

Dokumentacja IoT

Link do repozytorium: <https://github.com/MrSwistak/IoT---Projekt.git>

Opis klasy głównej, wraz z wykorzystanymi metodami. Dodatkowe, szczegółowe komentarze w pliku Line.cs, mówiące o połączeniu.

Kod jest implementacją klienta **OPC UA**, który odczytuje dane z serwera **OPC UA**, wysyła te dane do **Azure IoT Hub** i ustawia stan urządzeń (bliźniaków) na podstawie tych danych.

Klasa **Line** jest główną klasą, która wykonuje te operacje. Konstruktor tej klasy przyjmuje instancję **LineSettings**, która zawiera informacje o serwerze **OPC UA**, **Regex** do identyfikacji węzłów maszyn oraz ciąg połączenia **Azure IoT Hub**.

Metoda Up() jest wywoływana w celu wykonania następujących czynności:

1. Utwórz nowego klienta **OPC UA**, używając adresu serwera określonego w **LineSettings**.
2. Połącz klienta z serwerem.
3. Wywołaj metodę **GetDevicesIds()**, aby uzyskać identyfikatory urządzeń z **IoT Hub**.
4. Wywołaj metodę **Browse()**, aby przeglądać folder obiektów serwera i pobrać informacje dla określonych ID urządzeń.
5. Wywołaj metodę **ReadMachineNodesValues()**, aby odczytać wartości węzłów urządzenia, serializować stan urządzenia i ustawić bliźniaki asynchronicznie.
6. Jeśli błąd urządzenia nie jest **None**, zaktualizuj bliźniaka asynchronicznie.
7. Wywołaj metodę **ReadOpcData()**, aby asynchronicznie odczytać dane **OPC UA** z serwera dla wszystkich maszyn.

Metoda GetDevicesIds() tworzy pustą listę identyfikatorów urządzeń, sprawdza, czy łańcuchy połączeń i **Regex** nazwy urządzenia nie są puste, pobiera urządzenia jako bliźniaki z **IoT Hub**, używając łańcucha połączenia właściciela i maksymalnej liczby urządzeń określonej w **LineSettings**, i iteruje nad każdą stroną wyników, aby dodać identyfikatory urządzeń, które pasują do **Regex** nazwy urządzenia do listy.

