Univerzitet u Sarajevu
Elektrotehnički fakultet
Ugradbeni sistemi 2023 / 24

Izvještaj za laboratorijsku vježbu br. 7

Komunikacija – primjer korištenja MQTT protokola

Ime i prezime: **Ivona Jozić**
Broj indeks-a: **19357**

29. april 2024.

Sadržaj

1 Pseudokod i / ili dijagram toka

U pseudokodovima za laboratorijsku vježbu 7, data je ideja kako pristupiti rješavanju zadataka, a kompletna implementacija, kako zbog svoje kompleksnosti, tako i duzine, data je kao prilog u odjeljku 5 ovog izvještaja.

S obzirom da su 1. i 2. zadatak konceptualno isti, samo se izvršavaju u različitim okruženjima, u nastavku će bit dat zajednički pseudokod za oba zadatka.

1.1 Zadatak 1 i zadatak 2

```
while(1)
	ocitaj_i_ispisi_stanje_tastera()
	ocitaj_i_ispisi_stanje_potenciometra()

	if(stigla poruka za led1) azuriraj_led1()

	if(stigla poruka za led2) azuriraj_led2()

	if(stigla poruka za led3) azuriraj_led3()

	end_if

	sacekaj(0.1s)

	provjeri_dolazak_poruke()

end_while
```

1.2 Zadatak 3

Razlika u odnosu na prva dva zadatka i 3. zadatak je to što su u njemu dodana još dva elementa, RGB LED dioda i DHT11 senzor za mjerenje temperature i vlažnosti zraka, pa je pseudokod neznatno nadograđen.

```
ocitaj_i_ispisi_stanje_senzora_na_2s()

while(1)
	ocitaj_i_ispisi_stanje_tastera()
	ocitaj_i_ispisi_stanje_potenciometra()

	if(stigla poruka za led1) azuriraj_led1()

	if(stigla poruka za led2) azuriraj_led2()

	if(stigla poruka za led3) azuriraj_led3()

	if(stigla poruka za RGB)  azuriraj_RGB()

	end_if

	sacekaj(0.1s)

	provjeri_dolazak_poruke()

end_while
```

2 Analiza programskog rješenja

2.1 Zadatak 1

U prvom zadatku je bilo potrebno upotrebom Mbed simulatora i korištenjem desktop aplikacije MQTTX i mobilne aplikacije MQTT Dash ostvariti komunikaciju. Prvo su u Mbed simulator dodani potenciometar (p15), taster (p5), led1 (p6), led2 (p7), te led3 (p21). U aplikacijama su kreirani korisnici i dodani odgovarajući elementi odnosno supskripcije. U priloženom kodu za ovaj zadatak je promijenjen string za ostvarivanje komunikacije kao i jedinstveno ime klijenta. Nakon toga je pokrenut simulator, te je konekcija uspostavljena. Nakon svake promjene stanja potenciometra dolazilo je do ispisa njegove vrijednosti, kao i prilikom promjene stanja tastera. Omogućeno je upravljanje LED1 i LED2 diodama kroz aplikaciju, tako što je u MQTTX-u biran topik i slane su vrijednosti, odnosno pritiskom na odgovarajuća polja u MQTT Dash-u. Također, pomoću aplikacija je postavljan duty cycle LED3 diode. Sve promjene su se mogle uočiti i na elementima u simulatoru.

2.2 Zadatak 2

Drugi zadatak je skoro pa potpuno identičan kao i 1., sa razlikom u tome što se ovaj put umjesto u simulatoru, rješenje testiralo na picoETF razvojnom sistemu. S obzirom da picoETF traži programiranje u MicroPhythonu, prvo je bilo potrebno prilagoditi kod iz C++ tako da se može pokrenuti i na picoETF-u. Nakon što je izvršena prilagodba koda, bilo je moguće koristiti već kreiranog klijenta iz prvog zadatka za testiranje rješenja. Za razliku od prvog zadatka, bilo je neophodno dodati dio za WiFi konekciju kako bi rješenje funkcioniralo u skladu sa očekivanjem. Također, potrebno je dodati i biblioteku *simple.py* u kojoj se nalaze sve potrebne metode za komunikaciju.

