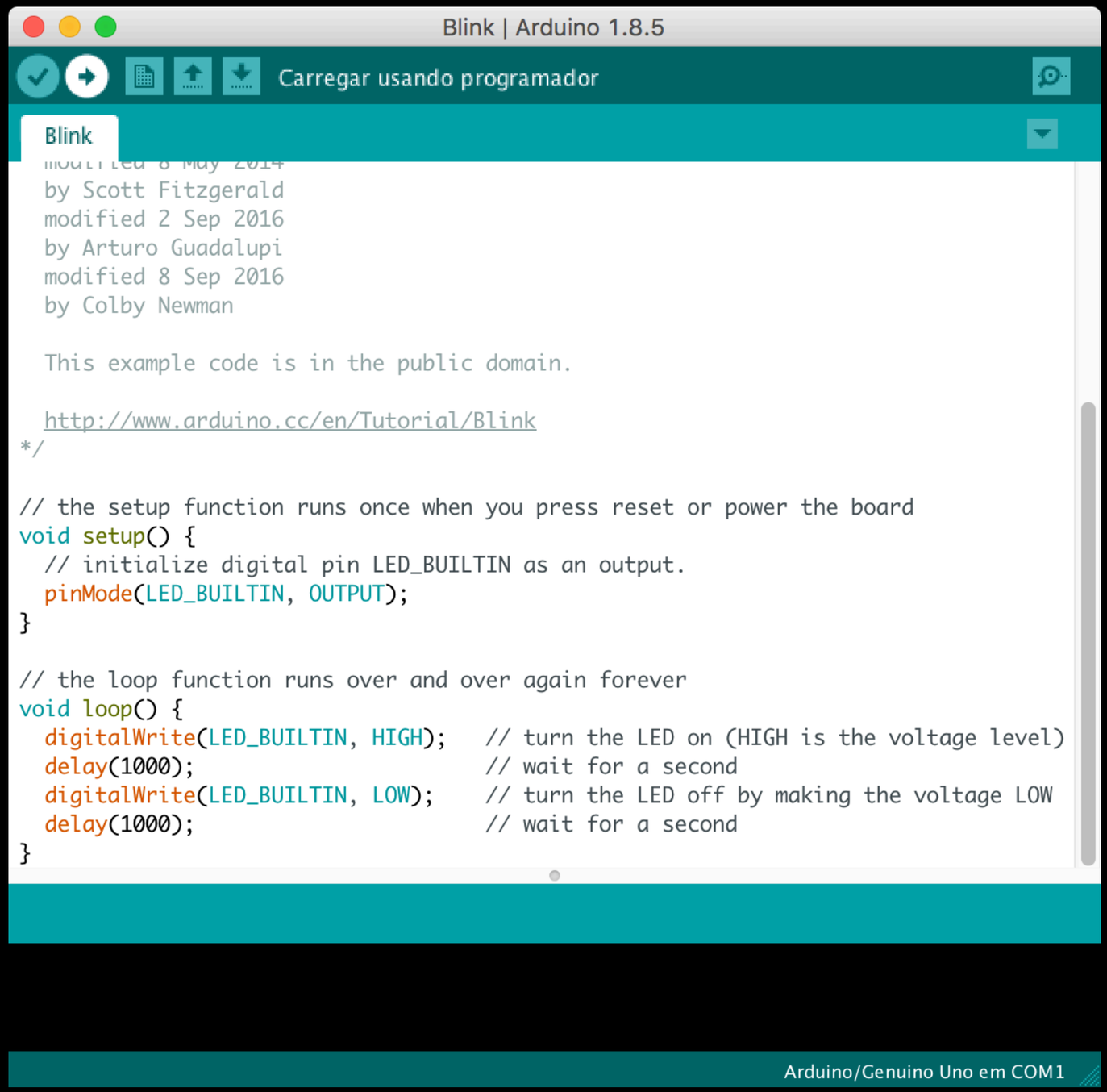


pino 3

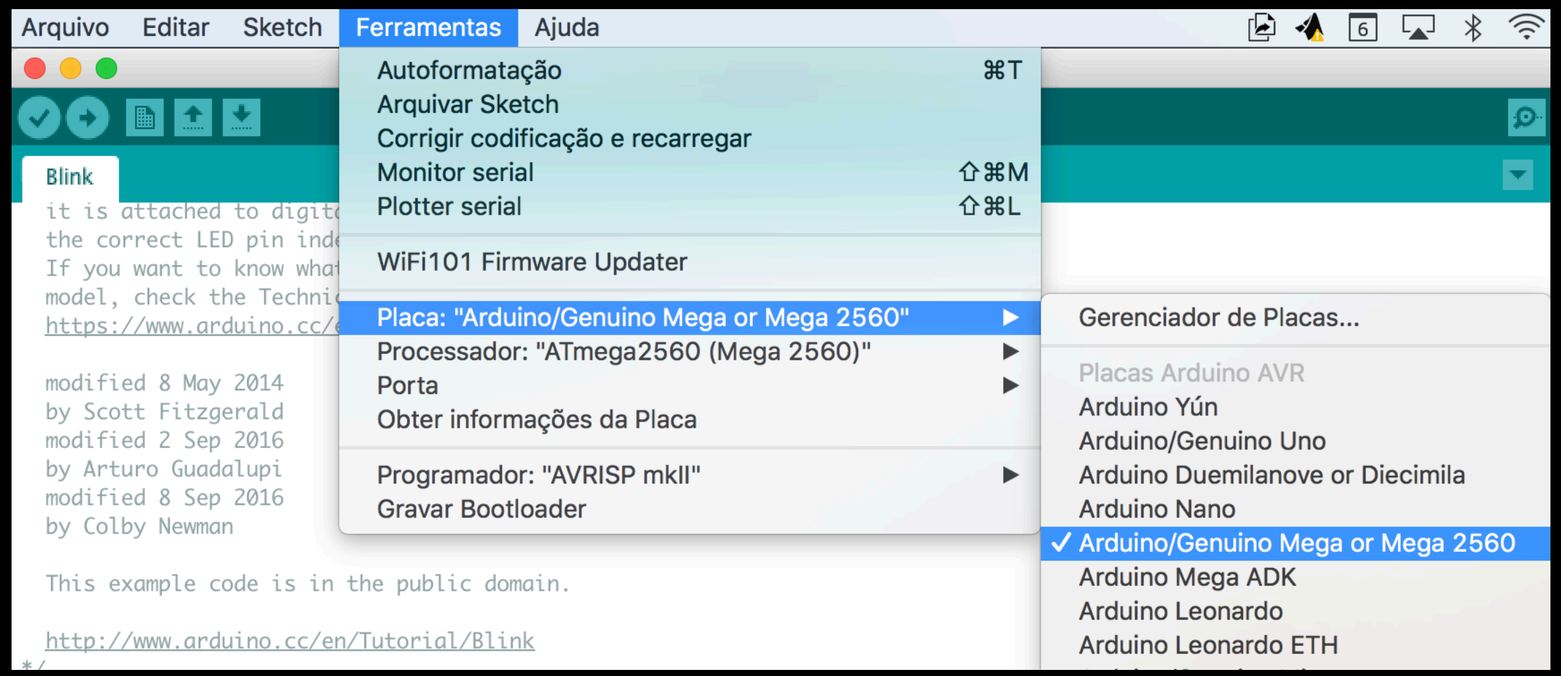
pinos 13, 12, 11 e 10

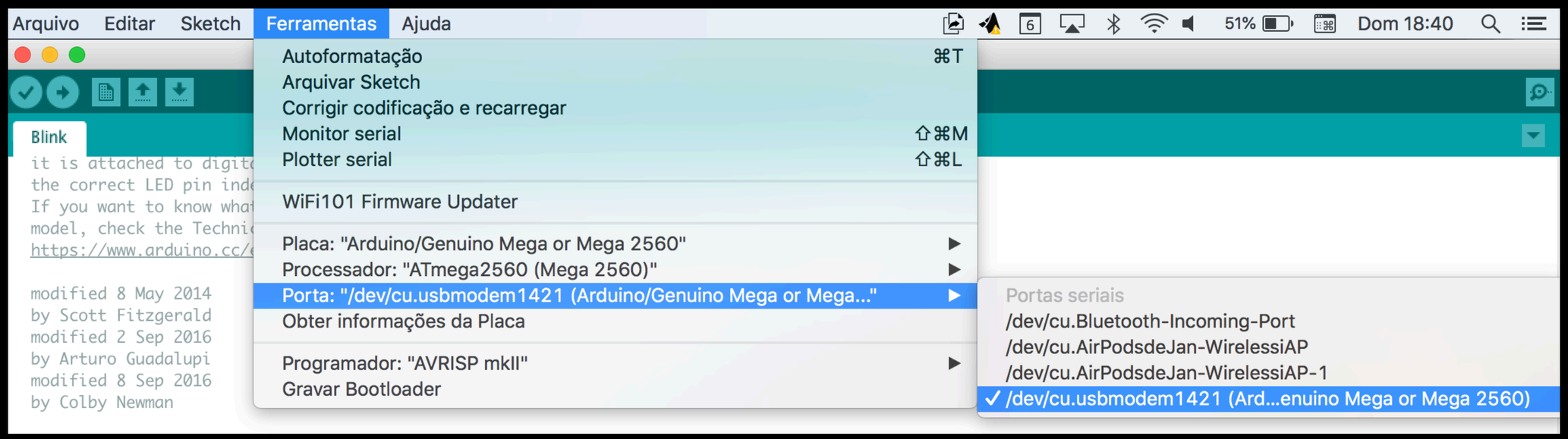
pinos A1, A2 e A3

Pinos Usados pelo Shield Multifunção



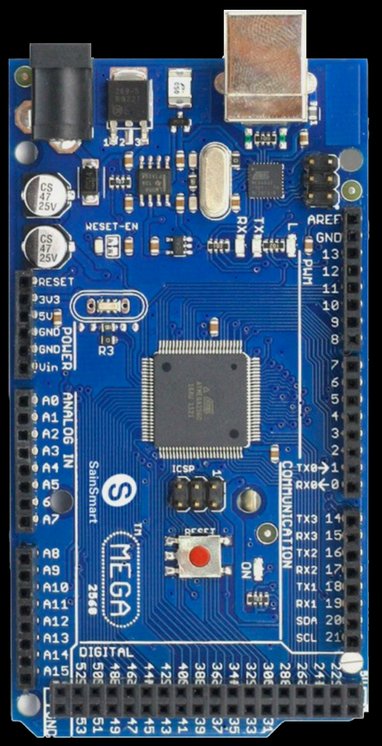
Editor de Código para Arduino





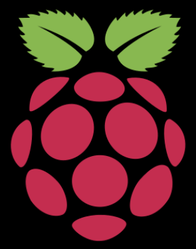
Configuração de Comunicação com o Arduino

```ino
// 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
}
```



escreve programa no computador → envia programa pela USB → executa programa

Programação do Arduino



```python
# bibliotecas
from gpiozero import LED, Button

# funções
def funcao1(x):
    return x + 2

# inicialização de componentes
led = LED(21)
botao = Button(11)
botao.when_pressed = funcao1

# loop infinito
while True:
    ...
```

```arduino
// bibliotecas
#include <GFButton.h>

// funções
int funcao1(int x) {
    return x + 2;
}

// inicialização de componentes
int led = 21;
GFButton botao(11);
void setup () {
    pinMode(led, OUTPUT);
    botao.setPressHandler(funcao1);
}

// loop infinito
void loop () {
    ...
}
```

Estrutura Geral de Código

Início do Programa

```
void setup () {
    ...
}
```

```
void loop () {
    ...
}
```

É possível reiniciar um programa no Arduino (RESET), mas não pará-lo.

Para remover um programa, é preciso inserir outro (ex: um em branco).

Rotina de Execução de Código em um Arduino

```
void setup () {
    // inicia comunicação com taxa de 9600 bit/s
    Serial.begin(9600);
}

void loop () {
    Serial.println("Hey, listen!");

    // aguarda tempo em MILISSEGUNDOS
    delay(1000);
}
```

Exemplo Básico de Setup/Loop para Imprimir Texto

teste | Arduino 1.8.5

teste

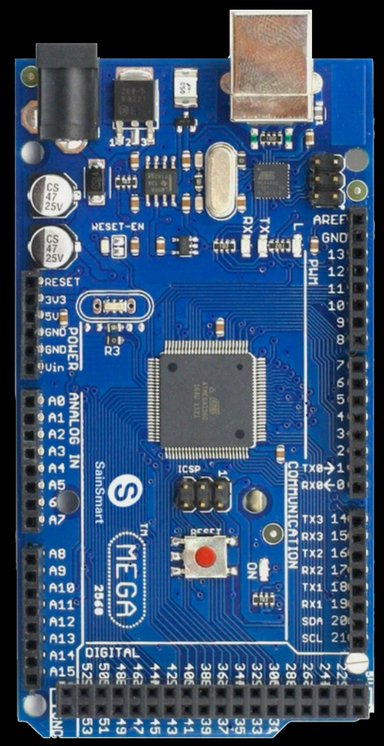
```
1  void setup () {
2      Serial.begin(9600);
3  }
4
5  void loop () {
6      Serial.println("Hey, listen!");
7      delay(1000); // aguarda tempo em MILISSEGUNDOS
8  }
9
10
```

Carregado.

