

# Dokumentacja

## Case study

Repozytorium projektu: <https://github.com/JoannaKacprzak/IoT>

## Wstęp

Kod konfiguruje system IoT, który odczytuje dane maszynowe za pomocą OPC UA i wysyła je do chmury za pomocą Azure IoT Hub.

Kod wykonuje między innymi następujące zadania:

- Odczytuje właściwości konfiguracyjne z pliku o nazwie "config.properties" za pomocą metody `readProperties()`.
- Ustawia właściwości z załadowanych właściwości konfiguracyjnych za pomocą metody `setupProperties()`.
- Pobiera listę urządzeń chmurowych z Azure IoT Hub za pomocą metody `getCloudDevices()`.
- Łączy się z klientem OPC UA za pomocą metody `connectToClient()`.
- Pobiera dane maszyny za pomocą klienta OPC UA i wysyła je do Azure IoT Hub za pomocą metody `readMachineData()`.
- Pętla przechodzi przez metodę `readMachineData()` co 5 sekund, aby stale wysyłać dane maszyny do Azure IoT Hub.

## Połączenie z urządzeniem

Połączenie z serwerem OPC UA jest nawiązywane w metodzie `connectToClient()`. Metoda `getMachines()` pobiera listę węzłów na serwerze OPC UA za pomocą metody `browse()` klasy `OpcUaClient`.

Metoda `readMachineData()` odczytuje dane z serwera OPC UA. W tej metodzie można użyć metody `readValue()`, aby odczytać wartość określonego węzła. Można również użyć metody `writeValue()` do zapisu danych do serwera OPC UA oraz metody `call()` do wywołania metod na serwerze OPC UA.

```
private void connectToClient() {
    try {
        // Utwórz nowego klienta OPC UA z podanym adresem URL serwera.
        this.client = OpcUaClient.create(this.serverURL);
        this.client.connect().get();
    } catch (UaException | ExecutionException | InterruptedException ex) {
        // Obsłuż wyjątki, które mogą wystąpić podczas procesu łączenia.
        System.out.println("UA Server connection problem: " + ex.getMessage());
    }
}
```

```

private void getMachines() { // Ta metoda pobiera listę urządzeń z serwera OPC UA.
    try {
        // Wykonaj żądanie przeglądania węzłów i pobierz wynik.
        BrowseResult result = this.client.browse(new BrowseDescription(
            Identifiers.ObjectsFolder,
            BrowseDirection.Both,
            Identifiers.References,
            true,
            UInteger.valueOf(NodeClass.Unspecified.getValue()),
            UInteger.valueOf(BrowseResultMask.BrowseName.getValue())
        )).get();

        // Przetwórz wynik przeglądania i utwórz obiekty Machine dla każdego węzła "Device".
        int index = 0;
        for (ReferenceDescription rd : result.getReferences()) {
            if (Objects.requireNonNull(rd.getBrowseName().getName()).contains("Device")) {
                UShort fullNodeId = rd.getNodeId().getNamespaceIndex();
                DeviceClient deviceClient = new DeviceClient(this.cloudDevices.get(index).getConnectionString(), IotHubClientProtocol.MQTT);
                Machine machine = new Machine(fullNodeId.intValue(), rd.getBrowseName().getName(), this.cloudDevices.get(index), deviceClient);
                this.machines.add(machine);
                index += 1;
                Thread thread = new Thread(new EmergencyStop(machine));
                thread.start();
            }
        }
    } catch (Exception e) {
        System.out.println("Machines browsing error: " + e.getMessage());
    }
}

private void readMachineData() { // Ta metoda odczytuje dane z węzłów serwera OPC UA i aktualizuje obiekty Machine.

    // Dla każdego obiektu Machine na liście machines wykonaj odczyt danych z węzłów serwera OPC UA i zaktualizuj wartości obiektu.
    for (Machine machine : this.machines) {
        machine.setProductionStatus(Integer.parseInt(Objects.requireNonNull(readNode(machine, STATUS_ENDPOINT)).toString()));
        machine.setWorkOrderId((String) readNode(machine, WORK_ORDER_ID_ENDPOINT));
        machine.setProductionRate(Integer.parseInt(Objects.requireNonNull(readNode(machine, PRODUCTION_RATE_ENDPOINT)).toString()));
        long goodCount = Long.parseLong(Objects.requireNonNull(readNode(machine, COUNT_GOOD_ENDPOINT)).toString()) - machine.getGoodCount();
        machine.setGoodCount(goodCount);
        long badCount = Long.parseLong(Objects.requireNonNull(readNode(machine, COUNT_BAD_ENDPOINT)).toString()) - machine.getBadCount();
        machine.setBadCount(badCount);
        machine.setTemperature(Double.parseDouble(Objects.requireNonNull(readNode(machine, TEMPERATURE_ENDPOINT)).toString()));
        machine.setDeviceError(Integer.parseInt(Objects.requireNonNull(readNode(machine, DEVICE_ERROR_ENDPOINT)).toString()));

        // Sprawdź, czy produkcja jest uruchomiona dla danego urządzenia i wyślij odpowiednie wiadomości.
        if (machine.getProductionStatus() == 1) {
            System.out.println("Trying to send data with D2C message for " + machine.getDeviceName());
            this.sendD2CMessage(machine);
            setTwin(machine);
        }
    }
}

private Object readNode(Machine machine, String endpoint) { // Ta metoda odczytuje wartość węzła serwera OPC UA dla danego urządzenia i punktu końcowego.

    // Tworzy się obiekt NodeId na podstawie pełnego identyfikatora węzła oraz punktu końcowego.
    NodeId nodeId = new NodeId(machine.getFullNodeId(), machine.getDeviceName() + endpoint);
    try {
        // Odczytuje się wartość węzła za pomocą klienta OPC UA.
        Variant variant = this.client.readValue(Double.MAX_VALUE, TimestampsToReturn.Server, nodeId).get().getValue();
        return variant.getValue();
    } catch (Exception e) {
        System.out.println("Node values reading error: " + e.getMessage());
        return null;
    }
}

