

Jakub Kośmider

402629

IoTProjekt

- Repozytory:

<https://github.com/KubaPlayer1/IoTProjekt>

Repozytorium zostało stworzone po zrobieniu projektu i dopiero zostały do niego dołączone pliki.

Folder: Agent, Pliki: Agent.cs, IoTDevice.cs, OpcDevice.cs.

Uwaga! Projekt nie jest dokończony, ponieważ ostatnie półtora tygodnia jestem chory na antybiotykach i niestety nie udało mi się go dokończyć w pełni na czas (zwolnienie dostałem w poniedziałek) do 23 grudnia.

-Funkcja Main():

```
1  using Opc.UaFx;
2  using Opc.UaFx.Client;
3  using Microsoft.Azure.Devices.Client;
4  using Newtonsoft.Json;
5  using System.Net.Mime;
6  using System.Text;
7
8  Odwołania: 3
9  public class Program
10 {
11     static string[] Links = File.ReadAllLines("Links.txt");
12     static string OpcConnectionString = Links[1];
13     static string DeviceConnectionString = Links[3];
14
15     public static DateTime maintenanceDate = DateTime.MinValue;
16
17     Odwołania: 0
18     static async Task Main(string[] arguments)
19     {
20         using var deviceClient = DeviceClient.CreateFromConnectionString(DeviceConnectionString, TransportType.Mqtt);
21         await deviceClient.OpenAsync();
22         var device = new IoTDevice(deviceClient);
23
24         await device.InitializeHandlers();
25
26         Opcdevice opcDevice = new Opcdevice();
27
28         Opcdevice.StartConnection();
29
30         var timer = new PeriodicTimer(TimeSpan.FromSeconds(1));
31         while (await timer.WaitForNextTickAsync())
32         {
33             Console.WriteLine(DateTime.Now);
34             await IoTDevice.SendTelemetry(Opcdevice.client);
35         }
36
37         Opcdevice.EndConnection();
38         Console.ReadLine();
39     }
40 }
```

Główna funkcja programu, to ona wywołuje wszystkie funkcje, oraz w niej przechowywane są linki połączeń z IIoTsim i IoT Hub.

Funkcja main obsługuje także wywoływanie funkcji InitializeHendlers w które zawarte są wywołania do DirectMetod i ReceivingMessages.

- IoT Explorer:

Telemetry:

Kod funkcji SendTelemetry & SendTelemetryMessage:

```
1 ocdhazam
public static async Task SendTelemetry(OpcClient opcClient)
{
    var node = opcClient.BrowseNode(OpcObjectTypes.ObjectsFolder);
    if (node.Children().Count() > 1)
    {
        foreach (var childNode in node.Children())
        {
            if (!childNode.DisplayName.Value.Contains("Server"))
            {
                var device = Convert.ToInt32(childNode.DisplayName.Value.Split(" ")[1]);
                var productionStatus = opcClient.ReadNode($"ns=2;s=Device {device}/ProductionStatus").Value;
                Console.WriteLine(productionStatus);
                var workorderId = opcClient.ReadNode($"ns=2;s=Device {device}/WorkorderId").Value;
                Console.WriteLine(workorderId);
                var goodCount = opcClient.ReadNode($"ns=2;s=Device {device}/GoodCount").Value;
                Console.WriteLine(goodCount);
                var badCount = opcClient.ReadNode($"ns=2;s=Device {device}/BadCount").Value;
                Console.WriteLine(badCount);
                var temperature = opcClient.ReadNode($"ns=2;s=Device {device}/Temperature").Value;
                Console.WriteLine(temperature);
                var telemetryData = new
                {
                    device = device,
                    productionStatus = productionStatus,
                    workorderId = workorderId,
                    goodCount = goodCount,
                    badCount = badCount,
                    temperature = temperature,
                };
                await SendTelemetryMessage(telemetryData, client);
                var deviceErrors = opcClient.ReadNode($"ns=2;s=Device {device}/DeviceErrors").Value;
                Console.WriteLine($"Device errors = {deviceErrors}");
                var productionRate = opcClient.ReadNode($"ns=2;s=Device {device}/ProductionRate").Value;
                Console.WriteLine($"Production rate = {productionRate}");
                await TwinAsync(deviceErrors, productionRate);
            }
        }
    }
    await Task.Delay(1000);
}
1 ocdhazam
public static async Task SendTelemetryMessage(dynamic telemetryData, DeviceClient client)
{
    var dataString = JsonConvert.SerializeObject(telemetryData);
    Microsoft.Azure.Devices.Client.Message eventMessage = new Microsoft.Azure.Devices.Client.Message(Encoding.UTF8.GetBytes(dataString));
    eventMessage.ContentType = MediaTypeNames.Application.Json;
    eventMessage.ContentEncoding = "utf-8";
    //Console.WriteLine($"[{DateTime.Now.ToLocalTime()}] Data: [{dataString}]");
    await client.SendEventAsync(eventMessage);
}
```

