Introdução

Revisão de Microcontroladores – Teoria

Jan K. S. – janks@puc-rio.br

ENG4051 – Projeto Internet das Coisas

Olá Novamente!

IoT

PUC
RIO

Disciplina de Projeto de Internet das Coisas na PUC-Rio

O que é IoT (Internet das Coisas)?

“A Internet das Coisas (IoT) descreve dispositivos com sensores, capacidade de processamento, software e outras tecnologias que se conectam e trocam dados com outros dispositivos e sistemas, via Internet ou outras redes de comunicação.”

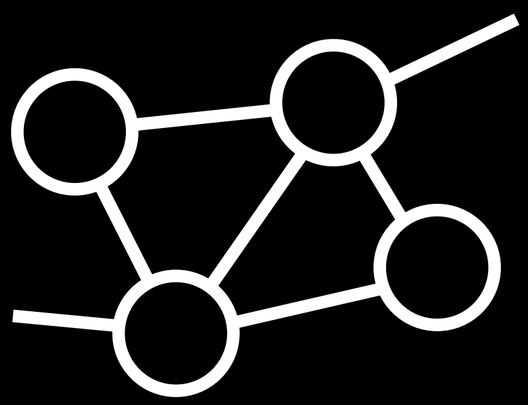
Wikipedia

	A	B	C	D	E	F
5	**Data**	**Luz 1**	**Luz 2**	**Luz 3**	**Luz 4**	**Luz 5**
6	September 17, 2018 at 12:12PM	0	0	0	0	
7	September 17, 2018 at 12:13PM	8	5	4	4	
8	September 17, 2018 at 12:14PM	8	5	4	8	
9	September 17, 2018 at 12:15PM	0	0	0	0	
10	September 17, 2018 at 12:16PM	0	0	0	0	
11	September 17, 2018 at 12:17PM	0	0	10	13	
12	September 17, 2018 at 12:18PM	51	39	18	0	
13	September 17, 2018 at 12:19PM	1	1	0	0	
14	September 17, 2018 at 12:21PM	0	0	0	0	
15	September 17, 2018 at 12:21PM	29	0	0	0	
16	September 17, 2018 at 12:22PM	29	0	0	3	
17	September 17, 2018 at 12:23PM	0	0	0	3	
18	September 17, 2018 at 12:24PM	0	0	0	0	

Página1

A gente já não viu Internet das Coisas
na G1 de Projeto de Microcontroladores?