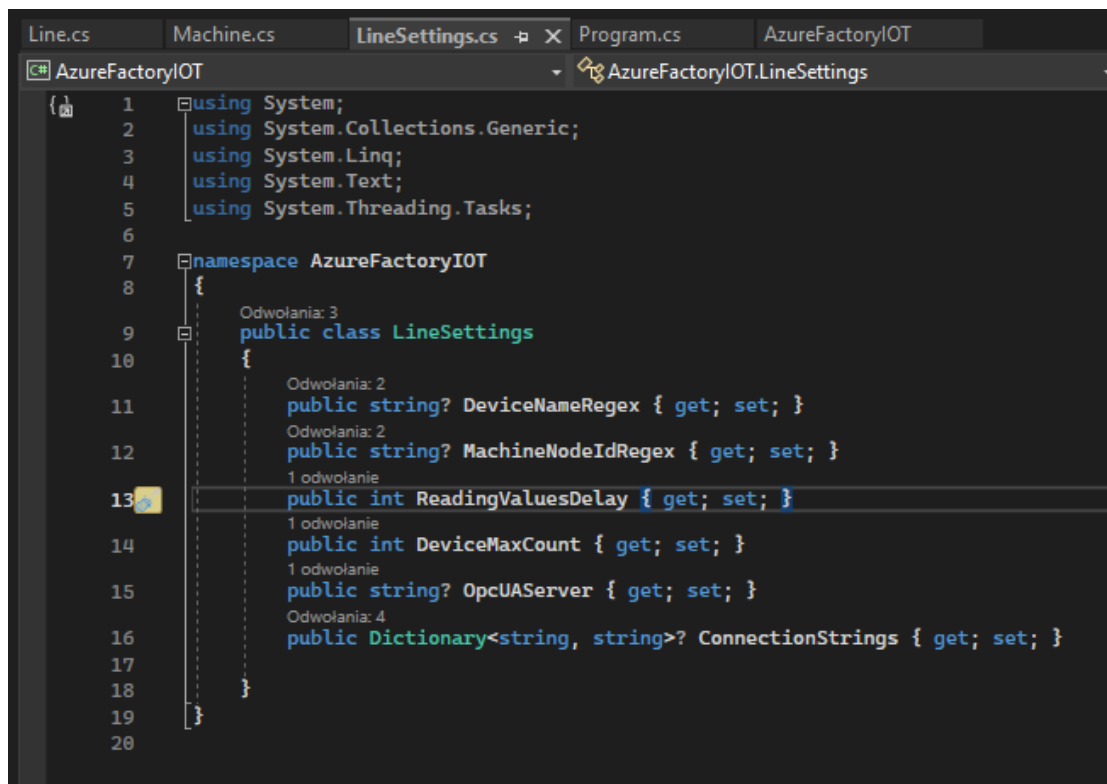
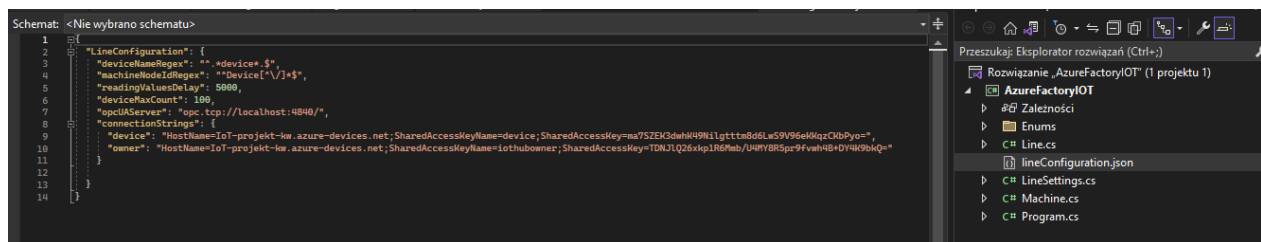
Metoda Browse() tworzy pustą listę maszyn, sprawdza, czy łańcuch **regex ID** węzła maszyny i łańcuchy połączeń nie są puste, sprawdza, czy ID bieżącego węzła pasuje do **regex ID** węzła maszyny, tworzy klienta urządzenia używając łańcucha połączenia określonego w **LineSettings**

używając ID urządzenia w ID bieżącego węzła i iteruje po każdym węźle-dziecku bieżącego węzła, aby stworzyć nowy obiekt maszyny i dodać go do listy.

Metoda **ReadMachineNodesValues()** odczytuje wartości węzłów maszyny, ustawia wartości odpowiednich właściwości w obiekcie maszyny i ustawia wartości odpowiednich bliźniaków asynchronicznie. Metoda wykorzystuje klienta **OPC UA** i obiekt maszyny jako parametry.

Metoda **ReadOpcData()** asynchronicznie odczytuje z serwera dane **OPC UA** dla wszystkich maszyn. Metoda wykorzystuje klienta **OPC UA** i listę maszyn jako parametry.