0 sketch usa
Variáveis glo

10 Arduino/Ge

envio de textos
via USB (serial)



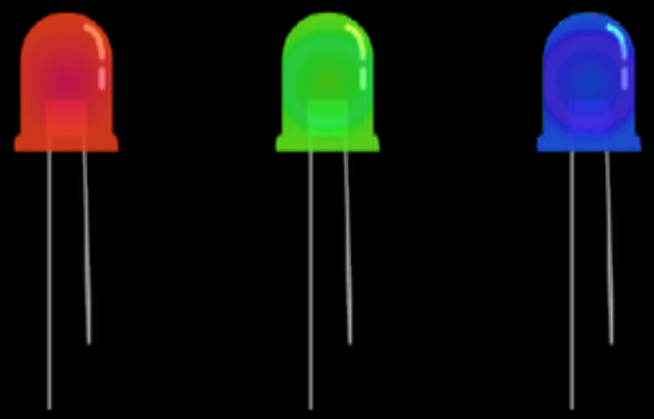
/dev/cu.usbmodem1421 (Arduino/Genuino Mega or Mega 2560)

Enviar

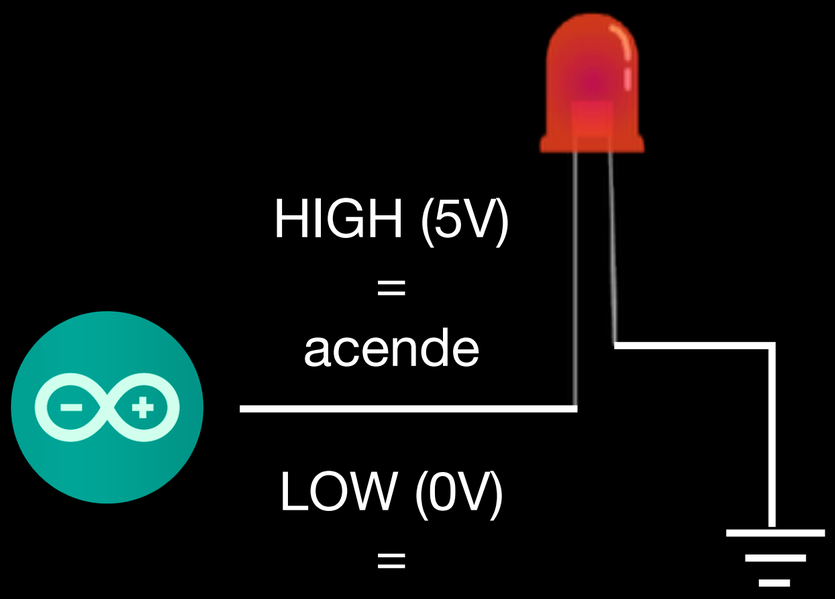
```
Hey, listen!
Hey, listen!
Hey, listen!
Hey, listen!
Hey, listen!
Hey, listen!
Hey, listen!
Hey, listen!
Hey, listen!
```

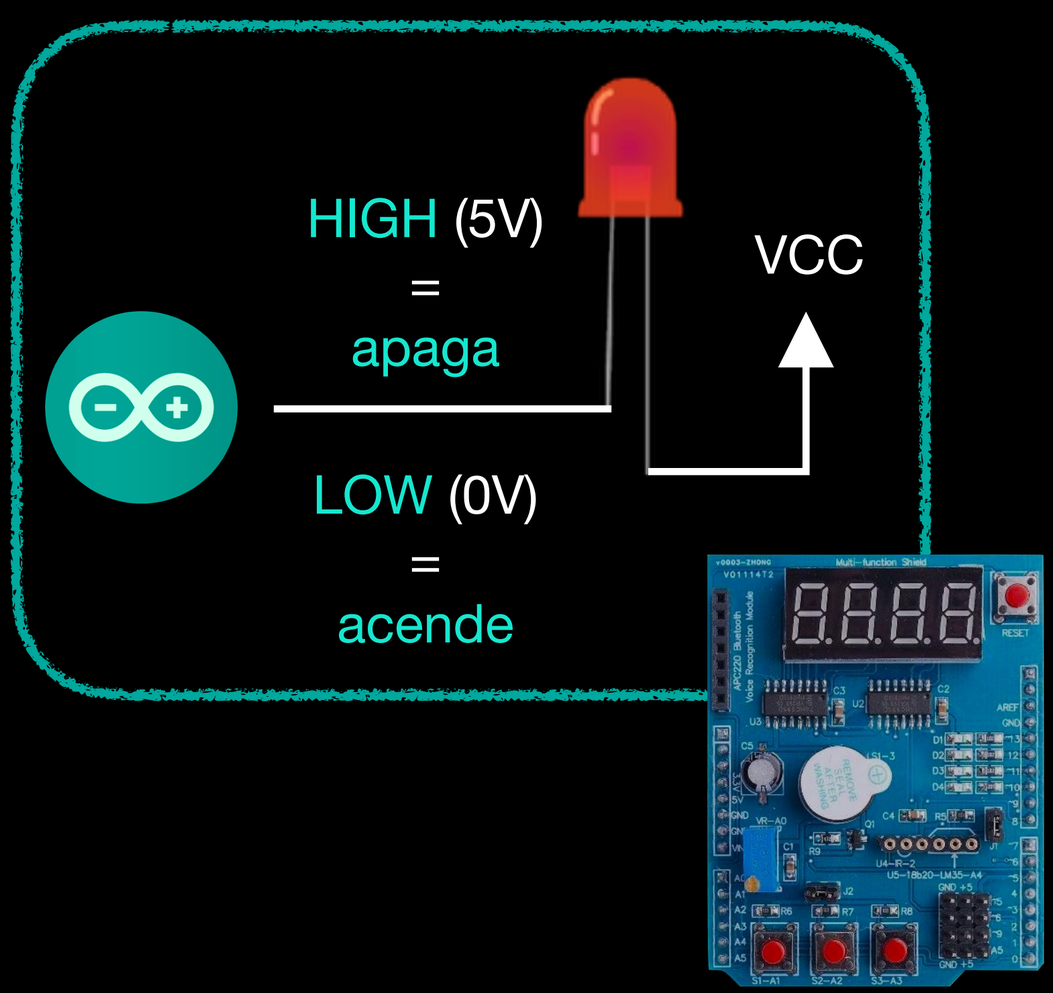
Auto-rolagem | Nenhum final-de-linha | 9600 velocidade | Deleta a saida

Impressão de Textos via Comunicação Serial



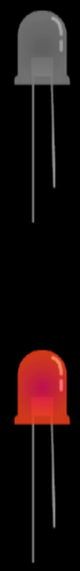
LEDs





Duas Formas de Usar um LED

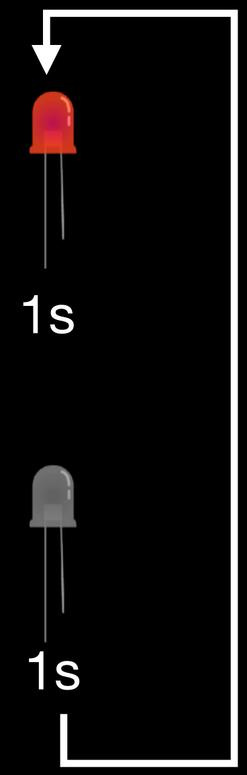
```ino
int led = 13; // LED 1 está no pino 13

void setup () {
    pinMode(led, OUTPUT);
    digitalWrite(led, HIGH);
}

void loop () {
    digitalWrite(led, LOW);
}
```



o primeiro `digitalWrite` não é necessário neste caso, mas é uma boa prática para casos gerais (por padrão, o pino começa com LOW)

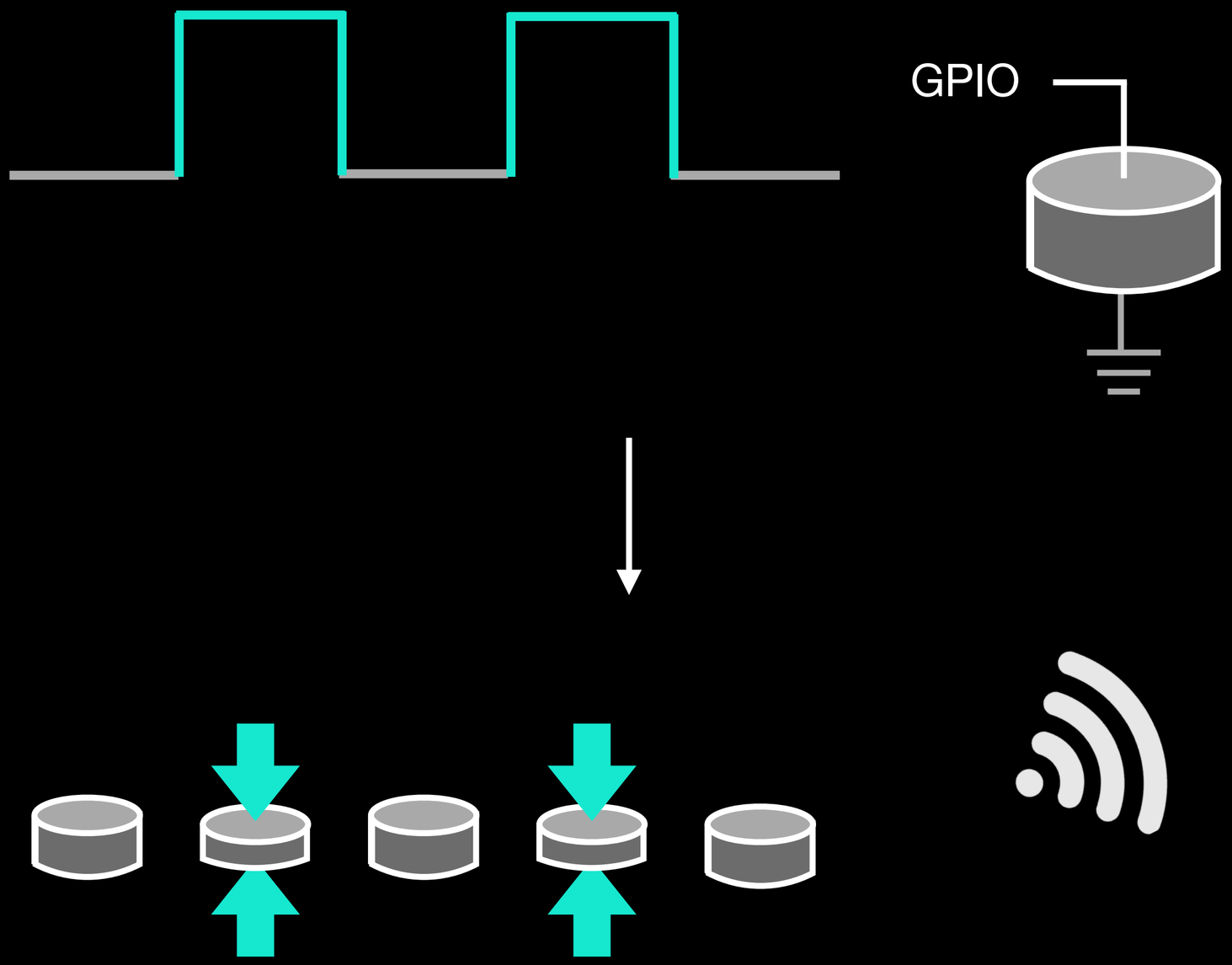
Apagando e Acendendo um LED

```
int led = 13;

void setup () {
    pinMode(led, OUTPUT);
    digitalWrite(led, HIGH);
}

void loop () {
    digitalWrite(led, LOW);
    delay(1000);

    digitalWrite(led, HIGH);
    delay(1000);
}
```



