

8. WYKORZYSTANIE PROTOKOŁU MQTT

Cele ćwiczenia:

- ♦ zapoznanie się z właściwościami protokołu MQTT,
- ♦ opanowanie umiejętności wykorzystania protokołu MQTT w aplikacjach do przesyłania wiadomości,
- ♦ opanowanie umiejętności wykorzystania protokołu MQTT do transmisji danych w systemach telemetrycznych.

Protokół MQTT

Message Queuing Telemetry Transport (MQTT) to prosty i lekki protokół transmisji wiadomości i danych; pierwsza wersja powstała w 1999 roku. Protokół został opracowany w firmach IBM oraz Circus Lin, obecnie jest rozwijany przez społeczność OpenSource (<http://mqtt.org/>). Najnowsza wersja, 5.0, została opublikowana w 2019 roku.

Właściwości protokołu:

- ♦ prosty i lekki protokół warstwy aplikacji,
- ♦ oparty na wzorcu publikacja/subskrypcja,
- ♦ przeznaczony do transmisji dla urządzeń niewymagających dużej przepustowości,
- ♦ poprzez ograniczenie prędkości transmisji zapewnia większą niezawodność niż przy prędkości wyższej,
- ♦ doskonale sprawdza się przy połączeniach maszyna-maszyna (M2M), w Internecie rzeczy, w urządzeniach mobilnych oraz tam, gdzie wymagana jest oszczędność przepustowości oraz energii.

W tabeli 8.1. przedstawiono porównanie protokołów MQTT z HTTP.

Tabela 8.1.
Porównanie protokołów HTTP i MQTT [12]

	MQTT	HTTP
<i>Przesyłana treść</i>	Dane lub krótkie wiadomości	Głównie dokumenty, często bardzo duże
<i>Wzorzec komunikacji</i>	Publikacja/Subskrypcja	Żądanie/Odpowiedź
<i>Złożoność</i>	Mała	Większa
<i>Rozmiar wiadomości</i>	Mały – jeden mały nagłówek	Duży, kilka nagłówków
<i>Jakość usługi</i>	Trzy poziomy	Jeden poziom
<i>Dystrybucja wiadomości</i>	Jeden do wielu(1:0,1,n): jedna publikacja jest rozsyłana do wielu subskrybentów albo do żadnego, jeśli żaden nie subskrybuje tematu	Jeden do jednego
<i>Biblioteki programistyczne (klienckie)</i>	Małe	Duże

Łączność z wykorzystaniem protokołu MQTT wymaga Brokera MQTT.

Broker MQTT

Broker MQTT pełni funkcję serwera, z którym łączą się klienci. Klienci mogą występować w dwóch rolach: publikatorów i subskrybentów. Publikatory mogą publikować wiadomości na jakiś temat, a subskrybenci mogą subskrybować tematy. Przesyłanie wiadomości odbywa się za pośrednictwem brokera.

Broker musi być widziany przez komputery klientów. Klienci nie muszą widzieć się wzajemnie. Częstym rozwiązaniem jest broker z publicznym adresem IP i klienci z dostępem do Internetu, bez konieczności posiadania publicznego IP. Klientami mogą być komputery w sieciach lokalnych i urządzenia przenośne. Publikator może przekazać wiadomość subskrybentowi bez znajomości jego adresu IP i wiedzy o jego istnieniu. Broker odbiera wiadomości od klientów publikujących, a następnie rozsyła je do klientów subskrybujących dany temat.

Przykładem zastosowania protokołu MQTT jest Facebook Messenger, za którego pomocą można przysyłać wiadomości między urządzeniami przenośnymi nieposiadającymi publicznych adresów IP.

Funkcje zarządzania połączeniami z brokerem MQTT

- ◆ Connect – ustanawia połączenie z brokerem,
- ◆ Subscribe – subskrybuje (nasłuchuje) zadany temat na brokerze,
- ◆ Publish – publikuje (wysyła) informację na podany temat, poprzez brokera, do wszystkich klientów subskrybujących dany temat,
- ◆ Unsubscribe – przestaje subskrybować dany temat,
- ◆ Disconnect – zamyka połączenie z brokerem.

Powyższe nazwy funkcji pojawiają się w nazwach metod bibliotek programistycznych realizujących funkcje klienta MQTT.

