Podstawy Systemów Mikroprocesorowych Laboratorium

Laboratorium 8: Wi-Fi i MQTT — podstawy systemów IoT

Autorzy: Dariusz Tefelski, Angelika Tefelska

Spis treści

1 Wprowadzenie 1

2 Wyposażenie i przygotowanie stanowiska 1

3 Zadanie 1: Połączenie Raspberry Pi Pico 2 W z Wi-Fi 2

4 Zadanie 2: MQTT — model publish/subscribe 3

5 Zadanie 3: Przygotowanie Adafruit IO 4

6 Zadanie 4: Pierwsza publikacja MQTT 4

7 Zadanie 5: Dane z DS18B20 w chmurze 6

8 Zadanie 6: Odbieranie poleceń z MQTT 7

9 Zadanie końcowe: Miniwęzeł IoT 9

10 Podsumowanie 11

1 Wprowadzenie

Cel ćwiczenia

Celem ćwiczenia jest wykorzystanie interfejsu Wi-Fi Raspberry Pi Pico 2 W do komunikacji z usługą sieciową oraz poznanie protokołu MQTT. W czasie laboratorium mikrokontroler będzie zarówno wysyłał dane pomiarowe do chmury, jak i odbierał z niej polecenia sterujące.

W poprzednich laboratoriach komunikacja odbywała się głównie pomiędzy urządzeniami znajdującymi się na jednym stanowisku. Tym razem Raspberry Pi Pico 2 W zostanie podłączone do sieci Wi-Fi i będzie wymieniało dane z brokerem MQTT. W laboratorium wykorzystamy platformę **Adafruit IO**.

Czym jest IoT?

Internet Rzeczy (*Internet of Things, IoT*) obejmuje urządzenia wyposażone w czujniki, elementy wykonawcze oraz interfejs sieciowy, które mogą wymieniać dane z innymi urządzeniami lub usługami.
Typowy system IoT składa się z:

- urządzenia sterującego, np. Raspberry Pi Pico 2 W,
- sieci komunikacyjnej, np. Wi-Fi,
- protokołu wymiany danych, np. MQTT,
- serwera lub brokera,
- aplikacji prezentującej dane lub umożliwiającej sterowanie.

2 Wyposażenie i przygotowanie stanowiska

Do wykonania ćwiczenia potrzebne są:

- płytka dydaktyczna z Raspberry Pi Pico 2 W,
- czujnik DS18B20 z Laboratorium 7,
- jedna z diod LED na płytce,
- serwomechanizm S90 (do zadania rozszerzającego),
- komputer z dostępem do Internetu,
- dostęp do sieci Wi-Fi 2,4 GHz,
- konto Adafruit IO przygotowane przez studenta lub prowadzącego.

W ćwiczeniu wykorzystamy:

- DS18B20 na `GP14`,
- diodę LED na `GP15`,

- serwomechanizm na `GP16` (w zadaniu rozszerzającym).

3 Zadanie 1: Połączenie Raspberry Pi Pico 2 W z Wi-Fi

Raspberry Pi Pico 2 W może pracować w różnych trybach sieciowych. W tym laboratorium wykorzystamy tryb **station**, w którym mikrokontroler łączy się z istniejącym punktem dostępowym Wi-Fi.

Zadanie 1.1 — Plik z danymi dostępowymi

Utwórz w pamięci Raspberry Pi Pico plik:

`secrets.py`

o zawartości:

```
WIFI_SSID = 'nazwa_sieci'
WIFI_PASSWORD = 'haslo_do_sieci'

AIO_USERNAME = 'nazwa_uzytkownika'
AIO_KEY = 'klucz_Adafruit_IO'
```

W kolejnych programach będziemy importować te wartości zamiast wpisywać je bezpośrednio do głównego kodu.
Na razie uzupełnij dane związane z nazwą sieci i hasłem przekazane przez prowadzącego. Zmiennej AIO* zostaw na razie puste.

Dlaczego osobny plik `secrets.py`?

Oddzielenie danych uwierzytelniających od głównego programu:

- ułatwia ponowne wykorzystanie kodu,
- zmniejsza ryzyko przypadkowego udostępnienia hasła,
- pozwala zmieniać sieć lub konto bez modyfikowania programu.

W prawdziwym projekcie pliku zawierającego hasła i klucze nie powinno się umieszczać w publicznym repozytorium.