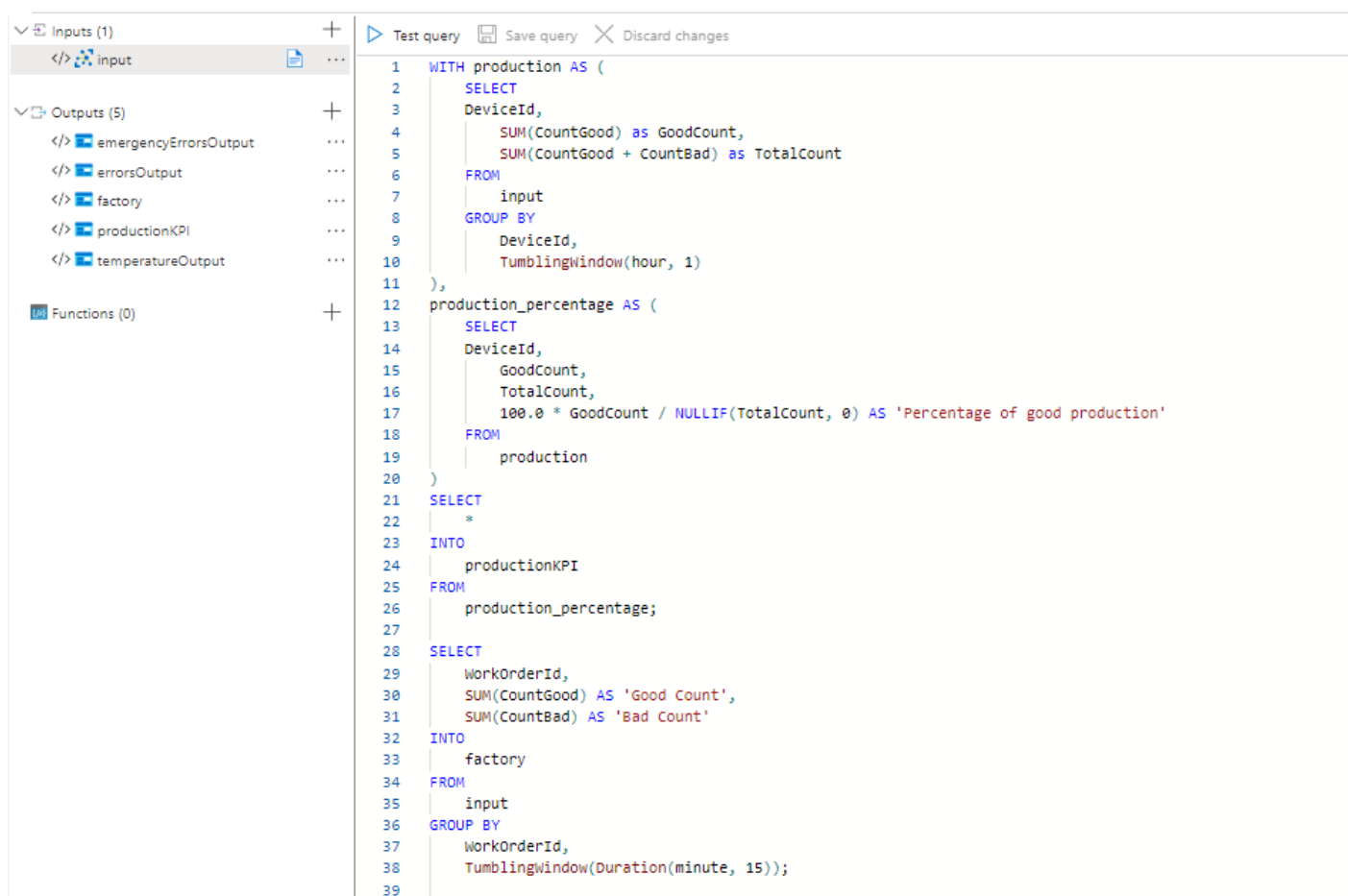
Konfiguracja agenta



Direct 2 Cloud Messages

Agent odczytuje dane z maszyn co 5 sekund, konwertuje je na json z danymi telemetrycznymi, które natychmiast wysyła do IoTHuba

Produkcja na zlecenie: suma Good Count dla każdego Workorder ID, suma Bad Count dla każdego Workorder ID oraz KPI produkcji - % dobrej produkcji (w stosunku do całkowitej produkcji) w 15-minutowych oknach.



The screenshot shows a SQL query editor with a sidebar on the left and a main query area on the right. The sidebar contains a tree view with 'Inputs (1)' and 'Outputs (5)'. The 'Inputs (1)' section shows a single input named 'input'. The 'Outputs (5)' section shows five outputs: 'emergencyErrorsOutput', 'errorsOutput', 'factory', 'productionKPI', and 'temperatureOutput'. The 'Functions (0)' section is empty. The main query area displays a SQL query with line numbers 1 through 39. The query is a T-SQL script that calculates production statistics. It starts with a 'WITH' clause defining a 'production' CTE, followed by a 'production_percentage' CTE. The main query selects from 'production' and 'production_percentage' to calculate the percentage of good production. It then inserts the results into 'productionKPI' and finally selects the data from 'input' to be sent to the 'factory' output.

```
1 WITH production AS (  
2     SELECT  
3         DeviceId,  
4         SUM(CountGood) AS GoodCount,  
5         SUM(CountGood + CountBad) AS TotalCount  
6     FROM  
7         input  
8     GROUP BY  
9         DeviceId,  
10        TumblingWindow(hour, 1)  
11 ),  
12 production_percentage AS (  
13     SELECT  
14         DeviceId,  
15         GoodCount,  
16         TotalCount,  
17         100.0 * GoodCount / NULLIF(TotalCount, 0) AS 'Percentage of good production'  
18     FROM  
19         production  
20 )  
21 SELECT  
22     *  
23 INTO  
24     productionKPI  
25 FROM  
26     production_percentage;  
27  
28 SELECT  
29     WorkOrderID,  
30     SUM(CountGood) AS 'Good Count',  
31     SUM(CountBad) AS 'Bad Count'  
32 INTO  
33     factory  
34 FROM  
35     input  
36 GROUP BY  
37     WorkOrderID,  
38     TumblingWindow(Duration(minute, 15));  
39
```