G1 de Projeto de Microcontroladores

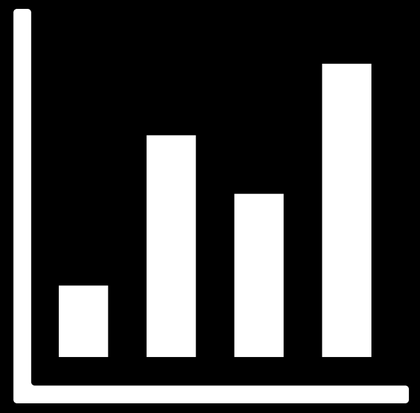


Múltiplas
Conexões

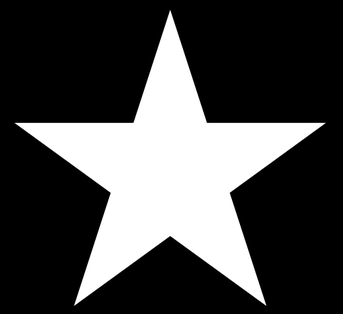
Resiliência a
Problemas

Autenticação
e Segurança

Baixo Consumo
Energético

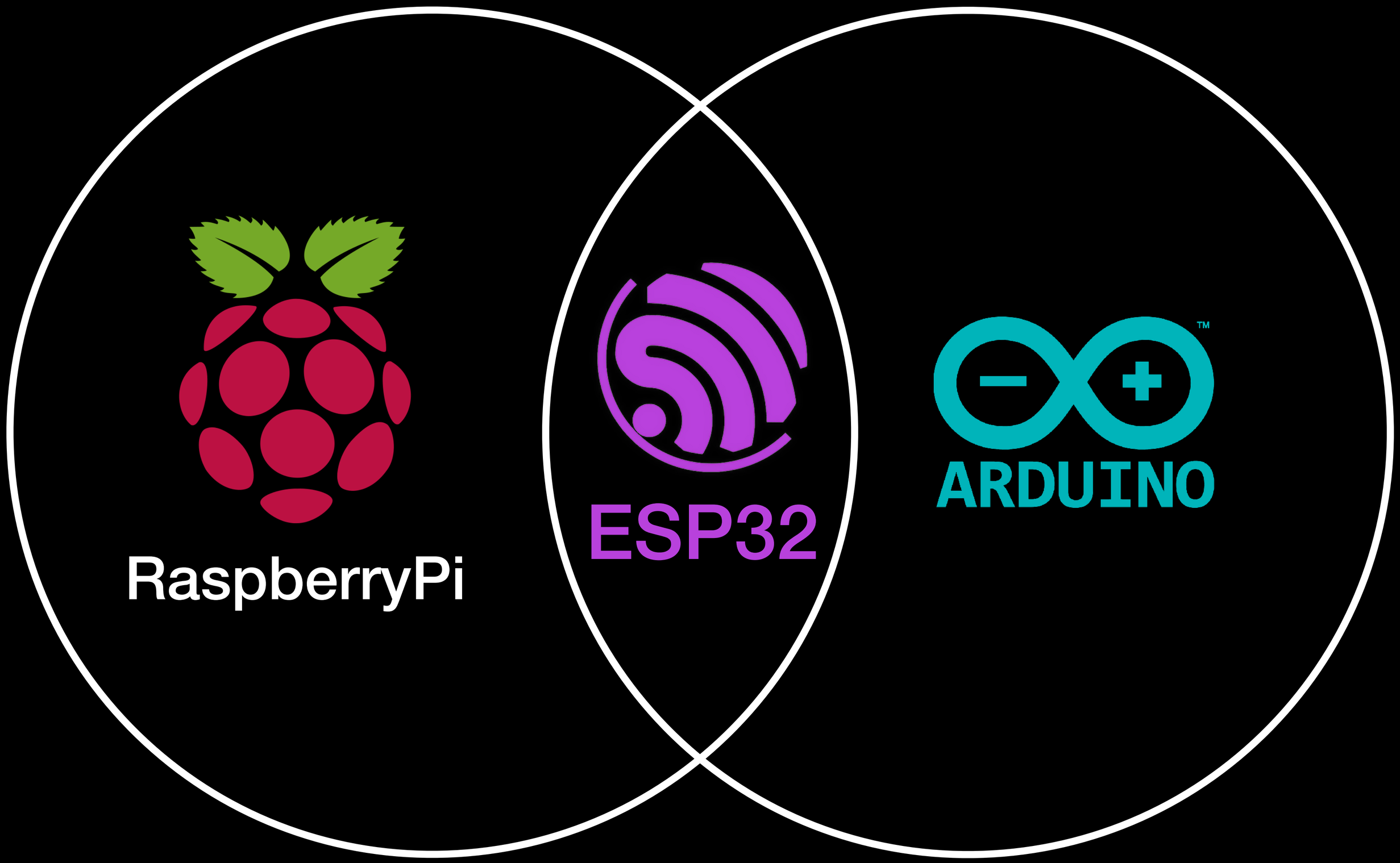


Análise dos
Dados

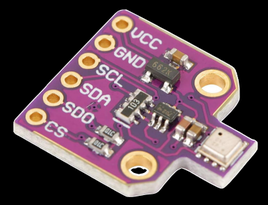


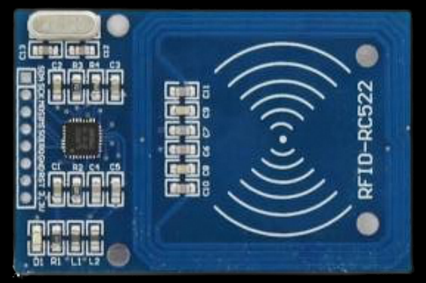
Aparência
Bonita

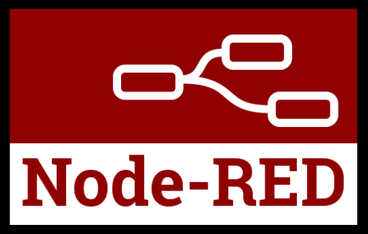
Requisitos Importantes num Produto Real



Microcontrolador Usado no Curso







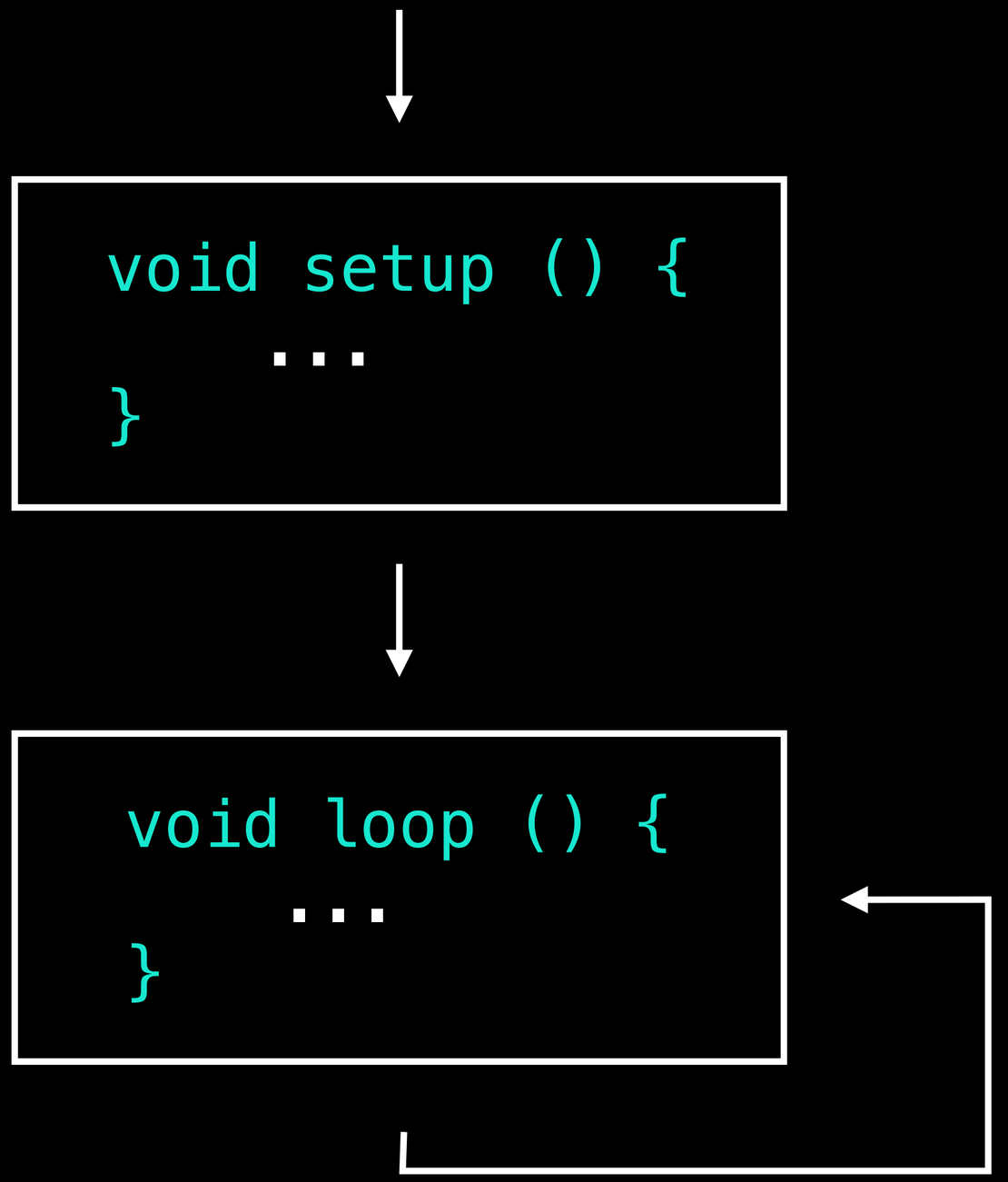
ESP32

Exemplos de Hardware e Software Vistos no Curso

Revisão de Microcontroladores

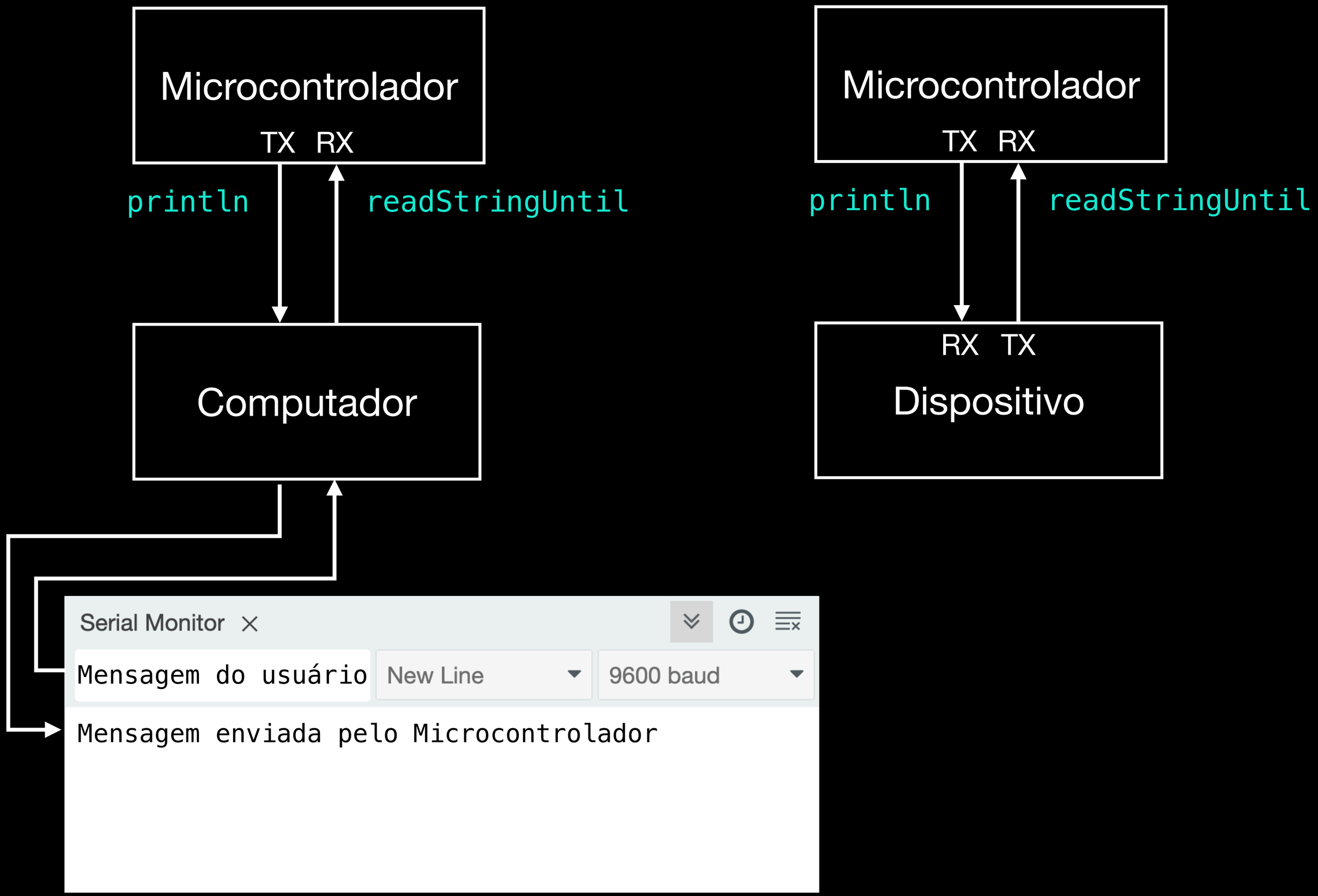
Arduino

Início do Programa



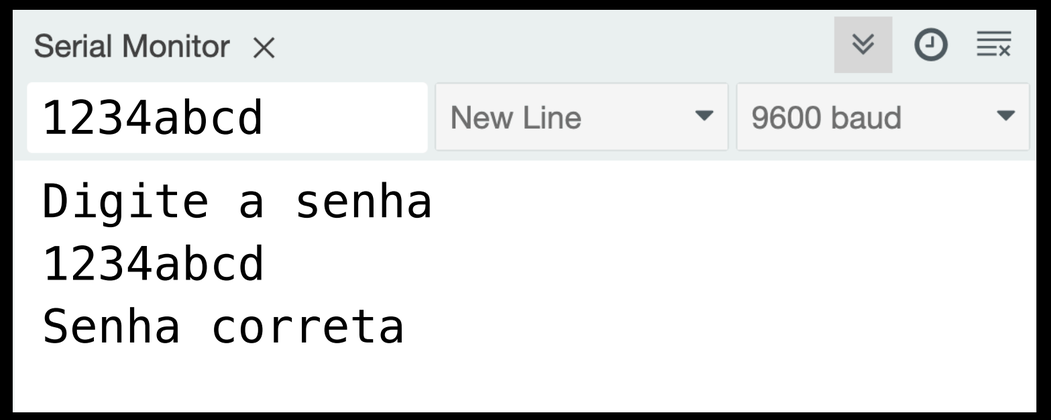
Rotina de Execução de Código em um Arduino

