**ML05\_ESP01\_MQTTklient-RPI\_MQTTbroker\_NodeRED**

**Téma:** Cvičení na fyzickém modulu ESP-01 demonstruje MQTT klienta, který bezdrátově komunikuje přes AP s MQTT brokerem na RPI. Druhého klienta tvoří prostředí Node-Red na RPI připojený k MQTT brokeru.

**Cíl hodiny:** Vytvořit úpravou existujícího kódu MQTT klienta na modulu ESP-01 přijímajícího zprávu topic z MQTT brokeru ovládajícího diodu LED. V prostředí Node-Red publikovat zprávu topic na MQTT broker pro ovládání diody LED. V druhé části cvičení vytvořit úpravu existujícího kódu MQTT klienta na modulu ESP-01 publikujícího zprávu topic na MQTT broker z teplotního čidla Dallas. V prostředí Node-Red přijímat zprávu topic z MQTT brokeru a zobrazovat ji (debug).

**Doba trvání:** 4x45min

**Pomůcky:** Rasberry Pi 3, AP Mikrotik, nepájivé pole, ESP-01, USB/UART převodník, Dallas (DS1820), 2x rezistor 4K7 Ω, 1x rezistor 220 Ω, tlačítko, LED dioda, propojovací vodiče, program ESP\_MQTT

# Upgrade Node-RED a přidání palety Dashboard

Pro možnost správy nabídky palet ve vývojovém prostředí Node-RED je nutno provést upgrade (připojením na internet) pomocí skriptu:

pi@raspberrypi:~ $ bash <(curl -sL <https://raw.githubusercontent.com/node-red/raspbian-deb-package/master/resources/update-nodejs-and-nodered>)

Poté doinstalujte paletu node-red-dashboard

Obsah obrázku snímek obrazovky, interiér, monitor, počítač

Popis byl vytvořen automaticky

# Úprava programu ESP\_MQTT

#include <ESP8266WiFi.h>

#include <WiFiClient.h>

#include <ESP8266WebServer.h>

#include <ESP8266mDNS.h>

#include <OneWire.h>

#include <DallasTemperature.h>

#include <PubSubClient.h>

//pro teploměr

const int ONE\_WIRE\_BUS = 0;

#define TEMPERATURE\_PRECISION 9

OneWire oneWire(ONE\_WIRE\_BUS);

DallasTemperature sensors(&oneWire);

DeviceAddress insideThermometer, outsideThermometer;

float teplota;

String termOUT;

String termIN;

volatile int posli;

const char \*ssid = "IoT";

const char \*password = "SPSvos123";

const char\* mqtt\_server = "192.168.100.100";

const int LED\_Pin = 2;

WiFiClient espClient;

PubSubClient client(espClient);

long lastMsg = 0;

char msg[50];

int value = 0;

MDNSResponder mdns;

ESP8266WebServer server(80);

void setup(void) {

pinMode(LED\_BUILTIN, OUTPUT);

digitalWrite(LED\_BUILTIN,LOW);

pinMode(LED\_Pin, OUTPUT);

digitalWrite(LED\_Pin, HIGH);

Serial.begin(115200);

setup\_wifi();

client.setServer(mqtt\_server, 1883);

client.setCallback(callback);

//teplomery

sensors.begin();

Serial.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Serial.print("Pocet teplomeru: ");

Serial.println(sensors.getDeviceCount(), DEC);

//zjisti adresy

oneWire.reset\_search();

if (!oneWire.search(insideThermometer)) Serial.println("Vnitrni teplomer nenalezen!");

if (!oneWire.search(outsideThermometer)) Serial.println("Vnejsi teplomer nenalezen!");

Serial.print("Adresa teplomeru 1: ");

printAddress(insideThermometer);

Serial.println();

//Serial.print("Adresa teplomeru 2: ");

//printAddress(outsideThermometer);

//Serial.println();

//nastav rozlišení

sensors.setResolution(insideThermometer, TEMPERATURE\_PRECISION);

sensors.setResolution(outsideThermometer, TEMPERATURE\_PRECISION);