Temperatura dla każdej maszyny (minimalna, maksymalna, średnia) w 5-minutowych oknach:

```
--
41 SELECT
42     WorkOrderId,
43     MIN(Temperature) as 'Minimum Temperature',
44     MAX(Temperature) as 'Maximum Temperature',
45     AVG(Temperature) as 'Average Temperature'
46 INTO
47     temperatureOutput
48 FROM
49     input
50 GROUP BY
51     WorkOrderId,
52     TumblingWindow(Duration(minute, 5));
--
```

Liczba pojedynczych błędów na maszynę w 30-minutowych oknach:

```
54 SELECT
55     DeviceId,
56     COUNT(*) AS ErrorCount
57 INTO
58     errorsOutput
59 FROM
60     input
61 WHERE
62     DeviceError != 0
63 GROUP BY
64     DeviceId,
65     TumblingWindow(minute, 30);
--
```

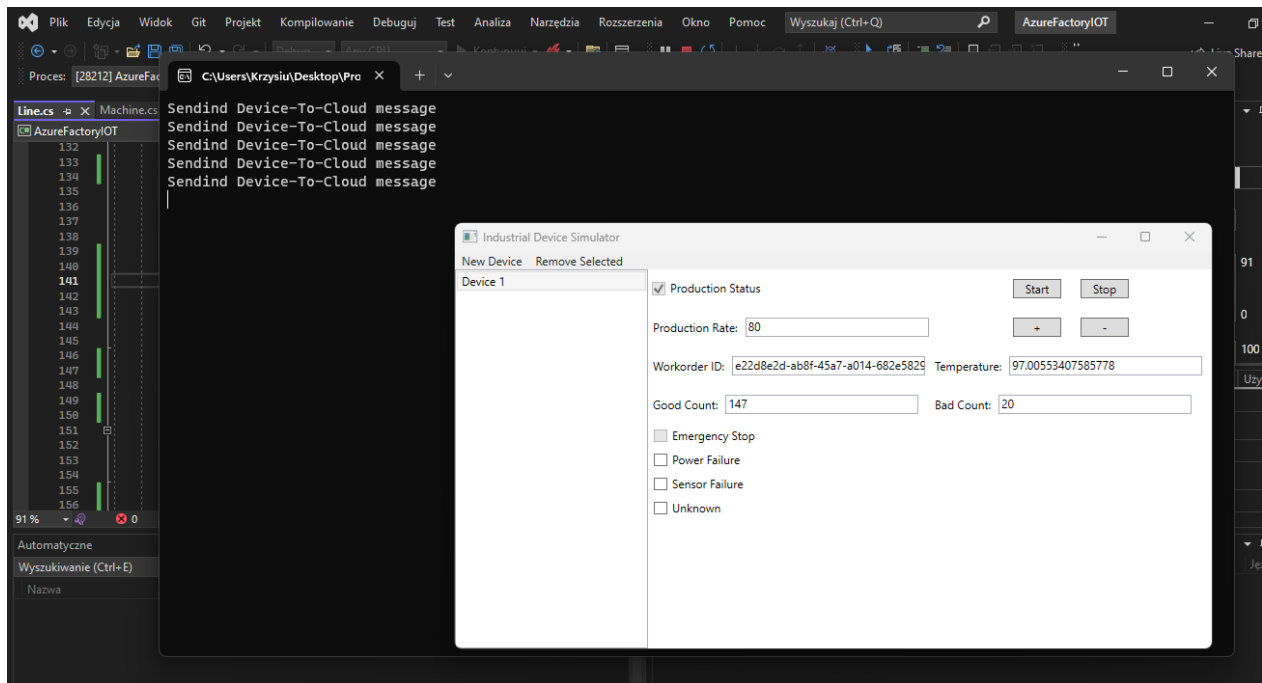
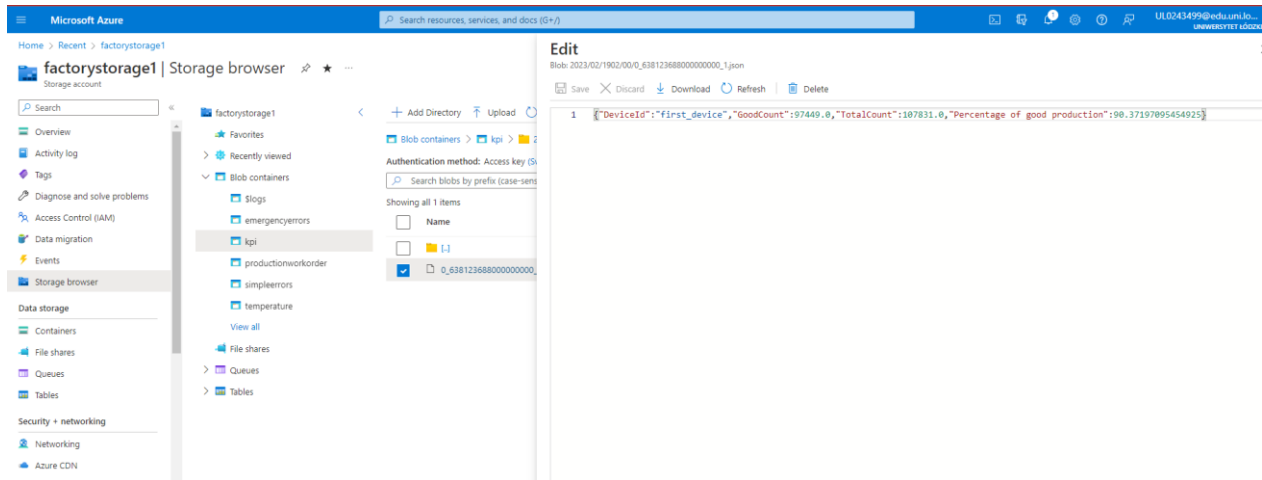
Sytuacje, gdy w ciągu 15 minut wystąpią więcej niż 3 błędy (na maszynę):

```
66 |
67 | SELECT
68 |     DeviceId,
69 |     COUNT(*) AS EmergencyErrors
70 | INTO
71 |     emergencyErrorsOutput
72 | FROM
73 |     input
74 | WHERE
75 |     DeviceError != 0
76 | GROUP BY
77 |     DeviceId,
78 |     TumblingWindow(minute, 15)
79 | HAVING
80 |     COUNT(*) > 3;
```

Ze storage:

The screenshot shows the Microsoft Azure portal interface. The left sidebar contains navigation options like Overview, Activity log, Tags, Diagnose and solve problems, Access Control (IAM), Data migration, Events, Storage browser, Data storage, Containers, File shares, Queues, Tables, Security + networking, Networking, and Azure CDN. The main area displays the 'factorystorage1' storage account. Under 'Blob containers', the 'emergencyerrors' container is selected. The 'Edit' pane on the right shows the JSON content of a blob, which contains two records: one for 'first_device' with 9 errors and one for 'second_device' with 9 errors.

```
1 [{"deviceId":"first_device","EmergencyErrors":9}]
2 [{"deviceId":"second_device","EmergencyErrors":9}]
```



