



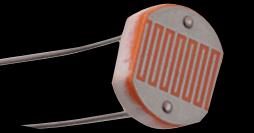
Projeto 03

Coisas Monitoradas – 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);  
    }  
}
```





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

```

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

WebServer servidor(80);

void setup () {
    reconnectWifi();
    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 () {
    reconnectWiFi();
    servidor.handleClient();
}

```

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

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!");
}

```



Servidor Web

Página Com Parâmetros

```

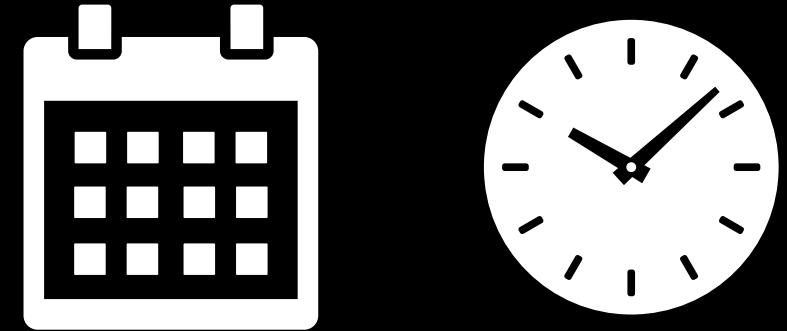
void pagina3 () {
    String texto = servidor.pathArg(0);
    int numero = servidor.pathArg(1).toInt();
    servidor.send(200, "text/html", "Dados ok");
}

```

Setup

```
#include <time.h>

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

```

#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();
}

```

```

  mqtt.publish("tópico", "conteúdo"); // retain = false, qos = 0
  mqtt.publish("tópico2/1234/parametro", "conteúdo 2", false, 1);
}

```

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("tópico", "conteúdo"); // retain = false, qos = 0
  mqtt.publish("tópico2/1234/parametro", "conteúdo 2", false, 1);
}

```

Listagem de Arquivos

```
File pasta = SD_MMC.open("/");
if(!pasta || !pasta.isDirectory()){
    Serial.println("Falha ao abrir diretório");
    while (true){};
}

File arquivo = pasta.openNextFile();
while(arquivo){
    if(!arquivo.isDirectory()){
        String nomeArquivo = String(arquivo.name());
        Serial.println(nomeArquivo);
    }
    arquivo = pasta.openNextFile();
}
```

Setup SD Card

```
#include <SD_MMC.h>

void setup () {
    SD_MMC.setPins(39, 38, 40);
    if (!SD_MMC.begin("/sdcard", true)) {
        Serial.println("Erro SD Card");
        while (true) {};
    }
}
```

Escrita SD Card

```
File arquivo = SD_MMC.open("/arquivo.txt", FILE_WRITE);
arquivo.println("Olá!");
arquivo.close();
```

Página com Foto

```
void paginaComFoto () {
    File arquivo = SD_MMC.open("/foto.jpeg", FILE_READ);
    if (!arquivo) {
        servidor.send(404, "text/plain", "Erro!");
        return;
    }
    servidor.sendHeader("Content-Type", "image/jpeg");
    servidor.sendHeader("Content-Length", String(arquivo.size()));
    servidor.sendHeader("Connection", "close");
    servidor.streamFile(arquivo, "image/jpeg");
    arquivo.close();
}
```



SD Card

Setup

```
#include <esp_camera.h>

camera_config_t config = {
    .pin_pdn = -1, .pin_reset = -1,
    .pin_xclk = 15, .pin_sccb_sda = 4,
    .pin_sccb_scl = 5,
    .pin_d7 = 16, .pin_d6 = 17,
    .pin_d5 = 18, .pin_d4 = 12,
    .pin_d3 = 10, .pin_d2 = 8,
    .pin_d1 = 9, .pin_d0 = 11,
    .pin_vsync = 6, .pin_href = 7,
    .pin_pclk = 13,
    .xclk_freq_hz = 20000000,
    .ledc_timer = LEDC_TIMER_0,
    .ledc_channel = LEDC_CHANNEL_0,
    .pixel_format = PIXFORMAT_JPEG,
    .frame_size = FRAMESIZE_SVGA,
    .jpeg_quality = 10, .fb_count = 2,
    .grab_mode = CAMERA_GRAB_LATEST
};
```

```
void setup () {
    // inicia Serial, WiFi, MQTT
    esp_err_t err = esp_camera_init(&config);
    if (err != ESP_OK) {
        Serial.printf("Erro na câmera: 0x%x", err);
        while (true);
    }
}
```

