



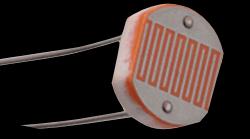
# Projeto 05

## Coisas Consumíveis – Resumo da Ópera

Jan K. S. – [janks@puc-rio.br](mailto:janks@puc-rio.br)

ENG4051 – Projeto Internet das Coisas

Sensor de Luz



```
int leitura = analogRead(pino);  
int porcentagemLuz = map(leitura, 0, 4095, 0, 100);
```

Millis

```
unsigned long instanteAnterior = 0;
```

```
void loop () {
```

```
    unsigned long instanteAtual = millis();
```

```
    if (instanteAtual > instanteAnterior + 1000) {
```

```
        Serial.println("+1 segundo");
```

```
        instanteAnterior = instanteAtual;
```

```
}
```

```
String texto1 = "Olá, mundo!";
```

```
int numero = 100 * 2;
```

```
String texto2 = String(numero);
```

```
int numero2 = texto2.toInt() + 42;
```

```
String texto3 = "aaa" + texto2;
```

```
bool ehIgual = texto2 == texto3;
```

```
bool comecaComOlá = texto1.startsWith("Olá");
```

```
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 \n";
```

```
texto4.replace("ab", "AB"); // "ABc ABC"
```

LED

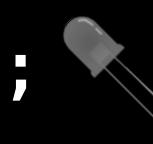
```
void setup () {
```

```
    pinMode(pinoLED, OUTPUT);
```

```
    digitalWrite(pinoLED, HIGH);
```

```
}
```

```
    digitalWrite(pinoLED, LOW);
```



Serial

```
void setup () {
```

```
    Serial.begin(115200); while(!Serial);
```

```
}
```

```
void loop () {
```

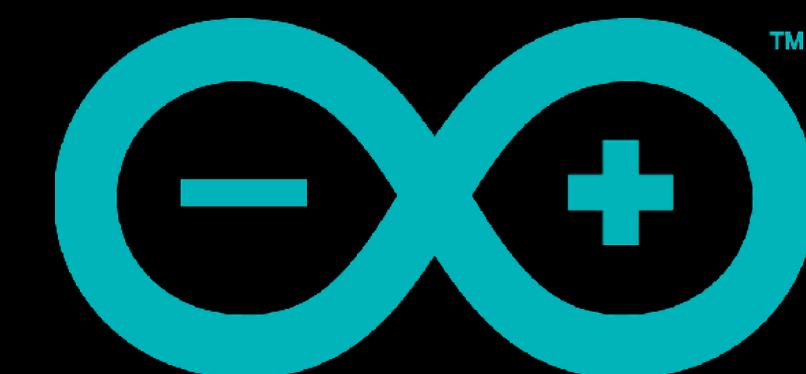
```
    if (Serial.available() > 0) {
```