Tematy – Topics

- ◆ Publikatory publikują wiadomości na dany temat, subskrybenci subskrybują wybrane tematy.
- ◆ Tematy nie muszą być wcześniej tworzone i mogą mieć dowolną nazwę.
- ◆ Tematy mają wielopoziomową strukturę hierarchiczną.
- ◆ / (slash) – rozdziela poziomy.
- ◆ # (hash character) – multi level wildcard.
- ◆ + (plus character) – single level wildcard; przykład: house/+ / main-light.
- ◆ \$SYS/broker/# – komunikaty brokera.
- ◆ Symbole + i # można stosować, subskrybując tematy, nie można ich stosować, publikując.

Jakość usługi – Quality of Service – QoS

Każde połączenie z brokerem musi mieć określoną jakość usługi QoS. Jakość usługi w protokole MQTT jest to umowa pomiędzy nadawcą (publikatorem) a odbiorcą (subskrybentem) dotycząca gwarancji dostarczenia wiadomości. Gdy klient wysyła / publikuje wiadomość, określa poziom QoS tej wiadomości. Gdy broker wysyła daną wiadomość do subskrybentów, obowiązuje QoS określony przez klienta, który opublikował wiadomość.

Zdefiniowane są trzy poziomy jakości usługi, określające, czy i ile razy wiadomość musi zostać dostarczona do subskrybenta:

♦ $QoS = 0$ – *At most once* (najwyżej raz) – wysłana wiadomość nie potrzebuje potwierdzenia dostarczenia – „Wyślij i zapomnij”. Jest to najprostsza metoda wysyłania wiadomości. Klient publikuje wiadomość na dany temat, ale nie otrzymuje potwierdzenia jej otrzymania przez brokera. Jest to odpowiedni wybór w przypadku niezawodnego połączenia klienta z brokerem. $QoS = 0$ gwarantuje najniższy koszt pod względem transferu danych.

♦ $QoS = 1$ – *At least once* (przynajmniej raz) – wysłana wiadomość musi zostać dostarczona przynajmniej raz, potwierdzenie jest wymagane. Wiadomość może zostać dostarczona do wielu odbiorców (może być odebrana kilkakrotnie, jest wysyłana do subskrybenta, dopóki broker nie otrzyma potwierdzenia odbioru). Można stosować, gdy wiadomość jest idempotentna.

♦ $QoS = 2$ – *Exactly once* (dokładnie raz) – wysłana wiadomość musi zostać dostarczona odbiorcy dokładnie jeden raz. Jest to najwyższy poziom QoS , w którym jest wykorzystywana sekwencja czterech komunikatów pomiędzy nadawcą i odbiorcą. $QoS = 2$ gwarantuje dostarczenie wiadomości dokładnie raz, ale z najwyższym kosztem transferu danych. Należy stosować, gdy wiadomość nie jest idempotentna.

Opcja Retain

♦ Domyślnie `Retain = false`: broker po przesłaniu opublikowanej wiadomości do wszystkich subskrybentów usuwa ją z pamięci, nowi subskrybenci jej nie dostaną.

♦ Jeśli `Retain = true`: broker zachowuje ostatnią wiadomość w temacie i dostarcza ją do nowych subskrybentów; przydatne do przekazania stanu systemu nowym subskrybentom.

Kiedy jakiś czujnik publikuje swój stan tylko wtedy, gdy ten stan się zmienia, opcja `Retain = true` powinna być ustawiona. Na przykład, jeśli czujnik zamkniętych drzwi publikuje stan tylko wtedy, gdy drzwi są zamykane bądź otwierane, to przy `Retain = false` nowy subskrybent może długo czekać, zanim się dowie, czy drzwi są otwarte, czy zamknięte.

Keep Alive

MQTT korzysta z połączenia TCP/IP. Połączenie jest zwykle otwarte, podobnie jak połączenie telefoniczne, aby klient mógł wysyłać i odbierać dane w dowolnym momencie. Jeśli żadne dane nie będą przesyłane przez otwarte połączenie przez określony czas, połączenie jest zamykane przez serwer. Aby podtrzymać połączenie, klient wysyła pakiet `PINGREQ` i oczekuje, że otrzyma pakiet `PINGRESP` od brokera. Ta wymiana wiadomości potwierdza, że połączenie jest otwarte i działa. Dopuszczalny czas przerwy między pakietami jest określany jako czas utrzymywania połączenia przy życiu `Keep Alive`.

