Univerzitet u Sarajevu
Elektrotehnički fakultet
Ugradbeni sistemi 2023 / 24

Izvještaj za laboratorijsku vježbu br. 7

Komunikacija – primjer korištenja MQTT protokola

Ime i prezime: **Ismar Muslić**
Broj index-a: **19304**

23. april 2024.

Sadržaj

1 Pseudokod

Zbog generalne kompleksnosti implementacije svih zadataka, pseudokod je dat kao intuitivna ideja kako pristupiti rješavanju problema, dok su konkretne implementacije svih zadataka date u prilogu ovog izvještaja.

Zadaci 1 i 2 dijele isti pseudokod jer izvršavaju identične funkcionalnosti, samo na različitim sistemima. Pseudokod zadatka 3 je nešto proširen, proporcionalno sa traženim dodatnim funkcionalnostima u odnosu na 2. zadatak.

1.1 Zadatak 1 i 2

```
while(true)
      ocitajISaljiStanjeTastera()
      ocitajISaljiStanjePotenciometra()
      provjeriDolazecePoruke()
      if(poruka za led1) podesiLed1()
      if(poruka za led2) podesiLed2()
      if(poruka za led3) podesiLed3()
      sleep(0.1s)
```

1.2 Zadatak 3

```
ocitajISaljiStanjeSenzoraSvakih2s()
while(true)
      ocitajISaljiStanjeTastera()
      ocitajISaljiStanjePotenciometra()
      provjeriDolazecePoruke()
      if(poruka za led1) podesiLed1()
      if(poruka za led2) podesiLed2()
      if(poruka za led3) podesiLed3()
      if(poruka za RGB) podesiRGB()
      sleep(0.1s)
```

2 Analiza programskog rješenja

2.1 Zadatak 1

Prvi zadatak na ovoj laboratorijskoj vježbi predstavljao je podešavanje komunikacije između Mbed simulatora i MQTTX desktop aplikacije, kao i MQTT Dash mobilne aplikacije za realizaciju datog programskog koda u Mbed OS-u koji upravalja komponentama spojenim na simulator. Od komponenata koje su spojene moguće je upravljati s LED diodama 1 i 2 (DigitalOut) kao i sa LED3 (PWMOut). Stanje tastera i potenciometra se ovisno o pritisku, odnosno podešenju otklona, šalje kao obavijest na odgovarajući topic u okviru MQTT protokola.

Potrebno je bilo odgovarajuće komponente povezati na virtualni sistem u okviru Mbed simulatora, te modificirati dati programski kod kako bi komunikacija bila unikatna između korisnika i sistema. Naime, bilo je potrebno u programskom kodu izmijeniti string koji predstavlja odgovarajuće „teme“ na kojima sistem komunicira preko MQTT protokola, kako bi se osiguralo da na istoj temi ne komunicira više sistema.

Zatim, u okviru MQTTX i MQTT Dash aplikacija se kreiraju konekcije na brokera, preko kojeg se upravlja LED diodama na sistemu, odnosno preko kojeg se primaju poruke o stanju potenciometra i tastera. Tako se o stanju tastera i potenciometra moglo svjedočiti u okviru aplikacija, s obzirom na to da na svaku promjenu sistem šalje poruku na odgovarajući topic. Također, omogućeno je upravljanje diodama LED1 i LED2 slanjem poruke sa stanjem na odgovarajući topic, kao i sa diodom LED3, postavljanjem vrijednosti duty – cycle-a također slanjem poruke na odgovarajući topic.

2.2 Zadatak 2

Zadatak 2 je konceptualno identičan kao i zadatak 1, s time da je potrebno prilagoditi ga za rad na razvojnom sistemu picoETF. Bilo je potrebno programski kod prilagoditi iz C++-a u MicroPython, te omogućiti WiFi konekciju na sistemu RP2040. Također je za korištenje klase *MQTTClient* koja je neophodna za podešavanje komunikacije MQTT protokolom bilo potrebno importovanje biblioteke *simple.py* koja je prebačena na RP2040. Nakon što je podešena WiFi konekcija i RP2040 uspješno povezan na internet, moguće je koristiti iste konekcije na aplikacijama MQTTX i MQTT Dash za ostvarivanje identične funkcionalnosti kao u prvom zadatku.