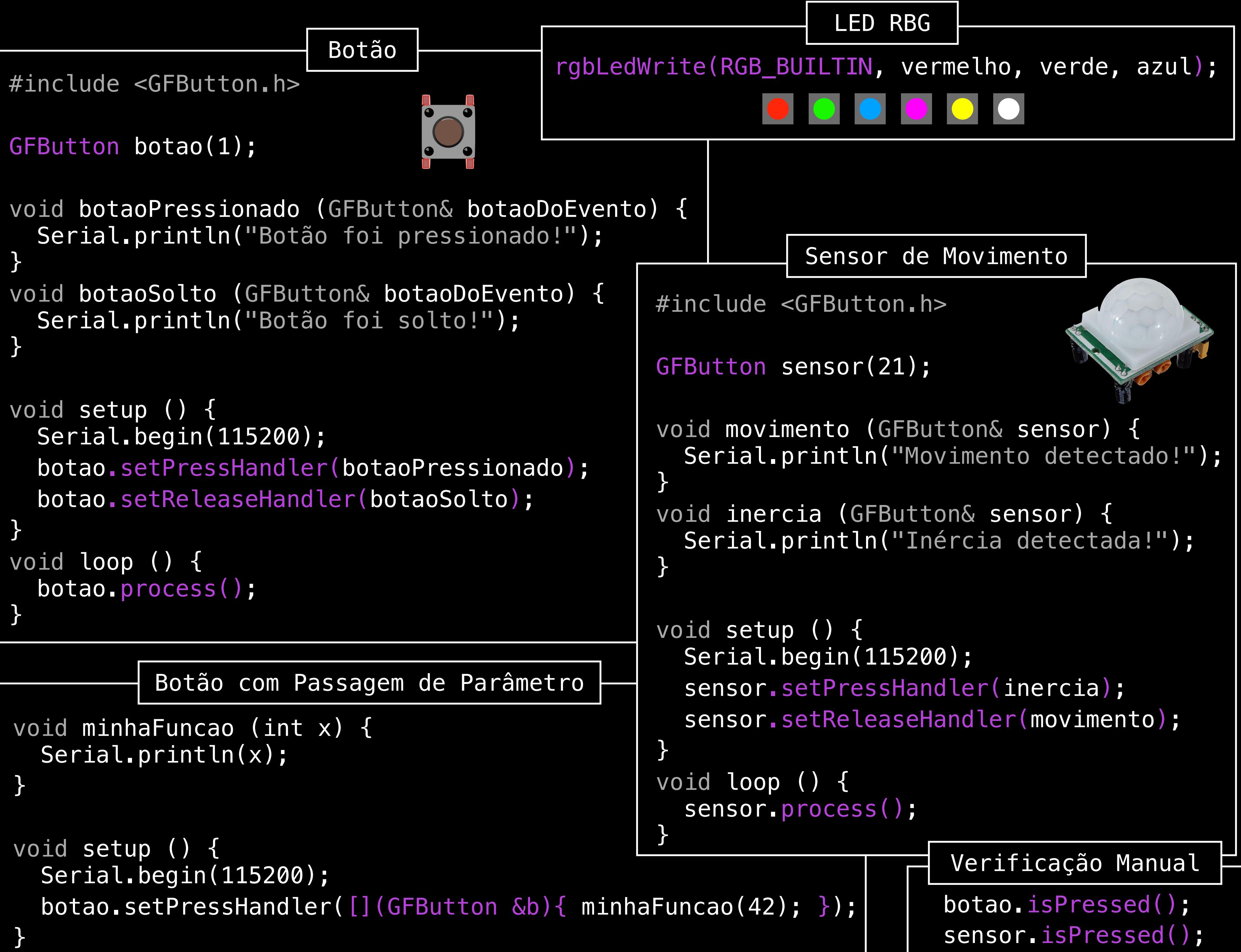
```
        String texto = Serial.readStringUntil('\n');
```

```
        Serial.println(texto);
```

```
}
```

```
}
```





WiFi

```
#include <WiFi.h>

void reconnectarWiFi() {
  if (WiFi.status() != WL_CONNECTED) {
    WiFi.begin("NOME DA REDE", "SENHA DA REDE");

    Serial.print("Conectando ao WiFi...");
    while (WiFi.status() != WL_CONNECTED) {
      Serial.print(".");
      delay(1000);
    }
    Serial.print("conectado!\nEndereço IP: ");
    Serial.println(WiFi.localIP());
  }
}

void setup () {
  Serial.begin(115200); delay(500);

  reconnectarWiFi();
}

void loop () {
  reconnectarWiFi();
}
```



Rede WiFi

Serialização

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

String textoJson;
serializeJson(dados, textoJson);
serializeJson(dados, Serial);

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

Json

```
#include <ArduinoJson.h>
```

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

```
String meuTexto = dados["texto"];
int meuNumero = dados["número"];
```

```
JsonDocument lista;
lista.add(10);
lista.add(20);
for (unsigned int i = 0; i < lista.size(); i++) {
  int elemento = lista[i];
  Serial.println(elemento);
}
```

O JSON

## Setup

```
#include <GxEPD2_BW.h>
#include <U8g2_for_Adafruit_GFX.h>

U8G2_FOR_ADAFRUIT_GFX fontes;
GxEPD2_290_T94_V2 modeloTela(10, 14, 15, 16);
GxEPD2_BW<GxEPD2_290_T94_V2, GxEPD2_290_T94_V2::HEIGHT> tela(modeloTela);

void setup() {
    tela.init();
    tela.setRotation(3);
    tela.fillRect(0, 0, 296, 128, GxEPD_WHITE);
    tela.display(true);

    fontes.begin(tela);
    fontes.setForegroundColor(GxEPD_BLACK);
}
```

## Desenhos

```
tela.drawLine(x1, y1, x2, y2, cor);
tela.fillCircle(x, y, raio, cor);
tela.drawCircle(x, y, raio, cor);
```

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

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

```
tela.display(true); // SEMPRE CHAMAR NO FINAL!
```

## Fontes de Símbolos

```
u8g2_font_open_iconic_all_4x_t
```



[https://github.com/olikraus/u8g2/wiki/fntpic/u8g2 font open iconic all 4x t.png](https://github.com/olikraus/u8g2/wiki/fntpic/u8g2_font_open_iconic_all_4x_t.png)