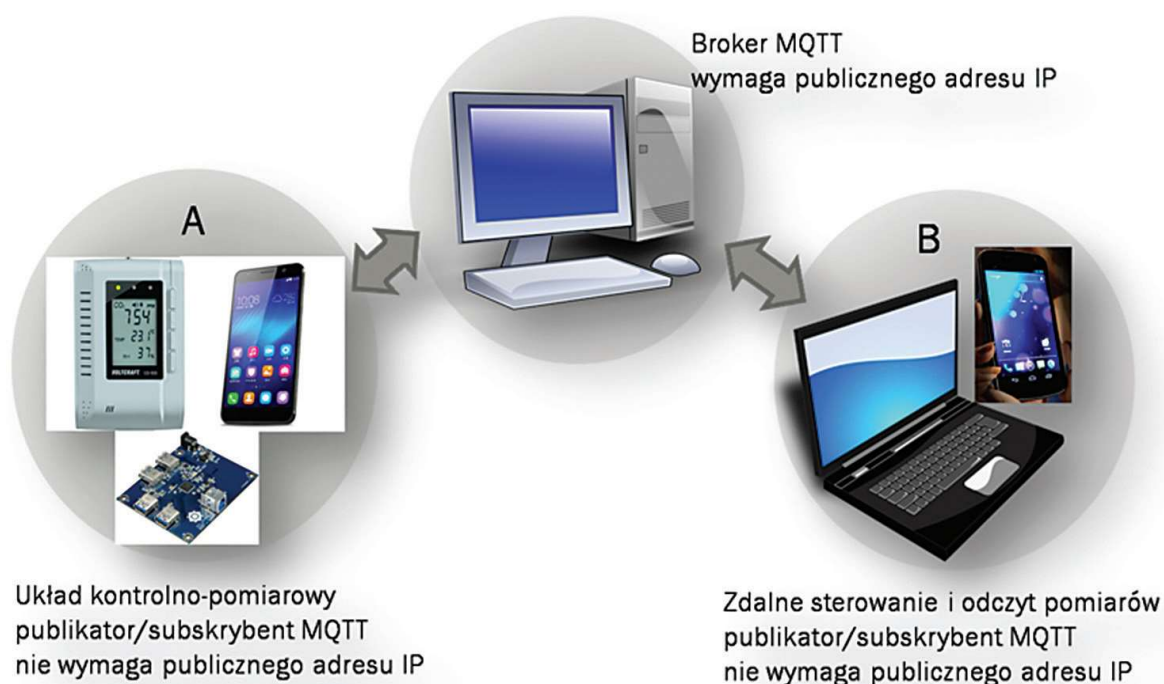
`Keep Alive` to przedział czasu mierzony w sekundach. Wyrażony jest przez 16-bitowe słowo, jest to maksymalny odstęp czasu, który może upłynąć między momentem, w którym klient kończy transmisję jednego pakietu kontrolnego, a momentem, w którym rozpoczyna wysyłanie następnego. Klient jest odpowiedzialny za upewnienie się, że odstęp czasu między wysyłanymi pakietami kontrolnymi nie przekracza wartości `Keep Alive`. W przypadku braku wysyłania jakichkolwiek innych pakietów kontrolnych klient MUSI wysłać pakiet `PINGREQ`. Może go w dowolnym momencie,

niezależnie od wartości Keep Alive, i użyć PINGRESP do ustalenia, czy sieć i serwer działają [12], [19].

System telemetryczny z wykorzystaniem protokołu MQTT

Oprócz komunikatorów takich jak Facebook Messenger istotnym zastosowaniem protokołu MQTT jest budowa systemów telemetrycznych (telemetria jest w nazwie protokołu). Elementy składowe systemu, np. przyrządy pomiarowe, mogą publikować wyniki pomiarów, urządzenia wykonawcze – subskrybować komendy sterujące, pulpit sterujący na zdalnej maszynie może subskrybować wyniki pomiarów i stan urządzeń oraz publikować komendy sterujące. Taki system może komunikować się przez Internet, ale tylko broker musi mieć publiczny adres IP. Pozostałe elementy muszą mieć dostęp do Internetu, np. za pomocą telefonii komórkowej, mogą też znajdować się w sieciach lokalnych.

Na rys. 8.1. przedstawiono prosty system telemetryczny wykorzystujący protokół MQTT do transmisji danych i wiadomości.