```

## Konfiguracja agenta

```

server.url = opc.tcp://localhost:4840/
connectionString.owner = HostName=iot-Kacprzak-standard-1234.azure-devices.net;SharedAccessKeyName=iothubowner;SharedAccessKey=Wj8n9F2p2b4eh+gPLI5rL+3EZmJUqrF4xnR51Fi6LE=
connectionString.device = HostName=iot-Kacprzak-standard-1234.azure-devices.net;SharedAccessKeyName=device;SharedAccessKey=S91V/qa060qtkDsD/jd+98jQRCzTZF0pBRD5B1X+w/w=

```

## Komunikaty D2C

Kod wysyła do IoT Hubu wiadomości D2C (device-to-cloud), które zawierają dane z maszyn. Częstotliwość wysyłania wiadomości zależy od stanu produkcji każdej maszyny. Agent odczytuje dane z węzłów serwera OPC UA i aktualizuje obiekty maszynowe o nowe wartości. Dla każdej maszyny, jeśli status produkcyjny wynosi 1 (czyli jest uruchomiona), agent wysyła do chmury wiadomość D2C zawierającą dane maszyny.

Formatem wiadomości jest JSON, który jest tworzony przy użyciu biblioteki Gson. Wiadomość zawiera obiekt MachineDTO, który reprezentuje dane z maszyny, w tym identyfikator zlecenia pracy, tempo produkcji, błąd urządzenia, temperaturę, good count i bad count.

Metoda sendD2CMessage tworzy nowy obiekt DeviceClient, aby połączyć się z IoT Hubem i wysłać wiadomość asynchronicznie. Metoda wykorzystuje protokół MQTT do wysłania wiadomości. Po wysłaniu wiadomości metoda zamyka połączenie klienta urządzenia.

Podsumowując typ wiadomości wysyłanych przez agenta do IoT Hub to wiadomości D2C, format to JSON, a częstotliwość zależy od stanu produkcji każdej maszyny.

### Podpunkt z Case study, spełnione wymagania

#### 2.2 Data calculations:

1. Production per workorder
  - Sum of Good Count per Workorder ID
  - Sum of Bad Count per Workorder ID

```
32  SELECT
33      workOrderId as 'WorkOrderId',
34      SUM(CountGood) AS 'GoodCount',
35      SUM(CountBad) AS 'BadCount',
36      System.Timestamp() AS Timestamp,
37      udf.guidGenerator(null) AS RowKey,
38      '1' AS PartitionKey
39  INTO
40      [production-sum]
41  FROM
42      [agentInput]
43  GROUP BY
44      workOrderId,
45      TumblingWindow(minute, 15);
```