Comunicação Serial (UART)



Comunicação entre Dispositivos via Serial (UART)

```ino
void setup () {
  Serial.begin(9600);
  Serial.println("Digite a senha");
}

void loop () {
  if (Serial.available() > 0) {
    String texto = Serial.readStringUntil('\n');
    Serial.println(texto); // imprime o que foi recebido
    if (texto == "1234abcd") {
      Serial.println("Senha correta!");
    }
    else {
      Serial.println("Senha incorreta!");
    }
  }
}
```

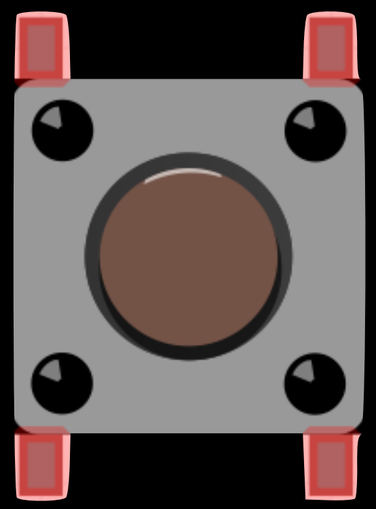


Escrita e Leitura Serial

```
String texto1 = "Olá, mundo!";

int numero = 100 * 2;
String texto2 = String(numero);                    // → "200"
int numero2 = texto2.toInt() + 2;                  // → 202

String texto3 = "aaa" + texto2;                    // → "aaa200"

bool ehIgual = texto2 == texto3;                   // → false
bool comecaComOla = texto1.startsWith("Olá"); // → true
```

```
char caracter = texto1[2];                         // → 'á'
int totalCaracteres = texto1.length();             // → 11

String trecho = texto1.substring(0, 3);            // → "Olá"
String trechoFinal = texto1.substring(5);          // → "mundo!"

String texto4 = "abc abc";
texto4.replace("ab", "AB");                        // → "ABc ABc"
```

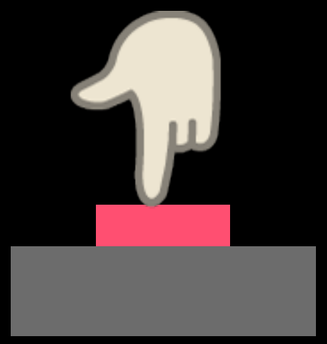
Exemplos de Funções para Strings



Botão

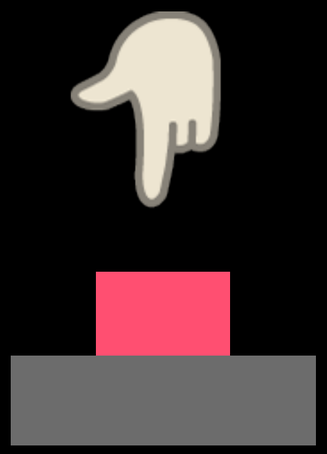
~~loop:~~
~~se botão estiver pressionado:~~
~~faça algo~~
~~caso contrário:~~
~~faça outra coisa~~

Quase sempre a gente quer
fazer algo apenas 1 vez
ao pressionar o botão



evento
"botão pressionado"

chama função
1 vez



evento
"botão liberado"

