

Projekt IoT – dokumentacja

Link do repozytorium: <https://github.com/WiktoriaSzkudlarek/IoT-project>

Uruchamianie aplikacji

Przed pobraniem aplikacji, na platformie Azure należy utworzyć:

- **Resource group**,
- **IoT Hub i 2 Device'y** (u mnie są o nazwach: Device1, Device2),
- **Storage account z 4 kontenerami typu Blob** (u mnie są o nazwach: telemetry, production-kpi, temperature-avg-min-max, device-errors),
- **Stream Analytics job z 1 Inputem oraz zapytania kierujące wyniki do 4 Outputów** (u mnie mają takie same nazwy jak kontenery).

Po pobraniu projektu z repozytorium należy wejść do folderu **ServiceSdkDemo.Console**, a następnie kliknąć na plik **ServiceSdkDemo.Console.sln**, który uruchomi projekt w Visual Studio (o ile jest zainstalowany).

W panelu Solution Explorer trzeba wyszukać i wcisnąć plik **config.json**, który ma następującą strukturę:

```
{
  "ServiceConnectionString": "",
  "OpcClientConnectionString": "opc.tcp://localhost:4840/",
  "StorageConnectionString": "",
  "Devices": [
    {
      "DeviceId": "1",
      "ConnectionString": ""
    },
    {
      "DeviceId": "2",
      "ConnectionString": ""
    }
  ],
  "Blobs": [
    {
      "BlobName": "production-kpi.json",
      "Method": "ReduceProductionRate"
    },
    {
      "BlobName": "device-errors.json",
      "Method": "EmergencyStop"
    }
  ]
}
```

Parametry będące pustymi znakami należy skopiować z portalu Azure.

- **ServiceConnectionString** znajduje się w: IoT Hub: Security settings > iothubowner > Primary connection string.
- **ConnectionString** znajduje się w: IoT Hub: Device management > Devices > [nazwa urządzenia] > Primary connection string.
- **StorageConnectionString** znajduje się w: Storage account: [nazwa kontenera] > Security + networking > Access keys > key1 > Connection string

Po wpisaniu connectionStringów, zapisaniu pliku config.json oraz **włączeniu symulatora i dodaniu odpowiedniej ilości urządzeń** można uruchomić projekt, który powinien włączyć dwa mniejsze projekty:

- ServiceSdkDemo.Console otwierające okienka dla agenta i urządzeń działających przez DeviceSdkDemo.Console
- oraz BlobStorageDemo.Desktop pozwalające na pobranie danych z chmury.

Projekt ServiceSdkDemo.Console uruchamia tyle okienek dla urządzeń, ile jest ich wpisanych w pliku konfiguracyjnym.

Połączenie z serwerem OPC UA odbywa się w projekcie DeviceSdkDemo.Console, gdzie tworzony jest obiekt klasy VirtualDevice i istnieje ono dla danego urządzenia do momentu zamknięcia okienka swojego lub agenta. Dane są pobierane co około 1 sekundę.

Komunikacja z platformą Azure

Wiadomości D2C

Aplikacja wysyła do platformy dane telemetryczne za pomocą D2C message, którego sekcja Body ma następujące parametry z przykładową zawartością:

```
"Body": {
  "device": "Device2",
  "productionStatus": 1,
  "workorderId": "26525968-b224-46eb-9de1-fe270f993039",
  "goodCount": 6,
  "badCount": 1,
  "temperature": 61.38280472731428
}
```

Ten mechanizm zostaje uruchamiany co ok. 1 sekundę i przechowywany jest w Blob Storage'u o nazwie **telemetry**.

Oprócz danych telemetrycznych, wysyłane są również raporty o zmianie błędów urządzeń, które przechowywane są w Blob Storage'u o nazwie **device-errors**.