2.3 Zadatak 3

Treći zadatak je ustvari svojevrsna nadogradnja drugog zadatka, s obzirom da je bilo moguće iskoristiti kod u potpunosti, te samo dodati odgovarajuće callback metode i supskripcije za RGB LED diodu (po jednu za svaku od boja), te za DHT11 senzor. Nakon što se dodaju odgovarajuće supskripcije i callback metode u kod, potrebno je kreirati novog korisnika u MQTT aplikacijama sa svim potrebnim elementima (bilo je moguće i nadograditi postojećeg klijenta sa nedostajućim elementima). Nakon što se program pokrene na razvojnom sistemu picoETF moguće su sve radnje kao i u prethodnim zadacima, te dodatne vezane za RGB LED diodu i DHT11 senzor. Boju RGB LED diode je moguće podešavati tako što se kroz aplikacije podešava duty cycle za svaku od boja, te je na taj način omogućeno kreiranje svake proizvoljne boje na RGB LED diodi. Što se tiče DHT11 senzora, korištenjem objekta Ticker je omogućeno da se njegovo stanje očitava na svake dvije sekunde, te se ono ispisuje u okviru odgovarajuće supskripcije. Stanje DHT11 senzora je moguće direktno očitavati uključivanjem biblioteke DHT u program.

Kao i u prethodnom zadatku, korišteni su importi iz datoteke *simple.py*.

3 Korišteni hardverski resursi

Za potrebe laboratorijske vježbe 7 korišten je razvojni sistemi picoETF. Pored njega korištene su LED diode, RGB LED dioda i taster koji su integrisani na razvojni sistem, potenciometar, te DHT11 senzor.

Za sistem picoETF:

- 8 LED dioda

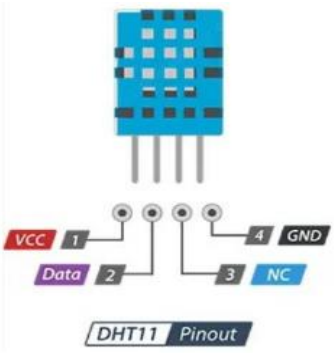
- RGB LED dioda

- taster

- rotacijski potenciometar

- DHT11 senzor za mjerenje temperature i vlažnosti zraka

ULAZI	IZLAZI
Rotacijski potenciometar (analogni; picoETF GP26/GP28)	LED1 (digitalni; picoETF GP4)
DHT11 (digitalni; picoETF GP28)	LED2 (digitalni; picoETF GP5)
Taster (digitalni; picoETF GP3)	LED3 (digitalni; picoETF GP6)
	RGB LED (digitalni; picoETF; Green - GP12, Blue - GP13, Red - GP14)



Slika 1. – Izlazni pinovi DHT11 senzora korištenog u 3. zadatku

4 Zaključak

Prilikom izvođenja Laboratorijske vježbe 7 nije bilo poteškoća s obzirom da su Mbed simulator, kao i razvojni sistema picoETF od ranije poznati. Dostupni su bili i primjeri implementacije koji su poslužili kao izvrsna vodilja. Također nije bilo problema sa povezivanjem aplikacija i uspostavljenjem konekcije. Cilj vježbe je bio upoznavanje sa MQTT protokolom, što je postignuto.

5 Prilog

U prilogu su dati kodovi koje se izvršavaju u Mbed simulatoru (1. zadatak) i na razvojnom okruženju picoETF (2. i 3. zadatak). Uz spomenuti simulator i razvojno okruženje, za testiranje zadataka neophodno je posjedovanje desktop aplikacije MQTTX, kao i mobilne aplikacije MQTT Dash koje je moguće preuzeti sa interneta. U aplikacijama je potrebno podesiti parametre konekcije, te izvršiti inicijalizaciju elemenata kao što je to obješnjeno u postavci ove laboratorijske vježbe.

