

Projekt z przedmiotu Internet of Things

Projekt zrealizował:
Wojciech
Rowiński
Numer 420152

2025

Spis treści

Wstęp	3
Połączenie z urządzeniem (serwerem OPC UA)	3
Sposób uruchomienia aplikacji	3
Sposób połączenia z serwerem	4
Sposób i częstotliwość odczytu i zapisu danych oraz wywoływania węzłów- metod	4
Konfiguracja agenta	5
Sposób, w jaki agent jest konfigurowany do komunikacji z danym serwerem OPC UA oraz instancją IoT Hub	5
Plik konfiguracyjny oraz zmienne środowiskowe	6
Rodzaj, format i częstotliwość wiadomości (D2C messages) wysyłanych przez agenta do IoT Hub	6
Rodzaj i format danych przechowywanych w Device Twin	8
Direct Methods zaimplementowane w agencie	9
Metoda EmergencyStop	9
Metoda ResetErrorStatus	11
Kalkulacje i logika biznesowa	12
Kalkulacje	12
Production KPIs	12
Temperature	13
Device errors	14
Logika biznesowa	15
Automatyczne zatrzymanie awaryjne przy wielu błędach urządzenia	15
Automatyczna redukcja Desired Production Rate przy spadku efektywności produkcji	17
Konkluzje	18

Wstęp

Niniejsza dokumentacja dotyczy aplikacji umożliwiającej połączenie urządzenia przemysłowego z chmurą Azure IoT z wykorzystaniem protokołu OPC UA. Projekt zakłada komunikację z jednym, konkretnym urządzeniem, jednak jego architektura została przygotowana w taki sposób, aby możliwe było rozszerzenie rozwiązania na większą liczbę urządzeń w przyszłości.

Aplikacja została zaimplementowana w języku C# i integruje się z usługami Azure, takimi jak IoT Hub, Device Twin, Azure Functions, Stream Analytics oraz Logic Apps. Główne zadania systemu to odczyt danych z serwera OPC UA, przesyłanie ich do chmury, synchronizacja ustawień z poziomu Device Twin, reagowanie na zdarzenia (np. błędy urządzenia) oraz wykonywanie prostych akcji w odpowiedzi na stan urządzenia.

Dokumentacja obejmuje sposób uruchamiania i konfiguracji systemu, opis zastosowanych technologii oraz ogólne zasady działania aplikacji w obecnej wersji.

Połączenie z urządzeniem (serwerem OPC UA)

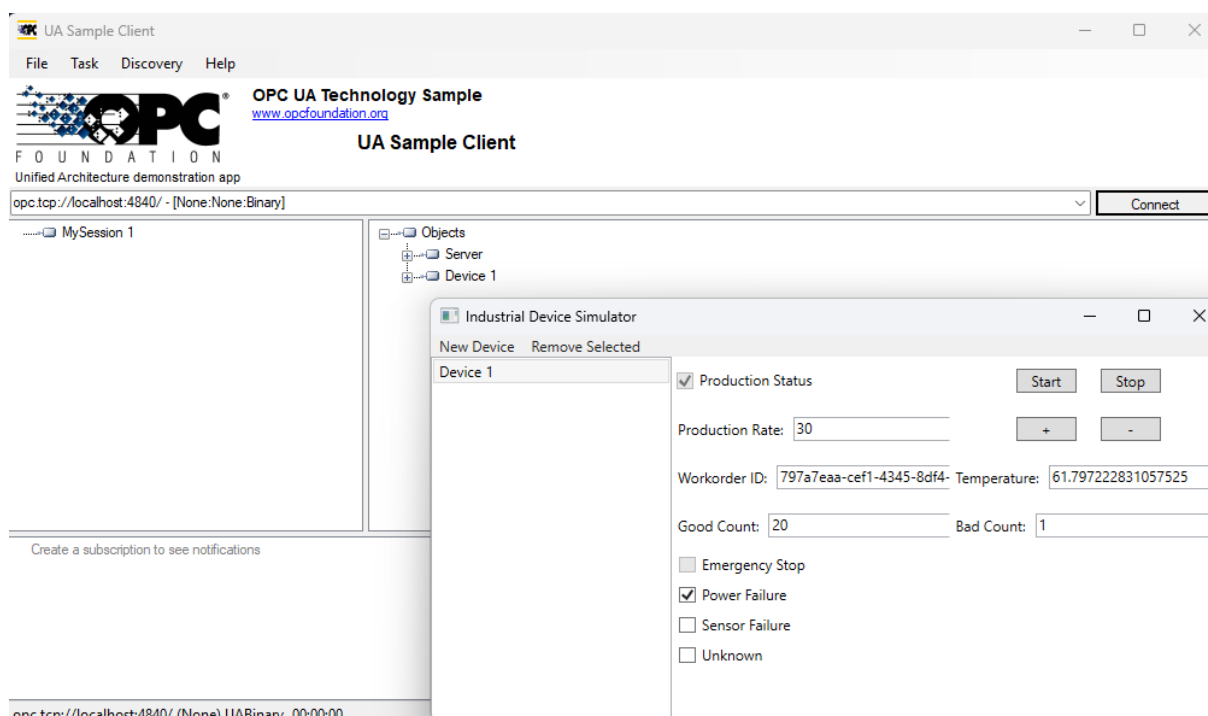
Sposób uruchomienia aplikacji

Aplikacja została przygotowana jako projekt w języku C# w środowisku Visual Studio. Przed jej uruchomieniem należy upewnić się, że wszystkie wymagane zależności zostały zainstalowane (np. pakiety NuGet do obsługi OPC UA, Azure IoT Hub i konfiguracji).

Aby uruchomić aplikację:

1. Skonfiguruj dane połączeniowe – w pliku konfiguracyjnym *keys.txt* należy uzupełnić dane połączeniowe do serwera OPC UA oraz do Azure IoT Hub (connection string urządzenia).
2. Zbuduj projekt – należy otworzyć rozwiązanie w Visual Studio i wykonać kompilację.
3. Uruchom aplikację – aplikację można uruchomić lokalnie z poziomu Visual Studio.

Aby aplikacja działała poprawnie, niezwykle ważne jest wcześniejsze uruchomienie symulatora urządzenia (Device Simulator), który pełni rolę serwera OPC UA. Następnie należy upewnić się, że symulator nasłuchuje na odpowiednim porcie (np. *opc.tcp://localhost:xxxx*) i że aplikacja klienta łączy się z nim za pomocą prawidłowego adresu.