2.3 Zadatak 3

Treći zadatak predstavlja nadogradnju prethodnog zadatka, s obzirom na to da je samo bilo potrebno dodati funkcionalnosti za očitavanje temperature i vlažnosti zraka sa digitalnog senzora DHT11, te upravljanje RGB LED diodom, odnosno svakoj od njenih komponenata pomoću PWM-a, kako bi se mogao dobiti kompletan spektar mogućih boja.

Prvo, dodani su novi objekti koji odgovaraju pinovima na koje su povezani DHT11 (digitalni ulaz, Single-Wire comm, GP28) i RGB LED dioda (PWM izlaz, GP5-GP7). Zatim su u kodu podešene nove „teme“ za komunikaciju sa ovim komponentama, te je izvršen „subscribing“ na iste. Zatim su definirane funkcije koje će služiti kao callback funkcije svaki put kada stigne poruka na odgovarajući topic za kontrolu stanja RGB diode, slično kako se to radilo sa LED3.

Što se tiče senzora DHT11, podešen je Ticker koji će svako 2 sekunde generisati interrupt, te slati poruku na odgovarajući topic, a koja sadrži podatke o trenutnom očitanju temperature i vlažnosti zraka.

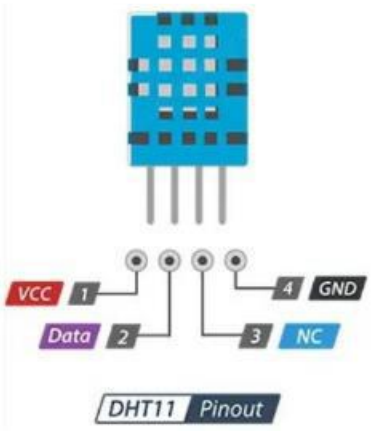
Na ovaj način je proširena implementacija 2. zadatka sa traženim funkcionalnostima u 3. zadatku. Detaljniji komentari o radu svakog dijela koda su dati u sklopu programskog koda u prilogu 5.3.

3 Korišteni hardverski resursi

Za potrebe ove laboratorijske vježbe korišteni sistem je picoETF, na njemu ugrađene LED diode, RGB LED dioda, kao i fizički taster. Osim toga, od dodatne opreme korišteno je sljedeće:

- Rotacijski potenciometar
- DHT11 Digitalni senzor temperature i vlažnosti zraka

ULAZI	IZLAZI
Potenciometar (analogni; GP26/GP28)	LED1-3 (digitalni; GP4-GP6)
DHT11 senzor (digitalni, GP28)	RGB LED (digitalni; GP12 – GP14)
Taster (ugrađen na picoETF) (digitalni, GP3)	



Slika 1 – Pinout DHT11 senzora korištenog u zadatku 3

4 Zaključak

Prilikom izrade laboratorijske vježbe 7 nije bilo poteškoća. Potrebno je bilo prilagoditi callback funkciju u MicroPython programskom kodu, s obzirom na to da klasa MQTTClient može za jedan objekt da postavi samo jednu callback funkciju. Iz tog razloga je korištena wrapper callback procedura, koja na osnovu datog topica prosljeđuje poruku odgovarajućoj proceduri za upravljanje datoj komponenti u sklopu cjelokupnog sistema.

Cilj ove laboratorijske vježbe je bio upoznavanje sa MQTT protokolom za komunikaciju, što je uspješno i postignuto.