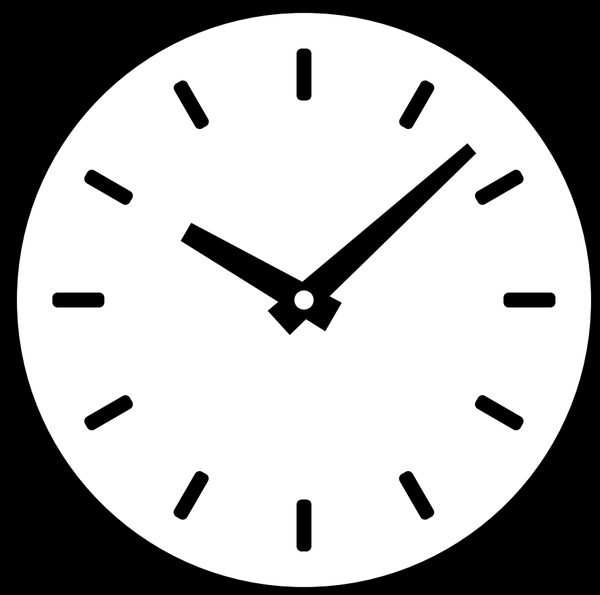
chama função
1 vez

Eventos de Apertar e Soltar um Botão

```ino
#include <GFButton.h>

GFButton botao(11); // conectado no pino 11

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

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

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

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

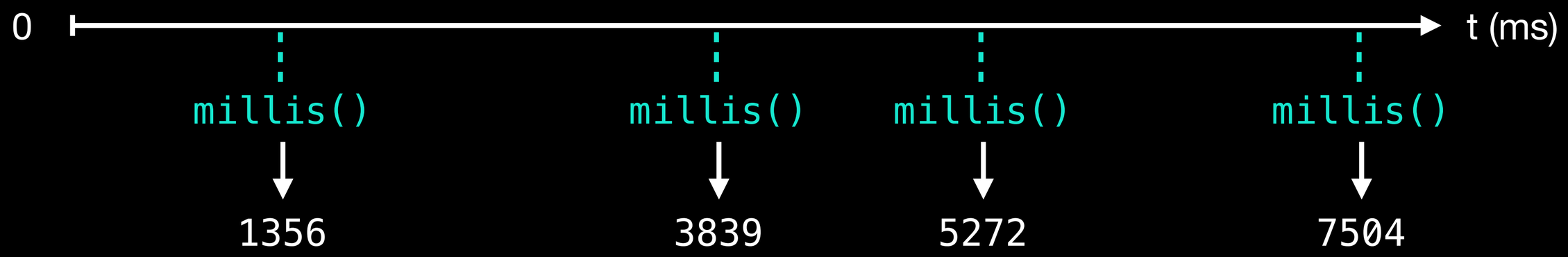
ATENÇÃO! Não se esqueça deste comando!

Eventos de um Botão

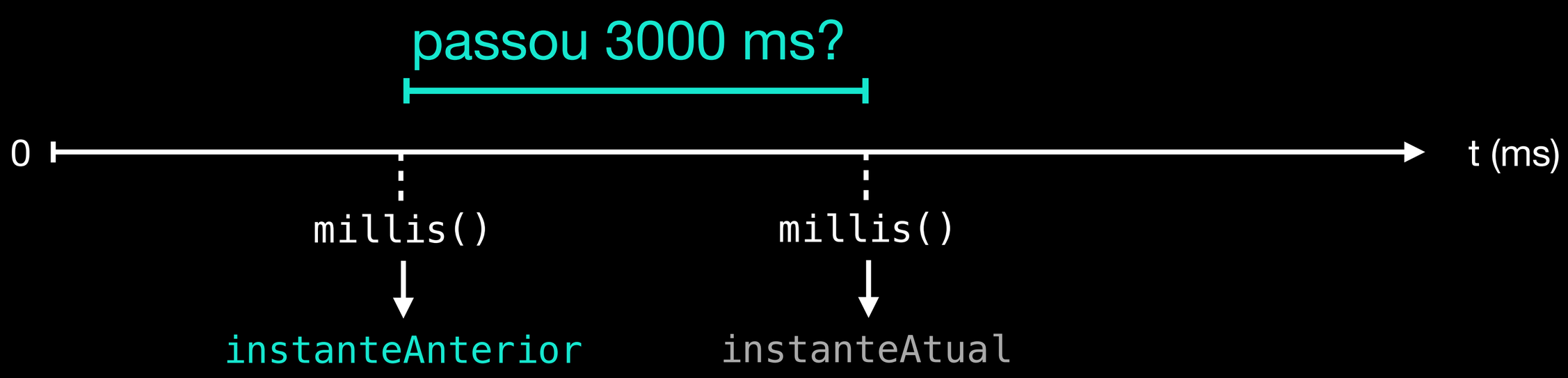


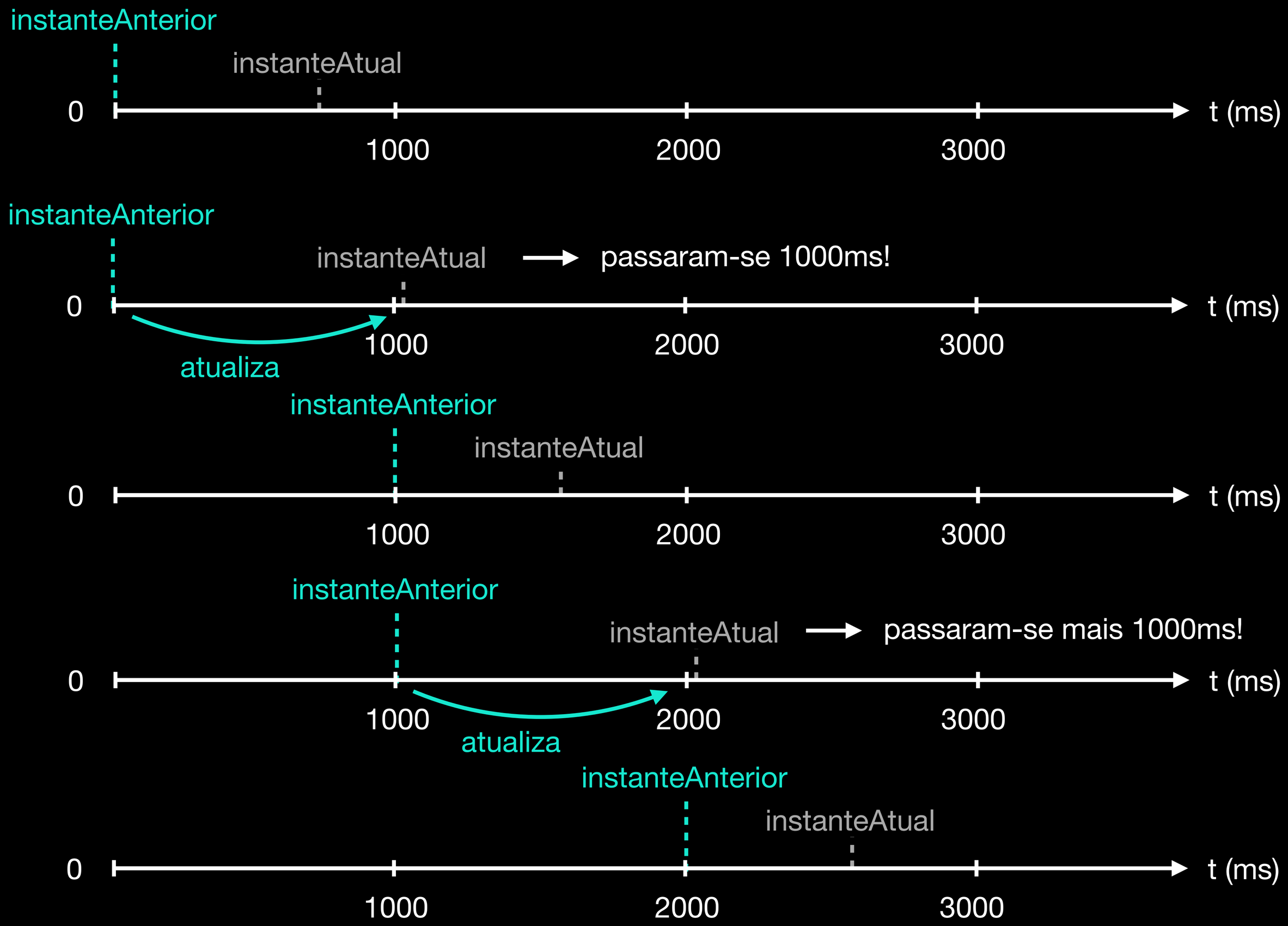
Monitoramento de Tempo

Instante de tempo (em ms)



Verificação da diferença entre dois instantes (em ms)





Monitoramento Periódico do Tempo

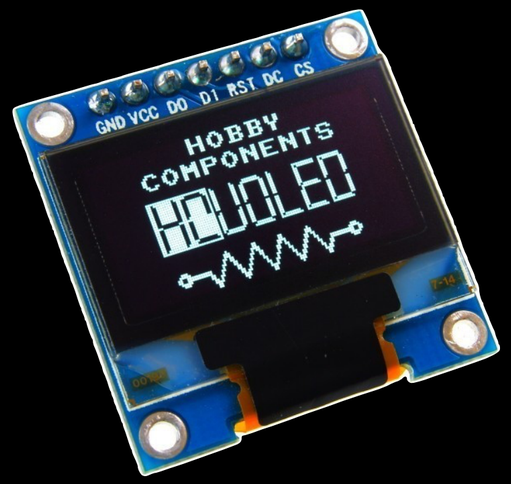
```ino
unsigned long instanteAnterior = 0; // TEM QUE SER unsigned long!

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

void loop () {
  unsigned long instanteAtual = millis();

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

Monitoramento Periódico de Tempo com a Função Millis

```ino
unsigned long instanteAnterior1 = 0;
unsigned long instanteAnterior2 = 0;
unsigned long instanteAnterior3 = 0;

void loop () {
  unsigned long instanteAtual = millis();
  if (instanteAtual > instanteAnterior1 + 1000) {
    instanteAnterior1 = instanteAtual;
    // faz alguma coisa a cada 1000 milissegundos...
  }
  if (instanteAtual > instanteAnterior2 + 200) {
    instanteAnterior2 = instanteAtual;
    // faz alguma coisa a cada 200 milissegundos...
  }

  if (instanteAtual > instanteAnterior3 + 8000) {
    instanteAnterior3 = instanteAtual;
    // faz alguma coisa a cada 8000 milissegundos...
  }
}
```

Vários Monitoramentos de Tempo com a Função Millis

Tela Gráfica



Adafruit_SSD1306

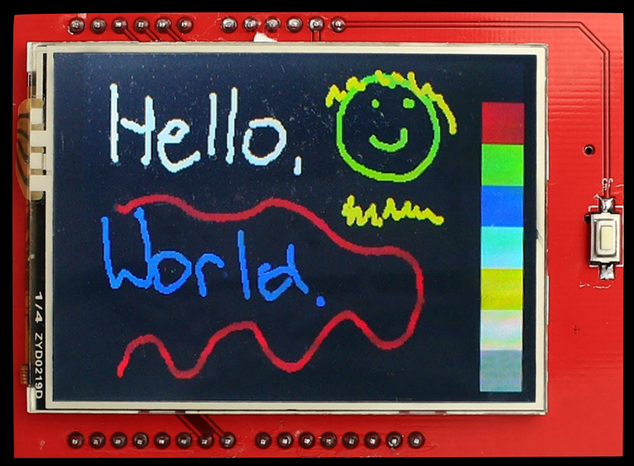
GxEPD2

TFT_eSPI

Adafruit_GFX

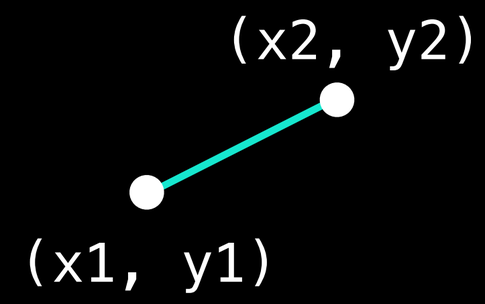
Adafruit_ILI9341

MCUFRIEND_kbv

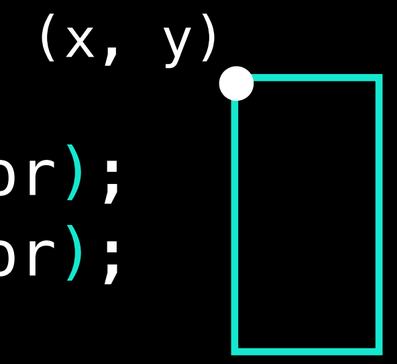


Ponto Comum entre Vários Tipos de Display

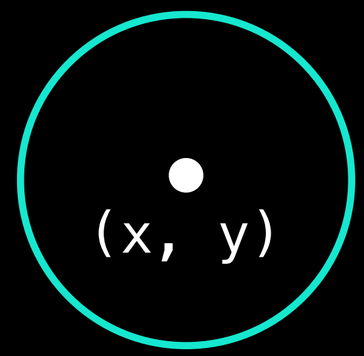
```
tela.drawLine(x1, y1, x2, y2, cor);
```