Device Twin

Nowe dane dotyczące wskaźnika produkcji, błędów urządzenia i daty ich ostatniego wystąpienia są zgłaszane do Device Twin.

```
49         var machines = Browse(client.BrowseNode(OpcObjectTypes.ObjectsFolder), devicesIds, client);
50
51         //Iteruj przez każdą maszynę i odczytaj wartości jej węzłów, serializuj stan maszyny i ustaw bliźniaki asynchronicznie
52         foreach (var machine in machines)
53         {
54             ReadMachineNodesValues(client, machine);
55             var options = new JsonSerializerOptions
56             {
57                 IgnoreReadOnlyProperties = true
58             };
59             string machineState = JsonSerializer.Serialize(machine, options);
60
61             await SetTwinAsync(machine);
62
63             //Jeśli błąd urządzenia nie jest None, zaktualizuj bliźniaka asynchronicznie
64             if (machine.DeviceError != Enums.DeviceErrorEnum.None)
65             {
66                 await UpdateTwinAsync(machine);
67             }
68         }
69     }
```

```
1 Odwołania: 2
2 public async Task UpdateTwinAsync(Machine machine)
3 {
4     // Utwórz nowy obiekt TwinCollection do przechowywania właściwości raportowanych
5     var reportedProperties = new TwinCollection();
6
7     // Dodaj właściwości raportowane "DeviceErrors" i "LastErrorDate" do kolekcji
8     reportedProperties["DeviceErrors"] = machine.DeviceError;
9     reportedProperties["LastErrorDate"] = DateTime.Now;
10
11     // Wywołaj metodę UpdateReportedPropertiesAsync na kliencie urządzenia, aby zaktualizować właściwości raportowane
12     await machine.DeviceClient.UpdateReportedPropertiesAsync(reportedProperties);
13 }
14
15 1 odwołanie
16 public async Task SetTwinAsync(Machine machine)
17 {
18     // Utwórz nowy obiekt TwinCollection do przechowywania właściwości raportowanych
19     var reportedProperties = new TwinCollection();
20
21     // Dodaj właściwości raportowane "DeviceErrors" i "ProductionRate" do kolekcji
22     reportedProperties["DeviceErrors"] = machine.DeviceError;
23     reportedProperties["ProductionRate"] = machine.Rate;
24
25     // Jeśli urządzenie ma błąd, dodaj właściwość raportowaną "LastErrorDate" do kolekcji
26     if (machine.DeviceError != 0)
27     {
28         reportedProperties["LastErrorDate"] = DateTime.Now;
29     }
30
31     // Wywołaj metodę UpdateReportedPropertiesAsync na kliencie urządzenia, aby zaktualizować właściwości raportowane
32     await machine.DeviceClient.UpdateReportedPropertiesAsync(reportedProperties);
33 }
34 }
```

W IoTHub:

Microsoft Azure

Home > IoT-projekt-kw | Devices > first_device >

Device twin

first_device

Save Refresh

The device twin for 'first_device' is shown below. You can add tags and desired properties to your device twin here. To remove a tag or desired property, set the value of the item to be removed to 'null'.

```
1  {
2    "deviceId": "first_device",
3    "etag": "AAAAAAAAAAE=",
4    "deviceEtag": "MTA1MTI1NDExNQ==",
5    "status": "enabled",
6    "statusUpdateTime": "0001-01-01T00:00:00Z",
7    "connectionState": "Disconnected",
8    "lastActivityTime": "2023-02-19T15:47:48.2233691Z",
9    "cloudToDeviceMessageCount": 0,
10   "authenticationType": "sas",
11   "x509Thumbprint": {
12     "primaryThumbprint": null,
13     "secondaryThumbprint": null
14   },
15   "modelId": "",
16   "version": 161,
17   "properties": {
18     "desired": {
19       "$metadata": {
20         "$lastUpdated": "2023-02-14T17:21:30.6680277Z"
21       },
22       "$version": 1
23     },
24     "reported": {
25       "DeviceErrors": 6,
26       "ProductionRate": 80,
27       "LastErrorDate": "2023-02-19T16:47:49.6348184+01:00",
28       "$metadata": {
29         "$lastUpdated": "2023-02-19T15:47:48.270284Z",
30         "DeviceErrors": {
31           "$lastUpdated": "2023-02-19T15:47:48.270284Z"
32         },
33         "ProductionRate": {
34           "$lastUpdated": "2023-02-19T15:41:26.9511443Z"
35         },
36         "LastErrorDate": {
37           "$lastUpdated": "2023-02-19T15:47:48.270284Z"
38         }
39       },
40       "$version": 160
41     }
42   },
43   "capabilities": {
44     "iotEdge": false
45   }
46 }
```