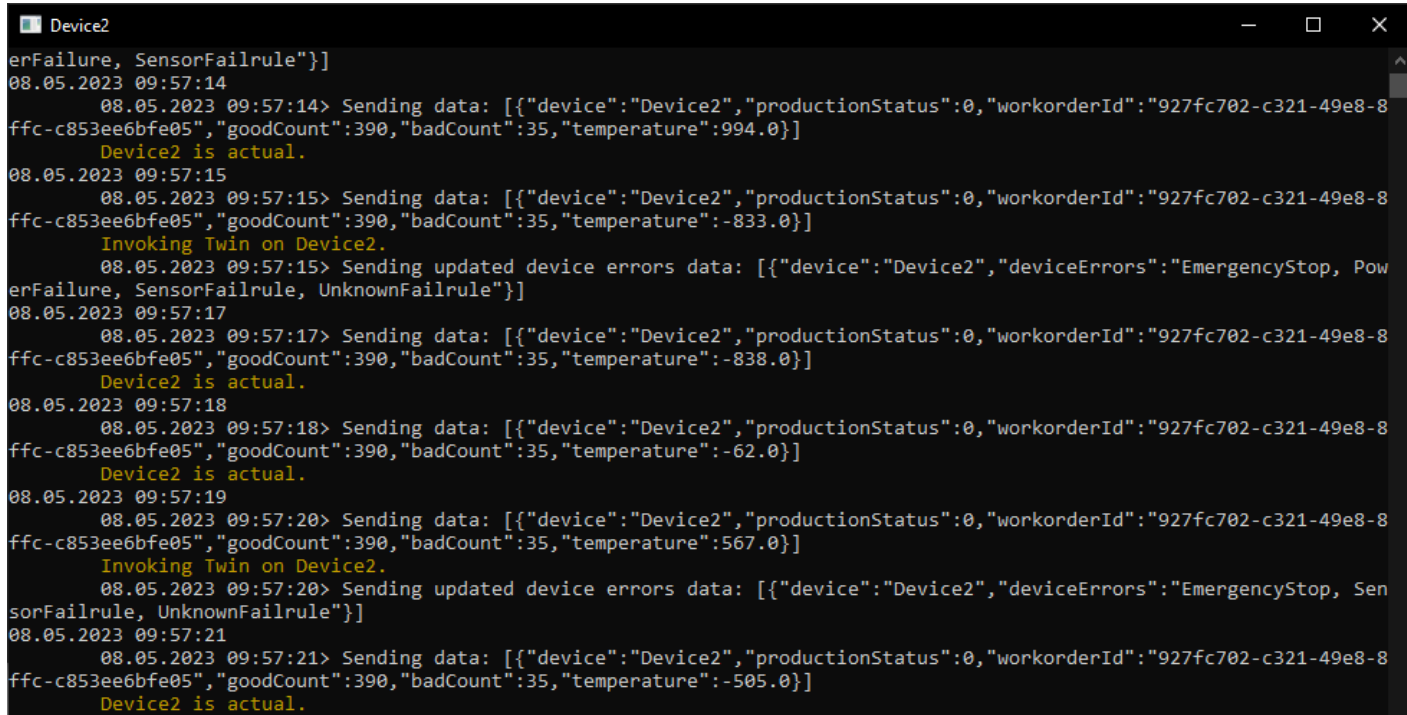
```
"Body": {
  "Count": 12,
  "Device": "Device2"
}
```

Device Twin

Za pomocą Device Twin przechowywane są raporty o aktualnym tempie produkcji oraz błędach urządzenia. Mechanizm się uruchamia jeśli zostanie odnotowana zmiana jednego z tych parametrów.

Poniżej znajduje się przykładowa treść dostarczana przez Device Twin oraz informacje na urządzeniu informujące o uruchomieniu się tej formy komunikacji:

```
"properties": {
  "desired": {
    "$metadata": {
      "$lastUpdated": "2023-05-06T08:21:13.923281Z"
    },
    "$version": 1
  },
  "reported": {
    "productionRate": 20,
    "deviceErrors": "None",
    "lastDeviceErrorsDate": "2023-05-06T00:00:00+02:00",
    "$metadata": {
      "$lastUpdated": "2023-05-06T13:21:29.5156727Z",
      "productionRate": {
        "$lastUpdated": "2023-05-06T13:21:29.5156727Z"
      },
      "deviceErrors": {
        "$lastUpdated": "2023-05-06T13:21:29.5156727Z"
      },
      "lastDeviceErrorsDate": {
        "$lastUpdated": "2023-05-06T09:39:54.80008Z"
      }
    },
    "$version": 13
  }
},
```



```
erFailure, SensorFailrule}}]
08.05.2023 09:57:14
08.05.2023 09:57:14> Sending data: [{"device":"Device2","productionStatus":0,"workorderId":"927fc702-c321-49e8-8ffc-c853ee6bfe05","goodCount":390,"badCount":35,"temperature":994.0}]
Device2 is actual.
08.05.2023 09:57:15
08.05.2023 09:57:15> Sending data: [{"device":"Device2","productionStatus":0,"workorderId":"927fc702-c321-49e8-8ffc-c853ee6bfe05","goodCount":390,"badCount":35,"temperature":-833.0}]
Invoking Twin on Device2.
08.05.2023 09:57:15> Sending updated device errors data: [{"device":"Device2","deviceErrors":"EmergencyStop, PowerFailure, SensorFailrule, UnknownFailrule"}]
08.05.2023 09:57:17
08.05.2023 09:57:17> Sending data: [{"device":"Device2","productionStatus":0,"workorderId":"927fc702-c321-49e8-8ffc-c853ee6bfe05","goodCount":390,"badCount":35,"temperature":-838.0}]
Device2 is actual.
08.05.2023 09:57:18
08.05.2023 09:57:18> Sending data: [{"device":"Device2","productionStatus":0,"workorderId":"927fc702-c321-49e8-8ffc-c853ee6bfe05","goodCount":390,"badCount":35,"temperature":-62.0}]
Device2 is actual.
08.05.2023 09:57:19
08.05.2023 09:57:20> Sending data: [{"device":"Device2","productionStatus":0,"workorderId":"927fc702-c321-49e8-8ffc-c853ee6bfe05","goodCount":390,"badCount":35,"temperature":567.0}]
Invoking Twin on Device2.
08.05.2023 09:57:20> Sending updated device errors data: [{"device":"Device2","deviceErrors":"EmergencyStop, SensorFailrule, UnknownFailrule"}]
08.05.2023 09:57:21
08.05.2023 09:57:21> Sending data: [{"device":"Device2","productionStatus":0,"workorderId":"927fc702-c321-49e8-8ffc-c853ee6bfe05","goodCount":390,"badCount":35,"temperature":-505.0}]
Device2 is actual.
```