//načti všechny teploměry

sensors.requestTemperatures();

//vytiskni data na seriák

printData(insideThermometer);

printData(outsideThermometer);

server.on("/", handleRoot);

server.on("/inline", []() {

server.send(200, "text/plain", "this works as well");

});

server.onNotFound(handleNotFound);

server.begin();

Serial.println("HTTP server started");

//digitalWrite(LED\_BUILTIN, HIGH);

}

void setup\_wifi() {

delay(10);

// We start by connecting to a WiFi network

Serial.println();

Serial.print("Connecting to ");

Serial.println(ssid);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

randomSeed(micros());

Serial.println("");

Serial.println("WiFi connected");

Serial.println("IP address: ");

Serial.println(WiFi.localIP());

if (mdns.begin("esp8266", WiFi.localIP())) {

Serial.println("MDNS responder started");

}

}

//pro server

void handleRoot()

{

digitalWrite(LED\_BUILTIN, 1);

char temp[450];

int sec = millis() / 1000;

int min = sec / 60;

int hr = min / 60;

uint8 index = termOUT.indexOf('.');

int restOUT = termOUT.substring(index+1).toInt();

int temperatureOUT = termOUT.toInt();

index = termIN.indexOf('.');

int restIN = termIN.substring(index + 1).toInt();

int temperatureIN = termIN.toInt();

snprintf(temp, 450,

"<html>\

<head>\

<meta http-equiv='refresh' content='5'/>\

<title>ESP8266 Demo</title>\

<style>\

body { background-color: #cccccc; font-family: Arial, Helvetica, Sans-Serif; Color: #000088; }\

</style>\

</head>\

<body>\

<h1>ESP1</h1>\

<p>Uptime: %02d:%02d:%02d</p>\

<p>Teplota 1: %02d.%01d&deg;C</p>\

<p>Teplota 2: %02d.%01d&deg;C</p>\

</body>\

</html>",

hr, min % 60, sec % 60, temperatureOUT,restOUT,temperatureIN,restIN

);

server.send(200, "text/html", temp);

digitalWrite(LED\_BUILTIN, 0);

}

void handleNotFound() {

digitalWrite(LED\_BUILTIN, 1);

String message = "File Not Found\n\n";

message += "URI: ";

message += server.uri();

message += "\nMethod: ";

message += (server.method() == HTTP\_GET) ? "GET" : "POST";

message += "\nArguments: ";

message += server.args();

message += "\n";

for (uint8\_t i = 0; i < server.args(); i++) {

message += " " + server.argName(i) + ": " + server.arg(i) + "\n";

}

server.send(404, "text/plain", message);

digitalWrite(LED\_BUILTIN, 0);

}

void callback(char\* topic, byte\* payload, unsigned int length) {

Serial.print("Message arrived [");

Serial.print(topic);

Serial.print("] "); //vypis na konzolu: "[topic..]"

Serial.print((char\*)payload);

/\*

for (int i = 0; i < length; i++) {

Serial.print((char)payload[i]); // vypis na konzolu: "payload"

}

\*/

Serial.println();

// Switch on the LED if an 1 was received as first character

if ((char)payload[0] == '1') {

digitalWrite(LED\_Pin, LOW); // Turn the LED on (Note that LOW is the voltage level

// but actually the LED is on; this is because

// it is acive low on the ESP-01)

} else {

digitalWrite(LED\_Pin, HIGH); // Turn the LED off by making the voltage HIGH

}

}

void reconnect() {

// Loop until we're reconnected

while (!client.connected()) {

Serial.print("Attempting MQTT connection...");

// Create a random client ID

String clientId = "ESP8266Client-";

clientId += String(random(0xffff), HEX);

// Attempt to connect

if (client.connect(clientId.c\_str())) {

Serial.println("connected");

// Once connected, publish an announcement...

client.publish("ESP1-Teplota", "topic: ESP1-Teplota");

// ... and resubscribe

client.subscribe("ESP1-LED");

} else {

Serial.print("failed, rc=");