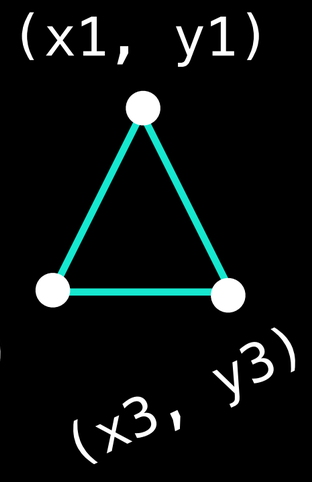

```
tela.fillRect(x, y, comprimento, altura, cor);
tela.drawRect(x, y, comprimento, altura, cor);
```



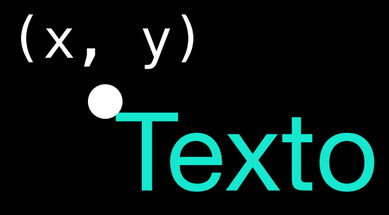

```
tela.fillCircle(x, y, raio, cor);
tela.drawCircle(x, y, raio, cor);
```




```
tela.fillTriangle(x1, y1, x2, y2, x3, y3, cor);
tela.drawTriangle(x1, y1, x2, y2, x3, y3, cor);
```

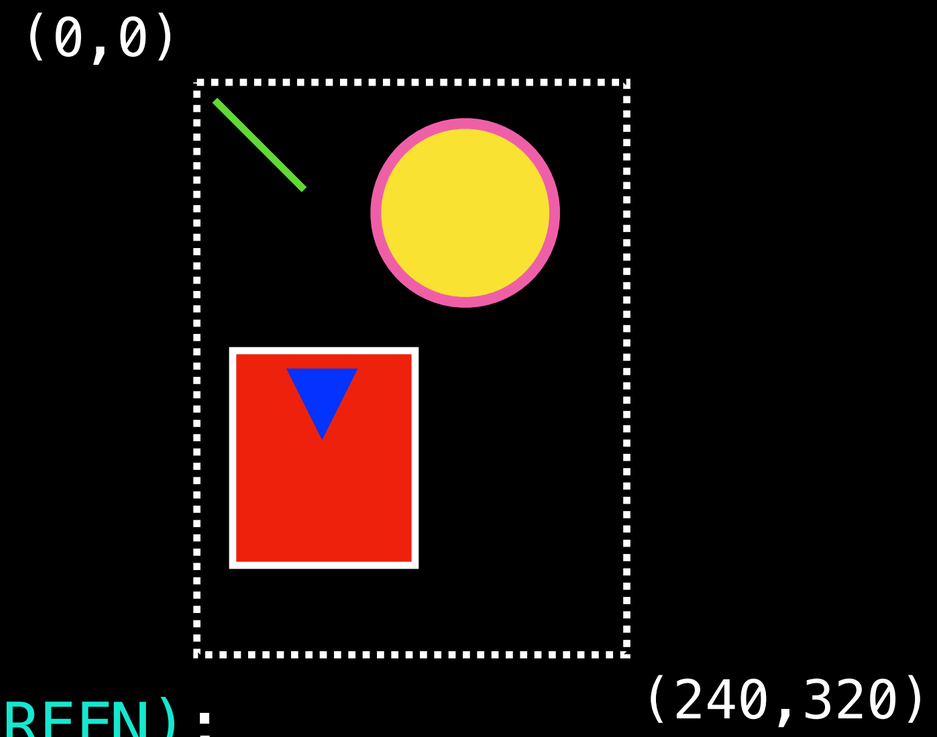



```
tela.setCursor(x, y);
tela.setTextColor(cor);
tela.setTextSize(escala);
tela.print("Texto");
```



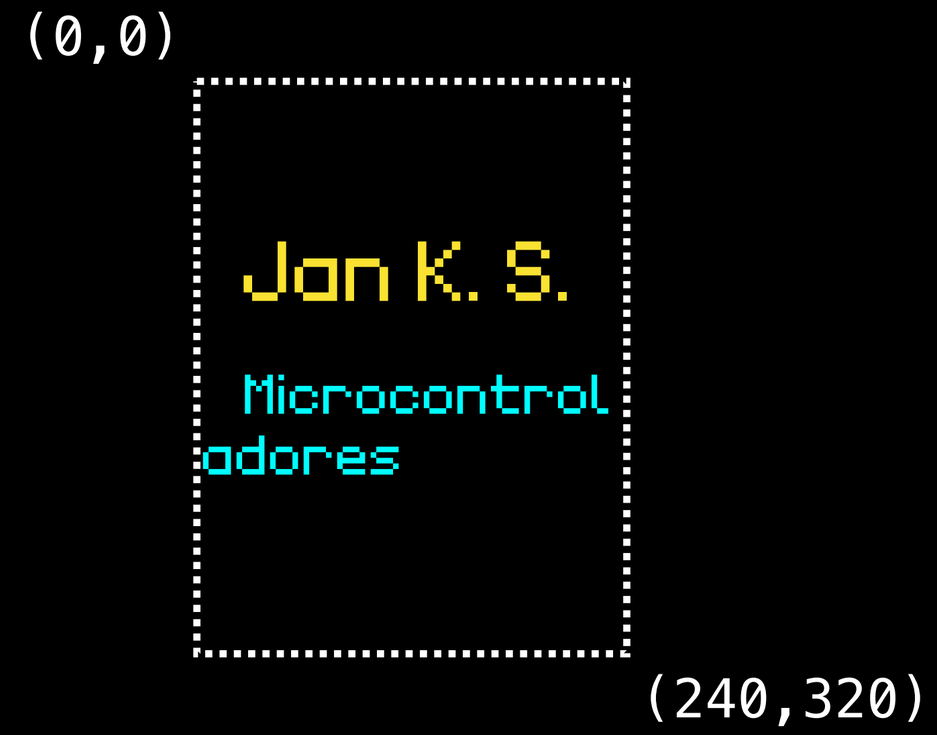
Funções de Desenho da Adafruitt_GFX

```ino
#include <Adafruit_ILI9341.h>

Adafruit_ILI9341 tela(53, 49, 48);

void setup () {
  tela.begin();
  tela.fillScreen(ILI9341_BLACK);

  tela.drawLine(10, 10, 60, 60, ILI9341_GREEN);

  tela.fillCircle(150, 70, 50, ILI9341_YELLOW);
  tela.drawCircle(150, 70, 50, ILI9341_PINK);

  tela.fillRect(20, 150, 100, 120, ILI9341_RED);
  tela.drawRect(20, 150, 100, 120, ILI9341_WHITE);

  tela.fillTriangle(50, 160, 70, 200, 90, 160, ILI9341_BLUE);
}

void loop () {
}
```