5 Prilog

5.1 Zadatak 1/Izvorni kod

```
#define TEMASUBLED1 "i&v/led1"
#define TEMASUBLED2 "i&v/led2"
#define TEMASUBLED3 "i&v/led3"
#define TEMAPUBPOT "i&v/potenciometar"
#define TEMAPUBTAST "i&v/taster"

#include "mbed.h"

#define MQTTCLIENT_QOS2 0

#include "MQTTNetwork.h"
#include "MQTTmbed.h"
#include "MQTTClient.h"
#include <string.h>

int arrivedcount = 0;

#define MQTT_CLIENT_NAME "mbedSim"
DigitalIn taster(p5);
DigitalOut led1(p6);
DigitalOut led2(p7);
AnalogIn pot(p15);
PwmOut led3(p21);

char* str;
double pot_value=-1;
bool taster_state=1;

void messageArrived_led1(MQTT::MessageData& md)
{
    MQTT::Message &message = md.message;
    printf("Message arrived: qos %d, retained %d, dup %d, packetid %d\r\n",
message.qos, message.retained, message.dup, message.id);
    printf("Payload %.*s\r\n", message.payloadlen, (char*)message.payload);
    ++arrivedcount;
    str=(char*)message.payload;
    led1=atoi(str);
}

void messageArrived_led2(MQTT::MessageData& md)
{
    MQTT::Message &message = md.message;
    printf("Message arrived: qos %d, retained %d, dup %d, packetid %d\r\n",
message.qos, message.retained, message.dup, message.id);
    printf("Payload %.*s\r\n", message.payloadlen, (char*)message.payload);
    ++arrivedcount;
    str=(char*)message.payload;
    led2=atoi(str);
}
```

```
void messageArrived_led3(MQTT::MessageData& md)
{
    MQTT::Message &message = md.message;
    printf("Message arrived: qos %d, retained %d, dup %d, packetid %d\r\n",
message.qos, message.retained, message.dup, message.id);
    printf("Payload %.*s\r\n", message.payloadlen, (char*)message.payload);
    ++arrivedcount;
    str=(char*)message.payload;
    led3=atof(str);
}


int main(int argc, char* argv[])
{
    printf("Ugradbeni sistemi\r\n");
    printf("Demonstracija korištenja MQTT protokola\r\n\r\n");

    SocketAddress a;

    NetworkInterface *network;
    network = NetworkInterface::get_default_instance();

    if (!network) {
        return -1;
    }
    MQTTNetwork mqttNetwork(network);

    MQTT::Client<MQTTNetwork, Countdown> client(mqttNetwork);

    const char* hostname = "broker.hivemq.com";
    int port = 1883;
    printf("Connecting to %s:%d\r\n", hostname, port);
    int rc = mqttNetwork.connect(hostname, port);
    if (rc != 0)
        printf("rc from TCP connect is %d\r\n", rc);
    else
        printf("Connected to broker!\r\n");

    MQTTPacket_connectData data = MQTTPacket_connectData_initializer;
    data.MQTTVersion = 3;
    data.clientID.cstring = MQTT_CLIENT_NAME;
    data.username.cstring = "";
    data.password.cstring = "";
    if ((rc = client.connect(data)) != 0)
        printf("rc from MQTT connect is %d\r\n", rc);

    if ((rc = client.subscribe(TEMASUBLED1, MQTT::QOS0, messageArrived_led1)) != 0)
        printf("rc from MQTT subscribe is %d\r\n", rc);
    else
        printf("Subscribed to %s\r\n", TEMASUBLED1);
```

```
    if ((rc = client.subscribe(TEMASUBLED2, MQTT::QOS0, messageArrived_led2)) != 0)
        printf("rc from MQTT subscribe is %d\r\n", rc);
    else
        printf("Subscribed to %s\r\n", TEMASUBLED2);

    if ((rc = client.subscribe(TEMASUBLED3, MQTT::QOS0, messageArrived_led3)) != 0)
        printf("rc from MQTT subscribe is %d\r\n", rc);
    else
        printf("Subscribed to %s\r\n", TEMASUBLED3);

    MQTT::Message message;

    char buf[100];
    while(1) {

        if (taster_state!=taster) {
            taster_state=taster;
            sprintf(buf, "{\"Taster\": %d}", taster.read());
            message.qos = MQTT::QOS0;
            message.retained = false;
            message.dup = false;
            message.payload = (void*)buf;
            message.payloadlen = strlen(buf);
            rc = client.publish(TEMAPUBTAST, message);
        }
        if (pot_value!=pot) {
            pot_value=pot;
            sprintf(buf, "{\"Potenciometar\": %f}", pot_value);
            message.qos = MQTT::QOS0;
            message.retained = false;
            message.dup = false;
            message.payload = (void*)buf;
            message.payloadlen = strlen(buf);
            rc = client.publish(TEMAPUBPOT, message);
        }

        rc = client.subscribe(TEMASUBLED1, MQTT::QOS0, messageArrived_led1);
        rc = client.subscribe(TEMASUBLED2, MQTT::QOS0, messageArrived_led2);
        rc = client.subscribe(TEMASUBLED3, MQTT::QOS0, messageArrived_led3);

        wait_us(100);
    }

}
```