2. Production KPIs
  - % of good production (vs total production) in 15-minute windows

```

12  ✓ production_percentage AS (
13  ✓      SELECT
14      |      deviceId,
15      |      GoodCount,
16      |      TotalCount,
17      |      100.0 * GoodCount / NULLIF(TotalCount, 0) AS 'Percentage',
18      |      System.Timestamp() AS Timestamp,
19      |      udf.guidGenerator(null) AS RowKey,
20      |      '1' AS PartitionKey
21  ✓      FROM
22      |      production
23      )
24  ✓ SELECT
25      |      *
26  ✓ INTO
27      |      [production-kpi]
28  ✓ FROM
29      |      production_percentage;
30

```

### 3. Temperature per machine

- Average temperature per machine in 5-minute windows
- Maximum temperature per machine in 5-minute windows
- Minimum temperature per machine in 5-minute windows

```

47  SELECT
48      |      workOrderId,
49      |      MIN(temperature) as 'MinTemperature',
50      |      MAX(temperature) as 'MaxTemperature',
51      |      AVG(temperature) as 'AvgTemperature',
52      |      System.Timestamp() AS Timestamp,
53      |      udf.guidGenerator(null) AS RowKey,
54      |      '1' AS PartitionKey
55  INTO
56      |      [temperature]
57
58  FROM
59      |      [agentInput]
60  GROUP BY
61      |      WorkOrderId,
62      |      TumblingWindow(Duration(minute, 5));

```

### 4. Errors per machine

- Number of individual errors per machine in 30-minute windows
- Situations, when there are more than 3 errors within 15 minutes (per machine)

```

63 SELECT
64     deviceId,
65     COUNT(*) AS ErrorsCount,
66     System.Timestamp() AS Timestamp,
67     udf.guidGenerator(null) AS RowKey,
68     '1' AS PartitionKey
69 INTO
70     [errors]
71 FROM
72     [agentInput]
73 WHERE
74     deviceError != 0
75 GROUP BY
76     deviceId,
77     TumblingWindow(minute, 30);
78
79 SELECT
80     deviceId,
81     COUNT(*) AS EmergencyErrors
82
83 INTO
84     [emergency-errors]
85 FROM
86     [agentInput]
87 WHERE
88     deviceError != 0
89 GROUP BY
90     deviceId,
91     TumblingWindow(minute, 15)
92 HAVING
93     COUNT(*) > 3;

```






## Storagefactory:

### Tables

Authentication method: Access key ([Switch to Azure AD User Account](#))

☐ Show system-generated tables

Showing all 5 items

<input type="checkbox"/>	Name	Url	
<input type="checkbox"/>	 emergencyErrors	https://storageforfactory.table.core.windows.net/emergencyErrors	...
<input type="checkbox"/>	 errors	https://storageforfactory.table.core.windows.net/errors	...
<input type="checkbox"/>	 productionKPI	https://storageforfactory.table.core.windows.net/productionKPI	...
<input type="checkbox"/>	 productionSum	https://storageforfactory.table.core.windows.net/productionSum	...
<input type="checkbox"/>	 temperature	https://storageforfactory.table.core.windows.net/temperature	...

Przykłady:

Tables > productionKPI

Authentication method: Access key (Switch to Azure AD User Account)

Add filter

Showing all 15 items

	PartitionKey	RowKey	Timestamp	GoodCount	TotalCount	deviceId	Percentage	
	1	1526e5aa-7058-4fc3-9f...	2023-02-18T16:00:43.73...	0.0	0.0	device_3		...
	1	1cda2d2d-ef40-45d3-ba...	2023-02-18T20:05:54.98...	1082.0	1199.0	device_3	90.2418682235196	...
	1	27bd9822-7298-41ba-af...	2023-02-18T19:46:37.45...	21.0	22.0	device_3	95.454545454545453	...
	1	3b3def04-f6ca-434f-bb...	2023-02-18T20:45:08.14...	6754.0	7511.0	device_1	89.9214485421382	...
	1	408fa2c7-82a7-4826-8e...	2023-02-18T13:00:06.75...	135348.0	150089.0	device_3	90.178494093504526	...
	1	437bde1c-43d8-4d87-b...	2023-02-18T20:31:19.68...	45403.0	50309.0	device_3	90.2482657178636	...
	1	566c69b2-daa6-4e11-8...	2023-02-18T20:15:54.57...	28867.0	31880.0	device_3	90.548933500627356	...
	1	5984fbcf-c3c5-48b2-88...	2023-02-18T15:15:06.71...	120.0	132.0	device_3	90.9090909090909	...
	1	5f7318f0-6766-412a-b2f...	2023-02-18T15:00:07.57...	619.0	672.0	device_3	92.113095238095241	...
	1	799d3aaf-5d8e-4d76-8c...	2023-02-18T20:31:19.68...	81670.0	90626.0	device_1	90.117626288261647	...
	1	951fa285-7c6e-4266-a0...	2023-02-19T13:15:07.04...	1848.0	2010.0	device_1	91.940298507462686	...
	1	9685927a-0eac-41b3-8c...	2023-02-19T13:30:11.64...	7855.0	8584.0	device_1	91.507455731593666	...
	1	9c814b84-7100-4314-b...	2023-02-18T12:45:06.20...	45611.0	50689.0	device_3	89.982047387007043	...
	1	de1b88dc-7abf-4174-84...	2023-02-18T16:15:51.97...	2986.0	3316.0	device_3	90.04825090470446	...
	1	f347968f-1c3c-43ad-96...	2023-02-18T14:45:06.61...	55.0	57.0	device_3	96.491228070175438	...