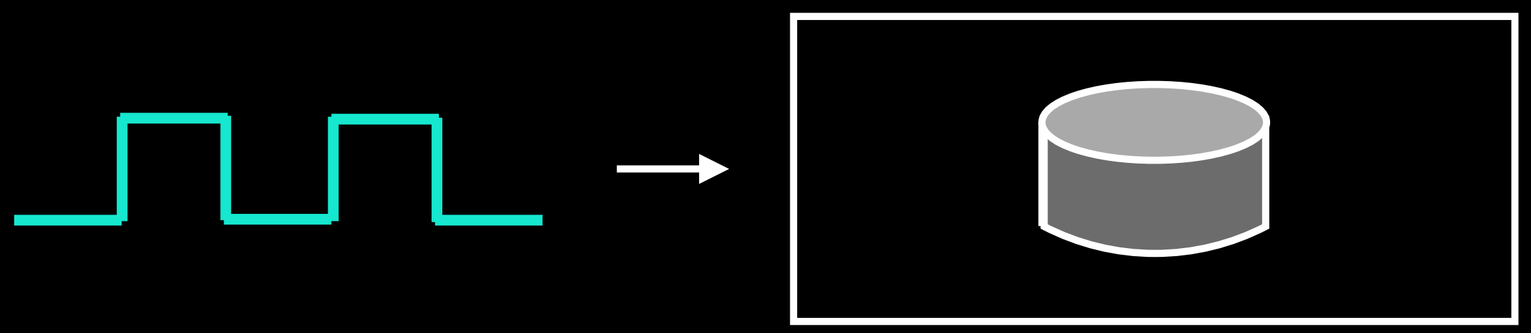
Piscando um LED

Campainha Passiva



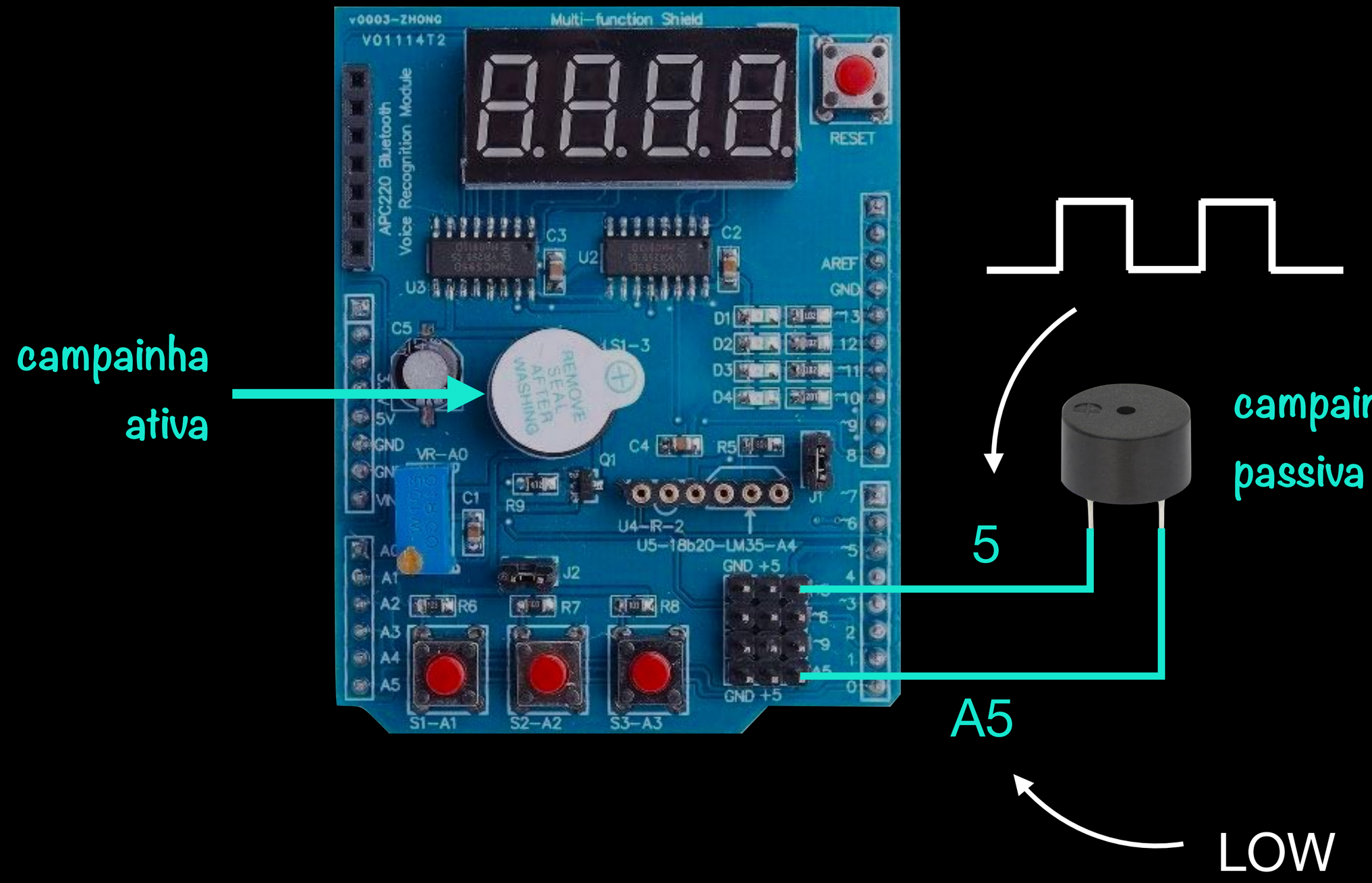
Geração de Som por Pulsos

campainha passiva



campainha ativa

Campainhas Ativas e Passivas

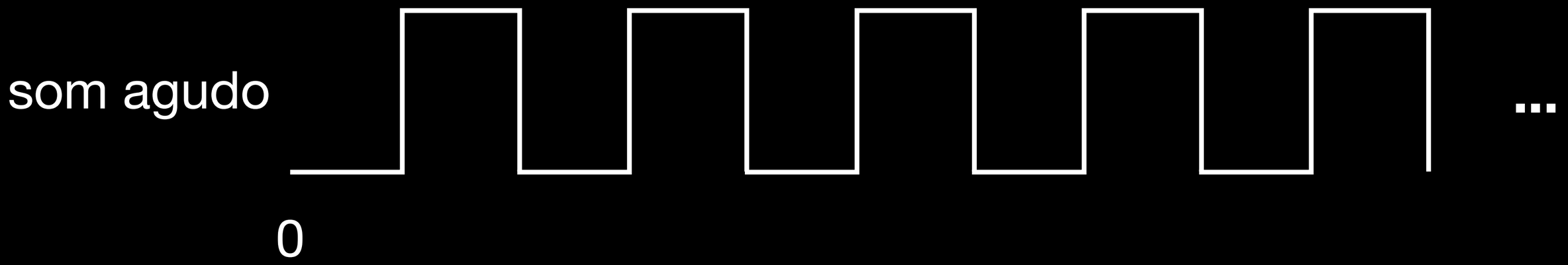


Campainha Passiva Conectada a Duas Portas de Uso Geral

pino

freqüência em Hz

```
tone(5, 440);
```

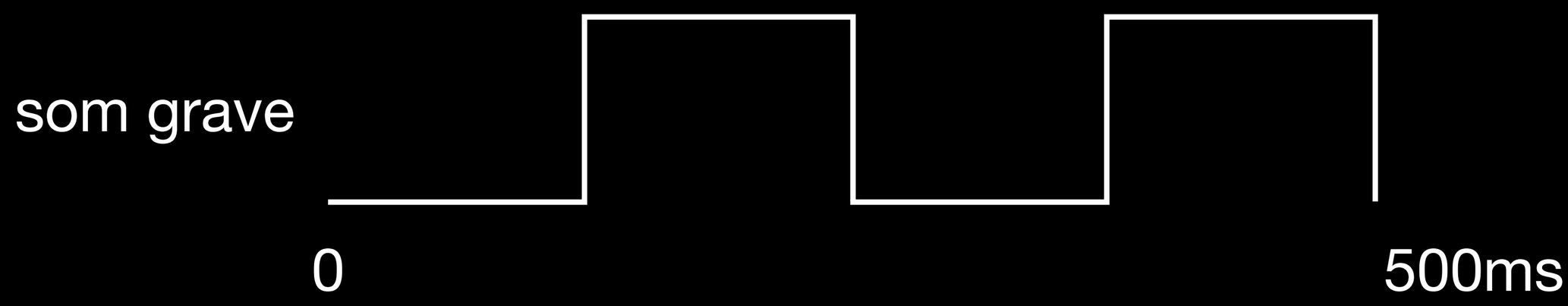



```
noTone(5)
```

(silêncio)

duração em ms (opcional)

```
tone(5, 220, 500);
```



Frequencia do Sinal para a Campainha

```
int terra = A5;
int campainha = 5;

void setup () {
    pinMode(terra, OUTPUT);
    digitalWrite(terra, LOW);

    pinMode(campainha, OUTPUT);

    // sinal de 220 Hz durante 500 milissegundos
    tone(campainha, 220.0, 500);
}
```

Toque da Campainha com Sinal

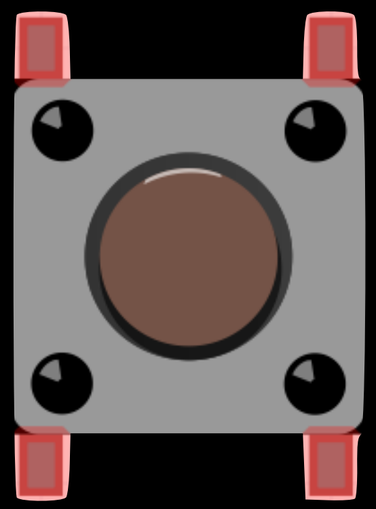
```
int terra = A5;
int campainha = 5;

void setup () {
    pinMode(terra, OUTPUT);
    digitalWrite(terra, LOW);

    pinMode(campainha, OUTPUT);

    tone(campainha, 220.0, 500);
    tone(campainha, 440.0, 500);
}
```

A função tone não trava
a execução do programa

Problema com Toque de Dois Tons Consecutivos

```
int terra = A5;
int campainha = 5;

void setup () {
    pinMode(terra, OUTPUT);
    digitalWrite(terra, LOW);

    pinMode(campainha, OUTPUT);
    // tone não trava a execução do programa
    tone(campainha, 220.0, 500);
    // portanto, temos que esperar um pouco...
    delay(500);
    // ... antes de tocar a próxima frequência
    tone(campainha, 440.0, 500);
}
```

Toque de Dois Tons Consecutivos com a Delay



Botão

HIGH
=
pressionado

VCC

LOW
=
solto

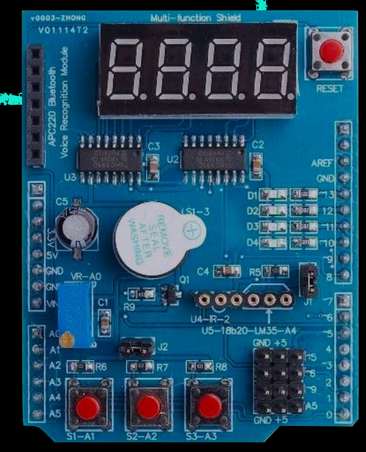
VCC

LOW
=
pressionado

VCC

HIGH
=
solto

VCC

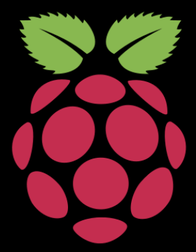


Duas Formas de Usar um Botão

pinos analógicos também podem ser usadas como portas digitais

```
int led = 13;
int botao = A1; // botão 1 está no pino Analógico 1

void setup () {
    pinMode(led, OUTPUT);
    pinMode(botao, INPUT);
}

void loop () {
    if ( digitalRead(botao) == LOW) ) {
        digitalWrite(led, LOW);
    }
    else {
        digitalWrite(led, HIGH);
    }
}
```

Configuração de um Botão



```python
# bibliotecas
from gpiozero import Button

# inicialização de componentes
botao = Button(11)
botao.when_pressed = funcao1
botao.when_released = funcao2
botao.when_held = funcao3
```

?

Eventos de Botão no Arduino?

Atlassian, Inc. bitbucket.org/geekfactory/gfbutton/overvie

GFButton

- Overview
- Source
- Commits
- Branches
- Pull requests
- Pipelines
- Downloads

GFButton Library

Arduino library to easily manage buttons and keys as objects on the arduino sketch. The GFButton class provides methods for polling and event based programming styles.

The objective of this library is to move the button logic outside of the arduino sketch in order to keep the code organized.

Basic library usage

The following example ilustrates the library usage. One button is used to turn on the led and other button is used to turn it off. When this example runs, pressing the ON button will print the message to the serial monitor many times, whereas the OFF button will print a message only once. See src/GFButton.h for full member documentation.

```
/**
   GeekFactory - "INNOVATING TOGETHER"
   Distribucion de materiales para el desarrollo e inn
   www.geekfactory.mx

   Basic example for the GFButton library. This exampl
   the polling (synchronous) API. This is the easiest
*/

#include "GFButton.h"

// Create two button instances on pins 2 & 3
GFButton buttonOn(2);
```

Jesus Ruben Santa Anna Zamudio · 2017-11-05

1 commit
Pushed to geekfactory/gfbutton
6967241 README.md edited online with ...

Jesus Ruben Santa Anna Zamudio · 2017-11-05

1 commit
Pushed to geekfactory/gfbutton
1abb33e Change version number

Jesus Ruben Santa Anna Zamudio · 2017-11-05

1 commit
Pushed to geekfactory/gfbutton
949c58f Added license information

Jesus Ruben Santa Anna Zamudio · 2017-11-05

2 commits
Pushed to geekfactory/gfbutton
a045beb Merge branch 'master' of https:/...
bc00663 Improve library description

Jesus Ruben Santa Anna Zamudio · 2017-11-05

1 commit
Pushed to geekfactory/gfbutton
f42e116 README.md edited online with ...