Serial.print(client.state());

Serial.println(" try again in 5 seconds");

// Wait 5 seconds before retrying

delay(5000);

}

}

}

void loop(void)

{

if (!client.connected()) {

reconnect();

}

client.loop();

long now = millis();

if (now - lastMsg > 2000) {

lastMsg = now;

++value;

snprintf (msg, 75, "%ld", posli);

Serial.print("ESP1-Teplota: ");

Serial.println(msg);

client.publish("ESP1-Teplota", msg);

digitalWrite(LED\_BUILTIN, LOW);

mdns.update();

server.handleClient();

sensors.requestTemperatures();

teplota = sensors.getTempC(insideThermometer);

posli = 10\*teplota;

Serial.println(posli);

digitalWrite(LED\_BUILTIN, HIGH);

termOUT = String(teplota, 1);

teplota = sensors.getTempC(outsideThermometer);

termIN = String(teplota, 1);

digitalWrite(LED\_BUILTIN, LOW);

// vytiskni data na seriák

printData(insideThermometer);

printData(outsideThermometer);

}

}

//pro teploměry

void printAddress(DeviceAddress deviceAddress)

{

for (uint8\_t i = 0; i < 8; i++)

{

// zero pad the address if necessary

if (deviceAddress[i] < 16) Serial.print("0");

Serial.print(deviceAddress[i], HEX);

}

}

void printData(DeviceAddress deviceAddress)

{

Serial.print("Adresa teplomeru ");

printAddress(deviceAddress);

Serial.print(":");

printTemperature(deviceAddress);

}

void printTemperature(DeviceAddress deviceAddress)

{

float tempC = sensors.getTempC(deviceAddress);

Serial.print("Teplota: ");

Serial.print(tempC);

Serial.write(176);

Serial.println("C");

}

# Vytvoření IoT sítě

1. Připojte RPI k AP a zapněte služby MQTT a Node-RED
2. Připojte se k bezdrátové síti IoT, ve Winbox/DHCP server by měla přibýt dynamicky zapůjčená adresa, nastavte ji jako statickou 192.168.100.100.
3. V konzole ověřte stav – spuštění aplikace node-red příkazem:  
   $ sudo systemctl status nodered
4. Ověřte stav MQTT brokeru příkazem:  
   sudo systemctl status mosquitto.service (případně restart)
5. Zapněte bezdrátový modul ESP1 zkontrolujte jeho bezdrátové připojení k AP, jednak v AP Mikrotik/DHCPserver/leases a také na RPI příkazem:  
   sudo ss -a -p | grep mosq
6. V prostředí Node-Red vytvořte flow umožňující komunikaci s bezdrátovým modulem ESP-01 a zpracování topiců ESP1-Teplota, ESP1-LED pro zobrazení v dashboardu.
7. V prohlížeči otevřete Node-Red na socketu: 192.168.100.100:1880 a vytvořte flow ESP1 podle obrázku (upravte název na ESP1, dashboard graficky upravte):  
   Obsah obrázku snímek obrazovky

   Popis byl vytvořen automaticky
8. V prohlížeči otevřete Node-Red na socketu: 192.168.100.100:1880/ui sledujte výpis teploty a ovládejte přepínačem diodu LED na ESP1-LED  
   Obsah obrázku snímek obrazovky, monitor

   Popis byl vytvořen automaticky

# Vypracování PL05 – ESP01\_MQTTklient-RPI\_MQTTbroker\_NodeRED

Při úspěšném zpracování pracovního listu by žáci měli zvládnout:

* Nahrát upravený program „ESP\_MQTT“ na ESP-01 pro komunikaci pomocí protokolu MQTT.
* Připojit RPI k AP a zprovoznit služby MQTT a Node-RED
* Ovládat LED diodu na ESP-01 z Node-RED (1-zapnout, 0-vypnout) a zobrazovat teplotu z čidla Dallas na ESP-01 v Node-Red (výpis na Debug).
* Graficky zobrazit zasílané hodnoty a povely pomocí dashboardu