Podstawy Systemów Wbudowanych Laboratorium

Laboratorium 5: Wi-Fi i MQTT — pierwszy węzeł IoT

Autorzy: Angelika Tefelska, Dariusz Tefelski

Spis treści

1 Wprowadzenie **1**

2 Wyposażenie stanowiska **1**

3 Zadanie 1: Połączenie Arduino z siecią Wi-Fi **2**
3.1 SSID, adres IP i RSSI 2
3.2 Dane dostępowe poza głównym programem 2

4 Zadanie 2: MQTT i Adafruit IO **3**
4.1 Broker, publisher, subscriber i feed 3

5 Zadanie 3: LM35 i publikowanie temperatury **4**
5.1 Pomiar temperatury czujnikiem LM35 4
5.2 Publikowanie danych do Adafruit IO 5

6 Zadanie 4: Sterowanie z Adafruit IO **6**

7 Zadanie końcowe: Dwukierunkowy węzeł IoT **7**

8 Podsumowanie **8**

1 Wprowadzenie

Cel ćwiczenia

Celem ćwiczenia jest wykorzystanie interfejsu Wi-Fi wbudowanego w Arduino UNO R4 WiFi oraz poznanie podstaw protokołu MQTT na przykładzie platformy Adafruit IO. W pierwszej części połączymy mikrokontroler z siecią bezprzewodową. Następnie Arduino będzie publikowało zmierzoną temperaturę do chmury oraz odbierało polecenia sterujące diodą LED. Na końcu połączymy oba kierunki komunikacji w jeden prosty węzeł IoT.

Arduino UNO R4 WiFi zawiera moduł ESP32-S3 realizujący komunikację bezprzewodową. W programie Arduino korzystamy z niego przez bibliotekę `WiFiS3`. Dzięki temu mikrokontroler może komunikować się nie tylko z elementami podłączonymi przewodami, lecz również z usługami działającymi w sieci.

W tym laboratorium wykorzystamy MQTT — lekki protokół często stosowany w systemach Internetu Rzeczy (IoT, *Internet of Things*). Jako broker oraz interfejs do przechowywania i prezentacji danych wykorzystamy platformę Adafruit IO.

Od czujnika do aplikacji

Typowy system IoT można uprościć do następującego łańcucha:

czujnik / element wykonawczy ↔ mikrokontroler ↔ Wi-Fi ↔ broker MQTT ↔ aplikacja użytkownika.

W tym ćwiczeniu Adafruit IO pełni jednocześnie rolę usługi MQTT, magazynu danych oraz prostego panelu użytkownika.

2 Wyposażenie stanowiska

Do wykonania ćwiczenia potrzebne są:

- Arduino UNO R4 WiFi,
- przewód USB,
- czujnik temperatury LM35,
- zewnętrzna dioda LED z rezystorem 330 Ω,
- płytka stykowa i przewody połączeniowe,
- komputer z Arduino IDE i dostępem do przeglądarki internetowej,
- konto Adafruit IO,
- dane sieci Wi-Fi wskazanej przez prowadzącego.

Co należy zapamiętać?



Klucz Adafruit IO (AIO Key) pełni rolę hasła dostępowego do Twojego konta i feedów. Nie umieszczaj go w sprawozdaniu, zrzutach ekranu ani kodzie publikowanym publicznie.

3 Zadanie 1: Połączenie Arduino z siecią Wi-Fi

3.1 SSID, adres IP i RSSI

Sieć Wi-Fi jest identyfikowana m.in. przez nazwę SSID. Po poprawnym połączeniu urządzenie otrzymuje adres IP, który pozwala innym elementom sieci kierować do niego pakiety.

Do oceny jakości połączenia można wykorzystać RSSI (*Received Signal Strength Indicator*). Wartość RSSI podawana jest zwykle w dBm i jest liczbą ujemną. Wartość bliższa zeru oznacza silniejszy odebrany sygnał, np. $-45\,\text{dBm}$ oznacza zwykle silniejszy sygnał niż $-80\,\text{dBm}$.

Zadanie 1.1 — Skanowanie dostępnych sieci



W Arduino IDE wybierz *Plik* → *Przykłady* → *WiFiS3* i odszukaj przykład skanujący dostępne sieci (`ScanNetworks`).
Uruchom przykład i przeanalizuj wynik w *Serial Monitor*. Zanotuj dla sieci wskazanej przez prowadzącego:

- jej SSID,
- RSSI,
- numer kanału, jeżeli przykład go wyświetla.