Direct Methods:

```
// Ustawienie metod bezpośrednich dla wywołań metod klienta urządzenia
machine.DeviceClient.SetMethodHandlerAsync("EmergencyStop", HandleEmergencyStop, machine);
machine.DeviceClient.SetMethodHandlerAsync("LowKpiDetected", HandleLowKPI, machine);
machine.DeviceClient.SetMethodHandlerAsync("ResetErrors", ResetErrors, machine);
machine.DeviceClient.SetMethodHandlerAsync("MaintenanceDate", MaintenanceDate, machine);
```



```

// Metoda poniżej obsługuje zdalne wywołanie w przypadku zdarzenia związanego z zatrzymaniem awaryjnym
1 odwołanie
private static async Task<MethodResponse> HandleEmergencyStop(MethodRequest request, object userContext)
{
    var machine = (Machine)userContext;
    Console.WriteLine(request.Name);
    Console.WriteLine(new string('-', 20));
    Console.WriteLine($"Emergency stop received for {machine.DeviceId}");
    Console.WriteLine(new string('-', 20));

    machine.OpcClient.CallMethod(machine.Id, machine.EmergencyStopNode);

    return new MethodResponse(0);
}

// Metoda poniżej obsługuje zdalne wywołanie w przypadku wykrycia niskiego współczynnika wydajności
1 odwołanie
private static async Task<MethodResponse> HandleLowKPI(MethodRequest request, object userContext)
{
    var machine = (Machine)userContext;
    Console.WriteLine(request.Name);
    Console.WriteLine(new string('-', 20));
    Console.WriteLine($"Low KPI detected for {machine.DeviceId}");
    Console.WriteLine(new string('-', 20));

    machine.OpcClient.WriteNode(machine.RateNode, machine.Rate - 10);

    return new MethodResponse(0);
}

```

```

//Metoda ResetErrors resetuje wszystkie błędy dla urządzenia poprzez wywołanie metody OPC UA.
1 odwołanie
private static async Task<MethodResponse> ResetErrors(MethodRequest request, object userContext)
{
    var machine = (Machine)userContext;
    Console.WriteLine(request.Name);
    Console.WriteLine(new string('-', 20));
    Console.WriteLine($"Reset all errors received for {machine.DeviceId}");
    Console.WriteLine(new string('-', 20));

    machine.OpcClient.CallMethod(machine.Id, machine.ResetErrorsNode);

    return new MethodResponse(0);
}

//Metoda MaintenanceDate aktualizuje zgłoszone właściwości bliźniaka urządzenia o bieżącą datę i godzinę ostatniej kontroli konserwacyjnej.
1 odwołanie
private static async Task<MethodResponse> MaintenanceDate(MethodRequest methodRequest, object userContext)
{
    var reportedProperties = new TwinCollection();
    reportedProperties["LastMaintenanceDate"] = DateTime.Now;

    await ((Machine)userContext).DeviceClient.UpdateReportedPropertiesAsync(reportedProperties);

    return new MethodResponse(0);
}

#endregion

```

Logika biznesowa

Logika biznesowa na podstawie schematów w Case Study.

Home > factoryfunctions

factoryfunctions | Functions ...

Function App

Search

+ Create Refresh Delete

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Microsoft Defender for Cloud

Events (preview)

Functions

Functions

Filter by name...

<input type="checkbox"/> Name ↑↓	Trigger ↑↓	Status ↑↓	Monitor ↑↓	
<input type="checkbox"/> EmergencyErrorTrigger	Blob	Enabled	Invocations and more	...
<input type="checkbox"/> LowKpiTrigger	Blob	Enabled	Invocations and more	...
<input type="checkbox"/> SimpleErrorTrigger	Blob	Enabled	Invocations and more	...

Home > factoryfunctions | Functions > EmergencyErrorTrigger

EmergencyErrorTrigger | Code + Test ...