Zadanie 1.2 — Połączenie z siecią

Uruchom:

```
import network
from utime import sleep
from secrets import WIFI_SSID, WIFI_PASSWORD

wlan = network.WLAN(network.STA_IF)
wlan.active(True)

```

```
wlan.connect(WIFI_SSID, WIFI_PASSWORD)

while not wlan.isconnected():
    print('Laczenie z Wi-Fi...')
    sleep(1)

print('Polaczono')
print(wlan.ifconfig())
```

Po poprawnym połączeniu odczytaj konfigurację zwróconą przez `ifconfig()`. Zidentyfikuj:

- adres IP Raspberry Pi Pico,
- maskę podsieci,
- bramę domyślną,
- adres serwera DNS.

Adres IP

Adres IP identyfikuje urządzenie w sieci. W typowej sieci lokalnej adres jest przydzielany automatycznie przez serwer DHCP.

4 Zadanie 2: MQTT — model publish/subscribe

MQTT (*Message Queuing Telemetry Transport*) jest lekkim protokołem komunikacyjnym często używanym w systemach IoT.

W odróżnieniu od komunikacji punkt–punkt urządzenia nie muszą znać swoich adresów. Wymiana danych odbywa się przez **broker**.

Broker, topic i payload

Najważniejsze pojęcia MQTT:

- **broker** — serwer pośredniczący w wymianie wiadomości,
- **topic** — nazwa kanału, np. `user/feeds/temperature`,
- **publisher** — klient publikujący dane,
- **subscriber** — klient subskrybujący wybrany topic,
- **payload** — właściwa treść wiadomości.

Jedno urządzenie może być jednocześnie publisherem i subscriberem.



5 Zadanie 3: Przygotowanie Adafruit IO

Załóż konto na Adafruit IO (https://io.adafruit.com/) i utwórz trzy feed-y:

`temperature` `led` `servo`

Następnie utwórz dashboard zawierający:

- wykres dla feed-u `temperature`,
- przełącznik *Toggle* dla feed-u `led`,
- suwak *Slider* dla feed-u `servo` w zakresie 0–180.

Topic Adafruit IO

Dla feedu `temperature` topic MQTT ma postać:

`USERNAME/feeds/temperature`

a dla feedu `led`:

`USERNAME/feeds/led`

oraz dla feedu `servo`:

`USERNAME/feeds/servo`.

W miejsce `USERNAME` należy wstawić nazwę użytkownika Adafruit IO.

Zadanie 3.1 — Uzupełnienie danych Adafruit IO

Po utworzeniu konta wróć do pliku `secrets.py` przygotowanego w zadaniu 1.1. Uzupełnij:

```
AIO_USERNAME = 'nazwa_uzytkownika'
AIO_KEY = 'klucz_Adafruit_IO'
```

Nazwę użytkownika i klucz znajdziesz w ustawieniach konta Adafruit IO. Po zapisaniu pliku `secrets.py` dane dostępowe nie powinny być już wpisywane bezpośrednio w kolejnych programach.

Uwaga

Adafruit IO ogranicza liczbę publikowanych danych. Na kontach bezpłatnych nie należy wysyłać danych w każdej iteracji bardzo szybkiej pętli. W tym laboratorium pomiary będziemy publikować co około 10 s.

6 Zadanie 4: Pierwsza publikacja MQTT

W Thonny zainstaluj pakiet:

`umqtt.simple`

za pomocą narzędzia *Manage packages*.

Zadanie 4.1 — Połączenie z Wi-Fi i brokerem MQTT

Utwórz program:

```
import network
import machine
import ubinascii

from utime import sleep
from umqtt.simple import MQTTClient

from secrets import (WIFI_SSID, WIFI_PASSWORD,
    AIO_USERNAME, AIO_KEY)

# --- Wi-Fi ---

wlan = network.WLAN(network.STA_IF)
wlan.active(True)

wlan.connect(WIFI_SSID, WIFI_PASSWORD)

while not wlan.isconnected():
    print('Laczenie z Wi-Fi...')
    sleep(1)

print('Wi-Fi OK')
print(wlan.ifconfig())

# --- MQTT ---

client_id = ubinascii.hexlify(machine.unique_id())

client = MQTTClient(client_id, 'io.adafruit.com', user=
    AIO_USERNAME, password=AIO_KEY, keepalive=60)

client.connect()

print('Polaczono z brokerem MQTT')
```

Zwróć uwagę na kolejność: **najpierw Raspberry Pi Pico musi połączyć się z siecią Wi-Fi, a dopiero potem może nawiązać połączenie z brokerem MQTT**.

Po co `client_id`?

Broker musi rozróżniać poszczególnych klientów MQTT. Dlatego każdy klient powinien posiadać unikatowy identyfikator.