Exemplo com Funções de Desenho para Formas

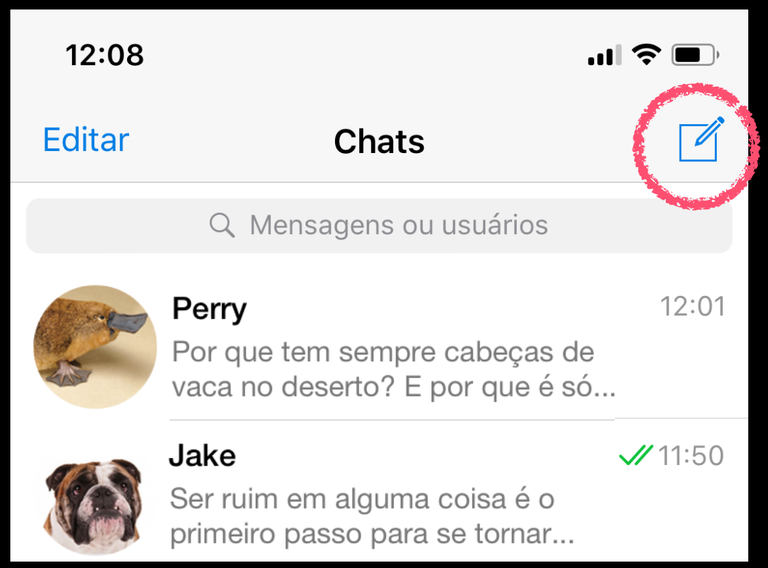
```arduino
#include <Adafruit_GFX.h>
#include <Adafruit_ILI9341.h>

Adafruit_ILI9341 tela(53, 49, 48);

void setup () {
  tela.begin();
  tela.fillScreen(ILI9341_BLACK);

  tela.setCursor(20, 100);
  tela.setTextColor(ILI9341_YELLOW);
  tela.setTextSize(4);
  tela.print("Jan K. S.");

  tela.setCursor(20, 160);
  tela.setTextColor(ILI9341_CYAN);
  tela.setTextSize(3);
  tela.print("Microcontroladores");
}

void loop () {
}
```

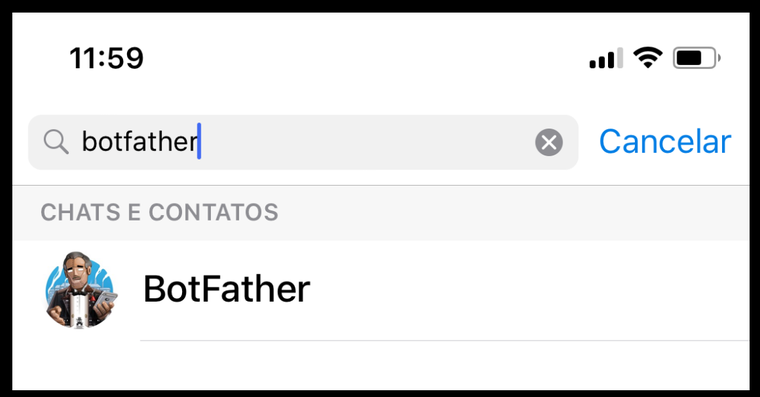


Exemplo com Funções de Desenho para Texto

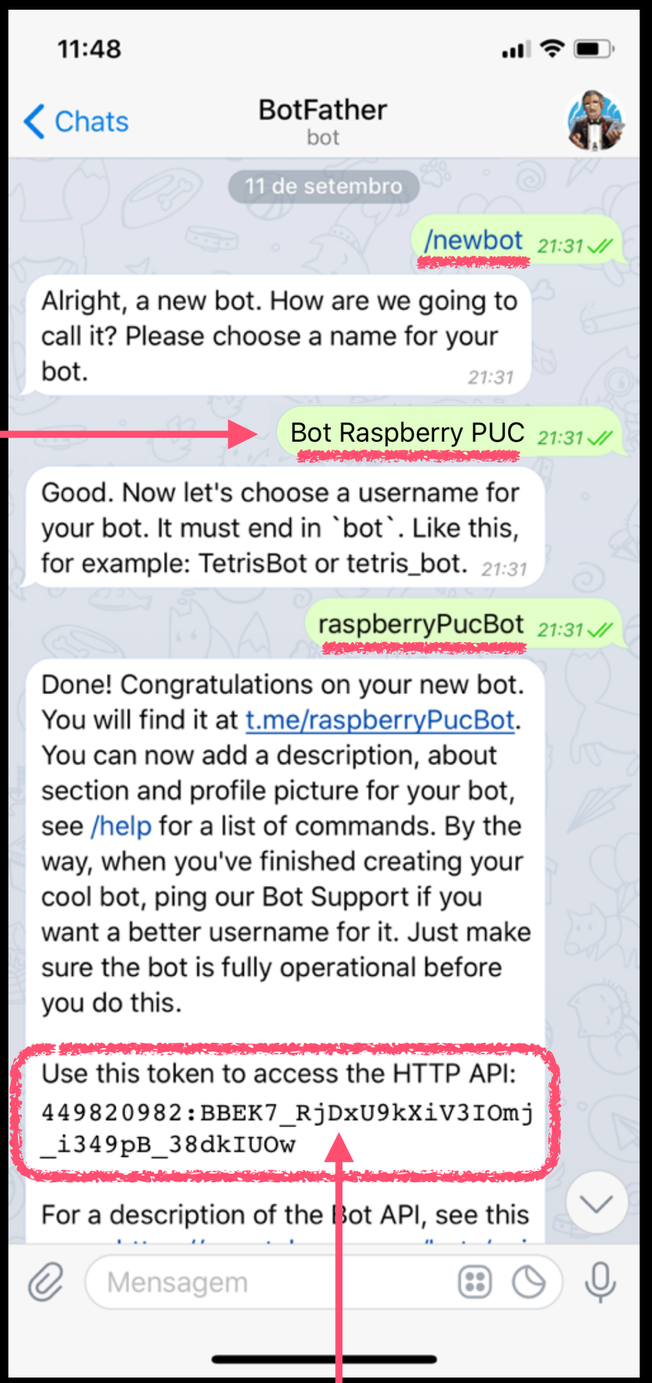
Telegram

1





invente um nome seu!

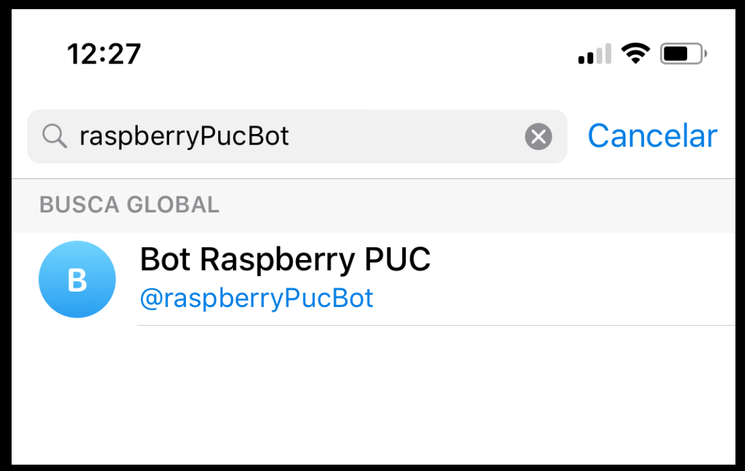


anote esta CHAVE!