Rys. 8.1. Przykładowy system telemetryczny wykorzystujący protokół MQTT do transmisji danych

Broker, za którego pośrednictwem uczestnicy systemu wymieniają dane, ma publiczny adres IP, pozostałe maszyny muszą mieć dostęp do Internetu, ale nie muszą mieć publicznego IP. Przykładem układu kontrolno-pomiarowego A może być Raspberry Pi, do którego są podłączone czujniki pomiarowe i przekaźniki do sterowania urządzeniami wykonawczymi. Układ publikuje dane pomiarowe, które mogą subskrybować zainteresowani odbiorcy, oraz subskrybuje komendy sterujące publikowane przez układ zdalnego sterowania B. Pulpit zdalnego sterowania może być realizowany przez aplikację działającą na dowolnej maszynie z dostępem do Internetu.

Realizacja systemu telemetrycznego w środowisku Node-RED

W wersji testowej system może być projektowany na pojedynczej maszynie, a po przetestowaniu poszczególne elementy mogą być przeniesione na różne maszyny. Do realizacji systemu można użyć środowiska Node-RED.

Broker MQTT

♦ Pierwszym brokerem może być bloczek *Aedes MQTT broker* (na palecie ma nazwę *aedes broker*) umieszczony na dowolnym grafie Node-RED; nie wymaga konfiguracji. Aby posłużyć się bloczkiem, należy zainstalować pakiet *node-red-contrib-aedes*.

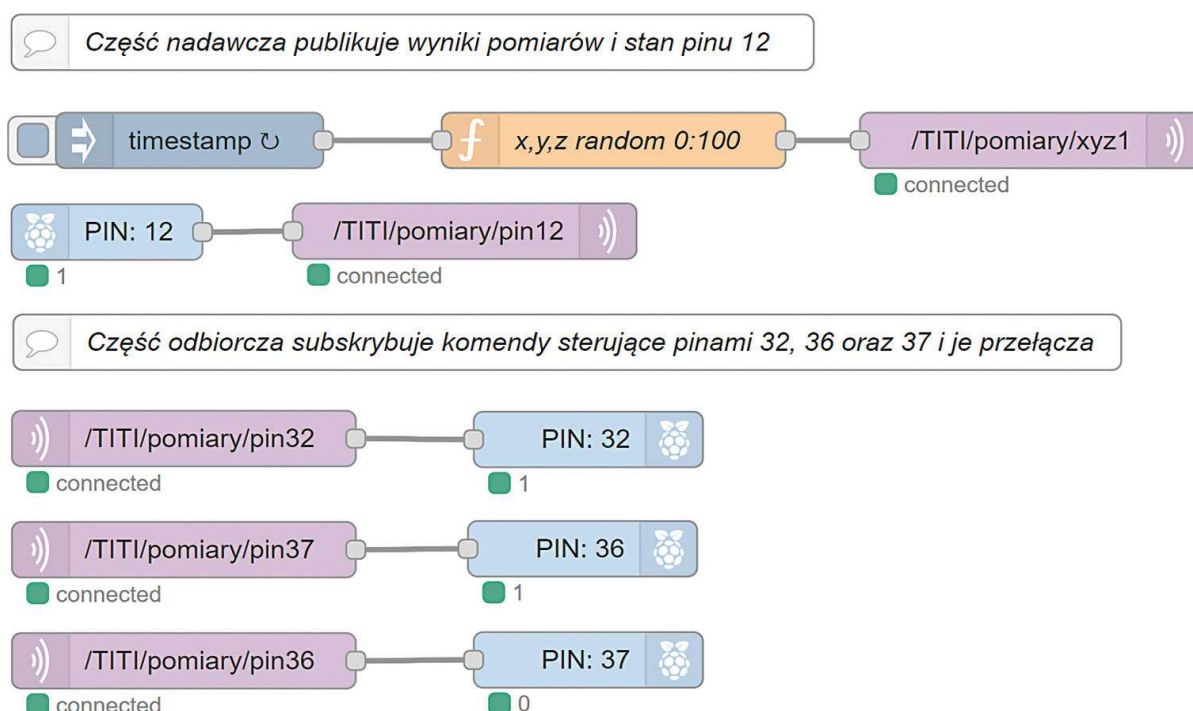
♦ Można zainstalować na komputerze w laboratorium lub innym broker MQTT, np. Eclipse Mosquitto™.

♦ Można skorzystać z brokera lukan.sytes.net.