Rysunek 1. Poprawnie skonfigurowany OPC UA Client i Device Simulator.

Sposób połączenia z serwerem

Po uruchomieniu aplikacji, klient OPC UA automatycznie nawiązuje połączenie z serwerem OPC UA na podstawie adresu URI zdefiniowanego w pliku konfiguracyjnym `keys.txt`. Adres serwera jest podawany w drugim wierszu pliku.

Aplikacja wykorzystuje bibliotekę OPC Foundation (`Opc.Ua.Client`) do realizacji połączenia i odczytu danych. Po zestawieniu sesji z serwerem, aplikacja subskrybuje określone węzły telemetryczne, takie jak status produkcji, licznik dobrych/słabych produktów oraz temperaturę. Połączenie jest utrzymywane przez cały czas działania aplikacji.

Sposób i częstotliwość odczytu i zapisu danych oraz wywoływania węzłów-metod

Aplikacja automatycznie odczytuje dane z serwera OPC UA poprzez subskrypcję wybranych węzłów telemetrycznych. Subskrypcja jest tworzona zaraz po nawiązaniu połączenia z serwerem, a odczyt wartości odbywa się cyklicznie co 1 sekundę (1000 ms).

Dane takie jak liczba dobrych i złych produktów, temperatura czy status produkcji są automatycznie przesyłane do usługi Azure IoT Hub jako komunikaty telemetryczne (D2C – Device-to-Cloud).

Każda zmiana dotycząca pojawienia się błędów na urządzeniu (*Power failure*, *Sensor failure*, *Unknown*) jest wysyłana do Azure IoT Hub jako osobne

zdarzenie oraz aktualizowana i zapisywana w Device Twin w części reported, co pozwala na śledzenie stanu błędów urządzenia w czasie rzeczywistym

Zapis do serwera OPC UA następuje w momencie, gdy otrzymywana jest aktualizacja właściwości „desired” w Device Twin (np. zmiana productionRate). Wówczas aplikacja zapisuje nową wartość do odpowiedniego węzła na serwerze OPC UA.

Wywoływanie metod (np. EmergencyStop) odbywa się poprzez mechanizm Direct Methods w Azure IoT Hub. Metoda może zostać wywołana zarówno ręcznie z poziomu narzędzia Azure IoT Hub (Device Preview), jak i automatycznie za pomocą dedykowanej funkcji Azure. Po odebraniu żądania aplikacja wykonuje operację na przypisanym węźle-metodzie w serwerze OPC UA.

Konfiguracja agenta

Sposób, w jaki agent jest konfigurowany do komunikacji z danym serwerem OPC UA oraz instancją IoT Hub

Agent jest konfigurowany za pomocą parametrów przekazywanych podczas uruchomienia aplikacji lub poprzez pliki konfiguracyjne, które zawierają m.in. adres serwera OPC UA oraz ciąg połączeniowy (connection string) do instancji Azure IoT Hub.

Aby nawiązać komunikację, aplikacja łączy się z serwerem OPC UA pod wskazanym adresem i porcie, co wymaga uruchomienia odpowiedniego symulatora urządzenia (device simulator) oraz działającego jako serwer OPC UA. Połączenie jest utrzymywane przez cały czas działania agenta, umożliwiając cykliczny odczyt i zapis danych.

Po stronie IoT Hub agent korzysta z przekazanego connection stringa, który pozwala na bezpieczne wysyłanie danych telemetrycznych oraz odbieranie poleceń (Direct Methods) i aktualizacji właściwości Device Twin. Konfiguracja ta umożliwia dwukierunkową komunikację pomiędzy urządzeniem (serwerem OPC UA) a chmurą Azure.

W przypadku zmiany parametrów połączenia lub innych ustawień można je modyfikować w pliku konfiguracyjnym lub jako zmienne środowiskowe przed uruchomieniem aplikacji. Dzięki temu agent może być łatwo dostosowany do pracy z różnymi instancjami serwera OPC UA oraz różnych urządzeń IoT Hub.

Należy dodać, że niektóre zmienne, takie jak nazwa lokalnego urządzenia, jego identyfikator w IoT Hub oraz nazwa kontenera w Azure Storage na *telemetryData*, są z góry określone w kodzie aplikacji czy funkcji, co wpływa

na sposób działania agenta i wymaga uwagi przy ewentualnej rozbudowie projektu lub zmianie konfiguracji.

Plik konfiguracyjny oraz zmienne środowiskowe

Aplikacja korzysta z pliku tekstowego, w którym zapisane są kluczowe ustawienia do komunikacji z serwerem OPC UA, Azure IoT Hub oraz Azure Storage. Przykładowa zawartość pliku wygląda następująco:

- Pierwsza linia to connection string urządzenia do Azure IoT Hub — umożliwia identyfikację i autoryzację urządzenia w chmurze.
- Druga linia zawiera adres serwera OPC UA, z którym aplikacja się łączy (lokalny host i port).
- Trzecia linia to connection string do konta Azure Storage, wykorzystywanego do przechowywania danych i logów

Dodatkowo, funkcje Azure używane w projekcie korzystają ze zmiennej środowiskowej o nazwie `IoTHubConnectionString`, która przechowuje connection string do Azure IoT Hub. Dzięki temu te dane nie są trwale wpisane w kod, co podnosi bezpieczeństwo i ułatwia zarządzanie konfiguracją.

Nazwa	Wartość	Ustawienie miejsca wdrożenia	Źródło	Usuń
APPLICATIONINSIGHTS_CONNECTION_STRING	Pokaż wartość		App Service	Usuń
AzureWebJobsStorage	Pokaż wartość		App Service	Usuń
FUNCTIONS_EXTENSION_VERSION	Pokaż wartość		App Service	Usuń
FUNCTIONS_WORKER_RUNTIME	Pokaż wartość		App Service	Usuń
IoTHubConnectionString	Pokaż wartość		App Service	Usuń
WEBSITE_CONTENTAZUREFILECONNECTIONSTRING	Pokaż wartość		App Service	Usuń
WEBSITE_CONTENTSHARE	Pokaż wartość		App Service	Usuń
WEBSITE_RUN_FROM_PACKAGE	Pokaż wartość		App Service	Usuń

Rysunek 2. Przykład zmiennej środowiskowej użytej w jednej z Funkcji Aplikacji.