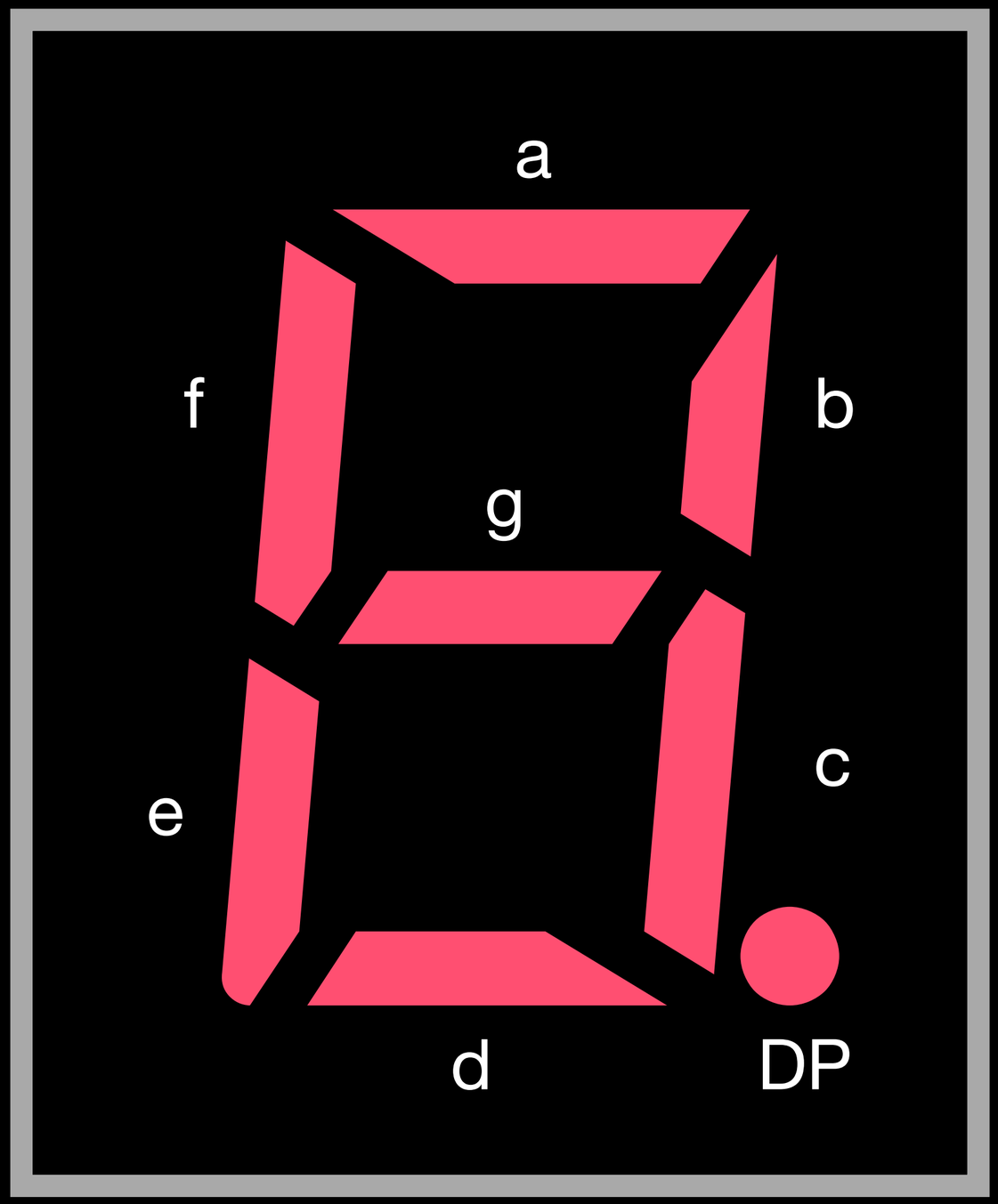
Biblioteca GFButton

```ino
#include <GFButton.h>

GFButton botao(A1);

void setup () {
    Serial.begin(9600);
    botao.setPressHandler(botaoPressionado);
    botao.setReleaseHandler(botaoSolto);
}

void loop () {
    botao.process();
}

void botaoPressionado (GFButton& botaoDoEvento) {
    Serial.println("Botão foi pressionado!");
}

void botaoSolto (GFButton& botaoDoEvento) {
    Serial.println("Botão foi solto!");
}
```

ATENÇÃO! Não esqueça desta chamada!

Eventos de um Botão

Displays de 7 Segmentos

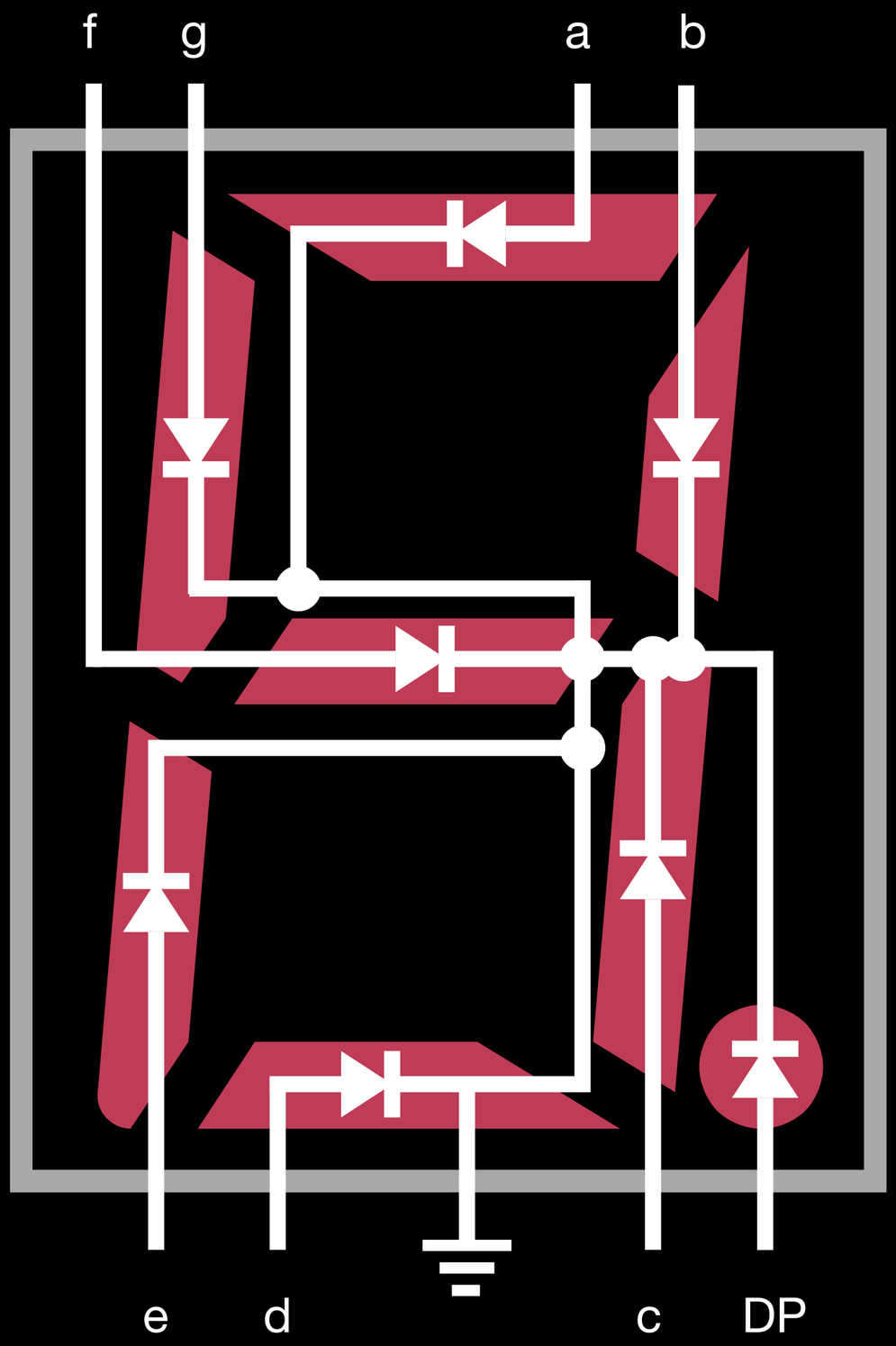


a	b	c	d	e	f	g	DP	
1	1	1	1	1	1	1	1	
0	0	0	0	0	0	0	0	
1	1	0	1	1	0	1	1	
0	1	1	1	1	0	0	0	

Elementos de um Display de 7 Segmentos

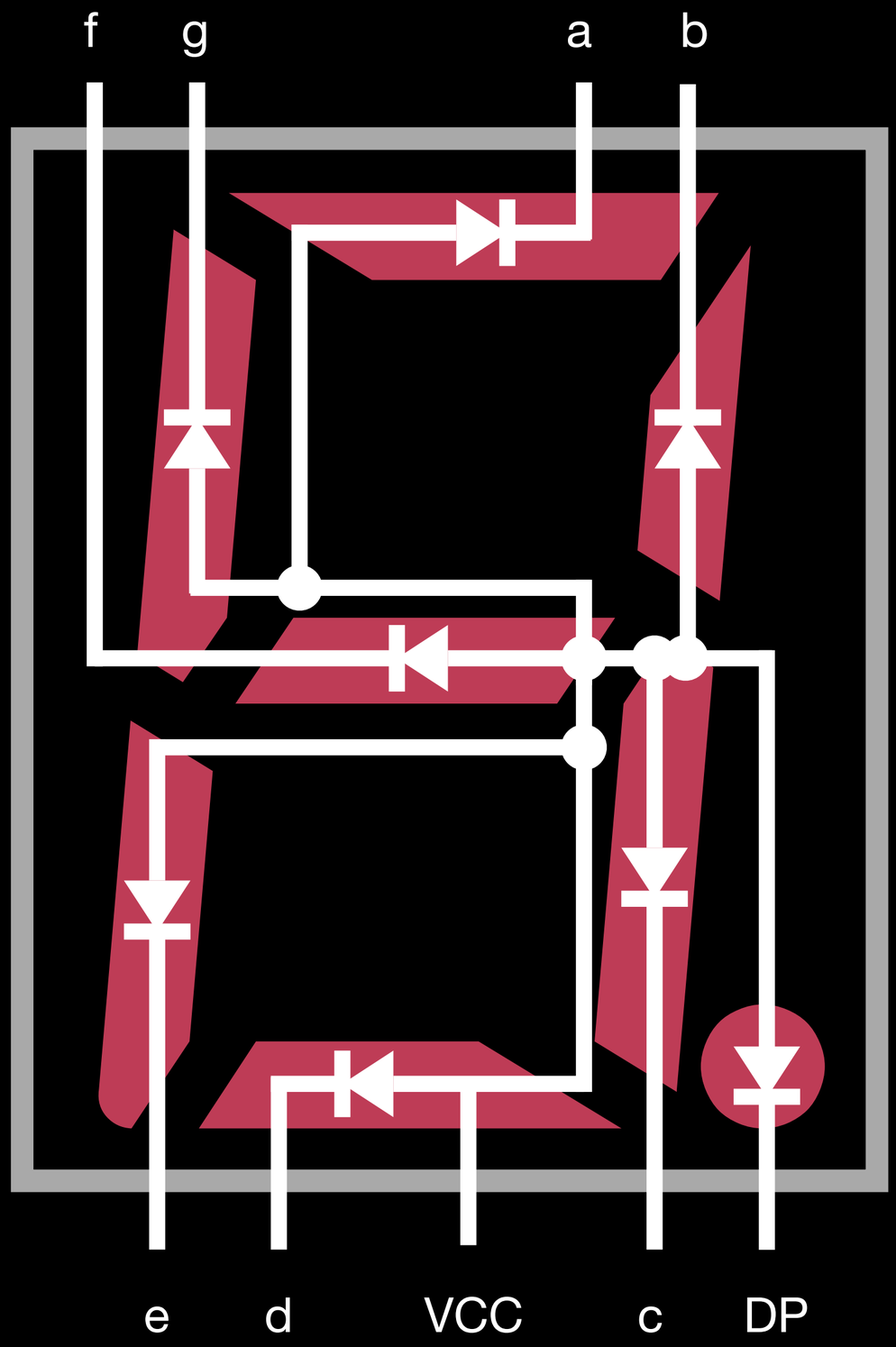
catodo comum

acende segmentos com HIGH (3.3V)

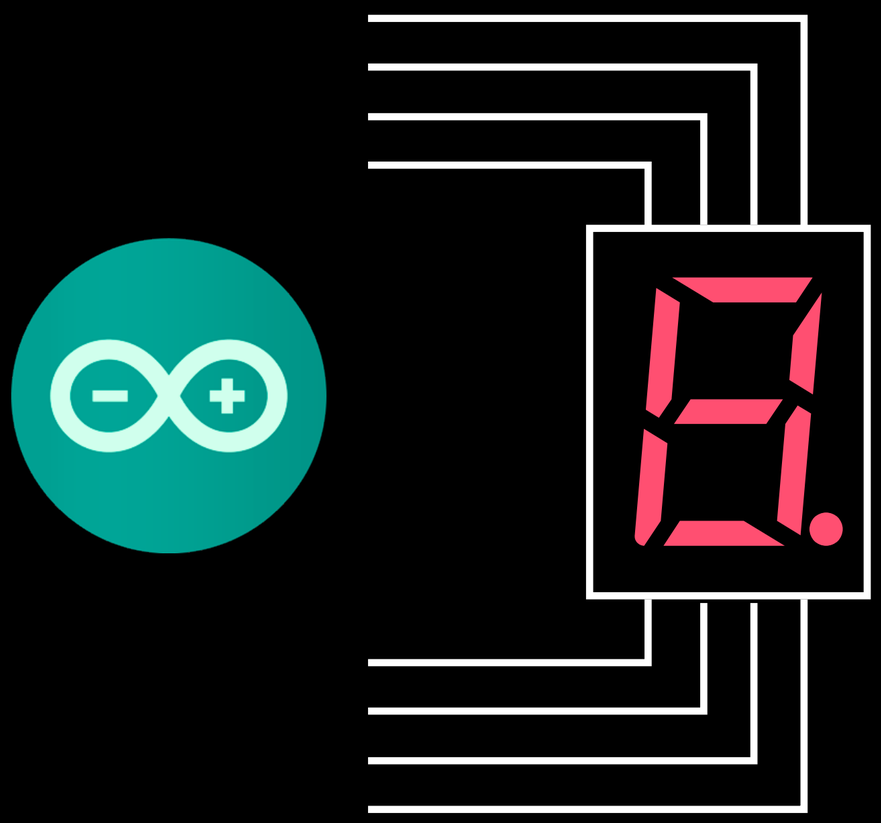


anodo comum

acende segmentos com LOW (0V)