♦ Można skorzystać z bezpłatnego publicznego brokera dostępnego w Internecie, np.: <https://www.hivemq.com/public-mqtt-broker/> lub <https://www.emqx.com/en/mqtt/public-mqtt5-broker>.

Funkcję układów A i B mogą pełnić komputery w laboratorium, smartfony lub Raspberry Pi z dostępem do Internetu. Na urządzeniach należy uruchomić i skonfigurować odpowiednie oprogramowanie.

Układ kontrolno-pomiarowy A – komputer PC lub Raspberry Pi z odpowiednio zaprojektowaną aplikacją Node-RED. Przykładowy graf przedstawiono na rys. 8.2.



Rys. 8.2. Przykładowy graf Node-RED symulujący działanie prostego układu kontrolno-pomiarowego

♦ Górna gałąź grafu symuluje działanie układu pomiarowego i publikuje wyniki pomiarów x, y, z w temacie /TITI/pomiary/xyz1.

♦ Dolna gałąź grafu subskrybuje temat /TITI/pomiary/pinXXi, odbiera komendy sterujące pinami 32, 36 i 37 Raspberry Pi.

♦ Do testów na komputerach PC, w których nie ma GPIO z pinami, subskrybowane tematy można wysyłać do debugera.

Bloczki Node-RED użyte w grafie na rys. 8.2. przedstawiono w tabeli 8.1.

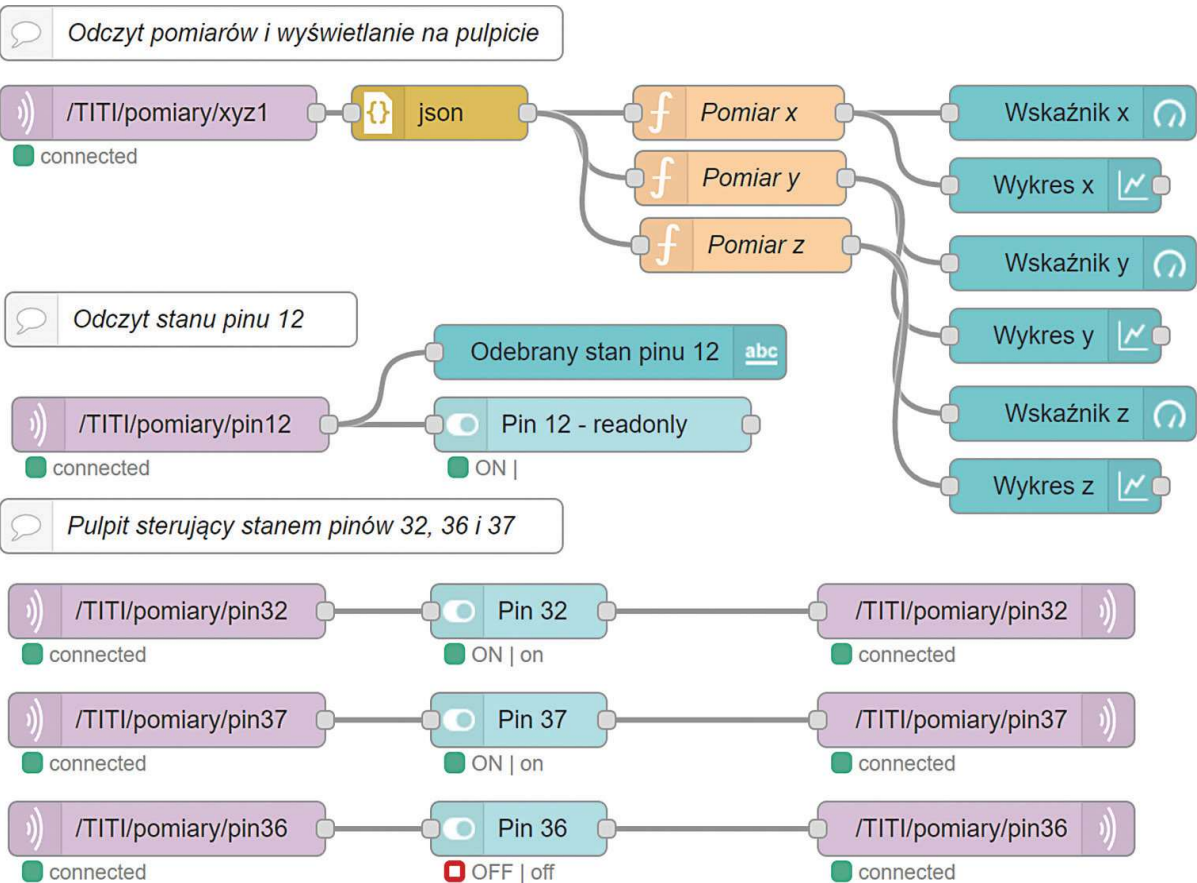
Tabela 8.2.