Porównaj RSSI z co najmniej jedną inną wykrytą siecią. Który sygnał jest silniejszy?

3.2 Dane dostępowe poza głównym programem

Nazwy sieci i hasła nie należy wpisywać wielokrotnie w kodzie ani publikować razem z programem. W przykładach Arduino dane dostępowe są często umieszczane w osobnym pliku `arduino_secrets.h`.

Utwórz obok głównego pliku szkicu plik `arduino_secrets.h` o zawartości:

```
#define SECRET_SSID "nazwa_sieci"
#define SECRET_PASS "haslo_do_sieci"
```

Wartości zastąp danymi przekazanymi przez prowadzącego.

Co należy zapamiętać?

Nie umieszczaj prawdziwego hasła do sieci w sprawozdaniu, zrzutach ekranu ani plikach udostępnianych publicznie.

Zadanie 1.2 — Połączenie z siecią Wi-Fi

Otwórz przykład `ConnectWithWPA` z biblioteki `WiFiS3`. Przeanalizuj sposób:

- dołączenia pliku `arduino_secrets.h`,
- nawiązywania połączenia przez `WiFi.begin()`,
- sprawdzania stanu połączenia,
- odczytu adresu IP,
- odczytu wartości RSSI.

Na podstawie przykładu przygotuj własny krótki program, który łączy się z siecią wskazaną przez prowadzącego, a po poprawnym połączeniu wyświetla w *Serial Monitor*:

- napis `Polaczono z Wi-Fi`,
- SSID,
- adres IP Arduino,
- RSSI w dBm.

4 Zadanie 2: MQTT i Adafruit IO

4.1 Broker, publisher, subscriber i feed

MQTT wykorzystuje model *publish/subscribe*. Klient może publikować wiadomości oraz subskrybować wybrane tematy. Centralnym elementem jest broker, który odbiera wiadomości od publisherów i przekazuje je subscriberom.

Broker	serwer pośredniczący w wymianie wiadomości MQTT
Publisher	klient publikujący wiadomość
Subscriber	klient subskrybujący wiadomości
Topic	tekstowa nazwa kanału MQTT
Payload	właściwa treść wiadomości, np. `23.6` lub `1`
Feed	strumień danych w Adafruit IO, odpowiadający określonemu tematowi MQTT

Adafruit IO udostępnia broker MQTT pod adresem `io.adafruit.com`. Do uwierzytelnienia używane są:

- nazwa użytkownika Adafruit IO,
- klucz AIO Key jako hasło.

Dla feedu o kluczu `temperature` odpowiadający mu temat MQTT ma postać:

`<nazwa_uzytkownika>/feeds/temperature`

Biblioteka Adafruit IO Arduino ukrywa szczegóły tworzenia tego topicu. W programie wystarczy odwoływać się do feedu po jego kluczu.

Zadanie 2.1 — Utworzenie feedu i pierwszego punktu danych

Zaloguj się do platformy:

https://io.adafruit.com

Jeżeli nie masz konta, utwórz je zgodnie ze wskazówkami prowadzącego. Następnie:

1. przejdź do zakładki *Feeds*,
2. wybierz *New Feed*,
3. utwórz feed o nazwie `temperature`,
4. otwórz utworzony feed i sprawdź jego klucz,
5. wybierz *Add Data* i wpisz przykładową wartość, np. `22.5`,
6. sprawdź, czy punkt pojawił się na wykresie wraz ze znacznikiem czasu.

Ręczne dodanie wartości służy jedynie do sprawdzenia feedu. W dalszej części wartości będą publikowane automatycznie przez Arduino z wykorzystaniem MQTT.

Zadanie 2.2 — Dashboard temperatury

Przejdź do zakładki *Dashboards* i utwórz nowy dashboard, np. `PSW Lab 5`. Dodaj blok prezentujący dane z feedu `temperature` w postaci wykresu liniowego. Ustaw czytelny tytuł i jednostkę °C. Sprawdź, czy ręcznie dodany wcześniej punkt danych jest widoczny na dashboardzie.