Funkcja:

`machine.unique_id()`

zwraca identyfikator związany z konkretnym mikrokontrolerem. `hexlify()` zamienia go na wygodny zapis szesnastkowy.

Zadanie 4.2 — Publikowanie danych

Dodaj:

```
topic_temp = (AIO_USERNAME + '/feeds/temperature')

temperature = 23.5

client.publish(topic_temp, str(temperature))

print('Wyslano:', temperature)
```

Sprawdź w Adafruit IO, czy wartość pojawiła się w feedzie `temperature`.

Topic i payload są przesyłane jako dane

MQTT nie wie, że wartość `23.5` oznacza temperaturę. Znaczenie danych wynika z:

- wybranego topicu,
- umowy przyjętej przez twórcę systemu.

Payload może zawierać tekst reprezentujący liczbę, napis, dane JSON lub dowolny ciąg bajtów.

7 Zadanie 5: Dane z DS18B20 w chmurze

Podłącz DS18B20 do `GP14` tak jak w Laboratorium 7.

Zadanie 5.1 — Publikowanie temperatury

Rozbuduj program tak, aby:

1. połączył się z Wi-Fi,
2. połączył się z brokerem MQTT,
3. skonfigurował czujnik DS18B20,
4. co około 10 s wykonywał pomiar temperatury,
5. publikował wynik do feedu `temperature`.

Pamiętaj o prawidłowej sekwencji:

```
ds.convert_temp()
sleep_ms(750)
temperature = ds.read_temp(rom)
```

Obserwuj wykres temperatury w dashboardzie. Ogrzej czujnik palcami i sprawdź, po jakim czasie zmiana pojawi się w chmurze.

System IoT ma opóźnienie

Wartość wyświetlana na dashboardzie nie pojawia się natychmiast po zmianie temperatury. Na całkowite opóźnienie wpływają m.in.:

- czas konwersji DS18B20,
- okres pomiędzy publikacjami,
- transmisja Wi-Fi,
- broker MQTT,
- odświeżanie dashboardu.

8 Zadanie 6: Odbieranie poleceń z MQTT

Teraz Raspberry Pi Pico będzie odbierało wiadomości z feedu `led`. W tym zadaniu wykorzystamy kompletny przykład, aby było widoczne, kiedy należy ustawić funkcję callback, wykonać subskrypcję i rozpocząć odbieranie wiadomości.

Zadanie 6.1 — Subskrypcja i sterowanie diodą

Uruchom program:

```
import network
import machine
import ubinascii

from machine import Pin
from utime import sleep
from umqtt.simple import MQTTClient

from secrets import (WIFI_SSID, WIFI_PASSWORD,
    AIO_USERNAME, AIO_KEY)

# --- Wi-Fi ---

wlan = network.WLAN(network.STA_IF)
wlan.active(True)

wlan.connect(WIFI_SSID, WIFI_PASSWORD)

while not wlan.isconnected():
    print('Laczenie z Wi-Fi...')
    sleep(1)

```

```
# --- LED ---

led = Pin(15, Pin.OUT, value=1)

# --- MQTT ---

client_id = ubinascii.hexlify(machine.unique_id())

client = MQTTClient(client_id, 'io.adafruit.com', user=
   AIO_USERNAME, password=AIO_KEY, keepalive=60)

topic_led = (AIO_USERNAME + '/feeds/led')

def mqtt_callback(topic, msg):
    print('Topic:', topic)
    print('Message:', msg)

    if msg == b'ON':
        led.value(0)

    elif msg == b'OFF':
        led.value(1)

client.set_callback(mqtt_callback)

client.connect()

client.subscribe(topic_led)

print('Oczekiwanie na polecenia MQTT...')

while True:
    client.check_msg()
    sleep(0.1)
```

Ustaw w widżecie Toggle:

- wartość ON: `ON`,
- wartość OFF: `OFF`.

Sprawdź, czy przełącznik na dashboardzie steruje diodą na stanowisku.

Zwróć uwagę na kolejność:

1. przygotowanie funkcji `mqtt_callback()`,
2. przypisanie jej za pomocą `set_callback()`,
3. połączenie z brokerem,
4. wykonanie `subscribe()`,
5. regularne wywoływanie `check_msg()` w pętli głównej.

Dlaczego porównujemy z `b'ON'`?

Biblioteka MQTT przekazuje odebraną wiadomość jako obiekt typu `bytes`, a nie `str`.
Dlatego:

```
msg == b'ON'
```

porównuje dwa obiekty tego samego typu.
Alternatywnie można użyć:

```
text = msg.decode()
```

i później porównywać `text == 'ON'`.