Tables > temperature

Authentication method: Access key (Switch to Azure AD User Account)

Add filter

Showing all 37 items

	PartitionKey	RowKey	Timestamp	AvgTemperature	MaxTemperature	MinTemperature	workOrderId	
	1	0c7b2853-181c-4f81-99...	2023-02-18T14:50:07.08...	83.8877699808883	83.8877699808883	83.8877699808883	edb63832-a03d-46ed-b...	...
	1	12a51344-48ea-4ad9-a...	2023-02-18T12:45:06.19...	75.5551362163418	96.921122564652649	60.844913544527643	fb916ebf-6783-4d99-8b...	...
	1	1e21e439-6418-4540-a...	2023-02-18T20:37:23.03...	60.419392747511672	60.419392747511672	60.419392747511672	57467577-519c-4bec-93...	...
	1	21e091ed-5e8b-499e-8...	2023-02-18T12:35:07.40...	72.834846627126964	89.7599391626537	61.600528233199391	fb916ebf-6783-4d99-8b...	...
	1	2555fcb0-46d6-40e7-84...	2023-02-18T14:45:07.20...	72.274655975259321	75.005183387149543	69.055085625918309	57e8da76-b56d-40ec-8f...	...
	1	28cc76f3-116a-4d38-a7...	2023-02-18T16:05:27.30...	74.961993036355267	81.951738450968	65.607097804185074	54b077ee-7631-4cf4-82...	...
	1	2bb9bbfd-ee9c-419c-a8...	2023-02-18T13:00:06.75...	79.26740297614991	90.976175373824532	60.114240564144012	fb916ebf-6783-4d99-8b...	...
	1	2c6b0dd9-84b3-4114-a...	2023-02-18T20:05:25.50...	69.994383604441552	84.423699717976731	60.3781892368589	42d9ab48-5a32-4440-8...	...
	1	36147698-35ea-4067-a...	2023-02-18T16:05:27.30...	72.711882194600832	86.210839378252288	60.898914858651693	a2180986-984c-4263-a1...	...
	1	39801e59-a4ff-4acf-a28...	2023-02-19T13:20:53.63...	81.7057280884133	89.835215206615544	76.682958979030573	2fb68f0f-adee-4260-bcc...	...
	1	39e74a61-1634-43b4-9...	2023-02-18T20:10:22.31...	125.40494337634613	137.71713234195317	113.09275441073908	f0e1ad56-731b-4b28-86...	...
	1	3ad46d95-6fce-4570-b5...	2023-02-18T20:00:19.63...	67.083035897278037	72.395122982269214	61.770948812286861	42d9ab48-5a32-4440-8...	...
	1	41732534-3853-42fe-97...	2023-02-18T12:50:05.73...	75.632461760039689	94.1232474044547	61.209232327015421	fb916ebf-6783-4d99-8b...	...
	1	4af25817-dccb-4804-97...	2023-02-18T20:25:08.78...	85.10992664961158	127.10531815473722	60.365990956546355	69033bca-a8d3-411c-b...	...
	1	4cbe8624-c0bf-45dc-b0...	2023-02-18T20:30:49.71...	38.319688409462636	109.11162667398668	-775.0	57467577-519c-4bec-93...	...
	1	515bc122-515e-4c60-a9...	2023-02-19T13:15:22.65...	75.120143369051391	105.82950410711413	60.811407450848115	6cf510d8-8b5c-468a-81...	...
	1	5383270d-bd2f-4d69-9...	2023-02-18T20:00:19.63...	75.807340125776449	75.807340125776449	75.807340125776449	14440e32-60b6-46ce-8...	...
	1	56aeecac-a8fb-4e9d-b7...	2023-02-18T20:10:22.31...	87.474414314222983	87.474414314222983	87.474414314222983	69033bca-a8d3-411c-b...	...