Rodzaj, format i częstotliwość wiadomości (D2C messages) wysyłanych przez agenta do IoT Hub

Wszystkie wiadomości są wysyłane w formacie JSON (tekst UTF-8).

Dane telemetryczne (telemetry data)

- Zawierają bieżące wartości pomiarów i statusów urządzenia: ID urządzenia, czas pomiaru, status produkcji, liczby dobrych i złych wyrobów, temperaturę, błędy urządzenia itd.

- Są to podstawowe, cyklicznie wysyłane dane telemetryczne (co sekundę).

Wed May 21 2025 18:48:22 GMT+0200 (czas środkowoeuropejski letni):

```
{
  "body": {
    "deviceId": "Device 1",
    "timestamp": "2025-05-21T16:48:24.0551795Z",
    "productionStatus": "1",
    "workorderId": "9820461b-94a8-401f-9186-6791c3e198ac",
    "productionRate": 70,
    "goodCount": 3744,
    "badCount": 410,
    "temperature": 68.38234388079064,
    "deviceErrors": 0
  },
  "enqueuedTime": "Wed May 21 2025 18:48:22 GMT+0200 (czas środkowoeuropejski letni)"
}
```

Rysunek 3. Przykładowa wiadomość D2C.

Dane telemetryczne są również zapisywane jako pliki JSON w kontenerze „telemetrydata” na Azure Blob Storage, co pozwala na ich archiwizację.

telemetrydata > Device 1

Metoda uwierzytelniania: Klucz dostępu (przełącz na konto użytkownika rozwiązania Microsoft Entra)

Wyszukaj obiekty blob według prefiksu (z uwzględnieniem wielkości liter) Pokaż tylko aktywne obiekty blob

Pokazywanie wszystkich 93 elementów

<input type="checkbox"/> Nazwa	Ostatnio zmodyfikowano	Warstwa dostępu	Typ obiektu bl...	Rozmiar	Stan dzierzawy
<input type="checkbox"/> [-]					...
<input type="checkbox"/> 20250522_100129.json	22.05.2025, 12:01:28	Gorąca (sugerowana)	Blokowy obiekt...	244 B	Dostępna
<input type="checkbox"/> 20250522_100141.json	22.05.2025, 12:01:39	Gorąca (sugerowana)	Blokowy obiekt...	243 B	Dostępna
<input type="checkbox"/> 20250522_100152.json	22.05.2025, 12:01:50	Gorąca (sugerowana)	Blokowy obiekt...	243 B	Dostępna
<input type="checkbox"/> 20250522_100203.json	22.05.2025, 12:02:01	Gorąca (sugerowana)	Blokowy obiekt...	243 B	Dostępna
<input type="checkbox"/> 20250522_100213.json	22.05.2025, 12:02:11	Gorąca (sugerowana)	Blokowy obiekt...	243 B	Dostępna
<input type="checkbox"/> 20250522_100224.json	22.05.2025, 12:02:22	Gorąca (sugerowana)	Blokowy obiekt...	243 B	Dostępna
<input type="checkbox"/> 20250522_100235.json	22.05.2025, 12:02:33	Gorąca (sugerowana)	Blokowy obiekt...	243 B	Dostępna

Rysunek 4. Folder zawierający zapisywane dane telemetryczne.

Edytuj ×

Blob: Device 1/20250522_100129.json

```
1 {"deviceId":"Device 1","timestamp":"2025-05-22T10:01:29.5787488Z","productionStatus":"1","workorderId":"316891d6-f957-4976-a59b-d094"
```

Rysunek 5. Przykładowy plik zawierający dane telemetryczne w formacie JSON.

Zdarzenia zmian błędów urządzenia (device error events)

- Wysyłane tylko gdy następuje zmiana stanu błędów urządzenia.
- Zawierają informacje o nowym statusie błędów i opis błędów

Thu May 22 2025 10:46:18 GMT+0200 (czas środkowoeuropejski letni):

```
{
  "body": {
    "event": "DeviceErrorsChanged",
    "deviceErrors": 2,
    "errorMessages": "Power Failure"
  },
  "enqueuedTime": "Thu May 22 2025 10:46:18 GMT+0200 (czas środkowoeuropejski letni)"
}
```

Rysunek 6. Przykładowa wiadomość D2C eventowo zgłaszająca pojawienie się błędu podczas pracy urządzenia.

Aplikacja informuje nas o poprawnym wystaniu danych poprzez wyświetlenie komunikatów.

```
Dane telemetryczne zostały wysłane do IoT Hub.
Zaktualizowano Device Twin (reported) – ProductionRate.
Wysłano dane telemetryczne o 22.05.2025 11:05:24
Dane telemetryczne zapisane do bloba: Device 1/20250522_110524.json
```

Rysunek 7. Komunikat informujący o wysłaniu danych telemetrycznych do IoT Huba, oraz zapisaniu ich w Blobach.

```
Błędy urządzenia zmieniono – wysłano zdarzenie.
```

Rysunek 8. Komunikat informujący o wysłaniu zdarzenia do IoT Huba podczas pojawienia się błędu na urządzeniu.

Rodzaj i format danych przechowywanych w Device Twin

Reported properties (stan raportowany przez urządzenie):

- deviceErrors — aktualny stan błędów urządzenia, kodowany jako liczba całkowita (bitowa flaga błędów). Każda zmiana błędów (deviceErrors) jest od razu aktualizowana w reported properties i wysyłana do IoT Hub.
- productionRate — aktualna, odczytana z urządzenia wartość prędkości produkcji (liczba całkowita).

Desired properties (właściwości oczekiwane, ustawiane z chmury):

- productionRate — wartość produkcji, którą chmura chce ustawić w urządzeniu.

Każda zmiana Device Twin jest odpowiednio komunikowana przez aplikację poprzez wyświetlenie odpowiedniej wiadomości.

```
Zaktualizowano Device Twin (reported) – DeviceError.
Zaktualizowano Device Twin (reported) – ProductionRate.
```

Rysunek 9. Komunikat dotyczący zaktualizowania wartości Device Twin (Reported).


```

"properties": {
  "desired": {
    "productionRate": 70,
    "$metadata": {
      "$lastUpdated": "2025-05-21T18:56:11.4939639Z",
      "$lastUpdatedVersion": 46,
      "productionRate": {
        "$lastUpdated": "2025-05-21T18:56:11.4939639Z",
        "$lastUpdatedVersion": 46
      }
    },
    "$version": 46
  },
  "reported": {
    "deviceErrors": 2,
    "productionRate": 70,
    "$metadata": {
      "$lastUpdated": "2025-05-22T10:30:54.0376235Z",
      "deviceErrors": {
        "$lastUpdated": "2025-05-22T10:29:59.4443309Z"
      },
      "productionRate": {
        "$lastUpdated": "2025-05-22T10:30:54.0376235Z"
      }
    },
    "$version": 4029
  }
},
}

```

Rysunek 10. Przykładowy podgląd w strukturę Device Twin.