Bloczki Node-RED użyte w grafie na rys. 8.2.

 mqtt in	Bloczek subskrybenta MQTT.
mqtt out 	Bloczek publikatora MQTT.
 aedes broker	Bloczek prostego brokera MQTT. Broker MQTT jest zaimplementowany w środowisku node.js. Aby korzystać z brokera, wystarczy umieścić na którymkolwiek grafie bloczek mosca in. Broker działa od razu z domyślną konfiguracją, którą można zmienić.

Układ zdalnego sterowania B – może być zrealizowany na komputerze PC lub urządzeniu przenośnym.

Na komputerze PC można stworzyć pulpit sterujący, wykorzystując program Node-RED. Do stworzenia interfejsu użytkownika można wykorzystać pakiet node-red-dashboard. Przykładowy graf realizujący graficzny interfejs użytkownika (panel sterujący) przedstawiono na rys. 8.3.



Rys. 8.3. Przykładowy graf Nade-RED realizujący graficzny interfejs użytkownika dla układu kontrolno-pomiarowego z rys. 8.2.

Ważne dla poprawnego działania grafu jest właściwe skonfigurowanie przełączników sterujących pinami o nazwach Pin 32, Pin 36 i Pin 37. Poprawną konfigurację przedstawiono na rys. 8.4.

→ Pass through `msg` if payload matches new state: ☐

☐ Indicator Switch icon shows state of the **input** ▼

☒ When clicked, send:

On Payload ▼ `a_z` 1

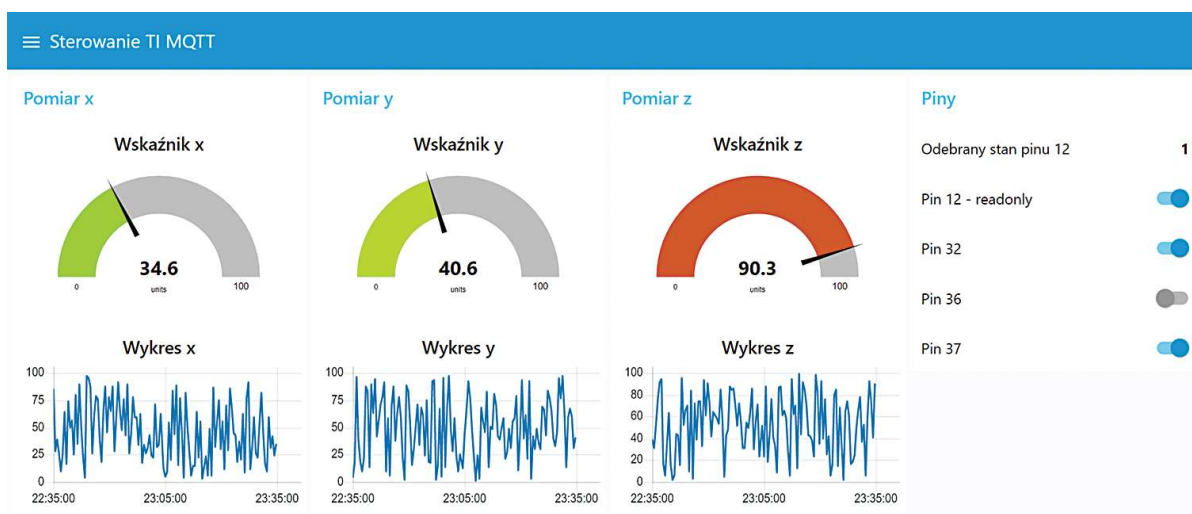
Off Payload ▼ `a_z` 0

Topic ▼ `a_z`

Rys. 8.4. Fragment okna dialogowego z poprawną konfiguracją przełącznika

Czerwonymi owalami zaznaczono pozycje, które trzeba zmienić. Błędna konfiguracja może spowodować zablokowanie brokera poprzez ciągłe wysyłanie publikacji. Inne bloczki można konfigurować bez obawy o zablokowanie brokera.

Na rys. 8.5. przedstawiono pulpit sterujący będący efektem działania grafu. Pulpit jest dostępny pod adresem <http://adres hosta:1880/ui>.