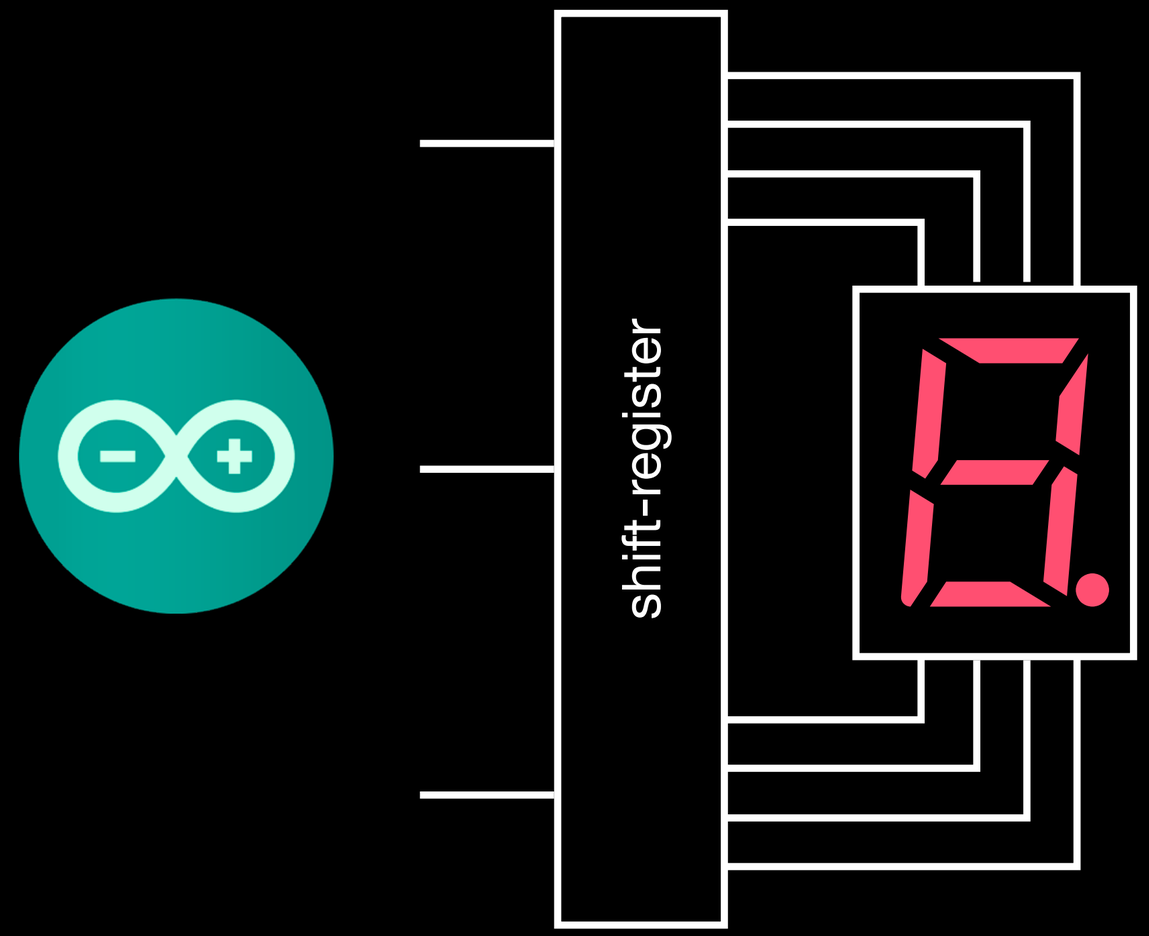


LEDs Dentro de um Display de 7 Segmentos



controle paralelo

8 conexões + terra/VCC



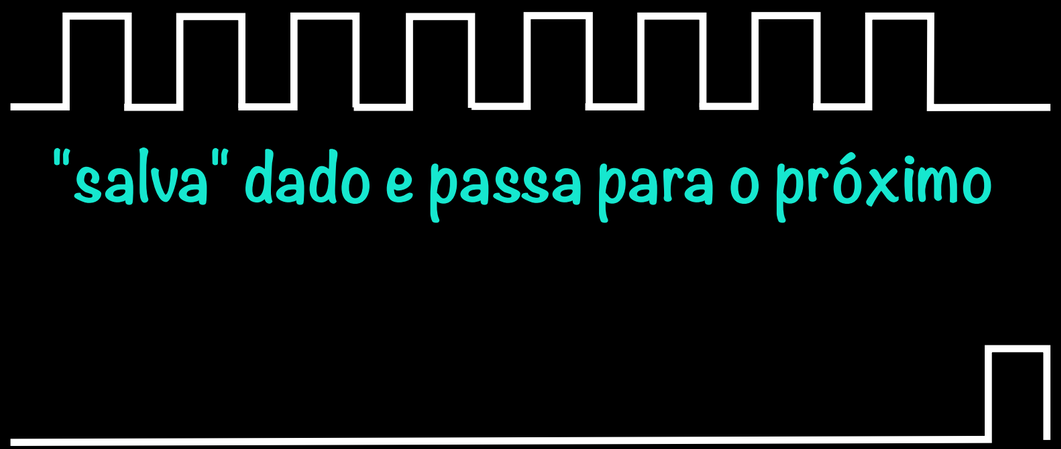
controle serial

3 conexões + terra/VCC

Controle Paralelo vs Serial

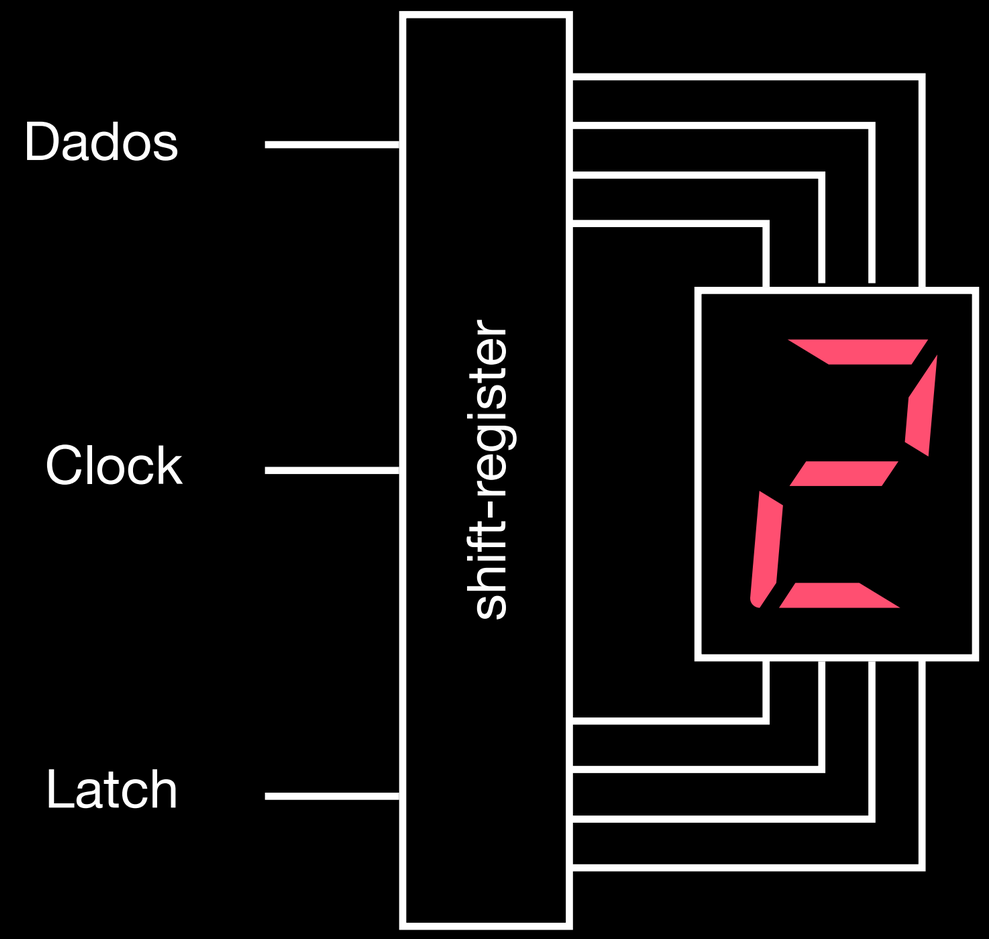
1 1 0 1 1 0 1 0

dados de cada segmento

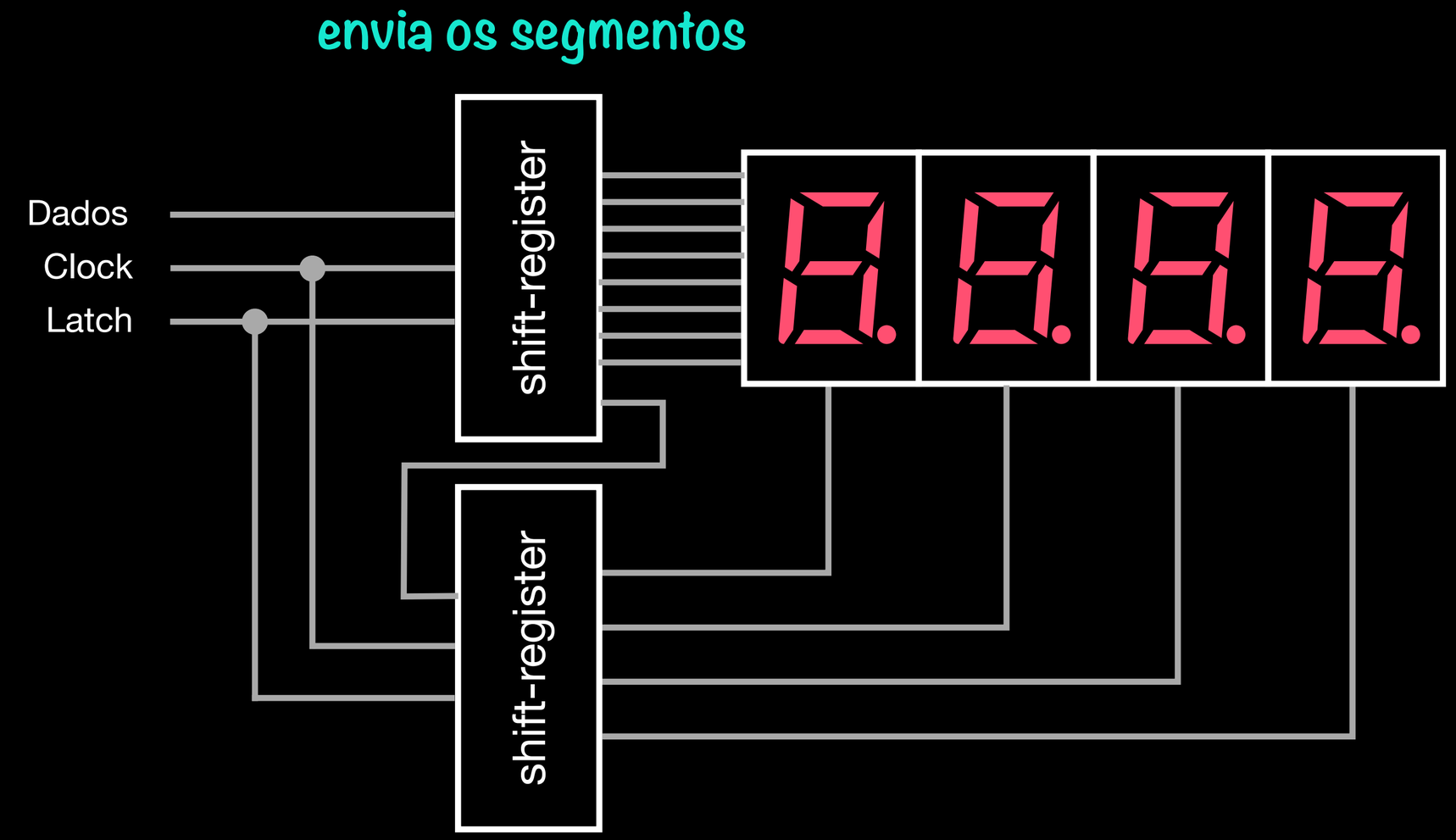


"salva" dado e passa para o próximo

envia dados para as saídas

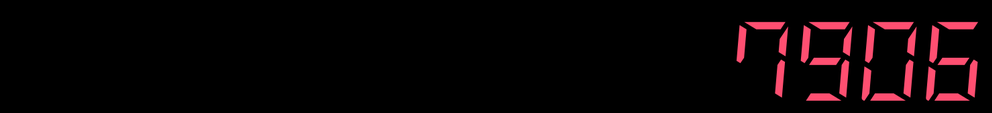


Envio Serial de Dados para o Display

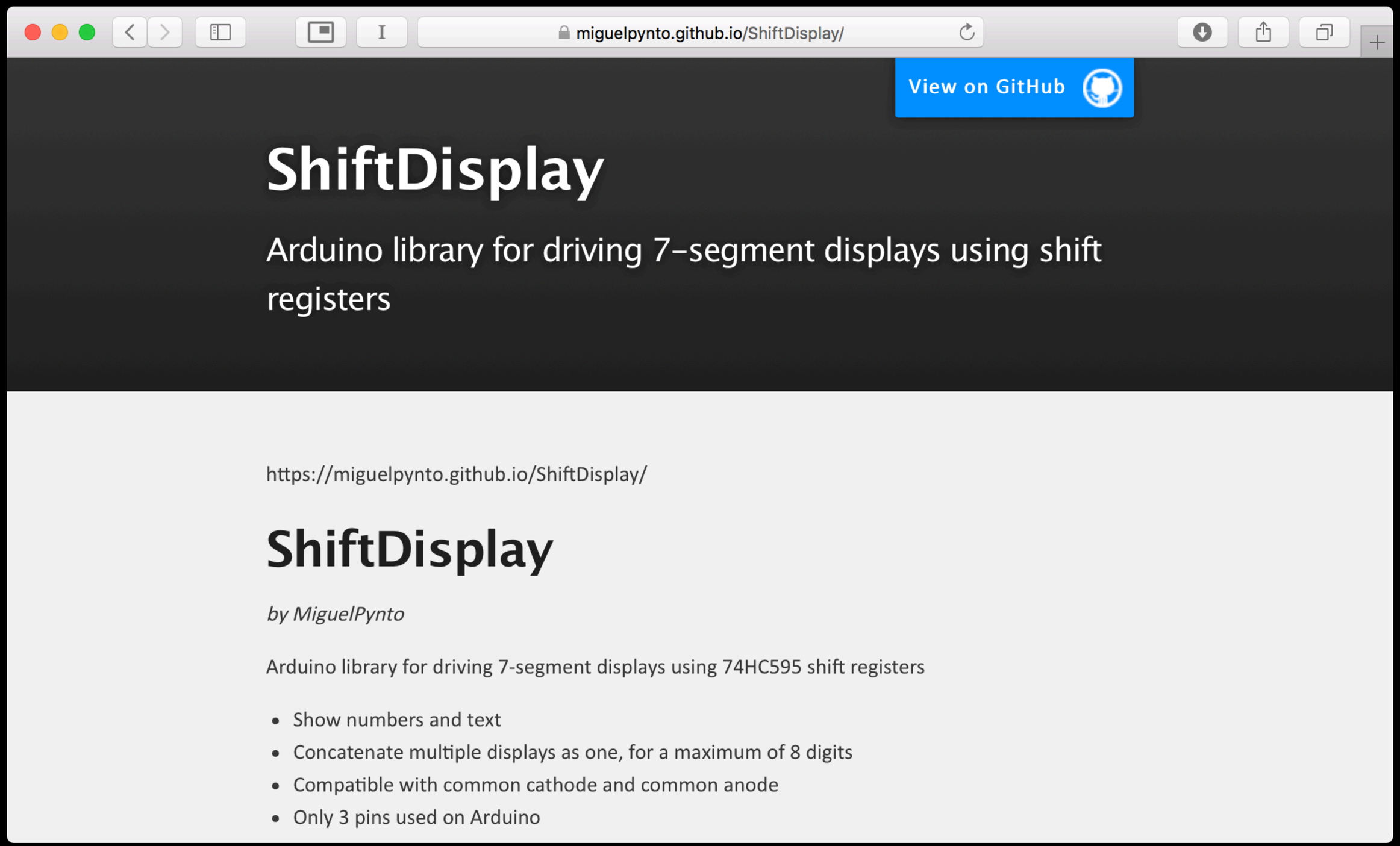


Controle Serial de Vários Displays

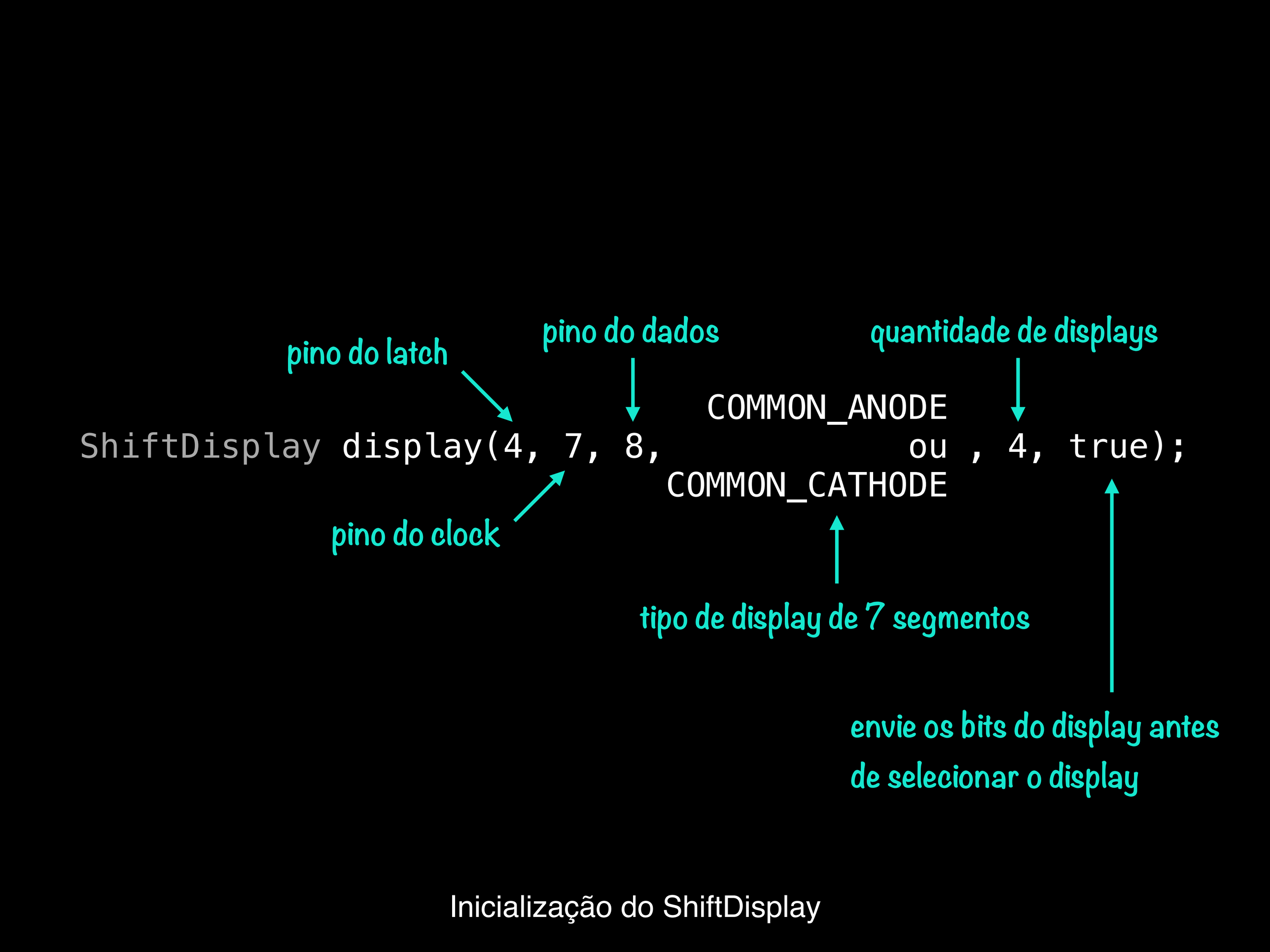
≈



Alternância Rápida de Displays para Exibir os Dígitos



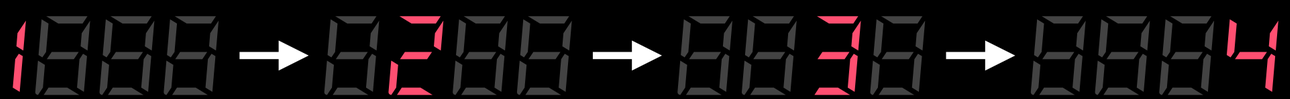
Biblioteca ShiftDisplay



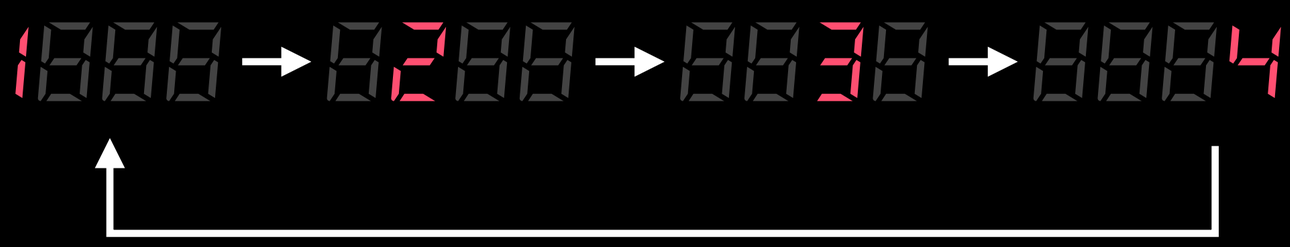
Inicialização do ShiftDisplay

```
// apenas salva o que será exibido
display.set(1234);
```

```
// alterna uma vez os dígitos
display.update();
```

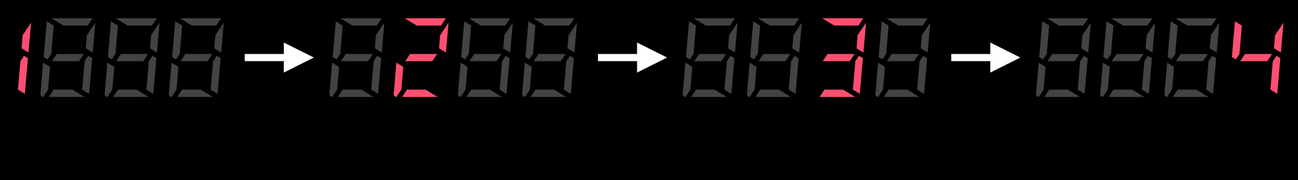


```
// alterna dígitos durante 1000 ms
