

Projekt IoT Azure - dokumentacja dla użytkownika

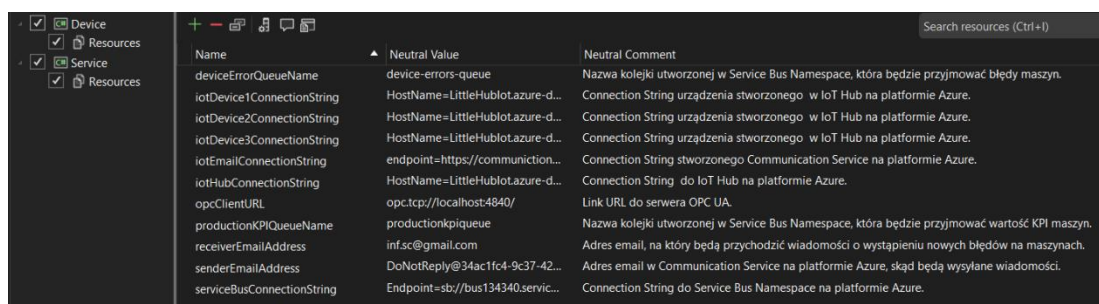
Aplikacja służy do zdalnego monitorowania i zarządzania liniami produkcyjnymi, łącząc fizyczne maszyny z **platformą IoT na Azure**. Poprzez protokół **OPC UA** zbiera dane z urządzeń, które są następnie przetwarzane i analizowane w chmurze.

Przed rozpoczęciem

1. Do działania na aplikacji jest potrzebne konto na platformie Azure.
2. W celu uruchomienia aplikacji Twój komputer musi mieć zainstalowany Visual Studio (<https://visualstudio.microsoft.com/pl/>).
 - Upewnij się, że masz dostęp do środowiska .NET 6.0 (możesz go doinstalować w Visual Studio Installer -> Modyfikuj -> Pojedyncze składniki -> Środowisko uruchomieniowe platformy .NET 6.0)
3. Pobierz i wypakuj folder z <https://github.com/abdukhamidova/IoTProjectAzure>

Uruchomienie aplikacji

1. Otwórz projekt w **Visual Studio**
 - Zlokalizuj plik o nazwie *IoTProjectAzure.sln* w pobranym folderze.
 - Kliknij dwukrotnie ten plik, projekt automatycznie otworzy się w programie Visual Studio.
2. Konfiguracja Connection Strings
 - W programie Visual Studio, znajdź widok Solution Explorer (zazwyczaj po prawej stronie), w którym zobaczysz drzewo plików projektu
 - Przejdź według ścieżki: **Device** → **Properties** → **Resources.resx** i otwórz plik.
 - Po lewej stronie edytora pliku **Resources.resx** znajdziesz widok z opcjami do zaznaczenia: Device i Service.
 - Upewnij się, że obie opcje są zaznaczone. Jeśli którakolwiek z nich nie jest zaznaczona, zaznacz ją.
 - Uzupełnij pola w kolumnie *Neutral Value* wskazanymi wartościami (wspieraj się komentarzami w kolumnie *Neutral Comment*).



Name	Neutral Value	Neutral Comment
deviceErrorQueueName	device-errors-queue	Nazwa kolejki utworzonej w Service Bus Namespace, która będzie przyjmować błędy maszyn.
iotDevice1ConnectionString	HostName=LittleHubIot.azure-d...	Connection String urządzenia stworzonego w IoT Hub na platformie Azure.
iotDevice2ConnectionString	HostName=LittleHubIot.azure-d...	Connection String urządzenia stworzonego w IoT Hub na platformie Azure.
iotDevice3ConnectionString	HostName=LittleHubIot.azure-d...	Connection String urządzenia stworzonego w IoT Hub na platformie Azure.
iotEmailConnectionString	endpoint=https://communication...	Connection String stworzonego Communication Service na platformie Azure.
iotHubConnectionString	HostName=LittleHubIot.azure-d...	Connection String do IoT Hub na platformie Azure.
opcClientURL	opc.tcp://localhost:4840/	Link URL do serwera OPC UA.
productionKPIQueueName	productionkpiqueue	Nazwa kolejki utworzonej w Service Bus Namespace, która będzie przyjmować wartość KPI maszyn.
receiverEmailAddress	inf.sc@gmail.com	Adres email, na który będą przychodzić wiadomości o wystąpieniu nowych błędów na maszynach.
senderEmailAddress	DoNotReply@34ac1fc4-9c37-42...	Adres email w Communication Service na platformie Azure, skąd będą wysyłane wiadomości.
serviceBusConnectionString	Endpoint=sb://bus134340.servic...	Connection String do Service Bus Namespace na platformie Azure.

(Rys. 1) Przykład uzupełnionego pliku Resources.resx

3. Kliknij na zielony trójkąt w górnym pasku **Menu**, aby uruchomić aplikację.

Połączenie z serwerem, pobieranie i przetwarzanie danych

1. Aplikacja łączy się z serwerem **OPC UA** dzięki linku URL, który był wcześniej przez Ciebie uzupełniony w *Resources.resx*.
2. Po uruchomieniu program czytuje 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.
3. Każde 3 sekundy kolejno pobiera od urządzeń dane i je przetwarza.