5.2 Zadatak 2/Izvorni kod

```
from machine import Pin, ADC, PWM
import network
import time
import simple

# Konfiguracija WiFi veze
wifi_ssid = "Lab220"
wifi_password = "lab220lozinka"

# Konfiguracija MQTT protokola
mqtt_server = "broker.hivemq.com"
mqtt_topic_led1 = b"i&v/led1"
mqtt_topic_led2 = b"i&v/led2"
mqtt_topic_led3 = b"i&v/led3"
mqtt_topic_pot =  b"i&v/potenciometar"
mqtt_topic_taster = b"i&v/taster"
mqtt_client_name = "mbedSim"

# Inicijalizacija mreže
print("Connecting to WiFi: ", wifi_ssid)
wifi = network.WLAN(network.STA_IF)
wifi.active(True)
wifi.connect(wifi_ssid, wifi_password)

# Čekanje na vezu
while not wifi.isconnected():
    pass

# Uspješno povezan na WiFi
print("Connected to WiFi")
print("IP address:", wifi.ifconfig()[0])

# Podešavanje pinova
taster = Pin(3, Pin.IN)
led1 = Pin(4, Pin.OUT)
led2 = Pin(5, Pin.OUT)
pot = ADC(Pin(28))
led3 = PWM(Pin(6))
led3.freq(1000)

# Callback funkcije za primljene poruke (kontrola LED dioda)
def message_arrived_led1(topic, msg):
    print("Message arrived on topic:", topic)
    print("Payload:", msg)
    led1.value(int(float(msg)))

def message_arrived_led2(topic, msg2):
    print("Message arrived on topic:", topic)
    print("Payload:", msg2)
    led2.value(int(float(msg2)))
```

```
def message_arrived_led3(topic, msg3):
    print("Message arrived on topic:", topic)
    print("Payload:", msg3)
    led3.duty_u16(int(float(msg3)*65535))

# Custom callback dispatcher procedura za prosljeđivanje poruke na tačan topic
def custom_dispatcher(topic, msg):
    if topic == mqtt_topic_led1:
        message_arrived_led1(topic, msg)
    elif topic == mqtt_topic_led2:
        message_arrived_led2(topic, msg)
    elif topic == mqtt_topic_led3:
        message_arrived_led3(topic, msg)

# Inicijalizacija varijabli za status potenciometra i tastera
taster_state = taster.value()
last_pot_value = pot.read_u16()

# Povezivanje na MQTT broker
client = simple.MQTTClient(client_id=mqtt_client_name,server=mqtt_server,port=1883)
client.connect()

# Subscribe na teme
client.set_callback(custom_dispatcher)
client.subscribe(mqtt_topic_led1)
client.set_callback(custom_dispatcher)
client.subscribe(mqtt_topic_led2)
client.set_callback(custom_dispatcher)
client.subscribe(mqtt_topic_led3)

# Main loop
while True:
    # Slanje stanja tastera
    if taster.value() != taster_state:
        taster_state = taster.value()
        buf = "{{\"Taster\": {}}}".format(taster_state)
        client.publish(mqtt_topic_taster, buf)

    # Slanje vrijednosti potenciometra
    pot_value = pot.read_u16()
    if pot_value != last_pot_value:
        last_pot_value = pot_value
        buf = "{{\"Potenciometar\": {}}}".format(pot_value)
        client.publish(mqtt_topic_pot, buf)

    # Provjera dolazećih poruka, na dolazeću poruku se poziva Dispatcher za callback
    client.check_msg()
    time.sleep(0.1)
```