display.show(1000);
```



ATENÇÃO: a função show trava
a execução do programa!

```
#include <ShiftDisplay.h>

ShiftDisplay display(4, 7, 8, COMMON_ANODE, 4, true);

void setup () {
    display.set(1234);
}

void loop () {
    display.update();
}
```



≈

Exibição de Dígitos Numéricos

```
#include <ShiftDisplay.h>

ShiftDisplay display(4, 7, 8, COMMON_ANODE, 4, true);

void setup () {
    display.set(1234);
    display.show(1000);
}

void loop () {
    display.set(4321);
    display.update();
}
```

1234

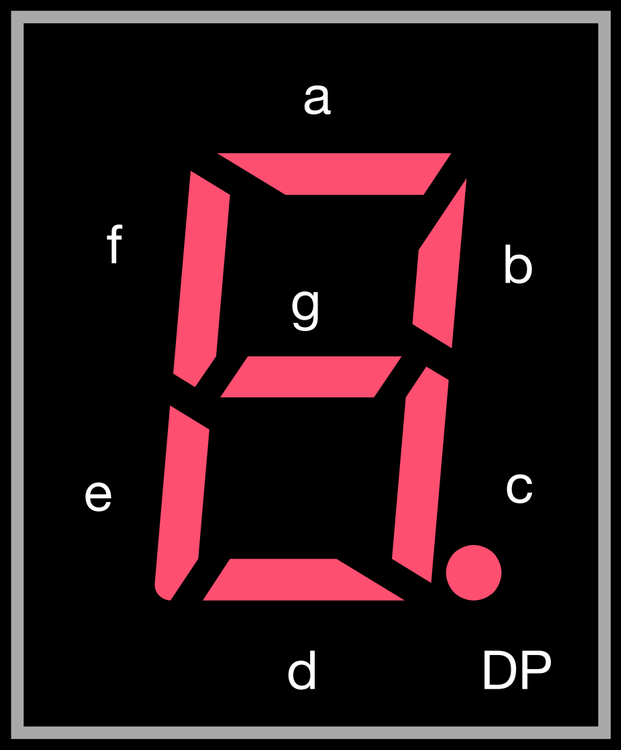
(exibe 1 vez durante 1000 ms)

4321

Exibição de Dígitos Numéricos: Show x Update

```
display.set(4321);                          4321
display.changeDot(0, true); // ponto no índice 0 4.321

display.set(4.56);                          0046
display.set(4.56, 2);       // 2 casas decimais  0456
display.set(4.56, 2, true); // zeros à esquerda  0456

display.set("ERRO");                        Erro
display.set("oi");                          00oi
display.set("oi", ALIGN_CENTER);            0oi0
```

