

Dokumentacja Projektowa – System IoT z OPC UA oraz platformą Azure

Przed rozpoczęciem

1. Do działania na aplikacji jest potrzebne konto na platformie Azure.
 2. Do uruchomienia aplikacji wymagane jest Visual Studio (<https://visualstudio.microsoft.com/pl/>).
 - Upewnij się, że masz dostęp do środowiska .NET 6.0 (można doinstalować w Visual Studio Installer -> Modyfikuj -> Pojedyncze składniki -> Środowisko uruchomieniowe platformy .NET 6.0).
 3. Pobierz i wypakuj folder z <https://github.com/mateusz2/loTProject>
-

Instrukcja uruchomienia aplikacji

Uzupełnij pola w kolumnie Wartość wskazanymi wartościami (wspieraj się komentarzami).

- Konfiguracja **Device - OPC UA Client - Agent** (Device → Properties → Resources.resx)

Nazwa	▲	Wartość	Komentarz
Device1ConnectionString			Connection String urządzenia stworzonego w IoT Hub.
Device2ConnectionString			Connection String urządzenia stworzonego w IoT Hub.
Device3ConnectionString			Connection String urządzenia stworzonego w IoT Hub.
EmailConnectionString			Connection String do Communication Service.
opcClientURL			Link URL do serwera OPC UA.
receiverEmail			Adres email, na który będą przychodzić wiadomości o wystąpieniu nowych błędów na maszynach.
senderEmail			Adres email w Communication Service na platformie Azure, skąd będą wysyłane wiadomości.

- Konfiguracja **Service** (Device → Properties → Resources.resx)

Nazwa	▲	Wartość	Komentarz
deviceErrorQueue			Nazwa kolejki utworzonej w Service Bus Namespace do błędów.
iotHubConnectionString			Connection String do IoT Hub na platformie Azure.
KPIQueueName			Nazwa kolejki utworzonej w Service Bus Namespace do zbierania wartości KPI urządzeń.
serviceBusConnectionString			Connection String do Service Bus Namespace na platformie Azure.

- Należy skonfigurować oba projekty jako startowe i uruchomić

☒ Wiele projektów startowych:

Projekt	Akcja
Device	Uruchomienie ▾
Service	Uruchomienie ▾

↑
↓

Połączenie z serwerem, pobieranie i przetwarzanie danych

- Aplikacja łączy się z serwerem OPC UA dzięki linku URL, który był wcześniej przez Ciebie uzupełniony w Resources.resx.
- Po uruchomieniu program wczytuje dostępne urządzenia (za urządzenie przyjmuje obiekt w węźle, którego nazwa pasuje do wzorca Device [liczba], np. Device 3.
Uwaga! Program działa dla maksymalnie 3 urządzeń jednocześnie.
- Każde 5 sekund kolejno pobiera od urządzeń dane i je przetwarza.

Sposoby komunikacji z platformą Azure

- Device-to-Cloud (D2C)**

- Telemetrie**

Co 5 sekund aplikacja wysyła dane telemetryczne do IoT Hub:

- Production Status – status produkcji, określający, czy urządzenie jest włączone
- Workorder ID – identyfikator urządzenia
- Good Count – liczbę wyprodukowanych produktów dobrej jakości
- Bad Count – liczbę produktów wadliwych
- Temperature – aktualną temperaturę urządzenia
- Device Errors – informacje o ewentualnych błędach urządzenia

```
{
  "body": {
    "ProductionStatus": 1,
    "WorkerId": "603f7982-7406-4386-aa0a-7466e9d9ac04",
    "Temperature": 81.01487461397274,
    "GoodCount": 16,
    "BadCount": 2
  },
  "enqueuedTime": "Tue Jan 21 2025 19:54:19 GMT+0100 (czas
środkowoeuropejski standardowy)"
}
```

- **Device twin reported properties**

Właściwości zgłaszane Device Twin służą do raportowania informacji o stanie urządzenia, takich jak dostępne możliwości, warunki lub stan długotrwałych przepływów pracy. Na przykład konfiguracja i aktualizacje oprogramowania.

```
"reported": {
  "ProductionRate": 40,
  "DeviceStatus": [
    "PowerFailure"
  ],
  "$metadata": {
    "$lastUpdated": "2025-01-21T19:04:07.5683281Z",
    "ProductionRate": {
      "$lastUpdated": "2025-01-21T19:04:07.5683281Z"
    },
    "DeviceStatus": {
      "$lastUpdated": "2025-01-21T19:00:45.3125487Z"
    }
  },
  "$version": 3087
}
```

- **Cloud-to-Device (C2D)**

- **Direct Methods**

Direct Methods to sposób komunikacji, który pozwala chmurze wysyłać polecenia bezpośrednio do urządzenia. Umożliwia to zdalne zlecenie działań wymagających szybkiej reakcji. Proces jest inicjowany z chmury, a urządzenie odpowiada w czasie rzeczywistym.