## Fontes de Texto

```
u8g2_font_helvB24_te
```

```
u8g2_font_helvB18_te
```

```
u8g2_font_helvB14_te
```

```
u8g2_font_helvB12_te
```

## Textos

```
fontes.setFont( u8g2_font_helvB24_te );
```

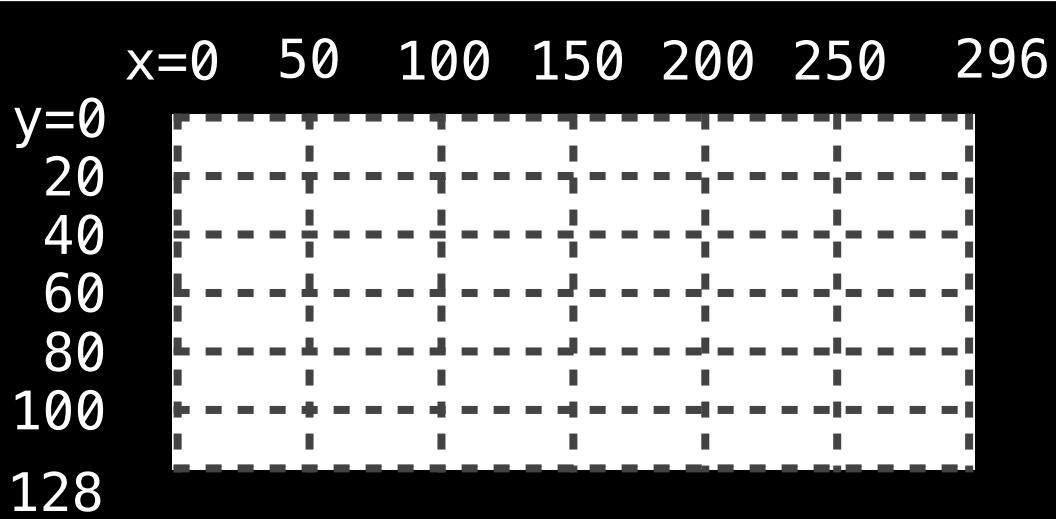
```
fontes.setFontMode(1);
```

```
fontes.setCursor(x, y);
```

```
fontes.print("Meu texto");
```

```
tela.display(true); // SEMPRE CHAMAR NO FINAL!
```

Display ePaper  
2.9" WeAct



```

#include <WiFi.h>
#include <WiFiClientSecure.h>
#include "certificados.h"
#include <MQTT.h>

WiFiClientSecure conexaoSegura;
MQTTClient mqtt(1000);

void setup() {
  Serial.begin(115200);
  delay(500);

  reconnectWiFi();
  conexaoSegura.setCACert(certificado1);

  mqtt.begin("mqtt.janks.dev.br", 8883, conexaoSegura);
  mqtt.onMessage(recebeuMensagem);
  mqtt.setKeepAlive(10);
  mqtt.setWill("tópico da desconexão", "conteúdo");

  reconnectMQTT();
}

void loop() {
  reconnectWiFi();
  reconnectMQTT();
  mqtt.loop();
}

```

Setup

Reconectar

```

void reconnectarMQTT() {
  if (!mqtt.connected()) {
    Serial.print("Conectando MQTT...");
    while(!mqtt.connected()) {
      mqtt.connect("SEU ID", "LOGIN", "SENHA");
      Serial.print(".");
      delay(1000);
    }
    Serial.println(" conectado!");

    mqtt.subscribe("topico1"); // qos = 0
    mqtt.subscribe("topico2/+/parametro", 1); // qos = 1
  }
}

```



Recebimento

```

void recebeuMensagem(String topico, String conteudo) {
  Serial.println(topico + ":" + conteudo);
}