Direct Methods zaimplementowane w agencie

Metoda EmergencyStop

Opis:

Ta metoda zatrzymuje produkcję. Po wywołaniu powoduje zmianę stanu na serwerze OPC UA, ustawiając status produkcji na „Zatrzymano”.

Sygnatura:

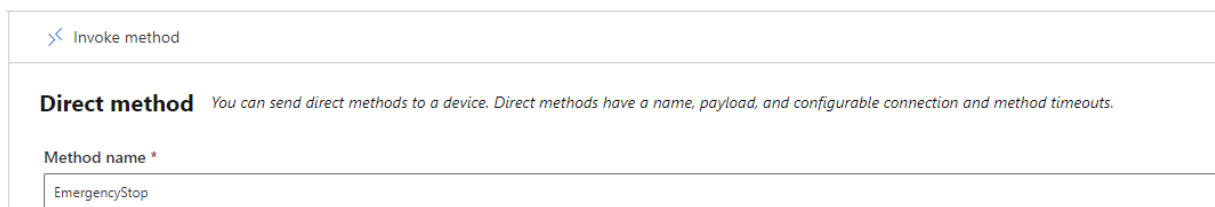
- Nazwa metody: *EmergencyStop*
- Parametry: brak (metoda nie wymaga parametrów)
- Zwracany rezultat:
 1. Maszyna zatrzymuje się jeżeli wszystko przebiegło pomyślnie podczas wywoływania metody i uzyskujemy komunikat o poprawnym działaniu.
 2. Podczas próby wywołania wystąpił błąd, który zostanie wyświetlony w aplikacji. Maszyna nie zatrzyma swojego działania.

Sposoby wywołania:

1. Wywołanie z poziomu Azure IoT Hub (np. Azure IoT Explorer)

- W narzędziu Azure IoT Explorer, wybieramy swoje urządzenie, a następnie:
 - ❖ Przechodzimy do zakładki „Direct Methods” (Metody bezpośrednie),
 - ❖ Wpisujemy nazwę metody: EmergencyStop,
 - ❖ Klikamy „Invoke”.
- System wyśle żądanie do urządzenia, które wykona zatrzymanie awaryjne.

2. Metoda ta może zostać wywołana przez funkcję aplikacji w momencie wykrycia 3 błędów na urządzeniu w przeciągu minuty. (Opisane w rozdziale: „Kalkulacje i logika biznesowa”)



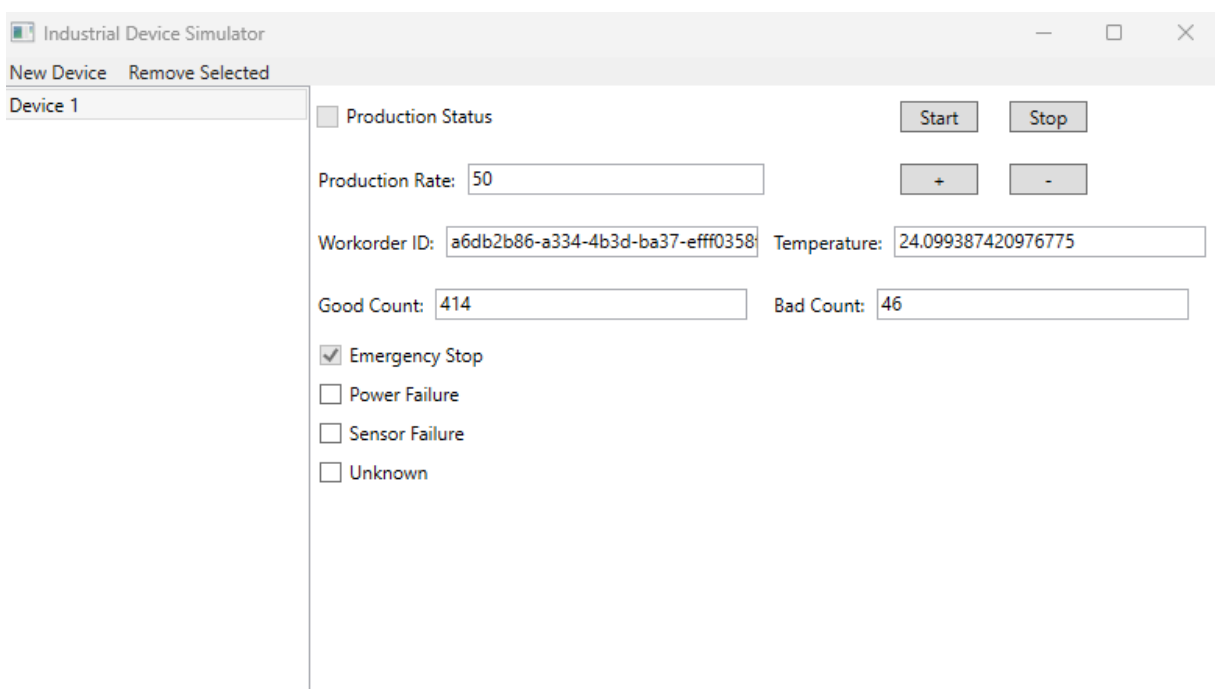
Invoke method

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

Method name *

EmergencyStop

Rysunek 11. Sposób wywołania metody EmergencyStop w IoT Explorer.



Industrial Device Simulator

New Device Remove Selected

Device 1

☐ Production Status Start Stop

Production Rate: 50 + -

Workorder ID: a6db2b86-a334-4b3d-ba37-efff0358 Temperature: 24.099387420976775

Good Count: 414 Bad Count: 46

☒ Emergency Stop

☐ Power Failure

☐ Sensor Failure

☐ Unknown

Rysunek 12. Urządzenie po wywołaniu metody bezpośredniej EmergencyStop.

**Metoda EmergencyStop została wywołana.
Status produkcji ustawiony na "Zatrzymano" na serwerze OPC UA.**

Rysunek 13. Komunikat po wywołaniu metody bezpośredniej EmergencyStop.