# Device Twin

Device Twin, jest używany do aktualizacji zgłoszonych właściwości maszyny.

```
private void setTwin(Machine machine) { // Metoda ta aktualizuje bliźniacze urządzenie o najnowsze właściwości maszyny
    try {
        machine.getDeviceClient().open();
        Set<Property> propertySet = new HashSet<>();
        // Dodaj właściwości maszyny do zestawu właściwości
        propertySet.add(new Property("Errors", machine.getDeviceError()));
        propertySet.add(new Property("ProductionRate", machine.getProductionRate()));
        // Dodaj właściwość LastErrorDate, jeśli błąd urządzenia nie jest zerowy
        if (machine.getDeviceError() != 0) {
            propertySet.add(new Property("LastErrorDate", Instant.now().getEpochSecond()));
        }
        try {
            // Wyślij zestaw właściwości, aby zaktualizować bliźniacze urządzenie
            machine.getDeviceClient().sendReportedProperties(propertySet);
        } catch (IOException e) {
            // Jeśli wystąpi błąd, ponawiamy próbę uruchomienia bliźniaczego urządzenia
            System.out.println("Error when updating device twin: " + e.getMessage());
            startTwin(machine);
        }
    }

    } catch (IOException e) {
        System.out.println("Error when updating device twin: " + e.getMessage());
        startTwin(machine);
    }
}
```

Metoda setTwin aktualizuje twin o najnowsze właściwości maszyny, takie jak liczba błędów i tempo produkcji, poprzez utworzenie zestawu właściwości i wysłanie go do device twin za pomocą metody sendReportedProperties obiektu DeviceClient. Jeśli podczas tego procesu wystąpi błąd, wywoływana jest metoda startTwin w celu ponownego uruchomienia bliźniaka urządzenia.

```
private void startTwin(Machine machine) { // Ta metoda uruchamia bliźniacze urządzenie dla podanej maszyny
    try {
        DeviceClient deviceClient = machine.getDeviceClient();
        // Otwórz klienta urządzenia i czekaj na zakończenie
        CompletableFuture<Boolean> future = CompletableFuture.supplyAsync(() -> {
            try {
                deviceClient.open();
            } catch (IOException e) {
                throw new RuntimeException(e);
            }

            return true;
        });
        future.get();
        // Uruchom bliźniacze urządzenie i poczekaj na zakończenie
        future = CompletableFuture.supplyAsync(() -> {
            try {
                deviceClient.startDeviceTwin((IoTHubStatusCode, o) -> {
                    System.out.println("Twin has been changed for " + machine.getDeviceName());
                }, null, (o, o2, o3) -> {
                }, null);
            } catch (IOException e) {
                throw new RuntimeException(e);
            }

            return true;
        });
        future.get();
        // Zamknij
        deviceClient.closeNow();
    } catch (IOException | ExecutionException | InterruptedException e) {
        System.out.println("Error when start device twin: " + e.getMessage());
        startTwin(machine);
    }
}
```

Metoda `startTwin` inicjalizuje device twin dla danej maszyny. Najpierw otwiera klienta urządzenia, a następnie uruchamia urządzenie bliźniacze przy pomocy metody `startDeviceTwin` obiektu `DeviceClient`. Metoda `startDeviceTwin` przyjmuje jako parametry kilka wywołań zwrotnych, które mogą być wykorzystane do obsługi odpowiedzi urządzenia twin. Jeżeli podczas tego procesu wystąpi błąd, metoda wywołuje się rekursywnie w celu ponownego uruchomienia urządzenia twin.