Wiadomości Device-to-Cloud (D2C)

Wiadomości D2C to komunikaty wysyłane przez urządzenia do Azure IoT Hub. Służą do przesyłania danych telemetrycznych, raportowania statusu urządzeń, a także zgłaszania błędów.

Aplikacja wysyła dwa rodzaje wiadomości D2C:

1. Telemetrie
2. Błędy urządzenia

1. Telemetrie

Co 3 sekundy aplikacja pobiera dane telemetryczne z urządzenia, obejmujące:

- **Production Status** – status produkcji, określający, czy urządzenie jest włączone.
- **Worker ID** – identyfikator urządzenia.
- **Production Rate** – wydajność produkcji wyrażoną w procentach.
- **Good Count** – liczbę wyprodukowanych produktów dobrej jakości.
- **Bad Count** – liczbę produktów wadliwych.
- **Temperature** – aktualną temperaturę urządzenia.
- **Device Errors** – informacje o ewentualnych błędach urządzenia.

Wszystkie telemetrie oprócz **Production Rate** są wysyłane do odpowiednika urządzenia w chmurze Azure.

```
{
  "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)"
}
```

(Rys. 2) Przykładowa telemetria

2. Błędy urządzenia

W przypadku zmiany statusu maszyny, spowodowanej na przykład wystąpieniem błędu, informacja o tym jest przesyłana do urządzenia na platformie Azure.

```
{
  "body": {
    "Message": "Status has changed from None to PowerFailure"
  },
  "enqueuedTime": "Tue Jan 21 2025 20:00:45 GMT+0100 (czas
środkowoeuropejski standardowy)"
}
```

(Rys.3) Przykładowa wiadomość o zmianie statusu urządzenia.

Device Twin

Device Twin to specjalny wirtualny model urządzenia w chmurze Azure. Pozwala na synchronizację i zarządzanie danymi pomiędzy urządzeniem fizycznym a aplikacją w chmurze. Składa się z dwóch kluczowych części: **Desired Properties** i **Reported Properties**.

- **Desired Properties** (pożądane własności) zawierają ustawienia, które aplikacja w chmurze chce wymusić na urządzeniu.
- **Reported Properties** (raportowane własności) to dane aktualne, które urządzenie przesyła do chmury, odzwierciedlając swój rzeczywisty stan.

Aplikacja wykorzystuje funkcjonalność Device Twin do przechowywania dwóch wartości maszyn:

1. Współczynnik produkcji
2. Błędy urządzenia

1. Wartość **Production Rate** jest zapisywana w **Reported Properties**, odzwierciedlając aktualny stan.

Maszyna na bieżąco monitoruje zmiany w **Desired Properties** i dynamicznie dostosowuje rzeczywistą wartość współczynnika produkcji, aby spełnić zadane wymagania.

2. Wartość **Device Error** (status błędów urządzenia) jest na bieżąco aktualizowana w **Reported Properties**.

```
"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
},
```

(Rys. 4) Przykład Desired Properties.

```
"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
}
```

(Rys. 5) Przykład Reported Properties.

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).

Direct Methods wywołuje się z poziomu urządzeń w IoT Hub:

- Znajdź w zakładkach **Device Management** -> **Devices** -> wybierz urządzenie, na którym chcesz wywołać metodę -> **Direct method**.
- W pasku *Method name* wpisz nazwę metody, którą zamierzasz wywołać:



Device ID

DeviceDemoSdk1

Method name * ⓘ

ResetErrorStatus

(Rys. 6) Przykład wywołania metody.

- Kliknij na *Invoke method*.

Uwaga! Metody, które są obsługiwane przez maszyny to:

1. **EmergencyStop**
2. **ResetErrorStatus**

W przypadku wywołania innej metody, aplikacja zwraca jedynie wiadomość o próbie jej uruchomienia.

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:

1. 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.

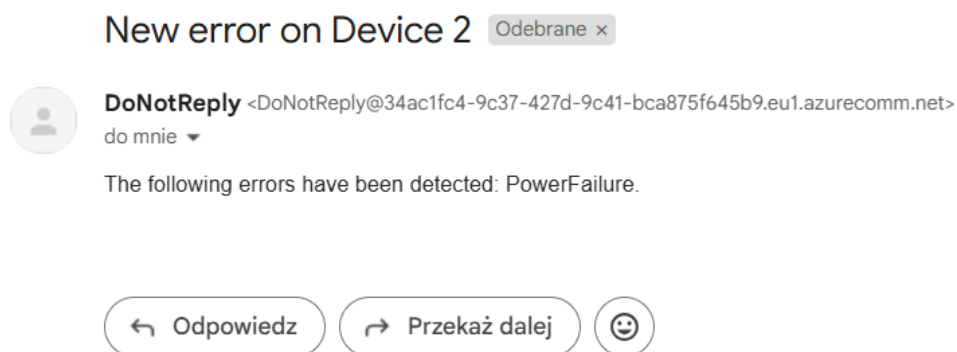
Jeżeli wskaźnik KPI spadnie poniżej 90%, system automatycznie obniża wartość pożądaney wydajności produkcji (Desired Production Rate) o 10 punktów procentowych.

2. 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.

3. 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*.



(Rys. 7) Przykład odebranej wiadomości e-mail.