07

Mesclando MQQT com Robótica

Transcrição

Nosso próximo passo, já que temos o MQTT funcionando. É integrar ele com o nosso projeto, na verdade mesclar. Isso por que iremos copiar o código do exemplo da biblioteca para dentro do nosso código no arquivo `controle` que atualmente se encontra dessa forma:

```
#include <Servo.h>

#define POS_INICIAL 90
#define X_MINIMO 30
#define Y_MINIMO 50
#define X_INTERVALO 12
#define Y_INTERVALO 4
#define LASER D1
#define PIR D2

Servo servoX;
Servo servoY;

void setup() {
    servoX.attach(D3);
    servoY.attach(D4);
    servoX.write(POS_INICIAL);
    servoY.write(POS_INICIAL);

    pinMode(LASER, OUTPUT);
    digitalWrite(LASER, LOW);
    // Serial.begin(9600);
}

void loop() {
    int trigger = digitalReader(PIR);
    if(trigger == HIGH){
        liga_laser();
        // Serial.println("Detectado")
    } else {
        desliga_laser();
        // Serial.println("NÃO Detectado");
    }
    delay(1000);
}

// FUNÇÕES AUXILIARES

void posiciona_motores() {
    int posicaoX = (random(0, (X_INTERVALO)) * 10 + X_MINIMO);
    int posicaoY = (random(0, (Y_INTERVALO)) * 10 + Y_MINIMO);
    servoX.write(posicaoX);
    servoY.write(posicaoY);
    // Serial.print(posicaoX);
```

```
    // Serial.print("  ,  ");
    // Serial.println(posicaoY);
    // delay(2000);
}

void liga_laser() {
    for (int i = 0; i < TEMPO_LASER/2; ++i){
        digitalWrite(LASER, HIGH);
        posiciona_motores();
        delay(2000);
    }
}

void desliga_laser() {
    digitalWrite(LASER, LOW);
}
```

Primeiro copiaremos os includes e logo em seguida as inicializações das variáveis de configuração seguidos dos objetos de `WiFiClient` e `PubSubClient`:

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

const char* ssid = "SEU SSID DE REDE";
const char* password = "SUA SENHA";
const char* mqtt_server = "Disponivel no CloudMQTT";
int mqtt_port = PORTA_DA_INSTANCIA;
const char* mqtt_user = USUARIO_DA_INSTANCIA;
const char* mqtt_password = SENHA_DA_INSTANCIA;

WiFiClient espClient;
PubSubClient client(espClient);
```

No bloco de `setup` precisamos das linhas que executam a função de *setup* da Wi-Fi e logo depois dela a que configura o servidor.

```
setup_wifi();
client.setServer(mqtt_server, mqtt_port);
```

Lembrando sempre de espelhar as seções do código, o que copiamos de um *setup* colamos no outro. Os próximos códigos a serem copiados vão abaixo de todas as funções auxiliares. A primeira delas é a de *setup* do Wi-Fi. Que faz a conexão.

```
void setup_wifi() {
    delay(10);
    Serial.println();
    Serial.print("Connecting to");
    Serial.println(ssid);

    Wifi.begin(ssid, password);

    while(Wifi.status() != WL_CONNECTED){
```

```
        delay(500);
        Serial.print(".");
    }

    Serial.println("");
    Serial.println("WiFI connected");
    Serial.println("IP adress: ");
    Serial.println(WiFi.localIP());
}
```

E a de reconexão também.

```
void reconnect(){
    while(!client.connected()){
        Serial.print("Attempting MQTT connection...");

        if(client.connect("ESP8266Client", mqtt_user, mqtt_password)){
            Serial.println("connected")

            client.public("outTopic", "hello world");

            client.subscribe("inTopic");
        } else {
            Serial.print("falied, rc=");
            Serial.print(client.state());
            Serial.println(" try again in 5 seconds ");

            delay(5000);
        }
    }
}
```

Agora vamos ao bloco `loop` para ajustar o que queremos fazer em nosso projeto. Primeiro copiamos as linhas que fazem a conexão do cliente com o servidor e também a que publica a mensagem.

```
if(!client.connected()){
    reconnect();
}

client.loop();
client.publish("outTopic", msg)
```

A ideia é que dentro do `loop`, após lermos o *status* do detector de presença, enviarmos esse *status* para o serviço do MQTT. Adaptando o `loop` teremos:

```
void loop() {
    if(!client.connected()){
        reconnect();
    }

    client.loop();
    int trigger = digitalReader(PIR);
    client.publish("outTopic", String(trigger).c_str());
```

```
    if(trigger == HIGH){
        liga_laser();
        // Serial.println("Detectado")
    } else {
        desliga_laser();
        // Serial.println("NÃO Detectado");
    }
    delay(1000);
}
```

Note que apenas estamos convertendo o valor retornado pelo `trigger` para texto. Por observação, é função `publish` aceita ainda mais um parâmetro, que é a retenção, ou seja, ele mantêm a última mensagem publicada. Para configurar esse recurso, caso o servidor aceite esse tipo de configuração, basta passar como parâmetro o valor `true` .