```

Envio

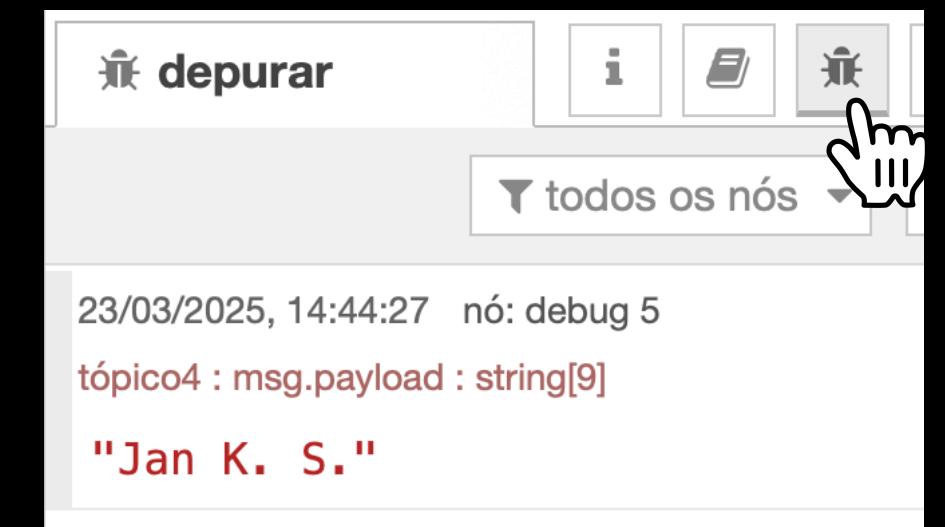
```

mqtt.publish("topico", "conteúdo"); // retain = false, qos = 0
mqtt.publish("topico2/1234/parametro", "conteúdo 2", false, 1);

```

```
mqtt → { topic: "tópico1/10", payload: "conteúdo"}  

{ topic: "tópico2", payload: "conteúdo"} → mqtt
```

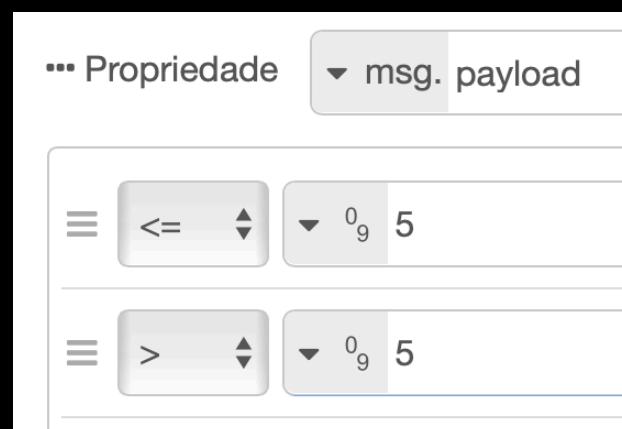


```
{ topic: "tópico4", payload: "Jan K. S."} → debug
```

```
inject → { topic: "tópico3", payload: "teste!"}
```

```
{ payload: "texto" } → change → { payload: "novo texto" }
```

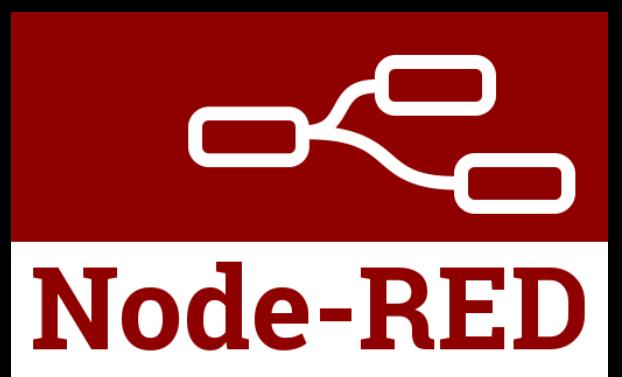
```
{ payload: 8 } → switch (não emite nada) → { payload: 8 }
```



```
{ payload: [10, 20, 30] } → split → { payload: 10 }  

{ payload: 20 }  

{ payload: 30 } → join → { payload: [10, 20, 30] }
```



```
receiver → { content: "Olá, Node-RED!", ... } { payload: <imagem> } → image
```

```
{ payload: 42 } → template → sender
```