Passo 1: Crie o Bot via BotFather e Anote a Chave

2

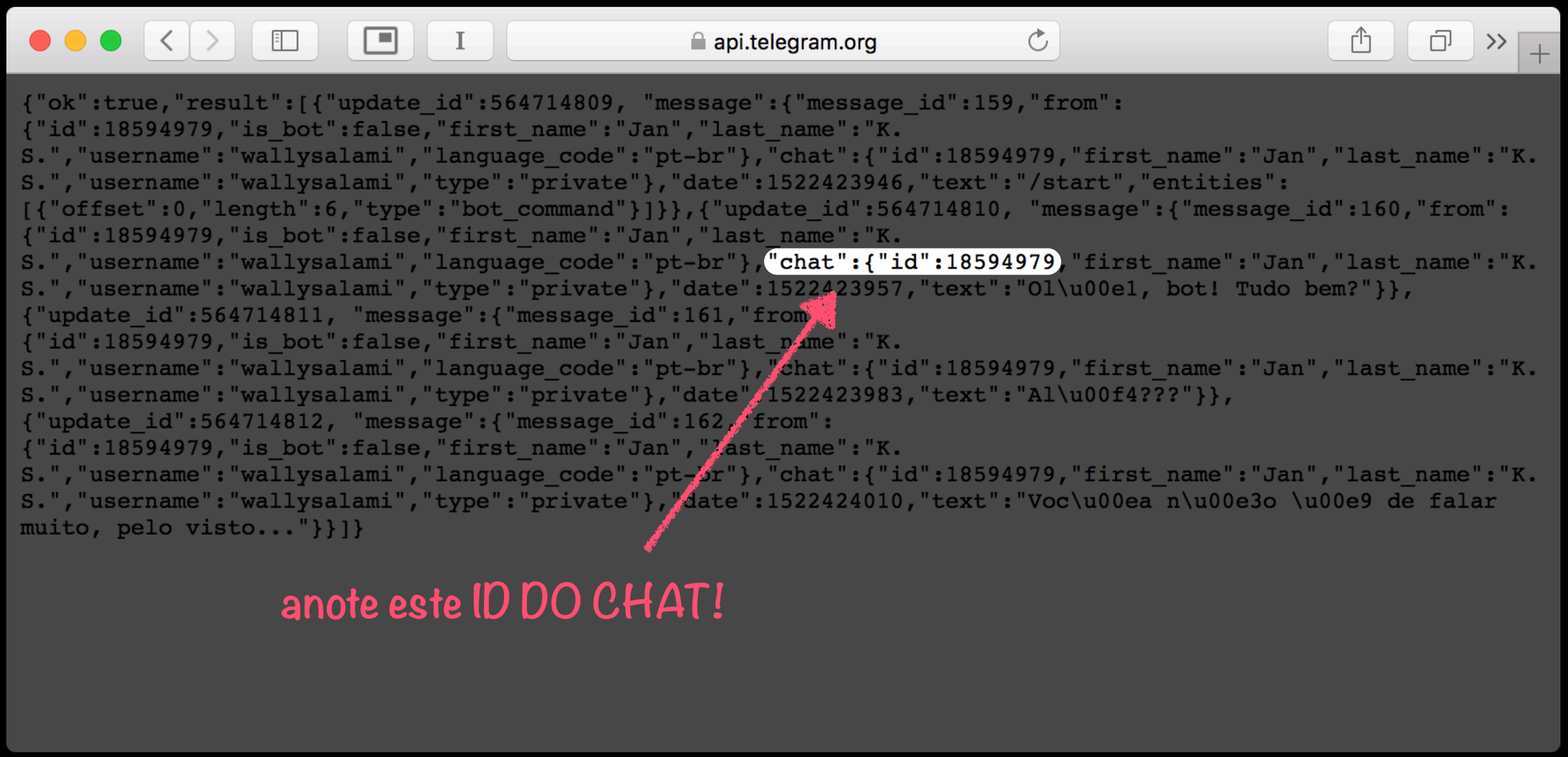
pesquise o nome
do SEU bot!



Passo 2: Converse com o Seu Bot

3

api.telegram.org/botSUA_CHAVE_SECRETA/getUpdates



Passo 3: Obtenha o Id do Seu Chat com o Bot

```
>>> chave = "COLOQUE A SUA CHAVE DO BOTFATHER AQUI!"
>>> id_da_conversa = "COLOQUE O ID DA SUA CONVERSA AQUI!"
>>> endereco_base = "https://api.telegram.org/bot" + chave

>>> from requests import post
>>> endereco = endereco_base + "/sendMessage"
>>> dados = {"chat_id": id_da_conversa, "text": "Oi!"}
>>> resposta = post(endereco, json=dados)
>>> print(resposta.text)

>>> endereco = endereco_base + "/sendPhoto"
>>> dados = {"chat_id": id_da_conversa}
>>> arquivo = {"photo": open("foto.jpg", "rb")}
>>> resposta = post(endereco, data=dados, files=arquivo)
```

Chamada POST para Envio de Mensagens de Texto e de Fotos

Foto Enviada pelo Bot na Conversa

```
>>> from urllib.request import urlretrieve
>>> link_do_arquivo = "https://www.pudim.com.br/pudim.jpg"
>>> arquivo_de_destino = "meu_arquivo.jpg"
>>> urlretrieve(link_do_arquivo, arquivo_de_destino)
```

Baixar Arquivos

Flask

web development,
one drop at a time

Framework Flask

```python
from flask import Flask
app = Flask(__name__)

@app.route("/")
def mostrar_pagina_principal():
  return "Bem-vindo!"

@app.route("/sobre")
def mostrar_pagina_sobre():
  return "Olá, meu nome é Jan!"

@app.route("/soma/<int:x>/<int:y>")
def calcular_soma(x, y):
  return "%d + %d = %d" % (x, y, x+y)

app.run(port=5000)
```

localhost:5000/

Bem-vindo!

localhost:5000/sobre

Olá, meu nome é Jan!

localhost:5000/soma/3/4

3 + 4 = 7

Rotas Dentro do Flask

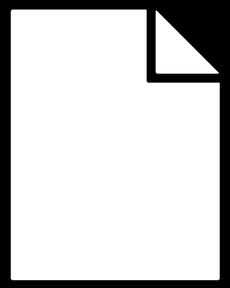
Código	Resultado
`<p>Parágrafo 1</p>` `<p>Parágrafo 2</p>`	Parágrafo 1 Parágrafo 2
`<strong>Negrito</strong>`	**Negrito**
`<img src="https://cdn-icons-png.flaticon.com/512/197/197386.png">`	

`<ul>`
`  <li>Item 1</li>`
`  <li>Item 2</li>`
`  <li>Item 3</li>`
`</ul>`
	• Item 1
	• Item 2
	• Item 3

`<a href="/pagina">`
`  Link para a Página`
`</a>`
	Link para a Página

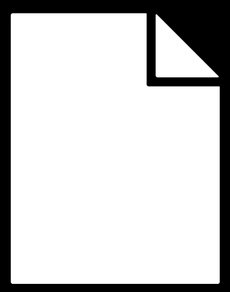
`Linha 1   Linha 2`	Linha 1 Linha 2

Exemplos de Tags HTML



servidor.py

```
[
  {
    "nome": "Globo",
    "código": "0431"
  },
  {
    "nome": "Band",
    "código": "0731"
  },
  ...
]
```



pagina.html

Funções

- Ligar/Desligar
- Dormir em 5 segundos
- Volume +
- Volume –

Canais

- TV Brasil
- Globo
- Band
- CNT
- SBT
- Record
- Globo
- CNT

Página HTML do Projeto 02

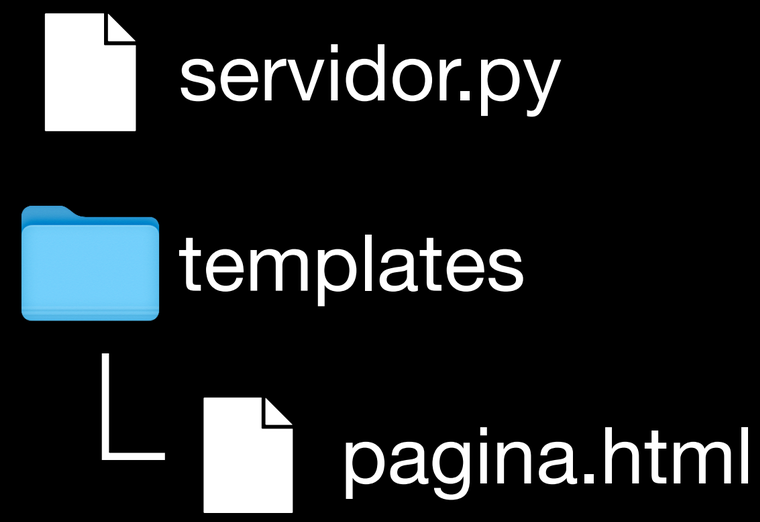
servidor.py

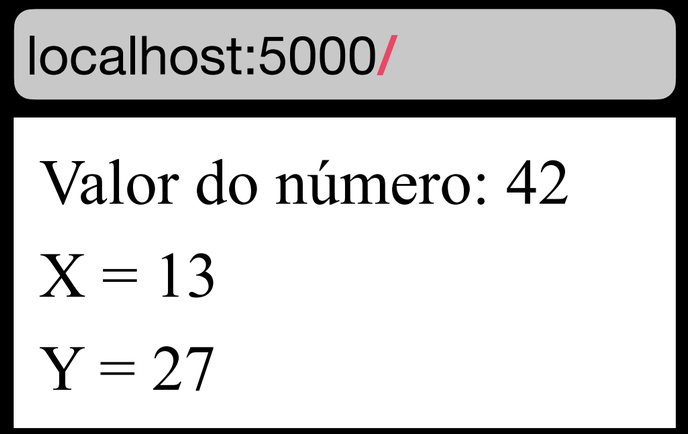
```python
from flask import Flask, render_template
app = Flask(__name__)

@app.route("/")
def mostrar_pagina_principal():
    return render_template(
        "pagina.html",
        numero=42,
        dicionario={"x": 13, "y": 27}
    )
```