Function

Save Discard Refresh Test/Run Test integration Upload

Overview

Developer

Code + Test

Integration

Monitor

Function Keys

```
factoryfunctions \ EmergencyErrorTrigger \ run.csx
1 using Microsoft.Azure.Devices;
2 using Newtonsoft.Json;
3
4 public static void Run(Stream myBlob, string name, ILogger log)
5 {
6     string connectionString = "HostName=IoT-projekt-kw.azure-devices.net;SharedAccessKeyName=iothubowner;SharedAccessKey=TDNjIQ26xkp1R6Hmb/U4HY8R5pr9Fvuh4B+DY4K9bkQ=";
7     ServiceClient serviceClient = ServiceClient.CreateFromConnectionString(connectionString);
8     var methodInvocation = new CloudToDeviceMethod("EmergencyStop");
9     using (StreamReader reader = new StreamReader(myBlob))
10     {
11         while (!reader.EndOfStream)
12         {
13             string line = reader.ReadLine();
14             var data = JsonConvert.DeserializeObject<DeviceRow>(line);
15             var response = serviceClient.InvokeDeviceMethodAsync(data.DeviceId, methodInvocation).Result;
16         }
17     }
18 }
19
20 public class DeviceRow{
21     public string DeviceId {get;set;}
22
23     public DeviceRow(){
24     }
25 }
26
27 }
```

Home > factoryfunctions | Functions > LowKpiTrigger

LowKpiTrigger | Code + Test ...

Function

Save Discard Refresh Test/Run Test integration Upload

Overview

Developer

Code + Test

Integration

Monitor

Function Keys

```
factoryfunctions \ LowKpiTrigger \ run.csx
1 using Microsoft.Azure.Devices;
2 using Newtonsoft.Json;
3
4 public static void Run(Stream myBlob, string name, ILogger log)
5 {
6     string connectionString = "HostName=IoT-projekt-kw.azure-devices.net;SharedAccessKeyName=iothubowner;SharedAccessKey=TDNjIQ26xkp1R6Hmb/U4HY8R5pr9Fvuh4B+DY4K9bkQ=";
7     ServiceClient serviceClient = ServiceClient.CreateFromConnectionString(connectionString);
8     var methodInvocation = new CloudToDeviceMethod("LowKpiDetected");
9     using (StreamReader reader = new StreamReader(myBlob))
10     {
11         while (!reader.EndOfStream)
12         {
13             string line = reader.ReadLine();
14             var data = JsonConvert.DeserializeObject<DeviceRow>(line);
15             if(100 * data.GoodCount / data.TotalCount < 90){
16                 var response = serviceClient.InvokeDeviceMethodAsync(data.DeviceId, methodInvocation).Result;
17             }
18         }
19     }
20 }
21
22 public class DeviceRow{
23     public string DeviceId {get;set;}
24     public double GoodCount {get;set;}
25     public double TotalCount {get;set;}
26
27     public DeviceRow(){
28     }
29 }
30
31 }
```

Logs

App Insights Logs Log Level Stop Copy

SimpleErrorTrigger | Code + Test

Function

Save Discard Refresh Test/Run Test integration Upload

Overview

Developer

Code + Test

Integration

Monitor

Function Keys

factoryfunctions \ SimpleErrorTrigger \

```
1 using Microsoft.Azure.Devices;
2 using Newtonsoft.Json;
3
4 public static void Run(Stream myBlob, string name, ILogger log)
5 {
6     string fromEmail = "your-email@your-domain.com";
7     string password = "your-email-password";
8     string smtpServer = "smtp.your-email-provider.com";
9     int smtpPort = 587;
10
11     string recipientEmail = "whereemailgoto@gmail.com";
12     string subject = "Errors";
13
14     var smtpClient = new SmtpClient(smtpServer, smtpPort)
15     {
16         Credentials = new NetworkCredential(fromEmail, password),
17         EnableSsl = true
18     };
19
20     using (StreamReader reader = new StreamReader(myBlob))
21     {
22         while (!reader.EndOfStream)
23         {
24             string line = reader.ReadLine();
25             var data = JsonConvert.DeserializeObject<DeviceRow>(line);
26
27             string message = $"Device error: {data.DeviceId}";
28
29             var mailMessage = new MailMessage
30             {
31                 From = new MailAddress(fromEmail),
32                 Subject = subject,
33                 Body = message
34             };
35
36             mailMessage.To.Add(recipientEmail);
37
38             var res = smtpClient.SendMailAsync(mailMessage).Result;
39         }
40     }
41 }
42
43 public class DeviceRow{
44     public string DeviceId {get;set;}
45     public DeviceRow(){
46     }
47 }
48
49
50
51
```

Logs