Co robi `check_msg()`?

Funkcja `check_msg()` sprawdza, czy broker przesłał nową wiadomość.
Jeżeli wiadomość jest dostępna, biblioteka wywołuje funkcję ustawioną przez:

```
set_callback(...).
```

Jeżeli wiadomości nie ma, program wykonuje dalsze instrukcje. Dzięki temu urządzenie może jednocześnie wykonywać pomiary i reagować na polecenia z sieci.

9 Zadanie końcowe: Miniwęzeł IoT

Zadanie 7.1 — Pomiar i zdalne sterowanie

Połącz wszystkie elementy laboratorium w jeden program.
System powinien:

1. połączyć Raspberry Pi Pico 2 W z Wi-Fi,
2. połączyć klienta z brokerem MQTT,
3. subskrybować feed `led`,
4. co około 10 s mierzyć temperaturę z DS18B20,
5. publikować temperaturę do feedu `temperature`,
6. pomiędzy pomiarami regularnie wykonywać `client.check_msg()`,
7. sterować diodą zgodnie z poleceniami z dashboardu,
8. wypisywać w konsoli informacje o wysłanych i odebranych wiadomościach.

Przetestuj jednocześnie:

- zmianę temperatury widoczną na wykresie,
- sterowanie diodą z poziomu przeglądarki.

Zadanie 7.2 — zdalne sterowanie serwomechanizmem

Podłącz serwomechanizm:

- sygnał sterujący do `GP16`,
- zasilanie do 5 V,
- GND do wspólnej masy układu.

Uwaga



Nie zasilaj serwomechanizmu z wyjścia 3,3 V Raspberry Pi Pico. Serwo powinno być zasilane z linii 5 V stanowiska, a jego masa musi być połączona z masą mikrokontrolera.

Na dashboardzie stwórz suwak przypisany do feedu `servo`. Ustaw zakres 0–180. Przypomnij sobie Laboratorium 3. Serwomechanizm sterujemy sygnałem PWM o częstotliwości około 50 Hz. Przyjmij:

- 0° — impuls około 1000 μs,
- 90° — impuls około 1500 μs,
- 180° — impuls około 2000 μs.

Możesz wykorzystać funkcję:

```
from machine import PWM

servo = PWM(Pin(16))
servo.freq(50)

def set_servo_angle(angle):
    angle = max(0, min(180, angle))

    pulse_us = 1000 + angle * 1000 // 180

    duty = int(
        pulse_us * 65535 / 20000
    )

    servo.duty_u16(duty)
```

Utwórz topic:

```
topic_servo = (
    AIO_USERNAME + '/feeds/servo'
)
```

i zasubskrybuj go:

```
client.subscribe(topic_servo)
```

Rozbuduj funkcję callback tak, aby rozpoznawała, z którego topicu przyszła wiadomość:

```
def mqtt_callback(topic, msg):
    if topic == topic_led.encode():
        if msg == b'ON':
            led.value(0)

```

```
        elif msg == b'OFF':
            led.value(1)

    elif topic == topic_servo.encode():
        angle = int(msg.decode())
        set_servo_angle(angle)
```

Przetestuj:

1. sterowanie diodą za pomocą przełącznika,
2. sterowanie położeniem serwa za pomocą suwaka,
3. jednoczesne publikowanie temperatury.

Zwróć uwagę, że w tym samym programie Raspberry Pi Pico jest jednocześnie:

- publisherem temperatury,
- subscriberem polecenia ON/OFF,
- subscriberem wartości liczbowej określającej położenie serwa.

10 Podsumowanie

Po tych zajęciach powinieneś umieć

- połączyć Raspberry Pi Pico 2 W z siecią Wi-Fi,
- odczytać konfigurację IP urządzenia,
- wyjaśnić pojęcia broker, topic, publisher, subscriber i payload,
- połączyć klienta MicroPython z brokerem MQTT,
- publikować dane pomiarowe,
- subskrybować topic i obsługiwać odebrane wiadomości,
- wykorzystać callback do sterowania elementem wykonawczym,
- odebrać przez MQTT wartość liczbową i wykorzystać ją do sterowania serwomechanizmem,
- zbudować prosty system IoT z komunikacją dwukierunkową.

Uwaga

Przed opuszczeniem stanowiska usuń z pamięci Raspberry Pi Pico plik `secrets.py` zawierający hasło i klucz Adafruit IO, jeżeli używane konto lub sieć nie są przeznaczone wyłącznie do ćwiczeń. Usuń także pozostałe pliki utworzone podczas laboratorium.