

Industrial IoT Projekt - Azure

Informacje podstawowe

Autor: System Administrator & DevOps

Projekt: Industrial IoT Production Monitoring System

Technologie: C#, .NET 6, Azure IoT Hub, Azure Functions, Stream Analytics

Repozytorium

[https://github.com/\[username\]/IndustrialIoT-Azure-Project](https://github.com/[username]/IndustrialIoT-Azure-Project)

Struktura projektu

```
IndustrialIoT-Solution/
├── IndustrialIoT.Agent/           # Console Application (OPC UA to IoT Hub)
│   ├── Program.cs
│   ├── IoTDevice.cs
│   ├── OpcDevice.cs
│   └── appsettings.json
├── IndustrialIoT.Functions/      # Azure Functions (Business Logic)
│   ├── BusinessLogicFunctions.cs
│   ├── AzureIoTService.cs
│   ├── Models.cs
│   └── Startup.cs
└── StreamAnalytics/             # ASA Queries
    ├── ProductionKPI.sql
    ├── TemperatureStats.sql
    └── DeviceErrors.sql
```

Agent Description

Agent jest aplikacją konsolową (.NET 6) uruchamianą on-premise w fabryce, która:

- Łączy się z OPC UA serverem (IloTSim) żeby odczytywać dane z urządzeń produkcyjnych
- Używa Azure SDK do komunikacji z Azure IoT Hub (dwukierunkowej)
- Wspiera monitorowanie wielu urządzeń poprzez konfigurację w `appsettings.json`
- Automatycznie wykrywa dostępne urządzenia na OPC UA serverze

Uruchomienie Agent:

```
bash
```

```
dotnet run --project IndustrialIoT.Agent
```

```
# Agent automatycznie połączy się z Device ID skonfigurowanym w appsettings.json
```

Funkcjonalności

Telemetry (D2C Messages)






Agent wysyła co 1 sekundę dane telemetryczne do IoT Hub:

Przykład wiadomości telemetrycznej:

```
json
```

```
{
  "deviceId": 1,
  "productionStatus": "Running",
  "workorderId": "550e8400-e29b-41d4-a716-446655440000",
  "goodCount": 150,
  "badCount": 5,
  "temperature": 23.5,
  "timestamp": "2024-01-15T10:30:00.000Z"
}
```

Dane wysyłane:

-  Production Status (Running/Stopped)
-  Workorder ID (GUID - puste gdy stopped)
-  Good Count (liczba dobrych produktów)
-  Bad Count (liczba złych produktów)
-  Temperature (°C - zawsze mierzone)

Direct Methods

Agent obsługuje następujące Direct Methods wywoływane z Azure:

EmergencyStop

- Zatrzymuje produkcję natychmiast
- Aktywuje flagę błędu Emergency Stop
- Zwraca status wykonania operacji

csharp

```
await _azureIoTService.TriggerEmergencyStopAsync(deviceId);
```

ResetErrorStatus

- Resetuje wszystkie flagi błędów
- Czyści stan Device Error na OPC UA
- Przygotowuje urządzenie do wznowienia pracy

csharp

```
await _azureIoTService.ResetErrorStatusAsync(deviceId);
```

Device Twin (Stan urządzenia)

Agent synchronizuje stan urządzenia z Device Twin:

Reported Properties (stan aktualny):

json





```
{  
  "productionRate": 85,  
  "deviceErrors": "PowerFailure, SensorFailure",  
  "lastUpdated": "2024-01-15T10:30:00.000Z"  
}
```

Desired Properties (stan pożądany):

json

```
{  
  "productionRate": 75 // Nowa wartość ustawiana przez Logikę biznesową  
}
```

Agent automatycznie:

-  Reportuje aktualne Production Rate i Device Errors
-  Reaguje na zmiany Desired Properties
-  Stosuje nowe Production Rate na OPC UA device
-  Wysyła Device Error events przy zmianie stanu błędów

Device Errors (flagi binarne):

- None = 0 (0000)
- Emergency Stop = 1 (0001)
- Power Failure = 2 (0010)
- Sensor Failure = 4 (0100)
- Unknown = 8 (1000)

Data Calculations (Stream Analytics)

Kalkulacje danych wykonywane w Azure Stream Analytics:

1. Production KPIs

Zapytanie ASA:

sql

```
SELECT
    deviceId,
    (SUM(goodCount) * 100.0) / (SUM(goodCount) + SUM(badCount)) as goodProductionPercentage,
    SUM(goodCount) as totalGoodCount,
    SUM(badCount) as totalBadCount,
    System.Timestamp() as windowEnd
INTO kpi-output
FROM iot-hub-input
WHERE messageType = 'telemetry'
GROUP BY deviceId, TumblingWindow(minute, 5)
```

2. Temperature Statistics

Zapytanie ASA:

sql

```
SELECT
    deviceId,
    AVG(temperature) as avgTemperature,
    MIN(temperature) as minTemperature,
    MAX(temperature) as maxTemperature,
    System.Timestamp() as windowEnd
INTO temperature-output
FROM iot-hub-input
WHERE messageType = 'telemetry'
GROUP BY deviceId, TumblingWindow(minute, 1)
```

3. Device Error Counting

Zapytanie ASA:

```
sql

SELECT
    deviceId,
    COUNT(*) as errorCount,
    System.Timestamp() as windowEnd
INTO error-output
FROM iot-hub-input
WHERE messageType = 'error' OR errorType = 'deviceError'
GROUP BY deviceId, TumblingWindow(minute, 1)
HAVING COUNT(*) > 3
```