Feed a MQTT

Adafruit IO przechowuje dane w feedach, ale komunikacja urządzenia z usługą może odbywać się przez MQTT. Publikacja wartości do feedu `temperature` oznacza w praktyce wysłanie wiadomości MQTT do topicu powiązanego z tym feedem.

5 Zadanie 3: LM35 i publikowanie temperatury

5.1 Pomiar temperatury czujnikiem LM35

LM35 jest analogowym czujnikiem temperatury. Jego napięcie wyjściowe zmienia się w przybliżeniu o 10 mV na każdy 1 °C. Oznacza to, że napięcie 250 mV odpowiada w przybliżeniu temperaturze 25 °C.

Zadanie 3.1 — Lokalny pomiar temperatury

Podłącz LM35:

LM35	Arduino UNO R4 WiFi
VCC	5V
GND	GND
VOUT	A0

Przy domyślnej 10-bitowej rozdzielczości ADC możesz obliczyć napięcie i temperaturę np. tak:

```
int raw = analogRead(A0);
float voltage = raw * 5.0 / 1023.0;
float temperatureC = voltage * 100.0;
```

Napisz krótki program wyświetlający w *Serial Monitor* surową wartość ADC, napięcie oraz temperaturę w °C.
Ogrzej czujnik palcami i sprawdź, czy wskazanie temperatury rośnie. Po odsunięciu dłoni zaobserwuj powrót temperatury w kierunku wartości początkowej.

To nie jest pomiar laboratoryjny

W ćwiczeniu traktujemy LM35 jako proste fizyczne źródło danych dla systemu IoT. Dokładność wyniku zależy m.in. od napięcia odniesienia ADC i nie jest głównym celem tego laboratorium.

5.2 Publikowanie danych do Adafruit IO

Do komunikacji z Adafruit IO wykorzystamy bibliotekę `Adafruit IO Arduino`. Biblioteka obsługuje połączenie z brokerem MQTT i pozwala publikować dane za pomocą obiektów reprezentujących feedy.

Zadanie 3.2 — Publikacja temperatury przez MQTT

W *Library Manager* zainstaluj bibliotekę `Adafruit IO Arduino`. Następnie wybierz *Plik* → *Przykłady* i odszukaj przykład `adafruitio_00_publish`.
Przeanalizuj:

- plik `config.h`,
- utworzenie obiektu feedu funkcją `io.feed()`,
- połączenie z usługą przez `io.connect()`,
- wywołanie `io.run()` w funkcji `loop()`,
- publikację wartości za pomocą metody `save()`.

W pliku z danymi poufnymi dopisz także dane Adafruit IO, np.:

```
#define SECRET_IO_USERNAME "twoja_nazwa_uzytkownika"
#define SECRET_IO_KEY      "twoj_AIO_Key"
```

Na ich podstawie skonfiguruj `config.h`. Główny program powinien zawierać

obiekt feedu:

```
AdafruitIO_Feed *temperature = io.feed("temperature");
```

Po połączeniu z Adafruit IO publikuj zmierzoną wartość temperatury, np.:

```
temperature->save(temperatureC);
```

Napisz program, który:

1. łączy się z Adafruit IO,
2. regularnie wykonuje `io.run()`,
3. co około 5 s mierzy temperaturę LM35,
4. wypisuje temperaturę w *Serial Monitor*,
5. publikuje tę samą wartość do feedu `temperature`.

Do odmierzania czasu wykorzystaj `millis()`, a nie długie `delay()`. Otwórz dashboard Adafruit IO i sprawdź, czy kolejne próbki pojawiają się na wykresie. Ogrzej czujnik dłonią i obserwuj zmianę wykresu.

Co należy zapamiętać?

Nie publikuj danych w każdej iteracji funkcji `loop()`. Usługi chmurowe ograniczają częstość przesyłania danych. Dla temperatury odstęp kilku sekund jest w zupełności wystarczający.

6 Zadanie 4: Sterowanie z Adafruit IO

MQTT działa w obu kierunkach. Arduino może nie tylko publikować dane, lecz również subskrybować feed i reagować na nowe wartości.

Zadanie 4.1 — Feed sterujący i przycisk na dashboardzie

W Adafruit IO utwórz drugi feed o nazwie `led`. Następnie na dashboardzie dodaj blok typu *Toggle* powiązany z tym feedem.
Ustaw wartości:

- wyłączone: `0`,
- włączone: `1`.