Salvar Foto no SD Card

```
void tirarFotoESalvarNoSDCard () {
    camera_fb_t* foto = esp_camera_fb_get();

    File arquivo = SD_MMC.open("/foto.jpeg", FILE_WRITE);
    arquivo.write(foto->buf, foto->len);
    arquivo.close();

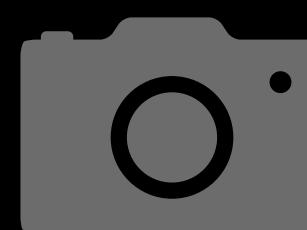
    esp_camera_fb_return(foto); // libera memória
}
```

Enviar Foto no MQTT

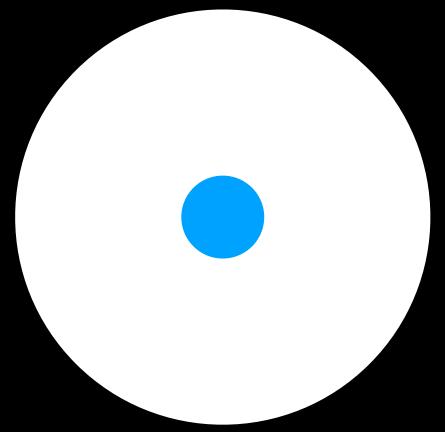
```
void tirarFotoEEviarParaMQTT () {
    camera_fb_t* foto = esp_camera_fb_get();

    if (mqtt.publish( "topico",
                      (const char*)foto->buf, foto->len)) {
        Serial.println("Foto enviada com sucesso");
    } else {
        Serial.println("Falha ao enviar foto");
    }

    esp_camera_fb_return(foto); // libera memória
}
```



Câmera



Beacon Bluetooth

```
class MeuRastreador: public BLEAdvertisedDeviceCallbacks {  
void onResult(BLEAdvertisedDevice dispositivoBluetooth) {  
    String dadosFabricante = dispositivoBluetooth.getManufacturerData();  
    if (dadosFabricante.length() != 25) { return; } // não é Beacon  
  
    BLEBeacon oBeacon = BLEBeacon();  
    oBeacon.setData(dadosFabricante);  
    String idDispositivo = oBeacon.getProximityUUID().toString();  
    if (idDispositivo == idDoMeuBeacon) {  
        scannerBluetooth->stop();  
        int potencia = dispositivoBluetooth.getRSSI();  
        float distancia = calcularDistancia(potencia);  
        Serial.printf("Beacon a %.1f metros!\n", distancia);  
    }  
}  
};
```

```
#include <BLEDevice.h>  
#include <BLEScan.h>  
#include <BLEBeacon.h>  
  
BLEScan* scannerBluetooth;  
String idDoMeuBeacon = "MEU ID!";  
void setup() {  
    BLEDevice::init("");  
    scannerBluetooth = BLEDevice::getScan();  
    scannerBluetooth->setAdvertisedDeviceCallbacks(new MeuRastreador());  
    scannerBluetooth->setActiveScan(true);  
    scannerBluetooth->setInterval(100);  
    scannerBluetooth->setWindow(99);  
};
```

```
float calcularDistancia(int potenciaSinal) {  
    if (potenciaSinal == 0) { return -1.0; } // erro  
    int referencia = -59; // dBm  
    float razao = potenciaSinal * 1.0 / referencia;  
    float distancia;  
    if (razao < 1.0) { distancia = pow(razao, 10); }  
    else {  
        distancia = (0.89976) * pow(razao, 7.7095) + 0.111;  
    }  
    return (float)(int)(distancia * 10 + 0.5) / 10.0;  
}
```

```
void loop() {  
    scannerBluetooth->start(1, true);  
    scannerBluetooth->clearResults();  
}
```