Metoda ResetErrorStatus

Opis:

Ta metoda resetuje status błędów urządzenia, ustawiając deviceErrors na None (brak błędów).

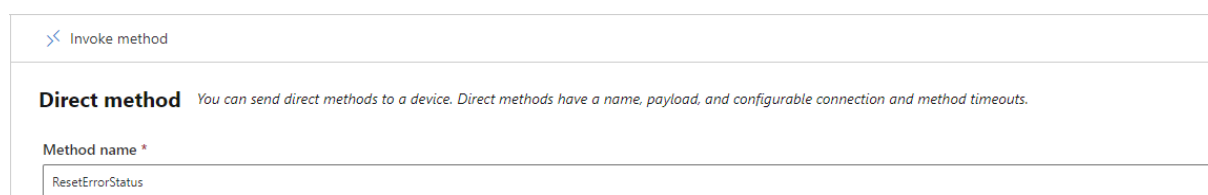
Sygnatura:

- Nazwa metody: *ResetErrorStatus*
- Parametry: brak (metoda nie wymaga parametrów)
- Zwracany rezultat:
 1. Jeśli wszystko przebiegło pomyślnie, urządzenie (maszyna) zresetuje swój stan błędu i otrzymamy potwierdzenie o poprawnym wykonaniu operacji. Komunikat zwrotny będzie wskazywał sukces.
 2. Jeżeli podczas próby wywołania metody wystąpił problem (np. urządzenie nie odpowiada, brak połączenia z OPC UA), operacja się nie powiedzie. Aplikacja pokaże odpowiedni komunikat błędu, a stan błędu na maszynie nie zostanie zresetowany.

Sposoby wywołania:

1. Wywołanie z poziomu Azure IoT Hub (np. Azure IoT Explorer)

- W narzędziu Azure IoT Explorer, wybieramy swoje urządzenie, a następnie:
 - ❖ Przechodzimy do zakładki „Direct Methods” (Metody bezpośrednie),
 - ❖ Wpisujemy nazwę metody: *ResetErrorStatus*,
 - ❖ Klikamy „Invoke”.
- System wyśle żądanie do urządzenia, które wykona zresetowanie wszystkich błędów na urządzeniu.



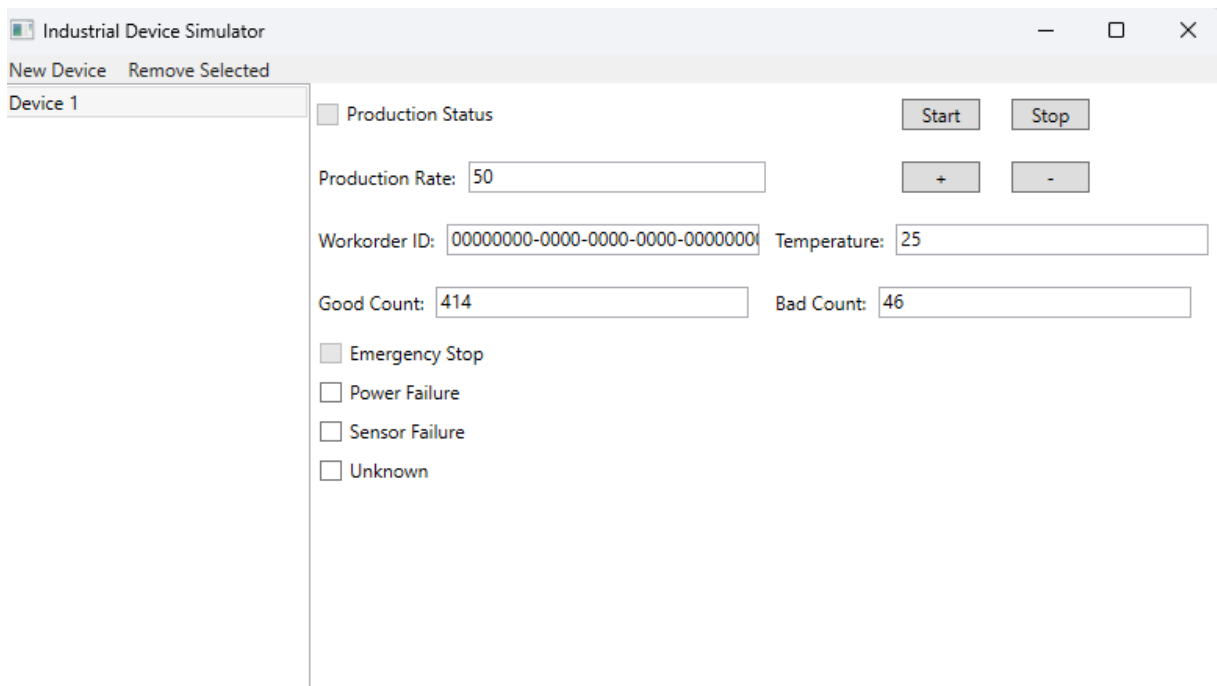
> Invoke method

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

Method name *

ResetErrorStatus

Rysunek 14. Sposób wywołania metody *ResetErrorStatus* w IoT Explorer



Rysunek 15. Urządzenie po wywołaniu metody bezpośredniej `ResetErrorStatus`.

```
Metoda ResetErrorStatus została wywołana.  
Device Error zostało usatwione na: None
```

Rysunek 16. Komunikat po wywołaniu metody bezpośredniej `ResetErrorStatus`.

Kalkulacje i logika biznesowa

Kalkulacje

Production KPIs

Opis kalkulacji:

Monitorowanie jakości produkcji w czasie rzeczywistym. KPI oblicza procent dobrych produktów względem całkowitej liczby wyprodukowanych jednostek (dobre + wadliwe), w 5-minutowych oknach czasowych, dla każdego urządzenia osobno.

Źródło danych:

Dane telemetryczne wysyłane przez urządzenia do Azure IoT Hub zawierają wartości:

- `goodCount` – liczba poprawnych produktów,
- `badCount` – liczba wadliwych produktów,
- `deviceId` – identyfikator urządzenia,
- `timestamp` – czas rejestracji.

Implementacja w Azure Stream Analytics:

W Azure Stream Analytics stworzono zapytanie przetwarzające strumień danych z IoT Hub oraz użyto:

- Okien czasowych typu Hopping Window co 5 minut (lub Tumbling, zależnie od konfiguracji),
- Obliczenia procentu dobrej produkcji.