Przełącz kilka razy przycisk na dashboardzie i sprawdź na stronie feedu `led`, że pojawiają się kolejne wartości 0 i 1.

Zadanie 4.2 — Zdalne sterowanie diodą

Otwórz przykład `adafruitio_01_subscribe` z biblioteki `Adafruit IO Arduino`. Przeanalizuj:

- utworzenie obiektu feedu,
- funkcję `onMessage()`,



- funkcję obsługującą odebraną wiadomość,
- wywołanie `get()` po połączeniu,
- rolę `io.run()` w głównej pętli programu.

Podłącz zewnętrzną diodę LED do pinu `D8`. Anodę diody połącz z pinem `D8` przez rezystor 470 Ω, a katodę z masą `GND`. Następnie zdefiniuj pin i feed w programie:

```
const int LED_PIN = 8;
AdafruitIO_Feed *led = io.feed("led");
```

W funkcji `setup()` skonfiguruj pin jako wyjście:

```
pinMode(LED_PIN, OUTPUT);
digitalWrite(LED_PIN, LOW);
```

W funkcji obsługującej wiadomość zamień odebraną wartość na liczbę i jawnie sprawdź jej wartość za pomocą instrukcji `if`:

```
void handleMessage(AdafruitIO_Data *data) {
    int state = data->toInt();

    if (state == 1) {
        digitalWrite(LED_PIN, HIGH);
    }
    else {
        digitalWrite(LED_PIN, LOW);
    }
}
```

Napisz program, który łączy się z Adafruit IO, subskrybuje feed `led` i steruje zewnętrzną diodą podłączoną do pinu `D8` za pomocą przycisku *Toggle* na dashboardzie.
Sprawdź również, czy po ponownym uruchomieniu Arduino metoda `led->get()` pozwala odtworzyć ostatni stan feedu.

7 Zadanie końcowe: Dwukierunkowy węzeł IoT

W ostatnim zadaniu połączymy publikowanie pomiaru i zdalne sterowanie w jednym programie.

Zadanie 5.1 — Temperatura i zdalne sterowanie



Przygotuj jeden program spełniający następującą specyfikację:

1. Arduino łączy się z Wi-Fi i Adafruit IO,
2. co około 5 s mierzy temperaturę LM35 i publikuje ją do feedu `temperature`,



3. przez cały czas odbiera zmiany feedu `led`,
4. wartość `1` włącza zewnętrzną diodę na pinie `D8`, a `0` ją wyłącza,
5. w każdej iteracji `loop()` regularnie wykonywane jest `io.run()`,
6. do odmierzania czasu publikacji zastosowane jest `millis()`, bez blokującego `delay()` w normalnej pracy programu.

Na dashboardzie powinny znajdować się co najmniej:

- wykres temperatury,
- przycisk *Toggle* sterujący zewnętrzną diodą LED.

Sprawdź, czy możliwe jest jednoczesne obserwowanie temperatury i sterowanie diodą bez zatrzymywania publikacji pomiarów.

Co dzieje się pod spodem?



W kodzie korzystamy z obiektów `AdafruitIO_Feed`, ale komunikacja z Adafruit IO odbywa się przez MQTT. Arduino jest równocześnie publisherem feedu temperatury oraz subscriberem feedu sterującego diodą. Broker pośredniczy między urządzeniem a dashboardem działającym w przeglądarce.

8 Podsumowanie

Po tych zajęciach powinieneś umieć



- wykorzystać bibliotekę `WiFiS3` do połączenia UNO R4 WiFi z siecią,
- odczytać adres IP i RSSI,
- wyjaśnić role brokera, publishera, subscribera, topicu i payloadu w MQTT,
- utworzyć feed i dashboard w Adafruit IO,
- wyjaśnić związek pomiędzy feedem Adafruit IO i topicem MQTT,
- zmierzyć temperaturę czujnikiem LM35 i przesłać ją do Adafruit IO,
- opublikować dane za pomocą metody `save()`,
- odebrać wiadomość z feedu i wykorzystać ją do sterowania wyjściem,
- użyć `io.run()` i `millis()` w programie wykonującym równocześnie publikację i odbiór danych.

Co należy zapamiętać?



Przed opuszczeniem stanowiska usuń prawdziwe hasła i klucze z plików, które zamierzasz udostępniać innym osobom. Odłącz Arduino i uporządkuj elementy stanowiska.