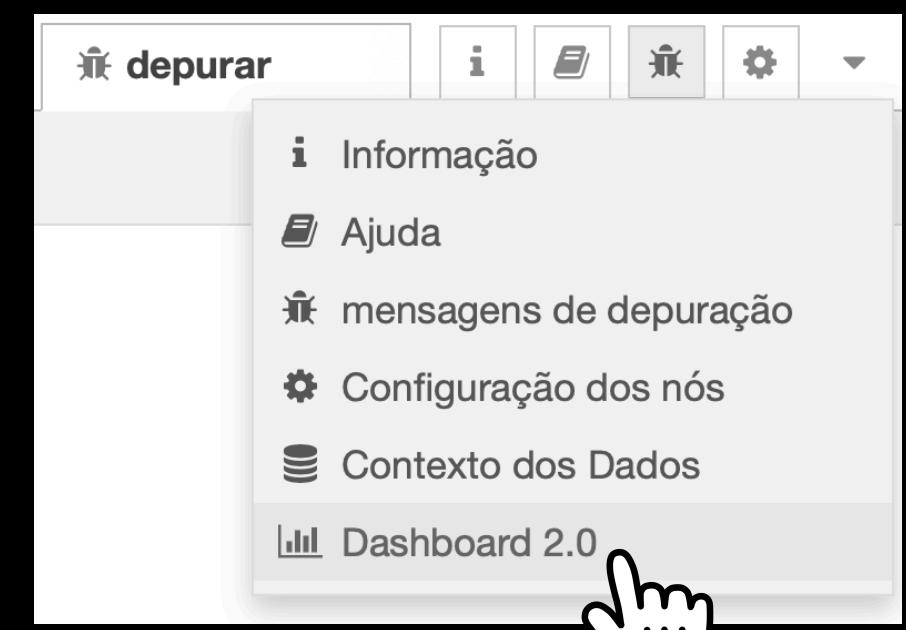
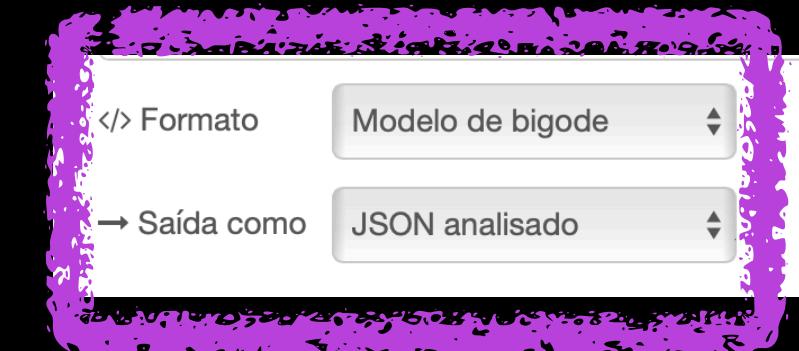
```
{  

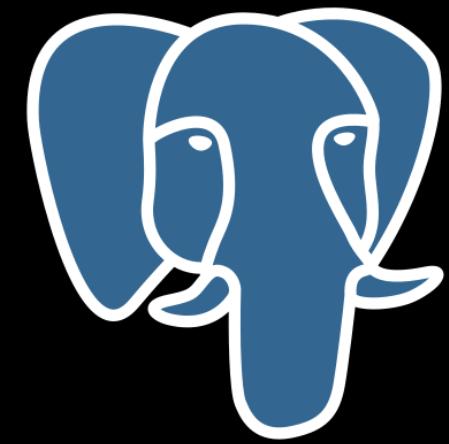
"content": "Valor = {{payload}}",  

"chatId": "ID DO SEU CHAT",  

"type": "message"  

}
```





# PostgreSQL

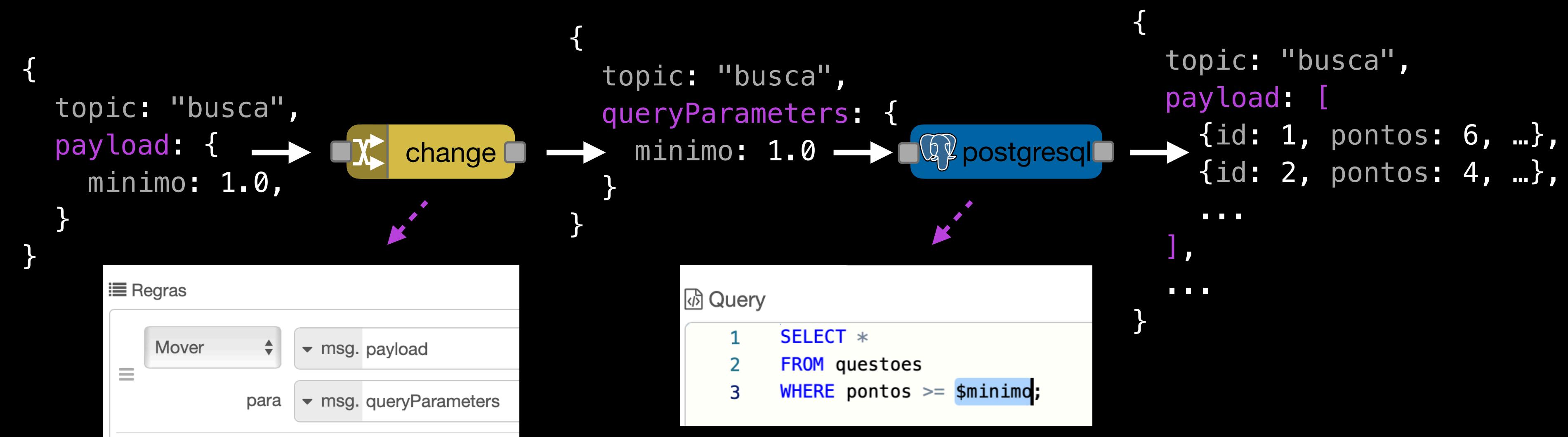
## Inserção de Dados

```
INSERT INTO provas (id, nome, inicio) VALUES
(1, 'P1 de Programação', '2024-04-22 17:00:00'),
(2, 'P2 de Programação', '2024-05-17 17:00:00');
```