## Przykład:

Home > [iot-kacprzak-standard-1234](#) | Devices > [device\\_1](#) >

### Device twin

device\_1

Save Refresh

The device twin for 'device\_1' is shown below. You can add tags and desired properties to your device twin here. To remove a tag or desired prop

```
1  {
2    "deviceId": "device_1",
3    "etag": "AAAAAAAAAAE=",
4    "deviceEtag": "ODM1MzU5MDE1",
5    "status": "enabled",
6    "statusUpdateTime": "0001-01-01T00:00:00Z",
7    "connectionState": "Disconnected",
8    "lastActivityTime": "2023-02-19T13:25:53.2603912Z",
9    "cloudToDeviceMessageCount": 0,
10   "authenticationType": "sas",
11   "x509Thumbprint": {
12     "primaryThumbprint": null,
13     "secondaryThumbprint": null
14   },
15   "modelId": "",
16   "version": 28,
17   "properties": {
18     "desired": {
19       "$metadata": {
20         "$lastUpdated": "2023-02-16T17:20:53.405958Z"
21       },
22       "$version": 1
23     },
24     "reported": {
25       "Errors": 0,
26       "ProductionRate": 40,
27       "LastErrorDate": 1676812443,
28       "$metadata": {
29         "$lastUpdated": "2023-02-19T13:22:33.8415593Z",
30         "Errors": {
31           "$lastUpdated": "2023-02-19T13:22:33.8415593Z"
32         },
33         "ProductionRate": {
34           "$lastUpdated": "2023-02-19T13:22:33.8415593Z"
35         },
36         "LastErrorDate": {
37           "$lastUpdated": "2023-02-19T13:14:04.1570988Z"
38         }
39       },
40       "$version": 27
41     }
42   },
43   "capabilities": {
44     "iotEdge": false
45   }
46 }
```



# Metoda bezpośrednia

Zaimplementowane są trzy metody bezpośrednie:

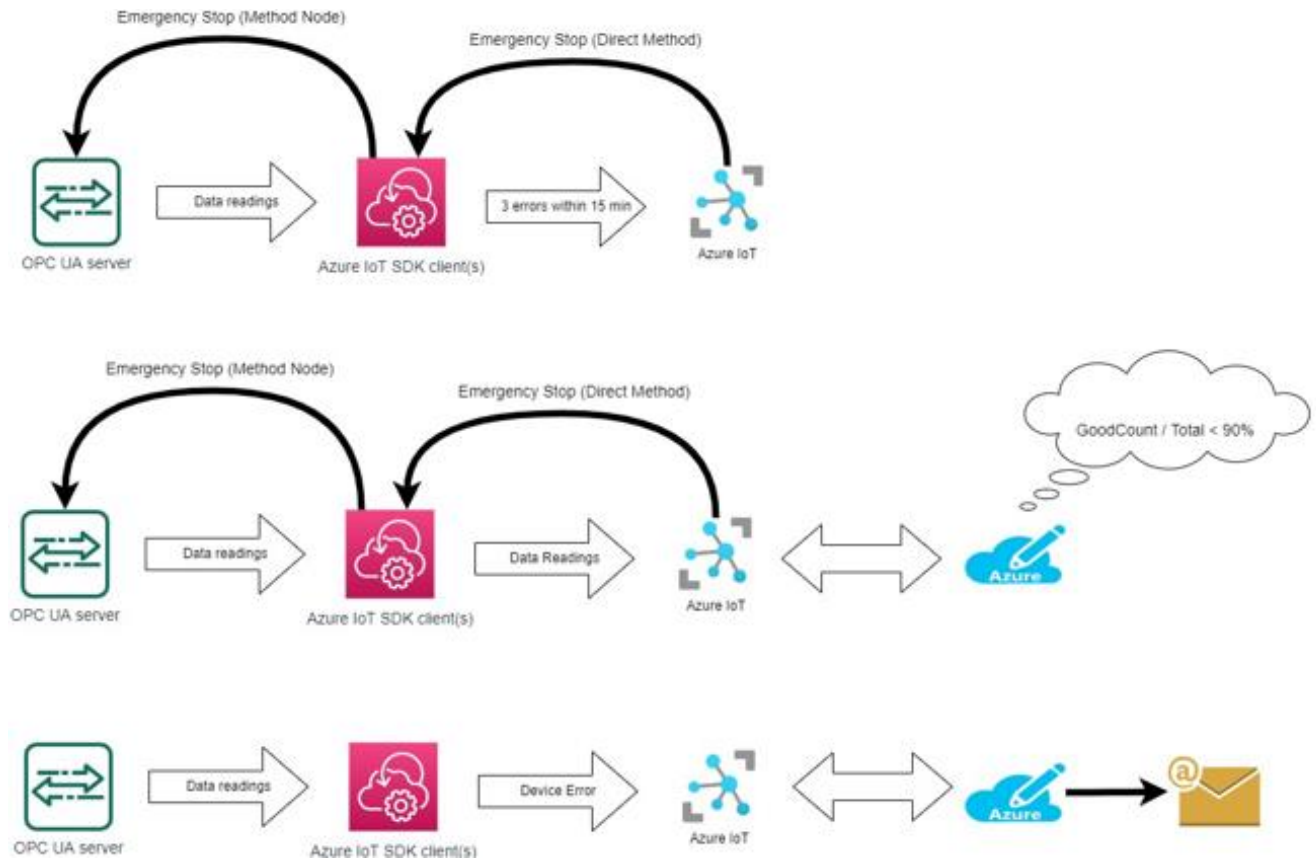
```
public class DeviceMethodCall implements DeviceMethodCallback {

    @Override
    public DeviceMethodData call(String s, Object o, Object o1) {
        Machine machine = (Machine) o1;
        CallMethodRequest request;
        switch (s) {
            case "EmergencyStop" -> { // Obsługa wywołania metody EmergencyStop.
                System.out.println("EMERGENCY STOP RECEIVED FOR " + machine.getDeviceName());
                request = new CallMethodRequest(
                    new NodeId(machine.getFullNodeId(), machine.getDeviceName()),
                    new NodeId(machine.getFullNodeId(), machine.getDeviceName() + EMERGENCY_STOP_ENDPOINT),
                    null);
                client.call(request);
            }
            case "ResetErrors" -> { // Obsługa wywołania metody ResetErrors.
                System.out.println("RESET ERRORS RECEIVED FOR " + machine.getDeviceName());
                request = new CallMethodRequest(
                    new NodeId(machine.getFullNodeId(), machine.getDeviceName()),
                    new NodeId(machine.getFullNodeId(), machine.getDeviceName() + RESET_ERRORS_ENDPOINT),
                    null);
                client.call(request);
            }
            case "ProductionReduce" -> { // Obsługa wywołania metody ProductionReduce.
                System.out.println("PRODUCTION REDUCE RECEIVED FOR " + machine.getDeviceName());
                Variant newValue = new Variant(machine.getProductionRate() - 10);
                NodeId nodeId = new NodeId(machine.getFullNodeId(), machine.getDeviceName() + PRODUCTION_RATE_ENDPOINT);
                try {
                    UaVariableNode variableNode = client.getAddressSpace().getVariableNode(nodeId);
                    variableNode.writeValue(newValue);
                } catch (UaException ignored) {
                }
            }
        }
        return new DeviceMethodData(200, "Ok");
    }
}
```

