

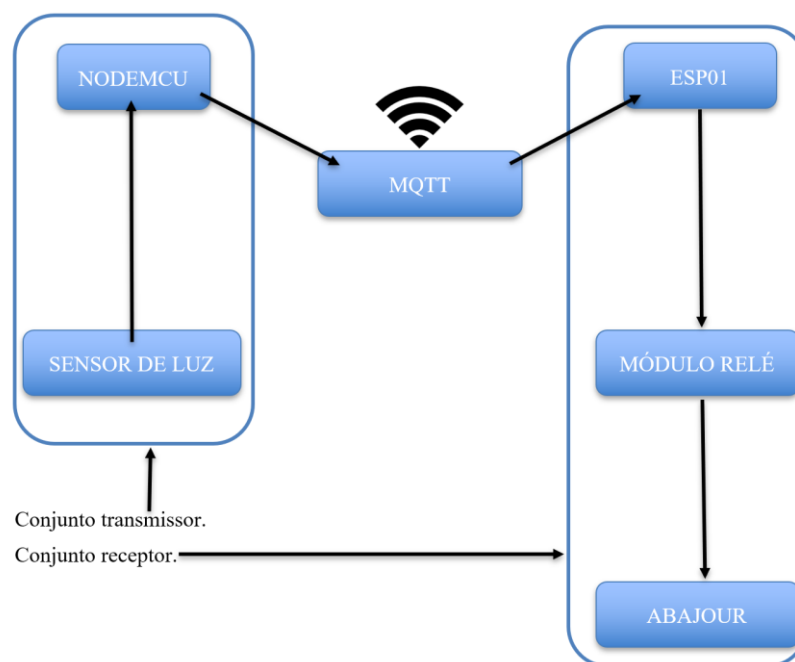


Interfaces, protocolos e módulos de comunicação

Este documento tem o objetivo de mostrar quais foram as interfaces, protocolos e módulos de comunicação utilizados no projeto, para que seja mais fácil para quem quiser reproduzir.

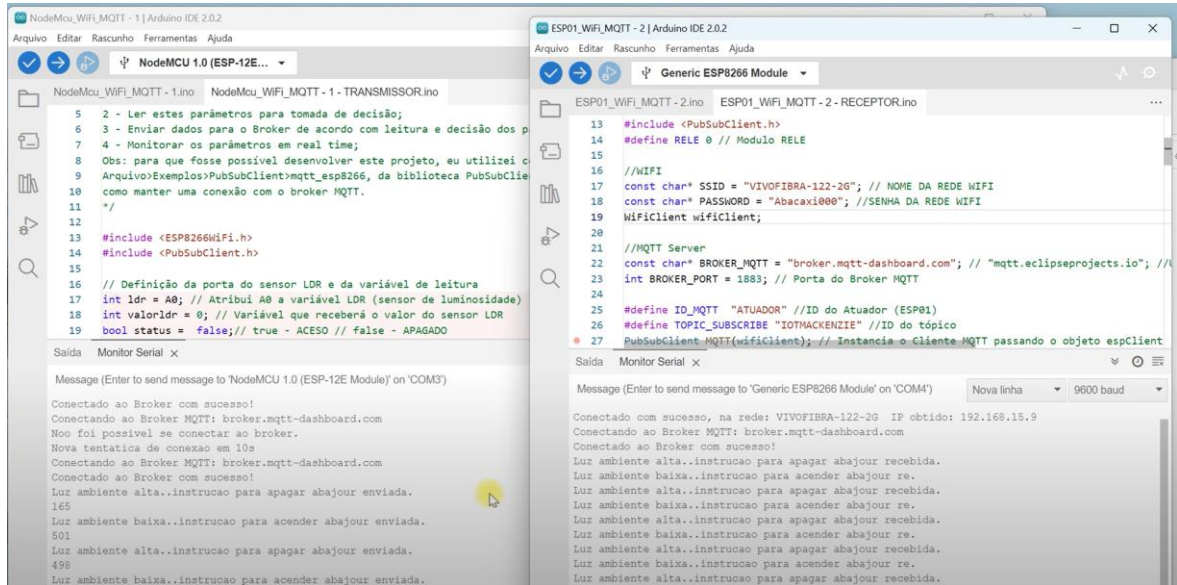
O projeto de Iluminação Noturna Automática utilizou a comunicação de dois dispositivos via internet, através dos protocolos TCP/IP e MQTT.