5.3 Zadatak 3/Izvorni kod

```
from machine import Pin, ADC, PWM, Timer
import network
import time
import simple
import dht

# Konfiguracija WiFi veze
wifi_ssid = "Asterix"
wifi_password = "123a456d78"

# Konfiguracija MQTT protokola
mqtt_server = "broker.hivemq.com"
mqtt_topic_led1 = b"i&v/led1"
mqtt_topic_led2 = b"i&v/led2"
mqtt_topic_led3 = b"i&v/led3"
mqtt_topic_pot =  b"i&v/potenciometar"
mqtt_topic_taster = b"i&v/taster"
mqtt_topic_rgbRed = b"i&v/rgbRed"
mqtt_topic_rgbGreen = b"i&v/rgbGreen"
mqtt_topic_rgbBlue = b"i&v/rgbBlue"
mqtt_topic_sensor = b"i&v/dht11"
mqtt_client_name = "mbedSim"

# Inicijalizacija mreže
print("Connecting to WiFi: ", wifi_ssid)
wifi = network.WLAN(network.STA_IF)
wifi.active(True)
wifi.connect(wifi_ssid, wifi_password)

# Čekanje na vezu
while not wifi.isconnected():
    pass

# Uspješno povezan na WiFi
print("Connected to WiFi")
print("IP address:", wifi.ifconfig()[0])

# Podešavanje pinova
taster = Pin(3, Pin.IN)
led1 = Pin(0, Pin.OUT)
led2 = Pin(1, Pin.OUT)
pot = ADC(Pin(26))
led3 = PWM(Pin(2))
led3.freq(1000)

rgbRed = PWM(Pin(5))
rgbGreen = PWM(Pin(6))
rgbBlue = PWM(Pin(7))
rgbRed.freq(1000)
rgbGreen.freq(1000)
rgbBlue.freq(1000)
```

```
sensor = dht.DHT11(Pin(28))

# Callback funkcije za primljene poruke (kontrola LED dioda)
def message_arrived_led1(topic, msg):
    print("Message arrived on topic:", topic)
    print("Payload:", msg)
    led1.value(int(float(msg)))

def message_arrived_led2(topic, msg):
    print("Message arrived on topic:", topic)
    print("Payload:", msg)
    led2.value(int(float(msg)))

def message_arrived_led3(topic, msg):
    print("Message arrived on topic:", topic)
    print("Payload:", msg)
    led3.duty_u16(int(float(msg)*65535))

def message_arrived_rgbRed(topic, msg):
    print("Message arrived on topic:", topic)
    print("Payload:", msg)
    rgbRed.duty_u16(int(float(msg)*65535))

def message_arrived_rgbGreen(topic, msg):
    print("Message arrived on topic:", topic)
    print("Payload:", msg)
    rgbGreen.duty_u16(int(float(msg)*65535))

def message_arrived_rgbBlue(topic, msg):
    print("Message arrived on topic:", topic)
    print("Payload:", msg)
    rgbBlue.duty_u16(int(float(msg)*65535))

# Custom callback dispatcher procedura za prosljeđivanje poruke na tačan topic
def custom_dispatcher(topic, msg):
    if topic == mqtt_topic_led1:
        message_arrived_led1(topic, msg)
    elif topic == mqtt_topic_led2:
        message_arrived_led2(topic, msg)
    elif topic == mqtt_topic_led3:
        message_arrived_led3(topic, msg)
    elif topic == mqtt_topic_rgbRed:
        message_arrived_rgbRed(topic, msg)
    elif topic == mqtt_topic_rgbGreen:
        message_arrived_rgbGreen(topic, msg)
    elif topic == mqtt_topic_rgbBlue:
        message_arrived_rgbBlue(topic, msg)

# Interrupt rutina za senzor
def sensorReport(pin):
    sensor.measure()
```

```
    temperature = sensor.temperature()
    humidity = sensor.humidity()
    publish = str("Sensor\nTemp: "+str(temperature)+"\nHumidity: "+str(humidity))
    buf = "{{\"Senzor\": \n{}}}".format(publish)
    client.publish(mqtt_topic_sensor, publish)

# Inicijalizacija varijabli za status potenciometra i tastera
taster_state = taster.value()
last_pot_value = pot.read_u16()

# Povezivanje na MQTT broker
client = simple.MQTTClient(client_id=mqtt_client_name,server=mqtt_server,port=1883)
client.connect()

# Subscribe na teme
client.set_callback(custom_dispatcher)
client.subscribe(mqtt_topic_led1)
client.subscribe(mqtt_topic_led2)
client.subscribe(mqtt_topic_led3)
client.subscribe(mqtt_topic_rgbRed)
client.subscribe(mqtt_topic_rgbGreen)
client.subscribe(mqtt_topic_rgbBlue)
client.subscribe(mqtt_topic_sensor)

# Inicijalizacija Timera za interrupte senzora
t = Timer(period=10000, callback=sensorReport, mode=Timer.PERIODIC)

# Main loop
while True:
    # Slanje stanja tastera
    if taster.value() != taster_state:
        taster_state = taster.value()
        buf = "{{\"Taster\": {}}}".format(taster_state)
        client.publish(mqtt_topic_taster, buf)

    # Slanje vrijednosti potenciometra
    pot_value = pot.read_u16()
    if pot_value != last_pot_value:
        last_pot_value = pot_value
        buf = "{{\"Potenciometar\": {}}}".format(pot_value)
        client.publish(mqtt_topic_pot, buf)

    # Provjera dolazećih poruka, na dolazeću poruku se poziva Dispatcher za callback
    client.check_msg()
    time.sleep(0.1)
```