



Projeto 01

Coisas Configuráveis – Resumo da Ópera

Jan K. S. – 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

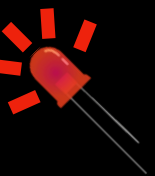
```
unsigned long instanteAnterior = 0;  
  
void loop () {  
    unsigned long instanteAtual = millis();  
    if (instanteAtual > instanteAnterior + 1000) {  
        Serial.println("+1 segundo");  
        instanteAnterior = instanteAtual;  
    }  
}
```

String

```
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 comecaComOla = 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);  
    }  
}
```



Botão

```
#include <GButton.h>
```

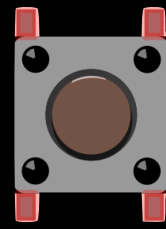
```
GButton botao(1);
```

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

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

```
void setup () {  
    Serial.begin(115200);  
    botao.setPressHandler(botaoPressionado);  
    botao.setReleaseHandler(botaoSolto);  
}
```

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



LED RGB

```
rgbLedWrite(RGB_BUILTIN, vermelho, verde, azul);
```



Sensor de Movimento

```
#include <GButton.h>
```

```
GButton sensor(21);
```

```
void movimento (GButton& sensor) {  
    Serial.println("Movimento detectado!");  
}
```

```
void inercia (GButton& sensor) {  
    Serial.println("Inércia detectada!");  
}
```

```
void setup () {  
    Serial.begin(115200);  
    sensor.setPressHandler(inercia);  
    sensor.setReleaseHandler(movimento);  
}
```

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



Botão com Passagem de Parâmetro

```
void minhaFuncao (int x) {  
    Serial.println(x);  
}
```

```
void setup () {  
    Serial.begin(115200);  
    botao.setPressHandler([](GButton &b){ minhaFuncao(42); });  
}
```

Verificação Manual

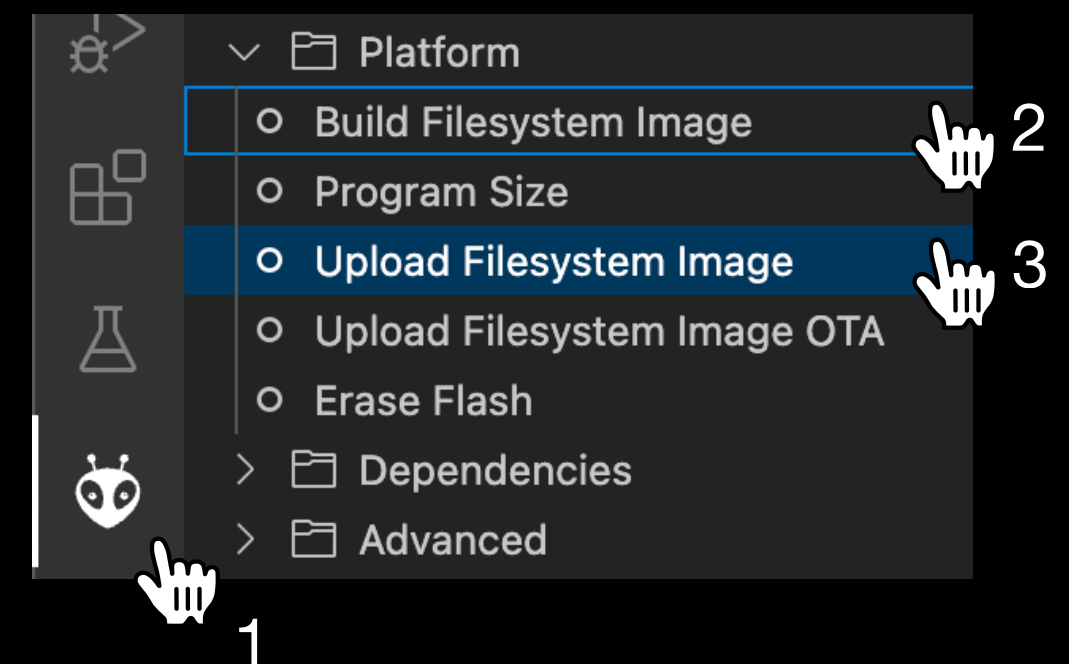
```
botao.isPressed();  
sensor.isPressed();
```

Setup

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

void setup() {
  if (!LittleFS.begin()) {
    Serial.println("LittleFS falhou!");
    while (true) {};
  }
}
```

Envio de Arquivos para ESP32



Leitura

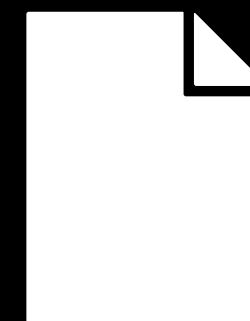
```
File arquivo = LittleFS.open("/arquivo.txt", "r");
if (!arquivo) {
  Serial.println("Arquivo falhou!");
  while (true) {};
}
String conteudo = arquivo.readString();
arquivo.close();
```

Escrita