O protocolo TCP/IP nos ajudou a expor nossos componentes na internet. Abaixo, segue uma ilustração macro do projeto para melhorar o entendimento da explicação:



O conjunto transmissor utiliza o protocolo TCP/IP de uma rede convencional doméstica para se conectar. Ele faz isso através do módulo wi-fi integrado ao componente microcontrolador NodeMcu. Uma vez conectado na internet, o próximo passo é se conectar com um broker através do protocolo MQTT, que é uma sigla de Message Queuing Telemetry Transport. Este protocolo tem por finalidade proporcionar a troca de mensagens leves entre sensores e dispositivos móveis que rodam sob a a rede TCP/IP. Através do MQTT é possível que o nosso conjunto de transmissão envie dados para que o nosso conjunto de recepção os receba e tome uma ação, que no nosso caso é ligar ou desligar o abajur.

Abaixo, segue uma ilustração real da comunicação dos componentes acontecendo em tempo real:



The image displays two side-by-side screenshots of the Arduino IDE 2.0.2, showing the code for two different microcontroller modules connected via MQTT.

Left Screenshot (NodeMCU 1.0 (ESP-12E...)):

```
NodeMcu_WIFI_MQTT - 1.ino NodeMcu_WIFI_MQTT - 1-TRANSMISSOR.ino
5 2 - Ler estes parâmetros para tomada de decisão;
6 3 - Enviar dados para o Broker de acordo com leitura e decisão dos p
7 4 - Monitorar os parâmetros em real time;
8 Obs: para que fosse possível desenvolver este projeto, eu utilizei c
9 Arquivo>Exemplos>PubSubClient>mqtt_esp8266, da biblioteca PubSubClie
10 como manter uma conexão com o broker MQTT.
11 */
12
13 #include <ESP8266WiFi.h>
14 #include <PubSubClient.h>
15
16 // Definição da porta do sensor LDR e da variável de leitura
17 int ldr = A0; // Atribui A0 a variável LDR (sensor de luminosidade)
18 int valorldr = 0; // Variável que receberá o valor do sensor LDR
19 bool status = false; // true - ACESO // false - APAGADO
```

Right Screenshot (Generic ESP8266 Module):

```
ESP01_WIFI_MQTT - 2.ino ESP01_WIFI_MQTT - 2-RECEPTOR.ino
13 #include <PubSubClient.h>
14 #define RELE 0 // Módulo RELE
15
16 //WIFI
17 const char* SSID = "VIVOFIBRA-122-26"; // NOME DA REDE WIFI
18 const char* PASSWORD = "Abacaxi000"; // SENHA DA REDE WIFI
19 WiFiClient wifiClient;
20
21 //MQTT Server
22 const char* BROKER_MQTT = "broker.mqtt-dashboard.com"; // "mqtt.eclipseprojects.io"; //
23 int BROKER_PORT = 1883; // Porta do Broker MQTT
24
25 #define ID_MQTT "ATUADOR" //ID do Atuador (ESP01)
26 #define TOPIC_SUBSCRIBE "IOTHACKENZIE" //ID do tópico
27 PubSubClient MQTT(wifiClient); // Instancia o Cliente MQTT passando o objeto espClient
```

Both screenshots show the serial monitor output, indicating successful connection to the MQTT broker and the transmission of data based on the LDR sensor readings.

Para este projeto, utilizamos um broker que fornece a comunicação via protocolo MQTT sem custos, por ser um trabalho acadêmico, porém, existem alguns serviços pagos com redundância, disponibilidade e outros recursos.