Kod definiuje klasę `DeviceMethodCall`, która implementuje interfejs `DeviceMethodCallback`. Metoda `call` tej klasy obsługuje wywoływanie różnych metod, takich jak `EmergencyStop`, `ResetErrors` i `ProductionReduce`, wykorzystując protokół OPC UA do wywoływania odpowiednich metod lub modyfikowania wartości określonych zmiennych. Kod drukuje również komunikaty do konsoli, aby wskazać wywoływane metody.

# Logika biznesowa

Na podstawie schematów w Case study:



factory-functions | Functions ...

Function App

Search

+ Create Refresh Delete

Filter by name...

Name ↑↓	Trigger ↑↓	Status ↑↓	Monitor ↑↓
ErrorMail	Timer	Enabled	Invocations and more
EventGridTrigger1	Timer	Enabled	Invocations and more
ProductionReduce	Timer	Enabled	Invocations and more

factory-functions \ ErrorMail \

```
1 public static void Run()
2 {
3     var connectionString = "https://storageforfactory.table.core.windows.net";
4     var accountName = "storageforfactory";
5     var accountKey = "CxCH0DQ9mhr4s8SKMP5X7X3nFzsdh9gW3pH8f8v1ckLif4rS1cUvNlJ9WcE0xC3WnLCeQMho8g4q+ASTXfbe1A==";
6     var tableName = "errors";
7
8     string fromEmail = "your-email@your-domain.com";
9     string password = "your-email-password";
10    string smtpServer = "smtp.your-email-provider.com";
11    int smtpPort = 587;
12
13    string recipientEmail = "whereemailgoto@gmail.com";
14    string subject = "Errors";
15
16    var smtpClient = new SmtpClient(smtpServer, smtpPort)
17    {
18        Credentials = new NetworkCredential(fromEmail, password),
19        EnableSsl = true
20    };
21
22    var client = new TableClient(new Uri(connectionString), tableName, new TableSharedKeyCredential(accountName, accountKey));
23
24    var fiveMinutesAgo = DateTime.UtcNow.AddMinutes(-30);
25
26    var filter = $"Timestamp ge datetime'{fiveMinutesAgo.ToString("yyyy-MM-ddTHH:mm:ssZ")}';";
27
28    var response = client.Query<TableEntity>(filter: filter);
29
30    foreach (var record in response)
31    {
32        string message = $"Device error: {record.GetString("deviceId")}";
33
34        var mailMessage = new MailMessage
35        {
36            From = new MailAddress(fromEmail),
37            Subject = subject,
38            Body = message
39        };
40
41        mailMessage.To.Add(recipientEmail);
42
43        var res = smtpClient.SendMailAsync(mailMessage).Result;
44    }
45 }
```