Također, savjetuje se korištenje vlastitog stringa (umjesto *„i&v“*) u inicijalizaciji TEMASUB… za Mbed, odnosno mqtt_topic_... za picoETF, kako bi se osiguralo da samo korisnik koji pokreće sistem ima upravljačke ovlasti nad istim.

5.1 Zadatak 1 / izvorni kod

Za ispravno izvršenje sljedećeg koda neophodno je u Mbed simulatoru dodati odgovarajuće komponente, kako i inicijalizirati iste u MQTT aplikacijama.

```
#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;

// QoS 0
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

Za ispravno izvršenje sljedećeg koda na razvojnom sistemu picoETF neophodno je izvršiti inicijalizaciju potrebnih elemenata u MQTT aplikacijama, te dodati *simple.py* file.

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

# WiFi konfiguracija
wifi_ssid = "Lab220"
wifi_password = "lab220lozinka"

# MQTT konfiguracija
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 mreze
print("Connecting to WiFi: ", wifi_ssid)
wifi = network.WLAN(network.STA_IF)
wifi.active(True)
wifi.connect(wifi_ssid, wifi_password)
```

```
# cekanje na konekciju
while not wifi.isconnected():
    pass

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

# inicijalizacija 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)

# MQTT poruka zaprimljena, callback metode za LED
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))

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)

taster_state = taster.value()
last_pot_value = pot.read_u16()

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

# Subscribe na topic-e
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)


while True:
    # stanje tastera - ispis
    if taster.value() != taster_state:
        taster_state = taster.value()
        buf = "{{\"Taster\": {}}}".format(taster_state)
        client.publish(mqtt_topic_taster, buf)

    # stanje potenciomentra - ispis
    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 MQTT poruke
    client.check_msg()

    time.sleep(1)
```

5.3 Zadatak 3 / izvorni kod

Za ispravno izvršenje sljedećeg koda na razvojnom sistemu picoETF neophodno je izvršiti inicijalizaciju potrebnih elemenata u MQTT aplikacijama, te dodati *simple.py* file.

```
from machine import Pin, ADC, PWM, Timer
import network
import time
import simple
import dht #neophodno za ocitavanje DHT11 senzora

# WiFi konfiguracija
wifi_ssid = "Lab220"
wifi_password = "lab220lozinka"

# MQTT konfiguracija
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 = "zadatak3"

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

# cekanje na konekciju
while not wifi.isconnected():
    pass

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

# inicijalizacija pinova
taster = Pin(3, Pin.IN)
led1 = Pin(4, Pin.OUT)
led2 = Pin(5, Pin.OUT)
pot = ADC(Pin(26))
led3 = PWM(Pin(6))
led3.freq(1000)

rgbRed = PWM(Pin(14))
rgbGreen = PWM(Pin(12))
rgbBlue = PWM(Pin(13))
rgbRed.freq(1000)
rgbGreen.freq(1000)
rgbBlue.freq(1000)

sensor = dht.DHT11(Pin(28))

# MQTT poruka zaprimljena, callback metode za LED
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))

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)

def sensorReport(pin):
    sensor.measure()
    temperature = sensor.temperature()
    humidity = sensor.humidity()
    publish = str("Temp: " + str(temperature) + "\nHumidity: " + str(humidity))
    buf = "{{\"Senzor\": \n{}}}".format(publish)
    client.publish(mqtt_topic_sensor, buf)

taster_state = taster.value()
last_pot_value = pot.read_u16()

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

# Subscribe na topic-e
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)

# Ticker za ocitavanje senzora na svake 2s
t = Timer(period=2000, callback=sensorReport, mode=Timer.PERIODIC)

while True:
    # stanje tastera - ispis
    if taster.value() != taster_state:
        taster_state = taster.value()
        buf = "{{\"Taster\": {}}}".format(taster_state)
        client.publish(mqtt_topic_taster, buf)

    # stanje potenciometra - ispis
    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 postojanja MQTT poruke
    client.check_msg()

    time.sleep(0.1)
```