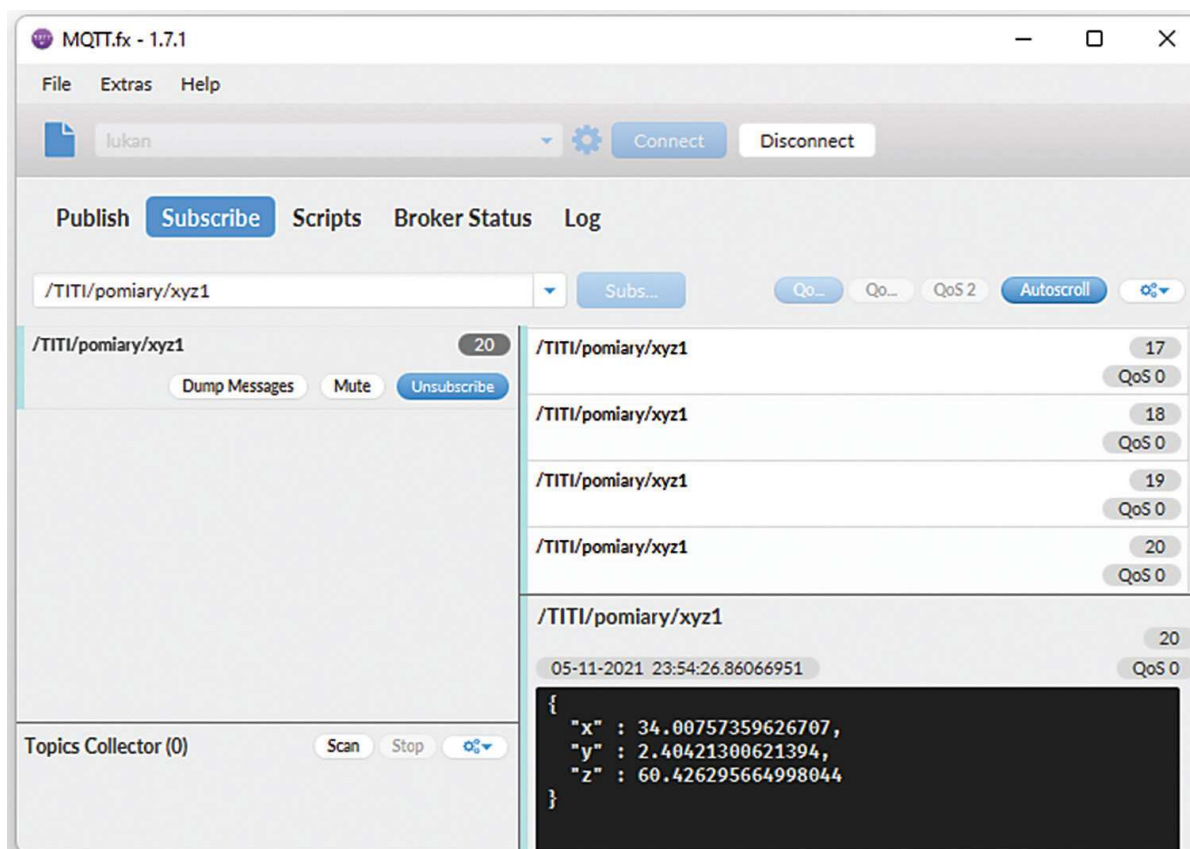


Rys. 8.5. Interfejs użytkownika zrealizowany przez graf z rys. 8.3.

Na komputer PC dostępnych jest wiele bezpłatnych klientów MQTT, których można wykorzystać do subskrypcji wyników pomiarów i publikowania komend sterujących. Na rys. 8.6. (s. 70) przedstawiono przykład wykorzystania programu MQTT.fx do subskrypcji wyników pomiarów i publikacji komend.