Dostępne metody

W kliencie dostępne są następujące metody:

C2D

Wysyłanie wiadomości w wybraną treść dla parametru text do wybranego urządzenia.

```
1 - C2D
2 - Direct Method
3 - Device Twin
4 - 'Business Logic'
0 - Exit
$>1
Type your message (confirm with enter):
$>test
Type your device id (confirm with enter):
$>Device2

08.05.2023 09:53:28> Sending data: [{"device":"Device2","productionStatus":1,"workorderId":"927fc702-c321-49e
ffc-c853ee6bfe05","goodCount":164,"badCount":14,"temperature":72.2000387051626}]
Device2 is actual.
08.05.2023 09:53:29
08.05.2023 09:53:29> Sending data: [{"device":"Device2","productionStatus":1,"workorderId":"927fc702-c321-49e
ffc-c853ee6bfe05","goodCount":165,"badCount":15,"temperature":82.7290043116441}]
Device2 is actual.
08.05.2023 09:53:31
08.05.2023 09:53:31> Sending data: [{"device":"Device2","productionStatus":1,"workorderId":"927fc702-c321-49e
ffc-c853ee6bfe05","goodCount":168,"badCount":15,"temperature":66.09538429021258}]
Device2 is actual.
08.05.2023 09:53:32
08.05.2023 09:53:32> Sending data: [{"device":"Device2","productionStatus":1,"workorderId":"927fc702-c321-49e
ffc-c853ee6bfe05","goodCount":169,"badCount":16,"temperature":81.88756146841807}]
Device2 is actual.
08.05.2023 09:53:33
08.05.2023 09:53:33> Sending data: [{"device":"Device2","productionStatus":1,"workorderId":"927fc702-c321-49e
ffc-c853ee6bfe05","goodCount":169,"badCount":16,"temperature":66.78649073186818}]
Device2 is actual.
08.05.2023 09:53:35
08.05.2023 09:53:35> Sending data: [{"device":"Device2","productionStatus":1,"workorderId":"927fc702-c321-49e
ffc-c853ee6bfe05","goodCount":171,"badCount":16,"temperature":82.55810504807215}]
Device2 is actual.
08.05.2023 09:53:35> C2D message callback - message received with Id=8bc35a93-9f3c-4497-b899-de581e8dc85c
Received message: {"text":"test"}
08.05.2023 09:53:35> Completed C2D message with Id=8bc35a93-9f3c-4497-b899-de581e8dc85c.
08.05.2023 09:53:36
08.05.2023 09:53:36> Sending data: [{"device":"Device2","productionStatus":1,"workorderId":"927fc702-c321-49e
ffc-c853ee6bfe05","goodCount":176,"badCount":17,"temperature":65.40128454523953}]
Device2 is actual.
```

Direct Method

Wykonanie wybranej metody, która wpłynie na wirtualną produkcję dla wybranego urządzenia.

- EmergencyStop – zatrzymanie produkcji na wybranym urządzeniu
- ResetErrorStatus – usuwanie błędów na wybranym urządzeniu
- ReduceProductionRate – zmniejszanie tempa produkcji o 10% na wybranym urządzeniu
- SendTelemetry – wysyłanie pojedynczej danych telemetrycznej z wybranego urządzenia

```
C:\Users\Wiko\Desktop\IoT\IoT-project\ServiceSdkDemo.Console\ServiceSdkDemo.Console\bin\Debug\net6.0\ServiceSdkDemo.Console.exe

1 - C2D
2 - Direct Method
3 - Device Twin
4 - 'Business Logic'
0 - Exit
$>2
Type your device id (confirm with enter):
$>Device1
Choose Method (Emergency Stop is default, hehe):
1 - Emergency Stop
2 - Reset Error Status
3 - Reduce Production Rate
4 - Send Telemetry
$>2
Method executed with status 0

1 - C2D
2 - Direct Method
3 - Device Twin
4 - 'Business Logic'
0 - Exit
$>_
```

Device Twin

Ta opcja działa tak jak na przykładowym kodzie z zajęć, czyli po podaniu nazwy urządzenia i nazwy właściwości, aktualizuje ją przypisując pseudolosową wartość.