Aplikacja wykorzystuje Direct Methods w dwóch sytuacjach:

1. Do wywołania metody Emergency Stop (awaryjne zatrzymanie)
2. Do wywołania metody Reset Error Stop (resetowanie zatrzymania spowodowanego błędem)

2025 > [Devices](#) > Line1 > Direct method

Invoke method

Direct method You can send direct methods to a device, configurable connection and method timeouts.

Method name *

EmergencyStop

Payload ⓘ

Connection timeout in seconds ⓘ

Response timeout in seconds ⓘ

Successfully invoked method 'EmergencyStop' on device 'Line1' with response {"status":200,"payload":{"result":"EmergencyStop triggered"}}.

Notification center 9:37:51 PM

10

10

- **Device twin desired properties**

Pożądane właściwości używane wraz ze zgłoszonymi właściwościami do synchronizowania konfiguracji lub warunków urządzenia. Aplikacje back-end mogą ustawiać pożądane właściwości, a aplikacja urządzenia może je odczytywać. Aplikacja urządzenia może również otrzymywać powiadomienia o zmianach w pożądanych właściwościach.

```
"desired": {
  "ProductionRate": 40,
  "EmergencyTrigger": 1,
  "$metadata": {
    "$lastUpdated": "2025-01-20T21:17:01.1763052Z",
    "$lastUpdatedVersion": 43,
    "ProductionRate": {
      "$lastUpdated": "2025-01-20T21:17:01.1763052Z",
      "$lastUpdatedVersion": 43
    },
    "EmergencyTrigger": {
      "$lastUpdated": "2025-01-20T21:17:01.1763052Z",
      "$lastUpdatedVersion": 43
    }
  },
  "$version": 43
},
```

Business Logic na platformie Azure

W celu zaimplementowania dalszych funkcjonalności, należy skonfigurować dodatkowe obiekty na platformie Azure:

1. Storage Account, a w nim kontener do przechowywania wyników przetwarzania temperatury maszyn
2. Stworzyć obiekt Stream Analytics Job
3. Przejść do widoku Job topology -> Query
4. Jako Inputs ustawić swój IoT Hub, natomiast jako Outputs dodać kolejki, które były już konfigurowane w Resources.resx oraz kontener z punktu 1
5. Otworzyć załączony plik asaQuery.txt i skopiować jego zawartość do pola przeznaczonego dla zapytań
6. Zapytanie należy odpowiednio przystosować do swoich urządzeń:
 - Zamienić nazwę w FROM [...] na własny Input (IoT Hub)
 - Zamienić nazwy w INTO [...] na odpowiadające zapytaniom obiekty (kolejkę KPI dla wartości KPI, kontener dla temperatury, kolejkę Error dla błędów urządzenia)

Na podstawie otrzymanych wiadomości platforma Azure realizuje kolejne funkcjonalności:

- **Monitorowanie wskaźnika KPI (kluczowe wskaźniki efektywności)**

Platforma przetwarza dane dotyczące liczby produktów dobrych (Good Count) oraz wadliwych (Bad Count), obliczając wskaźnik KPI, który przedstawia procentowy udział produktów dobrej jakości w całkowitej produkcji. Obliczenia te są realizowane w 5-minutowych oknach czasowych, co pozwala na regularne monitorowanie jakości produkcji w krótkich interwałach. Jeżeli wskaźnik KPI spadnie poniżej 90%, system automatycznie obniża wartość pożądanej wydajności produkcji (Desired Production Rate) o 10 punktów procentowych

- **Monitorowanie temperatury**

Co minutę z telemetrii są wyliczane wartości minimalne, maksymalne oraz średnie temperatury maszyny z ostatnich 5 minut. Wyniki będą magazynowane w przeznaczonym na to kontenerze na platformie Azure

- **Monitorowanie błędów urządzenia**

Każde wystąpienie nowego błędu jest zgłaszane do platformy Azure. Jeżeli w ciągu jednej minuty maszyna doświadczy więcej niż 3 błędów, aplikacja natychmiast wywoła metodę **Emergency Stop** (zatrzymanie awaryjne). Dodatkowo informacja o nowych błędach jest przekazywana drogą elektroniczną na email, który był wskazany w Resources.resx