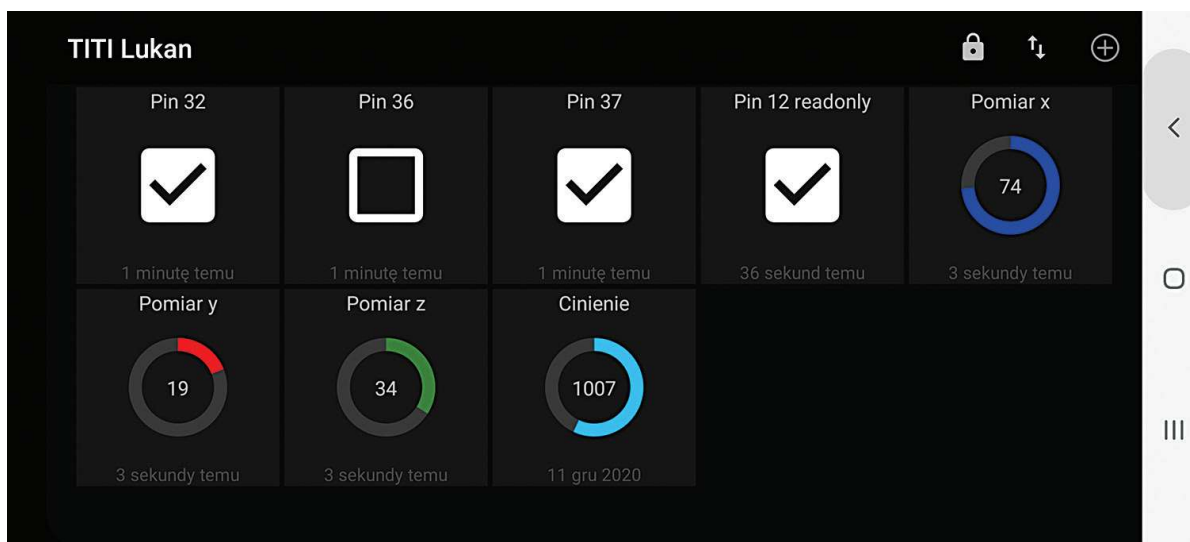
Program o podobnej funkcjonalności można zainstalować z adresu: <http://argo.am.gdynia.pl/www/Internet/MQTT/publish.htm>, kod programu w języku C# znajduje

się pod adresem: <https://github.com/Andrzej1959/MQTTclient>. Program MQTTclient do łączności z brokerem MQTT wykorzystuje bibliotekę: <https://github.com/chkr1011/MQTTnet>. Program można łatwo rozszerzyć o graficzny pulpit z wykresami wyników i przyciskami do sterowania pinami.



Rys. 8.6. Wykorzystania programu MQTT.fx do subskrypcji wyników pomiarów i publikacji komend

Pulpit sterujący można też zrealizować na urządzeniu przenośnym z dostępem do Internetu. Istnieje kilka programów klienta MQTT na smartfony z systemem Android. Na rys. 8.7. przedstawiono pulpit sterujący zrealizowany z użyciem programu MQTT Dash.



Rys. 8.7. Pulpit sterujący w telefonie na pulpicie programu MQTT Dash

Takich programów jest sporo i są dedykowane do Internetu rzeczy, w tym do współpracy z Node-RED. W ramach zadania można przetestować różne programy lub skorzystać z programu MQTT Dash jako wypróbowanego.

Jak wykonać ćwiczenie

- ♦ Ćwiczenie można wykonać indywidualnie lub w grupach dwuosobowych.
- ♦ Budując własny system z zastosowaniem wspólnego brokera, należy się posługiwać unikatowymi nazwami tematów, tak aby wzajemnie sobie nie przeszkadzać.
- ♦ Całość ćwiczenia można wykonać wyłącznie na komputerach w laboratorium, ale że system jest rozproszony, poszczególne elementy dobrze jest zrealizować na komputerach własnych, np. na laptopach czy Raspberry PI z dostępem do Internetu za pośrednictwem sieci komórkowej.
- ♦ Broker może być zainstalowany na komputerze w laboratorium lub na własnym komputerze z publicznym adresem IP.
- ♦ Pulpit sterujący można zrealizować na własnym urządzeniu przenośnym.
- ♦ Minimalne wymagania to system zrealizowany na platformie Node-RED na jednym (tylko indywidualnie), dwóch lub trzech komputerach w laboratorium.
- ♦ Można dowolnie rozszerzać funkcjonalność systemu, np. o pomiary rzeczywistych wartości fizycznych, sterowanie rzeczywistymi urządzeniami wykonawczymi, wpisywanie danych do bazy danych, np. za pomocą usługi WebApi z poprzedniego ćwiczenia.
- ♦ Cały system można wykonać poza laboratorium i przynieść na zajęcia gotowe działające oprogramowanie. Na zajęciach można je uruchomić na sprzęcie własnym lub dostępnym w laboratorium.