factory-functions \ EventGridTrigger1 \

```
1 using Azure;
2 using System;
3 using Microsoft.Azure.WebJobs;
4 using Microsoft.Extensions.Logging;
5 using Azure.Data.Tables;
6 using Microsoft.Azure.Devices;
7
8 public static void Run(TimerInfo myTimer, ILogger log)
9 {
10     string hubConnectionString = "HostName=IoT-Kacprzak-standard-1234.azure-devices.net;SharedAccessKeyName=iothubowner;SharedAccessKey=wJ8n9F2p2b4eh+gP1I5r1+3E2nJ0qrRfxnR51F16LE=";
11     ServiceClient serviceClient = ServiceClient.CreateFromConnectionString(hubConnectionString);
12     var methodInvocation = new CloudToDeviceMethod("EmergencyStop");
13
14     var connectionString = "https://storageforfactory.table.core.windows.net";
15     var accountName = "storageforfactory";
16     var accountKey = "CxCH0DQ9mhr4s8SKMP5X7X3nFzsdh9gW3pH8f8v1ckLif4rS1cUvNlJ9WcE0xC3WnLCeQMho8g4q+ASTXfbe1A==";
17     var tableName = "emergencyErrors";
18
19     var client = new TableClient(new Uri(connectionString), tableName, new TableSharedKeyCredential(accountName, accountKey));
20
21     var fiveMinutesAgo = DateTime.UtcNow.AddMinutes(-15);
22
23     var filter = $"Timestamp ge datetime'{fiveMinutesAgo.ToString("yyyy-MM-ddTHH:mm:ssZ")}';";
24
25     var response = client.Query<TableEntity>(filter: filter);
26
27     foreach (var record in response)
28     {
29         var agentResponse = serviceClient.InvokeDeviceMethodAsync(record.GetString("deviceId"), methodInvocation).Result;
30     }
31 }
```

factory-functions \ ProductionReduce \

```
1 using System;
2 using Azure.Data.Tables;
3 using Microsoft.Azure.Devices;
4
5 public static void Run(TimerInfo myTimer, ILogger log)
6 {
7     string hubConnectionString = "HostName=iot-kacprzak-standard-1234.azure-devices.net;SharedAccessKeyName=iothubowner;SharedAccessKey=w7j8n9F2p2b4eh+gP1I5r1+3EZn3UqrRfxnR51F16LE=";
8     ServiceClient serviceClient = ServiceClient.CreateFromConnectionString(hubConnectionString);
9     var methodInvocation = new CloudToDeviceMethod("ProductionReduce");
10
11     var connectionString = "https://storageforfactory.table.core.windows.net";
12     var accountName = "storageforfactory";
13     var accountKey = "CxCH0DQ9mhr4s8SKMP5X7X3nFzsdh9gW3pH8f8v1ckLi4rS1cUvNlJ9WcE8xC3WnLCeQW0o8g4q+ASTxfbe1A==";
14     var tableName = "productionKPI";
15
16     var client = new TableClient(new Uri(connectionString), tableName, new TableSharedKeyCredential(accountName, accountKey));
17
18     var fiveMinutesAgo = DateTime.UtcNow.AddMinutes(-15);
19
20     var filter = $"Timestamp ge datetime'{fiveMinutesAgo.ToString("yyyy-MM-ddTHH:mm:ssZ")}'";
21
22     var response = client.Query<TableEntity>(filter: filter);
23
24     foreach (var record in response)
25     {
26         var agentResponse = serviceClient.InvokeDeviceMethodAsync(record.GetString("deviceId"), methodInvocation).Result;
27     }
28 }
29
```