Dane wyjściowe:

- Wykorzystywane przez funkcję aplikacji do podejmowania decyzji, takich jak automatyczna zmiana tempa produkcji.
- Zapisywane do kontenera *kpi-production* w Azure Blob Storage, co umożliwia ich dalszą analizę historyczną – format JSON.

Podgląd danych wejściowych Wyniki testu Symulacja zadania (wersja zapoznawcza)

ⓘ Zasady czasu nie mają zastosowania do testowania zapytań. Wyniki widoczne w tym miejscu mogą się różnić od rzeczywistych wyników zadania. [Dowiedz się więcej o naszych narzędziach do tworzenia tutaj.](#)

↓ Pobierz wyniki

deviceId string	windowEnd datetime	goodProductionPercent float
"Device 1"	"2025-05-22T10:30:00.0000000Z"	91.14308342133052
"Device 1"	"2025-05-22T10:35:00.0000000Z"	91.17678212053808
"Device 1"	"2025-05-22T11:10:00.0000000Z"	89.4731296101159

Rysunek 17. Wynik testu w Stream Analytics.

Wygląd

Blob: 0_4543587cf1a24abc96af7e00719e9fff_1.json

📄 Zapisz ✕ Odrzuć ↓ Pobierz ↺ Odśwież 🗑 Usun

1	{ "deviceId": "Device 1", "windowEnd": "2025-05-21T18:33:00.0000000Z", "goodProductionPercent": 67.2566371681416 }
2	{ "deviceId": "Device 1", "windowEnd": "2025-05-21T18:34:00.0000000Z", "goodProductionPercent": 67.3076923076923 }
3	{ "deviceId": "Device 1", "windowEnd": "2025-05-21T18:35:00.0000000Z", "goodProductionPercent": 67.3076923076923 }
4	{ "deviceId": "Device 1", "windowEnd": "2025-05-21T18:36:00.0000000Z", "goodProductionPercent": 77.49171270718232 }
5	{ "deviceId": "Device 1", "windowEnd": "2025-05-21T18:37:00.0000000Z", "goodProductionPercent": 83.64499563995737 }

Rysunek 18. Przykładowy plik JSON zawierające dane wyliczone ze Stream Analytics.

Temperature

Opis kalkulacji:

Monitorowanie temperatury urządzeń w czasie rzeczywistym polega na tym, że co minutę obliczana jest średnia (AVG), minimalna (MIN) i maksymalna (MAX) temperatura z ostatnich pięciu minut, osobno dla każdego urządzenia.

Źródło danych:

Dane telemetryczne wysyłane przez urządzenia do Azure IoT Hub zawierają:

- temperature – aktualna temperatura urządzenia,
- deviceId – identyfikator urządzenia,
- timestamp – czas pomiaru.

Implementacja w Azure Stream Analytics:

W Azure Stream Analytics wykorzystano Sliding Window (okno przesuujące się co minutę, obejmujące 5 minut), aby analizować temperaturę.

Dane wyjściowe:

- Zapisywane do kontenera *datatemperature* w Azure Blob Storage, co umożliwia ich dalszą analizę historyczną – format JSON.

Podgląd danych wejściowych Wyniki testu Symulacja zadania (wersja zapoznawcza)

ⓘ Zasady czasu nie mają zastosowania do testowania zapytań. Wyniki widoczne w tym miejscu mogą się różnić od rzeczywistych wyników zadania. [Dowiedz się więcej o naszych narzędziach do tworzenia tutaj.](#)

↓ Pobierz wyniki

deviceId string	windowEnd datetime	avgTemperature float	minTemperature float	maxTemperature float
"Device 1"	"2025-05-22T10:27:00.0000000Z"	66.82290390147469	62.134673321873166	77.0286989932604
"Device 1"	"2025-05-22T10:28:00.0000000Z"	68.54194463462164	62.134673321873166	79.82677582518937
"Device 1"	"2025-05-22T10:29:00.0000000Z"	68.54194463462164	62.134673321873166	79.82677582518937
"Device 1"	"2025-05-22T10:30:00.0000000Z"	68.54194463462164	62.134673321873166	79.82677582518937
"Device 1"	"2025-05-22T10:31:00.0000000Z"	71.89633326369533	62.134673321873166	88.86389174372607

Rysunek 19. Wynik testu w Stream Analytics.

```
1 {"@25-05-22T12:01:00.0000000Z","avgTemperature":-385.0,"minTemperature":-385.0,"maxTemperature":-385.0}
2 {"@25-05-22T12:02:00.0000000Z","avgTemperature":105.95833333333333,"minTemperature":-650.0,"maxTemperature":953.0}
3 {"@25-05-22T12:03:00.0000000Z","avgTemperature":54.89212983051097,"minTemperature":-792.0,"maxTemperature":953.0}
4 {"@25-05-22T12:04:00.0000000Z","avgTemperature":44.32763555324535,"minTemperature":-792.0,"maxTemperature":953.0}
5 {"@25-05-22T12:05:00.0000000Z","avgTemperature":44.32763555324535,"minTemperature":-792.0,"maxTemperature":953.0}
6 {"@25-05-22T12:06:00.0000000Z","avgTemperature":49.151541570697546,"minTemperature":-792.0,"maxTemperature":953.0}
7 {"@25-05-22T12:07:00.0000000Z","avgTemperature":21.916472724122453,"minTemperature":-792.0,"maxTemperature":873.0}
8 {"@25-05-22T12:08:00.0000000Z","avgTemperature":25.179489675701447,"minTemperature":24.073014463699533,"maxTemperature":25.98981912305}
```

Rysunek 20. Przykładowy plik JSON zawierające dane wyliczone ze Stream Analytics.

Device errors

Opis kalkulacji:

Celem tej analizy jest bieżące monitorowanie błędów występujących na urządzeniach. System ma wykrywać sytuacje, w których dane urządzenie zgłasza więcej niż 3 błędy w czasie krótszym niż 1 minuta.

Źródło danych:

Dane pochodzą z telemetrycznych wiadomości D2C wysyłanych przez urządzenie do Azure IoT Hub. W każdej wiadomości znajduje się wartość pola *DeviceError*, które wskazuje na wystąpienie błędu.

Implementacja:

- Cała logika została zaimplementowana z użyciem Azure Stream Analytics, które odbiera dane telemetryczne z IoT Hub i analizuje je w oknach czasowych, szukając zbyt dużej liczby błędów w danym czasie.