## Busca de Dados

```
SELECT *
FROM questoes;
```

```
SELECT numero, pontos, enunciado
FROM questoes
WHERE pontos >= 2.0 AND id = 1
ORDER BY id_prova ASC, numero ASC;
```





# Timescale

Busca com Janelamento

```
SELECT
  time_bucket('1 hour', data_hora) AS time,
  AVG(luz) AS media_luz,
  SUM(movimento) AS soma_movimento
FROM dados
WHERE data_hora > NOW() - INTERVAL '1 day'
GROUP BY time
ORDER BY time ASC;
```

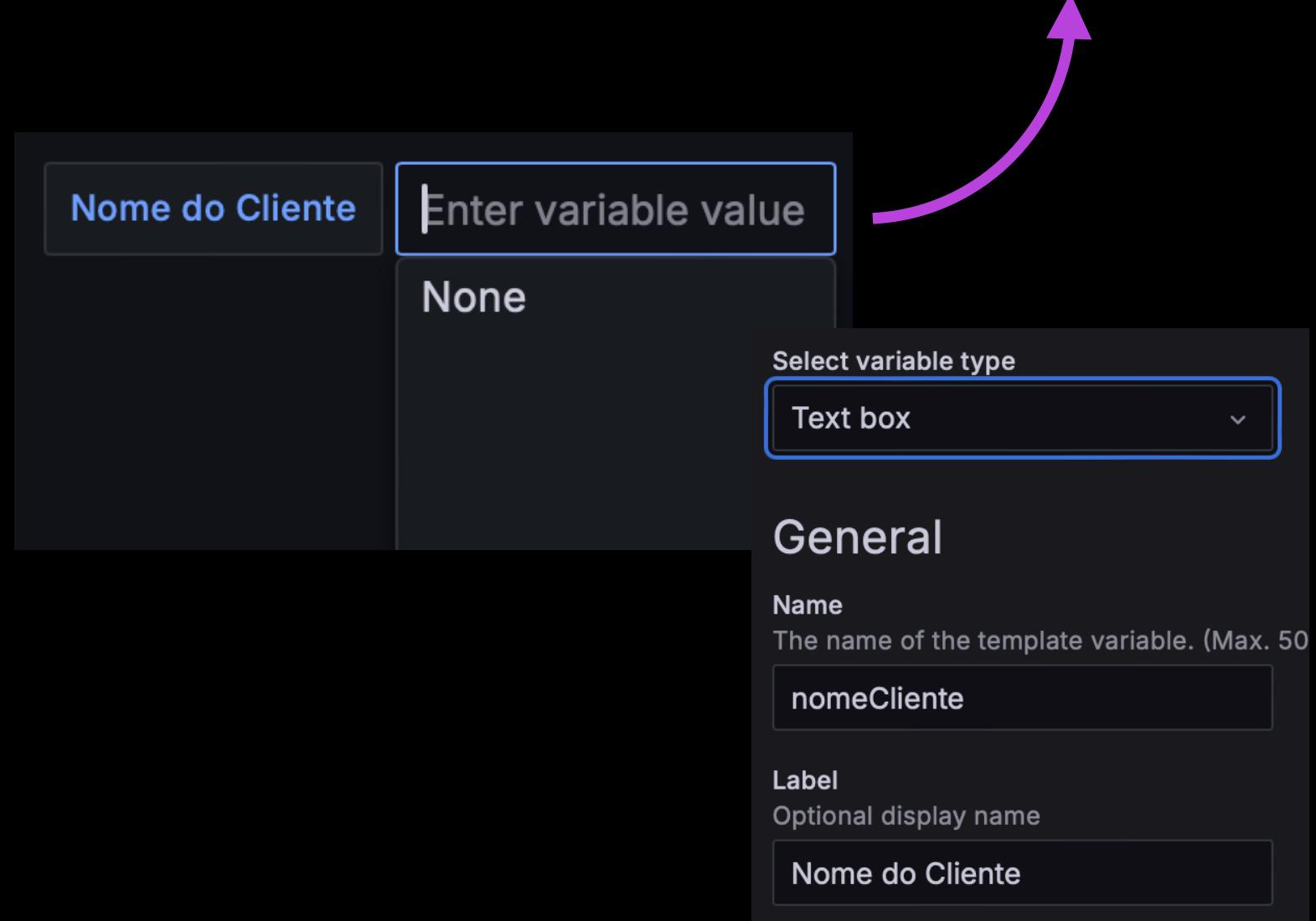
## Outras Funções de Janelamento

```
time_bucket('1 hour', data_hora) AS time
first(temperatura, time) AS primeira_temperatura
last(temperatura, time) AS ultima_temperatura
time_bucket_gapfill('1 hour', data_hora) AS time
```