Zaimplementowane kalkulacje

Za pomocą Stream Analytics job wykonywane są kalkulacje dotyczące:

- Wskaźnika wydajności produkcji – pokazuje KPI na dane urządzenie z podziałem na 5 minut, którego wyniki są przechowywane w Blobie **production-kpi**.

```
SELECT
    device AS Device, (( SUM(goodCount) / (SUM(goodCount) + SUM(badCount))) * 100) AS
Kpi
INTO
    [production-kpi]
FROM
    [iot]
WHERE
    device IS NOT NULL
GROUP BY
    device, TumblingWindow(minute, 5);
```

Input preview **Test results** Job simulation (preview)

[Download results](#)

Device <i>string</i>	Kpi <i>float</i>
"Device2"	82.76840874937386
"Device1"	96.74772927043657

Showing 2 rows from 'production-kpi'.

Ln 4, Col 5

- Temperatury – przechowuje średnią, minimalną i maksymalną temperaturę na dane urządzenie z podziałem na 5 minut, którego wyniki są przechowywane w Blobie **temperature-avg-min-max**.

```
SELECT
    device, AVG(temperature) AS temperatureAvg, MIN(temperature) AS temperatureMin,
MAX(temperature) AS temperatureMax
INTO
    [temperature-avg-min-max]
FROM
    [iot]
WHERE
    device IS NOT NULL
GROUP BY
    device, TumblingWindow(minute, 5);
```

Input preview **Test results** Job simulation (preview)

[Download results](#)

device <i>string</i>	temperatureAvg <i>float</i>	temperatureMin <i>float</i>	temperatureMax <i>float</i>
"Device2"	104.65190953490013	-835	922
"Device1"	63.4895660696113	60.00680880154655	68.03468591348994

Showing 2 rows from 'temperature-avg-min-max'.

Ln 4, Col 5

- Ilość zarejestrowanych błędów urządzenia – zlicza łączną liczbę napotkanych błędów w czasie 1 minuty i przechowuje wyniki w Blobie **device-errors**.

```
SELECT
    COUNT(deviceErrors) AS Count, device AS Device
INTO
    [device-errors]
FROM
    [iot]
WHERE
    deviceErrors IS NOT NULL
GROUP BY
    device, TumblingWindow(minute, 1);
```

Input preview **Test results** Job simulation (preview)

[Download results](#)

Count <i>bigint</i>	Device <i>string</i>
13	"Device2"

Showing 1 rows from 'device-errors'.

Ln 1, Col 1

- Jest również Blob o nazwie **telemetry**, który przechowuje wszystkie wiadomości wysłane przez C2D.

```
SELECT
    *
INTO
    [telemetry]
FROM
    [iot];
```

Input preview **Test results** Job simulation (preview)

[Download results](#)

device <i>string</i>	productionStatus <i>bigint</i>	workorderId <i>string</i>	goodCount <i>bigint</i>	badCount <i>bigint</i>	temperature <i>float</i>
"Device1"	1	"8e959195-e259-4b05...	43	2	63.620595965
"Device2"	1	"7626f6b7-f457-4ed6-...	78	11	66.563807468
"Device1"	1	"8e959195-e259-4b05...	44	2	65.790376615
"Device2"	1	"7626f6b7-f457-4ed6-...	79	11	65.426268347
"Device1"	1	"8e959195-e259-4b05...	45	2	63.873158465
"Device2"	1	"7626f6b7-f457-4ed6-...	80	11	71.545496432

Showing 113 rows from 'telemetry'.

Ln 4, Col 5

Przykładowe wyniki są zapisane w folderze **Blobs**.

Zaimplementowana logika biznesowa

Niestety nie byłam w stanie wymyślić jak wywoływać metody w chmurze, więc ten warunek jest wykonywany inaczej niż był zapisany w kryteriach.

Dane są pobierane z wybranych kontenerów, które później można pobrać do folderu **Blob**.

Aby „Logika biznesowa” zadziałała, trzeba pobrać pliki z kontenerów **device-errors** i **production-kpi** (domyślną nazwą pliku będzie [nazwa blobu].json). Następnie w okienku agenta, w „menu głównym” można wybrać opcję 4. Program odczyta „najświeższą” (pierwszą linijkę pliku) kalkulację z obu plików i na jej podstawie zadecyduje czy uruchomić metody EmergencyStop lub ReduceProductionRate.