- Dla każdego urządzenia liczone jest, ile razy w danym oknie wystąpiła wartość DeviceError różne od 0.

Dane wyjściowe:

- Wykorzystywane przez funkcję aplikacji do podejmowania decyzji, takich jak wywołanie metody *EmergencyStop*.
- Zapisywane do kontenera *alerts-error* w Azure Blob Storage, co umożliwia ich dalszą analizę historyczną – format JSON.

Podgląd danych wejściowych Wyniki testu Symulacja zadania (wersja zapoznawcza)

ⓘ Zasady czasu nie mają zastosowania do testowania zapytań. Wyniki widoczne w tym miejscu mogą się różnić od rzeczywistych wyników zadania. [Dowiedz się więcej o naszych narzędziach do tworzenia tutaj.](#)

↓ Pobierz wyniki

deviceId string	alertTime datetime	errorCount bigint
"Device 1"	"2025-05-22T11:07:00.0000000Z"	8

Rysunek 21. Wynik testu w Stream Analytics.

```

1 [{"deviceId":"Device 1","alertTime":"2025-05-22T12:02:00.0000000Z","errorCount":23}]
2 [{"deviceId":"Device 1","alertTime":"2025-05-22T12:03:00.0000000Z","errorCount":34}]
3 [{"deviceId":"Device 1","alertTime":"2025-05-22T12:04:00.0000000Z","errorCount":32}]

```

Rysunek 22. Przykładowy plik JSON zawierające dane wyliczone ze Stream Analytics.

Logika biznesowa

Automatyczne zatrzymanie awaryjne przy wielu błędach urządzenia

Opis logiki działania:

System monitoruje liczbę błędów zgłaszanych przez każde urządzenie. Jeśli w ciągu 1 minuty dane telemetryczne wykazują, że urządzenie zgłosiło więcej niż 3 błędy, uznaje się to za sytuację awaryjną. Jeżeli występuje taka sytuacja system:

- natychmiast wywołuje metodę bezpośrednią *EmergencyStop* na urządzeniu,
- urządzenie zostaje zatrzymane (np. linia produkcyjna przerywa pracę),
- akcja zostaje zalogowana i może zostać przeanalizowana później.

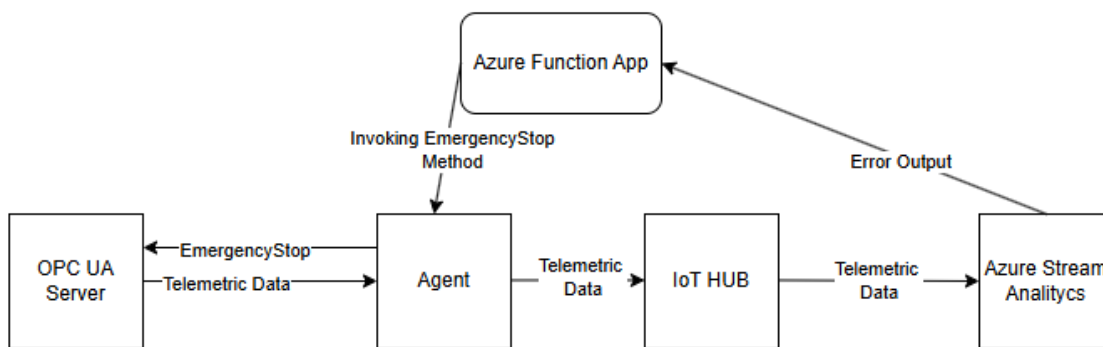
Implementacja techniczna:

1. Analiza danych telemetrycznych w czasie rzeczywistym:
 - Z urządzeń zbierana jest telemetria, w tym pole *DeviceError*, które przyjmuje wartość 1 w przypadku wystąpienia błędu.
 - Dane są przesyłane do Azure IoT Hub, a następnie przetwarzane w Azure Stream Analytics (ASA).
 - Dane są przesyłane za pomocą outputu do Azure Function.
2. Wywołanie metody *EmergencyStop* przez Azure Function:

- Strumień wynikowy z ASA (emergencyStopOutput) jest przekierowany jako output do funkcji w Azure Functions.
- Funkcja ta odbiera dane i wykonuje wywołanie metody EmergencyStop dla odpowiedniego urządzenia.

Działanie funkcji aplikacji:

- Pobiera nazwę urządzenia z danych wejściowych.
- Tworzy payload i używa IoT Hub Service SDK do wystąpienia żądania typu Direct Method.
- Wywołuje metodę EmergencyStop.
- Oczekuje na odpowiedź:
 - Jeżeli odpowiedź jest pozytywna (status == 200), funkcja loguje sukces.
 - Jeżeli wystąpił błąd (status != 200), funkcja może logować błąd.



Rysunek 23. Schemat działania danej logiki biznesowej.

Dienniki usługi App Insights ▾ Poziom dziennika ▾ □ Zatrzymaj Kopiuj Wyczyść Otwórz w metrykach na żywo Wyślij do nas swoją opinię

```

Połączono!
2025-05-22T12:02:12Z [Information] Executing 'Functions.EmergencyStopFunction' (Reason='This function was programmatically called via the host APIs.', Id=a9e6403a-4007-4edf-888d-7be0052248df)
2025-05-22T12:02:12Z [Information] EmergencyStopFunction triggered
2025-05-22T12:02:13Z [Information] EmergencyStop sent to Production01, status: 200
2025-05-22T12:02:13Z [Information] Executed 'Functions.EmergencyStopFunction' (Succeeded, Id=a9e6403a-4007-4edf-888d-7be0052248df, Duration=1214ms)
  
```

Rysunek 24. Komunikat w Azure Function Insight potwierdzający wykonanie funkcji.

**Metoda EmergencyStop została wywołana.
Status produkcji ustawiony na "Zatrzymano" na serwerze OPC UA.**

Rysunek 25. Komunikat potwierdzający działanie danej funkcji.