Business Logic (Azure Functions)

Zaimplementowane w Azure Functions z automatycznym trigger:

Reguła 1: Emergency Stop przy >3 błędach/minutę

```
csharp

// Filtrujemy Emergency Stop aby nie wywoływać kolejnego Emergency Stop
var countableErrors = errors & ~DeviceErrors.EmergencyStop;

if (errorCount > 3) {
    log.LogError($"🚨 EMERGENCY STOP: Device {deviceId} has {errorCount} errors!");
    await _azureIoTService.TriggerEmergencyStopAsync(deviceId);
}
```

Reguła 2: Obniżenie Production Rate gdy KPI < 90%

```
csharp

if (kpi.GoodProductionPercentage < 90.0) {
    log.LogWarning($"📉 Decreasing production rate for device {deviceId}");
    await _azureIoTService.DecreaseProductionRateAsync(deviceId);
}
```

Reguła 3: Email Alert przy każdym błędzie

```
csharp

// Wszystkie błędy (włącznie z Emergency Stop) generują email alert
log.LogWarning($"📧 EMAIL ALERT: Device {deviceId} has errors: {errors}");
```

Instrukcja uruchomienia

1. Wymagania wstępne

- .NET 6 SDK
- Azure Subscription
- IloTSim.Desktop (simulator OPC UA)

2. Azure Resources Setup

bash

IoT Hub

```
az iot hub create --name MyIoTHub --resource-group MyRG
```

Storage Account

```
az storage account create --name mystorageaccount --resource-group MyRG
```

Function App

```
az functionapp create --name MyFunctionApp --resource-group MyRG
```

Stream Analytics Job

```
az stream-analytics job create --name MyStreamJob --resource-group MyRG
```

3. Konfiguracja Agent

appsettings.json:

json

```
{
  "ConnectionStrings": {
    "AzureIoTHub": "HostName=MyIoTHub.azure-devices.net;DeviceId=Agent1;SharedAccessKey=...",
    "OpcServer": "opc.tcp://localhost:4840/"
  },
  "DeviceSettings": {
    "DeviceId": 1,
    "TelemetryIntervalMs": 1000,
    "EnableDebugLogging": true
  }
}
```

4. Deployment

```
bash
```

```
# Deploy Agent (on-premise)
```

```
dotnet publish IndustrialIoT.Agent -c Release
```

```
# Uruchom na maszynie w fabryce
```

```
# Deploy Functions
```

```
func azure functionapp publish MyFunctionApp
```

```
# Configure Stream Analytics Queries
```

```
# Import queries z plików *.sql do ASA Job w Azure Portal
```

Screenshoty działania

Agent Console Output:

```
=== Industrial IoT Agent Starting ===
```

```
✓ Agent connected to Device 1
```

```
✓ OPC UA Server: opc.tcp://localhost:4840/
```

```
✓ Azure IoT Hub connected
```

```
Press 'q' to quit...
```

```
[10:30:15] Telemetry sent - Status: Running, Temp: 23.5°C
```

```
[10:30:16] Device twin updated - Rate: 85%, Errors: None
```

```
[10:30:17] Telemetry sent - Status: Running, Temp: 23.7°C
```

Azure Functions Logs:

```
📬 Received message: {"deviceId":1,"productionStatus":"Running"...}
```

```
📊 Processed telemetry for device 1 - Status: Running, Temp: 23.5°C
```

```
📈 Processing KPI data from kpi-2024-01-15-10-30.json
```

```
📊 Device 1 KPI: 96.8% efficiency
```

```
✓ Device 1 efficiency is acceptable: 96.8%
```

Business Logic w akcji:

```
⚠️ Processing error aggregation data from errors-2024-01-15-10-35.json
```

```
📊 Device 1 countable error count in last minute: 4
```

```
🚨 EMERGENCY STOP TRIGGERED: Device 1 has 4 countable errors in last minute!
```

```
📧 URGENT EMAIL: Emergency stop triggered for device 1 due to excessive errors
```

```
🔑 Triggering emergency stop for device 1...
```

```
✓ Emergency stop executed successfully on Agent1
```

Kluczowe features implementacji

✅ Wymagania spełnione:

- **Agent:** Aplikacja konsolowa .NET 6 z pełną obsługą OPC UA i Azure IoT
- **Connectivity:** Dwukierunkowa komunikacja Agent ↔ Azure IoT Hub
- **Data Nodes:** Wszystkie wymagane węzły obsługiwane (telemetry + stan)
- **Direct Methods:** EmergencyStop + ResetErrorStatus zaimplementowane
- **Device Twin:** Synchronizacja Production Rate i Device Errors
- **Calculations:** Production KPI, Temperature stats, Error counting w ASA
- **Business Logic:** Wszystkie 3 reguły biznesowe w Azure Functions

🔧 Dodatkowe funkcje:

- **Konfiguracja:** appsettings.json dla łatwej konfiguracji
- **Error Handling:** Robust error handling i retry logic
- **Logging:** Szczegółowe logowanie z emoji dla czytelności
- **Scalability:** Design wspiera wielu Agents i urządzenia
- **Monitoring:** Application Insights integration ready

📈 Performance:

- **Telemetry:** 1 wiadomość/sekundę na urządzenie
- **Latency:** Real-time processing error events via EventHub
- **Throughput:** Batch processing KPI via Stream Analytics
- **Memory:** Efficient error tracking z automatic cleanup

Projekt w pełni realizuje wymagania Case Study dla Industrial IoT platform na Azure.