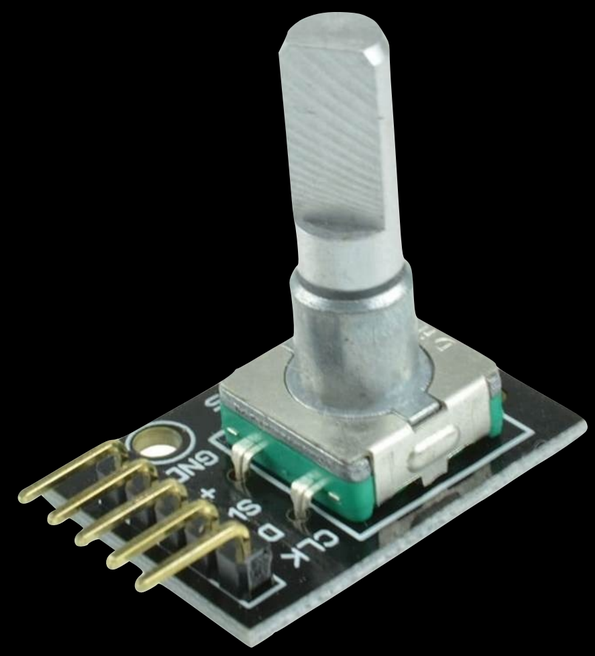
Outras Função de Exibição



a	b	c	d	e	f	g	DP	
1	1	1	1	1	1	1	1	
0	0	0	0	0	0	0	0	
1	1	0	1	1	0	1	1	
0	1	1	1	1	0	0	0	

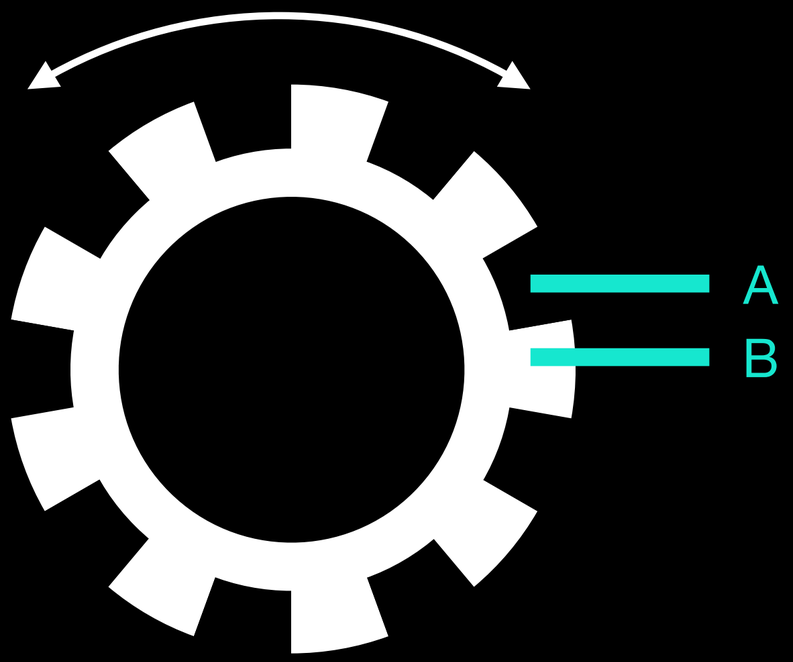
```
byte x[] = {0b11111111, 0b00000000, 0b01101111, 0b01111000};
display.set(x);
...
```

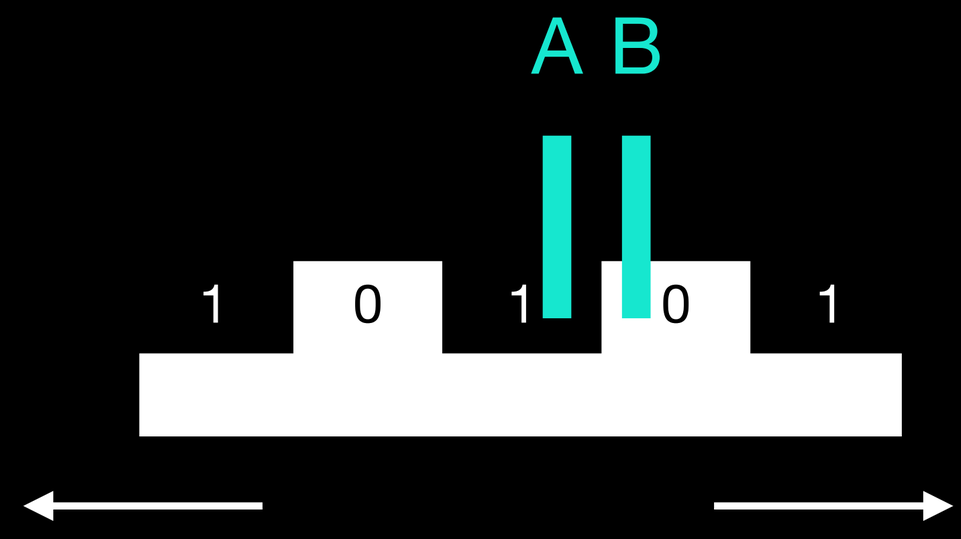
Configuração Manual de Segmentos



Encoder Rotativo

Exemplo de Encoder Rotativo para Controle Sonoro





A	B
1	0
0	0
0	1
1	1

A	B
1	0
1	1
0	1
0	0

Funcionamento Interno de um Encoder

www.mathertel.de/Arduino/RotaryEncoderLibrary.aspx

mathertel.de

Arduino | Diff | AJAXEngine | OpenAjax

www.mathertel.de > Arduino Projects > Arduino Rotary Encoder Library

Sitemap
Impressum
License Agreement

A Library for the Arduino environment for using a rotary encoder as an input.

Here you can find an Arduino compatible library for using rotary encoders.

I was searching a library for using a rotary encoder in my latest project and found a lot of information on this topic but none of the existing libraries did immediately match my expectations so I finally built my own.

This article likes to explain the software mechanisms used in detail so you can understand the coding and might be able to adjust it to your needs if you like. There are various aspects when writing a library for rotary encoders and you can also find a lot of the sources I analyzed at the bottom of this article.

Download

You can download the library and examples directly from the github repository that you can find at:

- https://github.com/mathertel/RotaryEncoder

Use the "Download zip file" button to get all the files and put them into your Sketches /libraries folder.

Rotary Encoder signals

The signals a rotary encoder produces (and what can be handled by this library) are based on a 2-bit gray

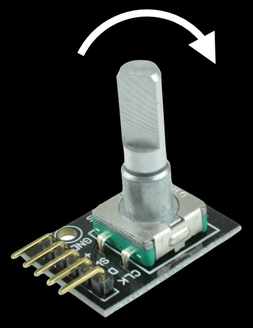
Biblioteca RotaryEncoder

verifica estado das saídas, para contar os giros

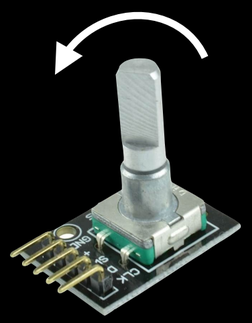
```
encoder.tick()
```

retorna a posição atual de giro

```
int x = encoder.getPosition()
```



0 → 1 → 2 → 3 → 4 → ...



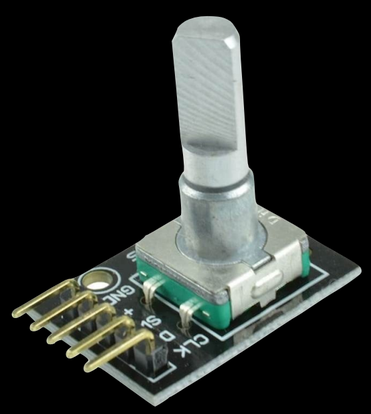
3 → 2 → 1 → 0 → -1 → ...

Ideia Geral da Biblioteca RotaryEncoder

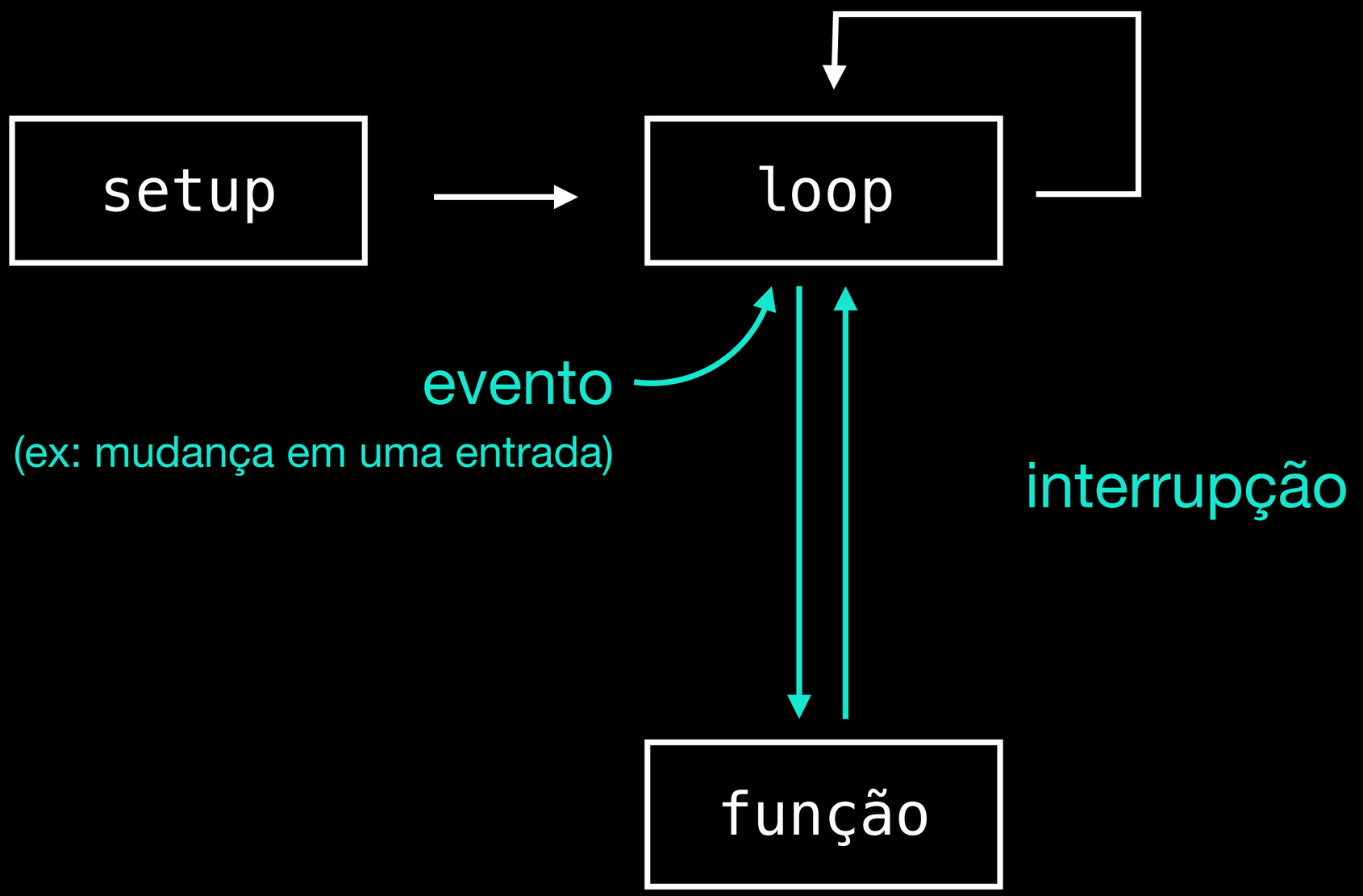
```
void loop() {
    display.update();
    encoder.tick();
}
```

demora 4 milissegundos

O giro rápido não vai funcionar assim...