Automatyczna redukcja Desired Production Rate przy spadku efektywności produkcji

Opis logiki działania:

System monitoruje procent dobrych produktów w całkowitej produkcji (good production rate) dla każdego urządzenia w minutowych oknach czasowych. Jeżeli procent dobrych produktów spadnie poniżej 90%, system automatycznie zmniejsza wartość Desired Production Rate o 10 punktów. W momencie pojawienia się takiej sytuacji system:

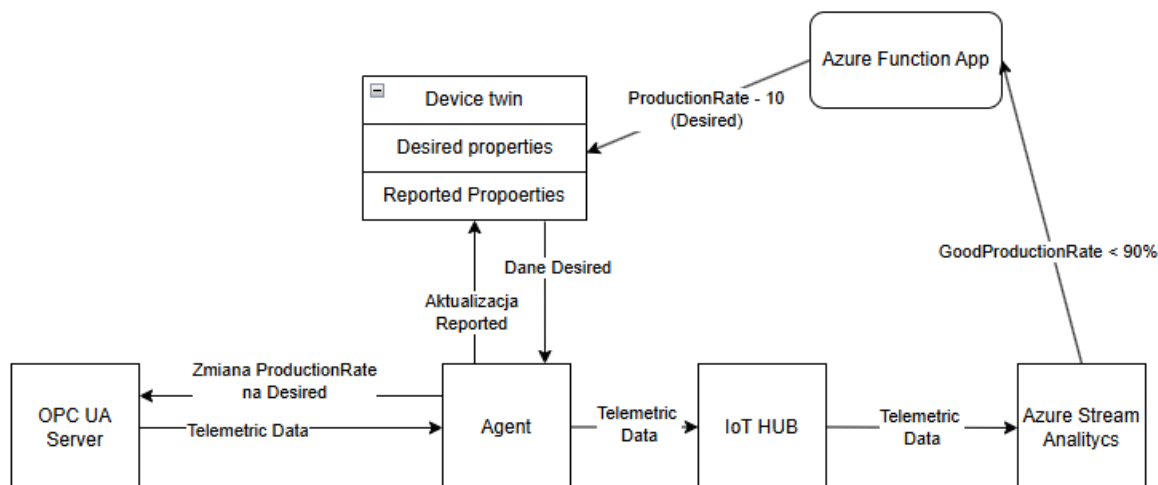
- Zmienia wartość Desired Production Rate w Twin Device (desired property).
- Zmiana jest synchronizowana z urządzeniem, które powinno dostosować swoją produkcję.
- Akcja jest logowana i może być monitorowana.
-

Implementacja techniczna:

1. Analiza danych telemetrycznych w czasie rzeczywistym:
 - Dane o liczbie dobrych produktów (GoodCount) i łącznej produkcji (GoodCount + BadCount) są przesyłane do Azure IoT Hub.
 - Azure Stream Analytics (ASA) analizuje dane w minutowych oknach, obliczając procent dobrych produktów dla każdego urządzenia.
 - ASA filtruje urządzenia, u których ten procent jest poniżej 90%.
 - Wynik zapytania jest wysyłany jako output do Azure Function.
2. Aktualizacja Desired Production Rate poprzez Azure Function:
 - Funkcja odbiera dane o urządzeniu z ASA (np. deviceId i aktualny procent dobrych produktów).
 - Funkcja pobiera aktualną wartość Desired Production Rate z Twin Device (desired properties).
 - Funkcja zmniejsza Desired Production Rate o 10 punktów (z zachowaniem minimalnych limitów, jeśli są ustalone).
 - Funkcja aktualizuje Desired Property w Device Twin za pomocą Azure IoT Hub Service SDK.
 - Zmiana Desired Production Rate jest automatycznie przekazywana do urządzenia i skutkuje dostosowaniem produkcji.

Działanie funkcji aplikacji:

- Pobiera nazwę urządzenia oraz bieżący procent dobrej produkcji z danych wejściowych.
- Pobiera aktualną wartość Desired Production Rate z Device Twin.
- Oblicza nową wartość Desired Production Rate, odejmując 10 punktów.
- Wysyła aktualizację Desired Property do IoT Hub (Device Twin).
- Loguje sukces aktualizacji lub ewentualne błędy.



Rysunek 26 . Schemat działania danej logiki biznesowej

[Dzienniki usługi App Insights](#)
[Poziom dziennika](#)
[Zatrzymaj](#)
[Kopiuj](#)
[Wyczyść](#)
[Otwórz w metrykach na żywo](#)
[Wyślij do nas swoją opinię](#)

```

Połączono!
2025-05-22T13:01:10Z [Information] Executing 'Functions.DecreaseProductionRateFunction' (Reason:'This function was programmatically called via the host APIs.', Id=0703b6c8-b88b-4dd4-9cfa-b32ebd9552b)
2025-05-22T13:01:11Z [Information] DecreaseProductionRateFunction triggered
2025-05-22T13:01:11Z [Information] Twin for device Production01:
Reported: {"$metadata":{"$lastUpdated":"2025-05-22T13:01:09.7816404Z"},"deviceErrors":{"$lastUpdated":"2025-05-22T13:01:09.7816404Z"},"version":4000,"deviceErrors":{"$lastUpdated":"2025-05-22T13:01:09.7816404Z"},"productionRate":{"$lastUpdated":"2025-05-22T13:01:09.7816404Z"},"productionRate":70}
2025-05-22T13:01:11Z [Information] Updated desired productionRate to 60 for device Production01
2025-05-22T13:01:11Z [Information] Executed 'Functions.DecreaseProductionRateFunction' (Succeeded, Id=0703b6c8-b88b-4dd4-9cfa-b32ebd9552b, Duration=990ms)
  
```

Rysunek 27. Komunikat w Azure Function Insight potwierdzający wykonanie funkcji.

```

Odebrano desired productionRate: 60
Wartość 60 została zapisana do węzła ns=2;s=Device 1/ProductionRate
Dane telemetryczne zostały wysłane do IoT Hub.
Zaktualizowano reported productionRate: 60
  
```

Rysunek 28. Komunikat potwierdzający działanie danej funkcji.

Konkluzje

Projekt wykorzystuje mechanizmy Azure IoT Hub oraz Device Twin do efektywnego zarządzania i monitorowania urządzeń przemysłowych w czasie rzeczywistym. Dzięki dwukierunkowej synchronizacji Desired i Reported Properties, system umożliwia zdalne sterowanie parametrami urządzeń, takimi jak productionRate, oraz automatyczną reakcję na zdarzenia, np. obniżenie produkcji w przypadku spadku jakości lub zatrzymanie awaryjne przy wykryciu błędów.

Dodatkowo, rozwiązanie jest skalowalne i elastyczne — w przyszłości można je łatwo rozszerzyć o funkcje powiadomień e-mail, które będą informować odpowiednie osoby o krytycznych zdarzeniach oraz o obsłudze większej liczby urządzeń, co pozwoli na zarządzanie całym parkiem maszynowym w sposób zautomatyzowany i zdalny.