---

pagina.html

```html
<p>Valor do numero: {{numero}}</p>
<p>X = {{dicionario["x"]}}</p>
<p>Y = {{dicionario["y"]}}</p>
```





Renderização de Página HTML com Passagem de Parâmetros

```
<input type="text">
```

IoT

```
<input type="password">
```

•••••

```
<textarea></textarea>
```

What is love
Baby don't hurt me, don't hurt me
No more

```
<input type="radio" value="sim" id="sim">
<label for="sim">Sim</label>

<input type="radio" value="não" id="não">
<label for="não">Não</label>
```

Sim
Não

```
<select>
    <option>Sim</option>
    <option>Talvez</option>
    <option selected>Não</option>
</select>
```

Não

Sim
Talvez
✔ Não

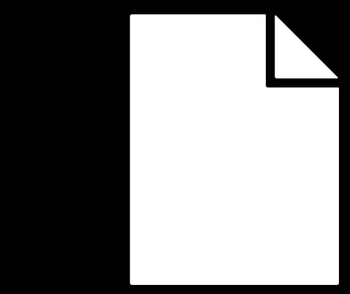
```
<input type="checkbox" id="ok">
<label for="ok">Concordo</label>
```

Concordo

```
<button type="submit">
    Enviar
</button>
```

Enviar

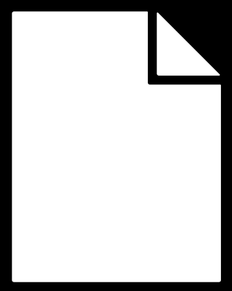
Exemplos de Elementos HTML para Formulários



pagina.html

request.form

```
{
  "usuario": "janks",
  "senha": "1234abcd"
}
```



servidor.py

(processa os dados recebidos)

Exemplo de Formulário no HTML

pagina.html

```html
<form method="POST">
  <input placeholder="Usuario" type="text" name="usuario">
  <input placeholder="Senha" type="password" name="senha">

  <button type="submit">Entrar</button>
</form>
```

---

servidor.py

```python
from flask import Flask, render_template, redirect, request
app = Flask(__name__)

@app.route("/novo", methods=["GET", "POST"])
def pagina_com_formulario():
    if request.method == "POST":
        usuario = request.form["usuario"]
        senha = request.form["senha"]

        # faz alguma coisa com os dados...

        return redirect("/outra-rota")

    return render_template("formulario.html")

app.run(port=5000)
```

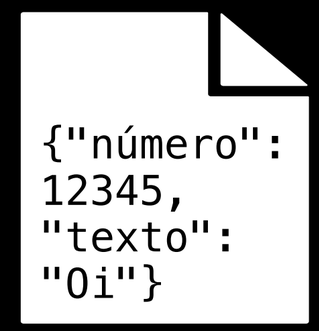
Exemplo de Formulário no HTML

JSON

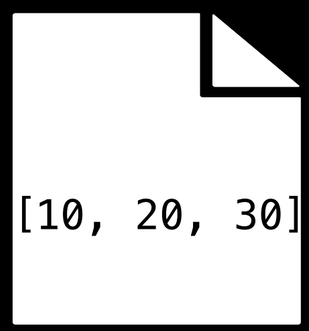
Texto do Título

```
>>> import json
>>> dados = {}
>>> dados["número"] = 12345
>>> dados["texto"] = "Oi!"
>>> with open("arquivo.json", "w", encoding="utf-8") as arquivo:
...     json.dump(arquivo, dados)
...

>>> with open("arquivo2.json", "r", encoding="utf-8") as arquivo:
...     lista = json.load(arquivo)
...
>>> for elemento in lista:
...     print(elemento)
...
10
20
30
```

arquivo.json



arquivo2.json



```
>>> import json
>>> dados = {}
>>> dados["número"] = 12345
>>> dados["texto"] = "Oi!"
>>> texto_json = json.dumps(dados)
>>> texto_json
'{"número": 12345, "texto": "Oi!"}'
```

```
>>> texto_json2 = '[10, 20, 30]'
>>> lista = json.loads(texto_json2)
>>> for elemento in lista:
...     print(elemento)
...
10
20
30
```

Seria legar poder mexer com JSON no Arduino também...

Armazenamento e Carregamento de Dados em Strings JSON

github.com/bblanchon/ArduinoJson

README | MIT license

ArduinoJson

build passing | build passing | oss-fuzz fuzzing | coverage

stars 6.8k | sponsors 4

ArduinoJson is a C++ JSON library for Arduino and IoT (I

Features

- JSON deserialization
  - Optionally decodes UTF-16 escape sequences
  - Optionally supports comments in the input
  - Optionally filters the input to keep only desired

Languages

C++ 98.5% | CMake 1.1%
Shell 0.4% | Dockerfile 0.0%
Makefile 0.0% | Awk 0.0%

sketch_mar2a | Arduino IDE 2.3.4

Arduino Mega or Mega 2...

LIBRARY MANAGER

arduinojson

Type: All

Topic: All

ArduinoJson by Benoit Blanchon <blog.benoitblanchon.fr>

7.3.1 installed

A simple and efficient JSON library for embedded C++. ⭐ 6849 stars on GitHub! Supports serialization, deserialization,...

More info

7.3.1 INSTALL

sketch_mar2a.ino

```
1  void setup() {
2    // put your setup code
3
4  }
5
6  void loop() {
7    // put your main code h
8
9  }
```

Output

```
Installing ArduinoJson@7.3.1
Replacing ArduinoJson@7.2.0 with
Installed ArduinoJson@7.3.1
```

indexing: 23/27 Ln 1, Col 1 Arduino Mega or Mega 2560 on /dev/cu.usbserial-56B6002841

Biblioteca ArduinoJson

```cpp
#include <ArduinoJson.h>
                    // em alguma função do código...
// JsonDocument representando um dicionário
JsonDocument dados;
dados["número"] = 12345;
dados["texto"] = "IoT";

String meuTexto = dados["texto"]; // → "IoT"
int meuNumero = dados["número"];  // → 12345

String outroTexto = dados["campo inexistente"]; // → "null"
int outroNumero = dados["campo inexistente"];   // → 0
```

```cpp
// JsonDocument representando uma lista
JsonDocument lista;
lista.add(10);
lista.add(20);
lista.add(30);

for (unsigned int i = 0; i < lista.size(); i++) {
  int elemento = lista[i];
  Serial.println(elemento);
}
```

```
10
20
30
```

JsonDocument Representando Dicionários e Listas

```arduino
#include <ArduinoJson.h>

JsonDocument dados;
dados["número"] = 12345;
dados["texto"] = "IoT";

Serial.println(dados["texto"]);
```

```
error: call of overloaded 'println(...)' is ambiguous
```

Compilador não sabe se você quer o dado como int, float, String... ficou ambíguo.

```arduino
String meuTexto = dados["texto"];
Serial.println(meuTexto);
```

Agora o compilador sabe que você quer o dado como String.

Erro Comum com a JsonDocument na Serial

```cpp
#include <ArduinoJson.h>

JsonDocument dados;
dados["número"] = 12345;
dados["texto"] = "IoT";

String textoJson;
serializeJson(dados, textoJson); // → "{"número": 12345, ... }"

serializeJson(dados, Serial);    // ou imprime direto na Serial
```

```cpp
String texto_json2 = "[10, 20, 30]";
JsonDocument lista;
deserializeJson(lista, texto_json2);

for (unsigned int i = 0; i < lista.size(); i++) {
  int elemento = lista[i];
  Serial.println(elemento);
}
```

```
10
20
30
```

Armazenamento e Carregamento de JSON em Strings