## Filtro de Tempo do Grafana

```
SELECT
  time_bucket('1 minute', data_hora) AS time,
  AVG(luz) AS media_luz,
FROM dados
WHERE $__timeFilter(data_hora)
GROUP BY time
ORDER BY time ASC;
```



```
SELECT saldo
FROM clientes
WHERE nome = '$nomeCliente'
```

```

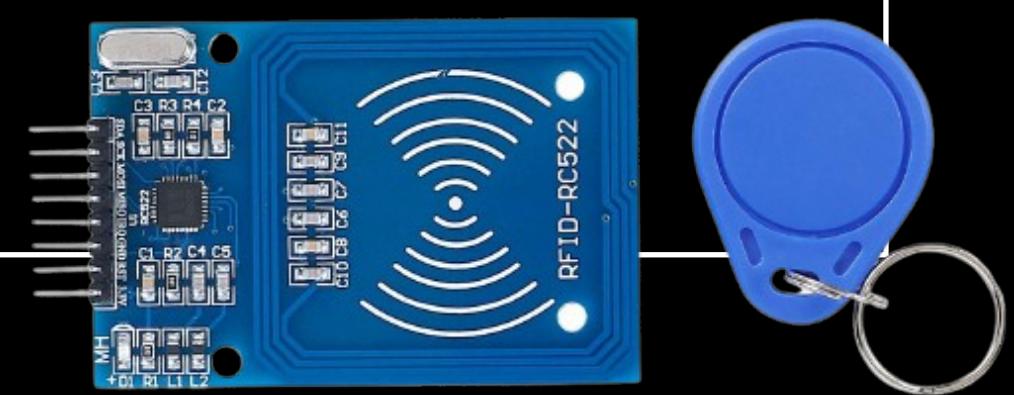
#include <SPI.h>
#include <MFRC522.h>

MFRC522 rfid(46, 17);
MFRC522::MIFARE_Key chaveA =
{{0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF}};

void setup() {
  Serial.begin(115200); delay(500);
  SPI.begin();
  rfid.PCD_Init();
}

```

## RFID



### Setup

```

String lerRFID() {
  String id = "";
  for (byte i = 0; i < rfid.uid.size; i++) {
    if (i > 0) {
      id += " ";
    }
    if (rfid.uid.uidByte[i] < 0x10) {
      id += "0";
    }
    id += String(rfid.uid.uidByte[i], HEX);
  }
  id.toUpperCase();
  return id;
}

```

### Leitura do UID

```

void loop() {
  if (rfid.PICC_IsNewCardPresent() &&
    rfid.PICC_ReadCardSerial()){
    String id = lerRFID();
    Serial.println("UID: " + id);

    String texto = lerTextoDoBloco(6);
    Serial.println("Bloco 6: " + texto);

    rfid.PICC_HaltA();
    rfid.PCD_StopCrypto1();
  }
}

```

### Leitura de um Bloco

```

String lerTextoDoBloco(byte bloco) {
  byte tamanhoDados = 18;
  char dados[tamanhoDados];
  MFRC522::StatusCode status = rfid.PCD_Authenticate(
    MFRC522::PICC_CMD_MF_AUTH_KEY_A,
    bloco, &chaveA, &(rfid.uid)
  );
  if (status != MFRC522::STATUS_OK) { return ""; }
  status = rfid.MIFARE_Read(bloco,
    (byte*)dados, &tamanhoDados);
  if (status != MFRC522::STATUS_OK) { return ""; }
  dados[tamanhoDados - 2] = '\0';
  return String(dados);
}