Ideia Geral da Biblioteca RotaryEncoder



Interrupção do Loop

Modelo de Arduíno	Pinos
Uno, Nano, Mini	2, 3
Mega, Mega2560, MegaADK	2, 3, 18, 19, 20, 21
Micro, Leonardo	0, 1, 2, 3, 7
Zero	all digital pins, except 4
MKR1000 Rev.1	0, 1, 4, 5, 6, 7, 8, 9, A1, A2
Due, 101	todos

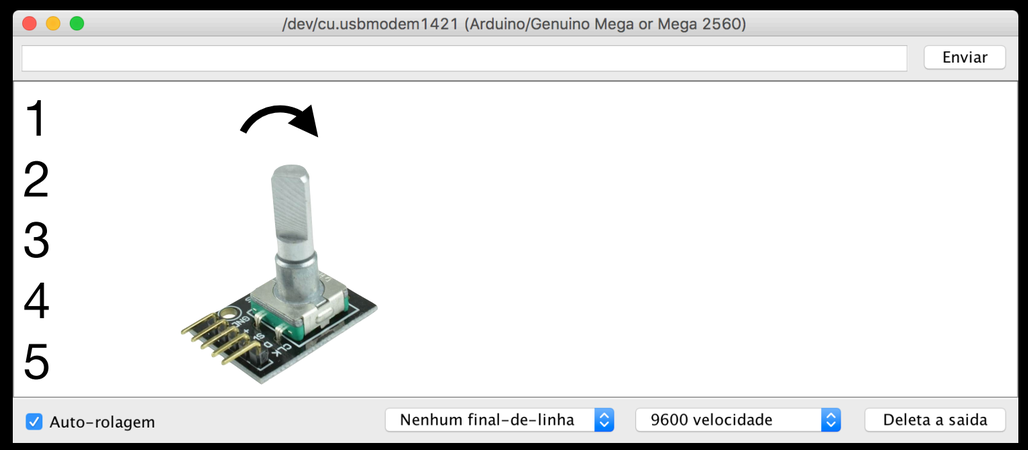
Pinos com Recurso de Interrupção

```
int origem = digitalPinToInterrupt(pino);
attachInterrupt(origem, funcaoParaChamar, TIPO);
```

Tipo	Evento
RISING	entrada passa de LOW para HIGH
FALLING	entrada passa de HIGH para LOW
CHANGE	qualquer mudança na entrada

Função para Configurar Interrupção

```ino
#include <RotaryEncoder.h>

RotaryEncoder encoder(20, 21);
int posicaoAnterior = 0;
void setup() {
    Serial.begin(9600);
    int origem1 = digitalPinToInterrupt(20);
    attachInterrupt(origem1, tickDoEncoder, CHANGE);
    int origem2 = digitalPinToInterrupt(21);
    attachInterrupt(origem2, tickDoEncoder, CHANGE);
}

void tickDoEncoder() {
    encoder.tick();
}

void loop() {
    int posicao = encoder.getPosition();
    if (posicao != posicaoAnterior) {
       Serial.println(posicao);
    }
    posicaoAnterior = posicao;
}
```

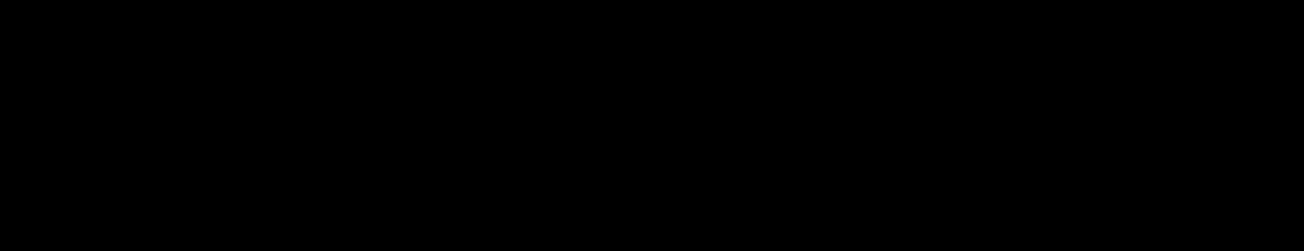


Exemplo da RotaryEncoder com Interrupção

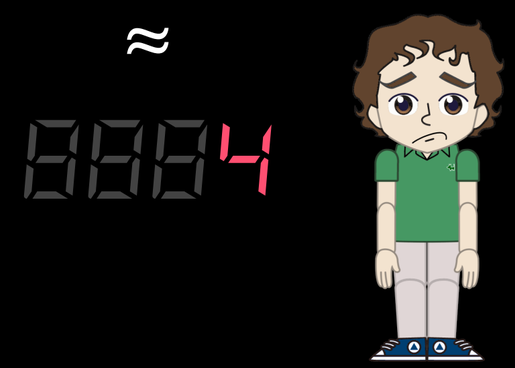
"Software"

```ino
#include <ShiftDisplay.h>

ShiftDisplay display(4, 7, 8, COMMON_ANODE, 4, true);
void setup () {
    Serial.begin(9600);
    display.set(1234);
}
```

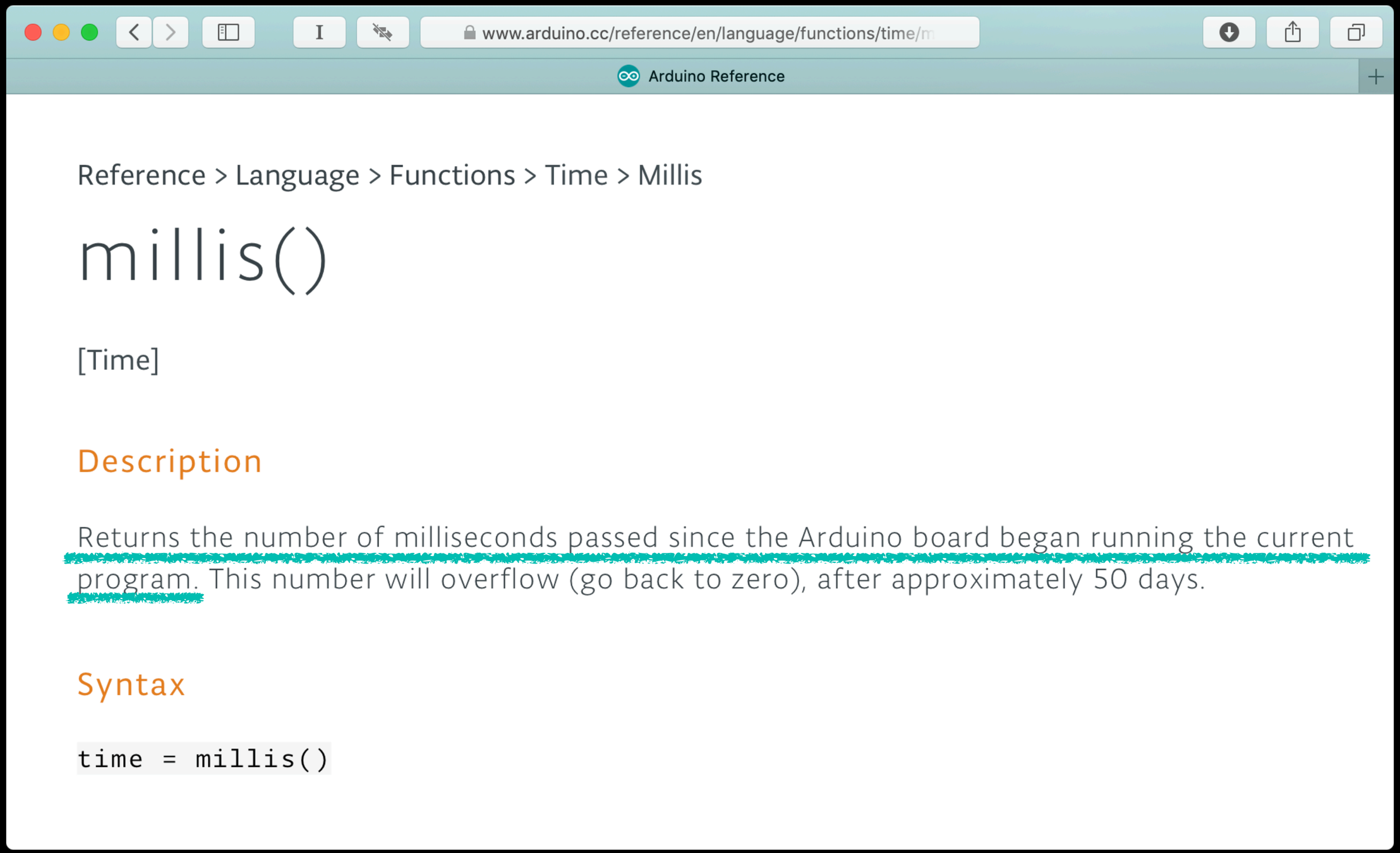
```ino
void loop () {
    display.update();
    delay(1000);
    Serial.println("+1 segundo");
}
```



≈



Exemplo de Problema com a Função Delay



Contagem de Milissegundos com a Millis

Exemplo de Contagem de Milissegundos com a Millis

playground.arduino.cc/Code/DatatypePractices/

Arduino Playground - Datatype Practices

HOME STORE SOFTWARE EDU RESOURCES COMMUNITY HELP

ion index

Arduino reference lists these datatypes:

Datatype	RAM usage
void keyword	N/A
boolean	1 byte
char	1 byte
unsigned char	1 byte
int	2 byte
unsigned int	2 byte
word	2 byte
long	4 byte
unsigned long	4 byte
float	4 byte

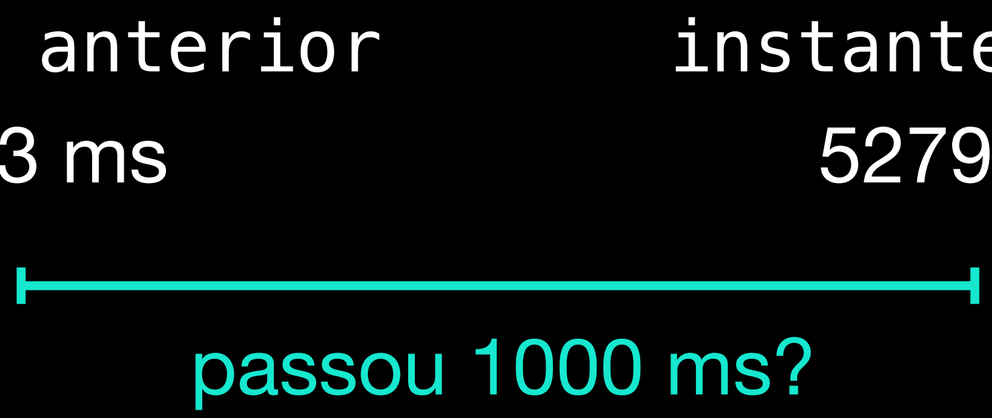
máximo = 32767

máximo = 65535

máximo =4294967295

Usamos unsigned long para poder contar tempo sem esbarrar no limite de 2 bytes.

Capacidade dos Números Inteiros no Arduino




```
loop:
    se tiver passado 1 segundo desde o instante anterior:
        imprime texto na serial
        salva instante de tempo atual
```

Pseudo-Algorítmo para Contagem de Tempo

```
#include <ShiftDisplay.h>

ShiftDisplay display(4, 7, 8, COMMON_ANODE, 4, true);
unsigned long instanteAnterior = 0;

void setup () {
    Serial.begin(9600);
    display.set(1234);
}

void loop () {
    display.update();

    unsigned long instanteAtual = millis();
    if (instanteAtual > instanteAnterior + 1000) {
        Serial.println("+1 segundo");
        instanteAnterior = instanteAtual;
    }
}
```

Contagem de Tempo com a Função Millis

```
unsigned long instanteDaContagem1 = 0;
unsigned long instanteDaContagem2 = 0;
unsigned long instanteDaContagem3 = 0;

void loop () {
    unsigned long instanteAtual = millis();
    if (instanteAtual > instanteDaContagem1 + 1000) {
        ...
    }

    if (instanteAtual > instanteDaContagem2 + 200) {
        ...
    }

    if (instanteAtual > instanteDaContagem3 + 8000) {
        ...
    }

    ...
}
```

Várias Contagens de Tempo com a Função Millis

Resumo da Opera

Funcionalidad	Comandos

Revisão de C++	`int inteiro = 2; float decimal = 4.5; bool booleano = true;`
	`char texto[] = "Olá"; int listaDeInteiros[] = {1, 2, 3, 4};`
	`if (x > 0 && y > 0) {`
	`  z = 1;`
	`}`
	`else if (x < 0 || y < 0) {`
	`  z = 2;`
	`}`
	`for (int i = 0; i < 5; i++) {`
	`  Serial.println(i);`
	`}`
	`float soma (float x) {`
	`  return x + 2;`
	`}`

Print Serial	`Serial.begin(9600); Serial.println("Olá"); Serial.println(2);`
LEDs documentação	`int led = 13; pinMode(led, OUTPUT);` `digitalWrite(led, LOW); digitalWrite(led, HIGH);`

GFButton
documentação
	`#include <GFButton.h>`
	`GFButton botao(A1); botao.isPressed(); botao.process();`
	`botao.setPressHandler(funcao); botao.setReleaseHandler(funcao);`

ShiftDisplay
documentação
	`#include <ShiftDisplay.h>`
	`ShiftDisplay display(4, 7, 8, COMMON_ANODE, 4, true);`
	`ShiftDisplay display(4, 7, 8, COMMON_CATHODE, 4, true);`
	`display.set(1234); display.set(4.21, 2); display.set("Erro");`
	`display.update(); display.show(1000); display.changeDot(0, true)`

Resumo para o Projeto 06 + 07 – Parte 1

Funcionalidade	Comandos

Campainha Passiva
documentação
	`int campainhaPassiva = 5;`
	`pinMode(campainhaPassiva, OUTPUT);`
	`int frequencia = 220; int duracaoEmMs = 500;`
	`tone(campainhaPassiva, frequencia);`
	`tone(campainhaPassiva, frequencia, duracaoEmMs);`
	`noTone(campainhaPassiva);`

Contagem de Tempo
documentação
	`unsigned long instanteAnteriorDeDeteccao = 0;`
	
	`if (millis() > instanteAnteriorDeDeteccao + 10) {`
	`instanteAnteriorDeDeteccao = millis();`
	`}`

Encoder Rotativo
documentação
	`#include <RotaryEncoder.h>`
	`RotaryEncoder encoder(20, 21);`
	`int origem1 = digitalPinToInterrupt(20);`
	`attachInterrupt(origem1, tickDoEncoder, CHANGE);`
	`int origem2 = digitalPinToInterrupt(21);`
	`attachInterrupt(origem2, tickDoEncoder, CHANGE);`
	`encoder.tick(); int posicao = encoder.getPosition();`
	`encoder.setPosition(posicao);`