```
File arquivo = LittleFS.open("/arquivo.txt", "w");
arquivo.println("IoT");
arquivo.print("PUC-Rio ");
arquivo.println(2020 + 4);
arquivo.close();
```

Arquivo JSON

```
JsonDocument dados;
deserializeJson(dados, arquivo); // leitura
serializeJson(dados, arquivo);   // escrita
```



Arquivos no LittleFS

WiFi

```
#include <WiFi.h>

void reconectarWiFi() {
  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);

  reconectarWiFi();
}

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



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);
}
```



Servidor

```
#include <WebServer.h>
#include <uri/UriBraces.h>
```

```
WebServer servidor(80);
```

```
void setup () {
  reconectarWifi();
```

```
  servidor.on("/inicio", HTTP_GET, pagina1);
  servidor.on("/contato", HTTP_GET, pagina2);
  servidor.on("/contato", HTTP_POST, tratarDados);
  servidor.on(UriBraces("/parametros/{}/{ }"), HTTP_GET, pagina3);
  servidor.begin();
```

```
}
```

```
void loop () {
  reconectarWiFi();
  servidor.handleClient();
}
```

Tratamento de Dados POST

```
void tratarDados () {
  String email = servidor.arg("email");
  String mensagem = servidor.arg("mensagem");
  // faz alguma coisa com esses dados...
  // redireciona para uma outra página
  servidor.sendHeader("Location", "/inicio");
  servidor.send(303);
}
```

Página Simples

```
void pagina1 () {
  servidor.send(200, "text/html", "Bem-vindo!");
}
```

Página com HTML

```
void pagina2 () {
  File arquivo = LittleFS.open("/pagina.html", "r");
  if (!arquivo) {
    servidor.send(500, "text/html", "Erro no HTML");
    return;
  }
  String html = arquivo.readString();
  arquivo.close();
  html.replace("{{nome}}", "Jan");
  servidor.send(200, "text/html", html);
}
```



Servidor Web

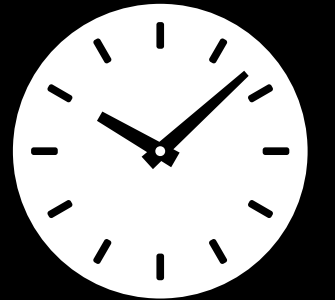
Página Com Parâmetros

```
void pagina3 () {
  String texto = servidor.pathArg(0);
  int numero = servidor.pathArg(1).toInt();
  servidor.send(200, "text/html", "Dados ok");
}
```

Setup

```
#include <time.h>

void setup() {
  Serial.begin(115200);
  reconectarWiFi();
  configTzTime("<-03>3", "a.ntp.br", "pool.ntp.org");
}
```



Data / Hora (NTP)

Data/Hora Atual

```
struct tm tempo;
getLocalTime(&tempo);
```

Pedaços Individuais

```
int hora = tempo.tm_hour;
int minuto = tempo.tm_min;
int segundo = tempo.tm_sec;
int dia = tempo.tm_mday;
int mes = tempo.tm_mon + 1;
int ano = tempo.tm_year + 1900;
int diaDaSemana = tempo.tm_wday;
```

Impressão na Serial, Display, Arquivo, etc

```
Serial.println(&tempo, "%d/%m/%Y %H:%M:%S");
display.print(&tempo, "%d/%m/%Y %H:%M:%S");
arquivo.println(&tempo, "%d/%m/%Y %H:%M:%S");
```

Conversão para String

```
char buffer[100];
strftime(buffer, sizeof(buffer), "%d/%m/%Y %H:%M:%S", &tempo);
String tempoString = String(buffer);
```

Setup

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <WiFiClientSecure.h>
#include <ArduinoJson.h>
#include "certificados.h"
```

```
String chaveTelegram = "COLOQUE A SUA CHAVE DO BOTFATHER AQUI!";
String idDoChat = "COLOQUE O ID DA SUA CONVERSA AQUI!";
String enderecoBase = "https://api.telegram.org/bot" + chaveTelegram;
WiFiClientSecure conexaoSegura;
```

```
void setup() {
  Serial.begin(115200);
  reconectarWiFi();

  conexaoSegura
    .setCACert(certificado1);
  conexaoSegura
    .setCACert(certificado2);
}
```



Telegram (POST)

Envio de Mensagem por POST

```
void enviarMensagemTelegram(String mensagem) {
  JsonDocument dados;
  dados["chat_id"] = idDoChat;
  dados["text"] = mensagem;
  String dadosString;
  serializeJson(dados, dadosString);

  String enderecoMensagem = enderecoBase + "/sendMessage";
  HTTPClient requisicao;
  requisicao.begin(conexaoSegura, enderecoMensagem);
  requisicao.addHeader("Content-Type", "application/json");
  int codigoDoResultado = requisicao.POST(dadosString);
  String resposta = requisicao.getString();
  Serial.println(resposta);

  if (codigoDoResultado != 200) {
    Serial.println("Erro ao enviar mensagem!");
  }
}
```