```

```

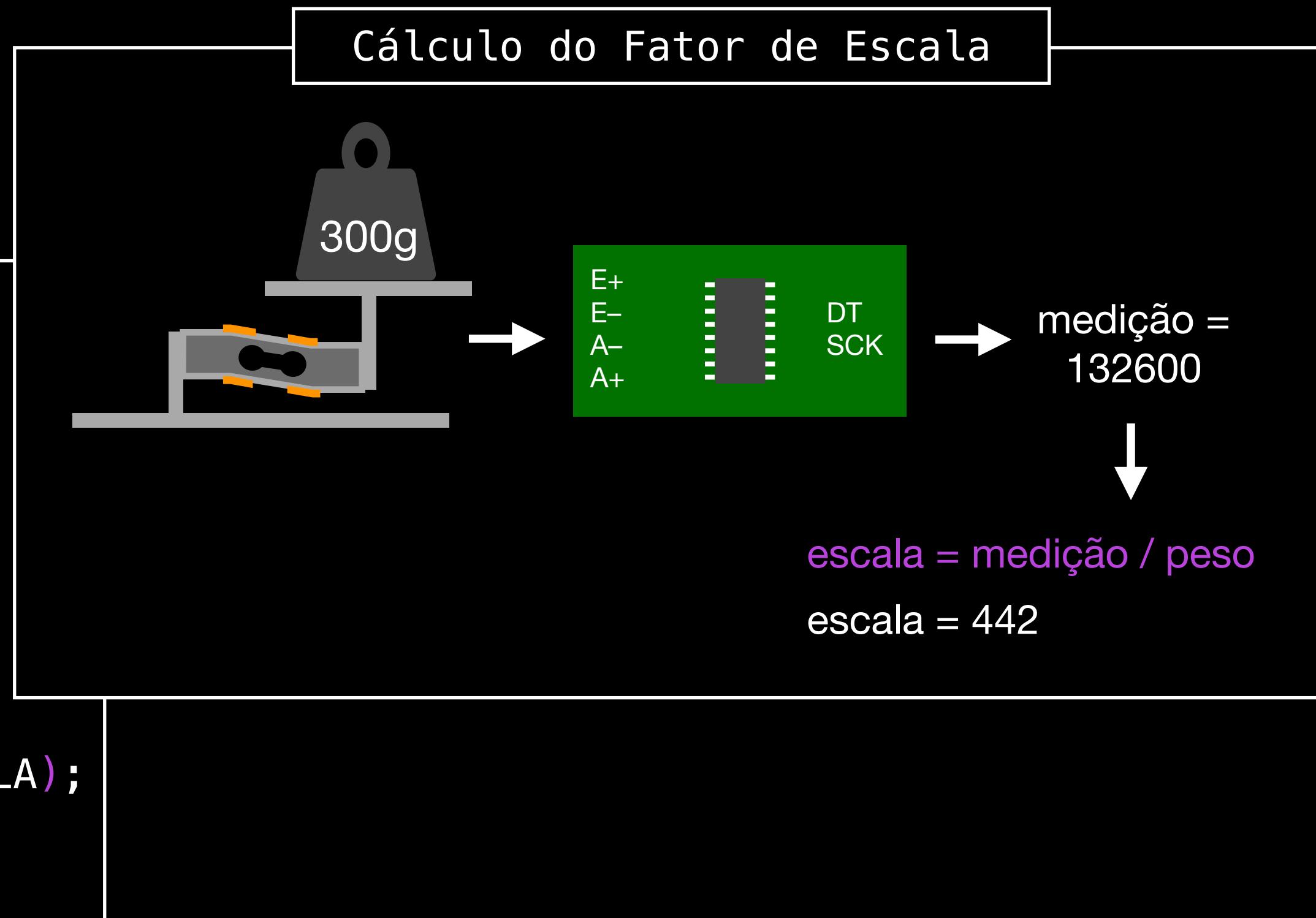
Setup
#include <HX711.h>

HX711 balanca;

void setup() {
  Serial.begin(115200); delay(500);

  balanca.begin(6, 7);
  balanca.set_scale(SEU_FATOR_DE_ESCALA);
  balanca.tare(5);
}

```



**Leitura do Peso**

```

// 1 medição (em gramas)
float pesoMedido = balanca.get_units(1);

// média de 5 medições seguidas
float pesoMedio = balanca.get_units(5);

// define tara (peso inicial para ignorar) com 5 medições
balanca.tare(5);

```



**Célula de Carga  
(Balança)**