Telemetria jest wysyłana co sekundę.

```
var timer = new PeriodicTimer(TimeSpan.FromSeconds(1));
while (await timer.WaitForNextTickAsync())
{
    Console.WriteLine(DateTime.Now);
    await IoTDevice.SendTelemetry(Opcdevice.client);
}
```

Przykładowa wiadomość telemetry D2C:

Thu Dec 22 2022 18:28:33 GMT+0100 (czas środkowoeuropejski standardowy):

```
{
  "body": {
    "device": 2,
    "productionStatus": 1,
    "workorderId": "5a34428e-32e6-4259-9825-bf20d63eb0e5",
    "goodCount": 68,
    "badCount": 26,
    "temperature": -761
  },
  "enqueuedTime": "Thu Dec 22 2022 18:28:33 GMT+0100 (czas środkowoeuropejski standardowy)",
  "properties": {}
}
```

Device Twin:

```
public static async Task TwinAsync(dynamic deviceErrors, dynamic productionRate)
{
    await UpdateTwinValueAsync("deviceErrors", deviceErrors);
    await UpdateTwinValueAsync("productionRate", productionRate);
}

Odwolania 2
public static async Task UpdateTwinValueAsync(string valueName, dynamic value)
{
    var twin = await client.GetTwinAsync();

    var reportedProperties = new TwinCollection();
    reportedProperties[valueName] = value;

    await client.UpdateReportedPropertiesAsync(reportedProperties);
}

Odwolania 3
public static async Task UpdateTwinValueAsync(string valueName, DateTime value)
{
    var twin = await client.GetTwinAsync();

    var reportedProperties = new TwinCollection();
    reportedProperties[valueName] = value;

    await client.UpdateReportedPropertiesAsync(reportedProperties);
}

1 odwołanie
private async Task OnDesiredPropertyChange(TwinCollection desiredProperties, object userContext)
{
    Console.WriteLine($"~\tDesired property change:\n\t{JsonConvert.SerializeObject(desiredProperties)}");
    Console.WriteLine($"~\tSending current time as reported property");
    TwinCollection reportedProperties = new TwinCollection();
    reportedProperties["DateTimeLastDesiredPropertyChangeReceived"] = DateTime.Now;

    await client.UpdateReportedPropertiesAsync(reportedProperties).ConfigureAwait(false);
}
```

```
13      "secondaryThumbprint": null
14    },
15    "modelId": "",
16    "version": 677,
17    "properties": {
18      "desired": {
19        "test1": "new_value",
20        "test2": "new_value4",
21        "test3": "new_value2",
22      },
23      "$metadata": {
24        "$lastUpdated": "2022-12-22T17:37:57.6117392Z",
25        "$lastUpdatedVersion": 5,
26      },
27      "test1": {
28        "$lastUpdated": "2022-12-22T17:37:57.6117392Z",
29        "$lastUpdatedVersion": 5
30      },
31      "test2": {
32        "$lastUpdated": "2022-12-22T17:37:57.6117392Z",
33        "$lastUpdatedVersion": 5
34      },
35      "test3": {
36        "$lastUpdated": "2022-12-22T17:37:57.6117392Z",
37        "$lastUpdatedVersion": 5
38      }
39    },
40    "$version": 5
41  }
```

Direct Metod:

```

async Task EmergencyStop(string deviceId)
{
    Opcdevice.Stop(deviceId);
    await Task.Delay(1000);
}

1 odwołanie
private async Task<MethodResponse> EmergencyStopHandler(MethodRequest methodRequest, object userContext)
{
    Console.WriteLine($"{methodRequest.Name}");

    var payload = JsonConvert.DeserializeObject<AnonymousType>(methodRequest.DataAsJson, new { machineId = default(string) });

    await EmergencyStop(payload.machineId);

    return new MethodResponse(0);
}

1 odwołanie
async Task ResetErrorStatus(string deviceId)
{
    Opcdevice.Reset(deviceId);
    await Task.Delay(1000);
}

1 odwołanie
private async Task<MethodResponse> ResetErrorStatusHandler(MethodRequest methodRequest, object userContext)
{
    Console.WriteLine($"{methodRequest.Name}");

    var payload = JsonConvert.DeserializeObject<AnonymousType>(methodRequest.DataAsJson, new { machineId = default(string) });

    await ResetErrorStatus(payload.machineId);

    return new MethodResponse(0);
}

1 odwołanie
async Task Maintenance()
{
    Opcdevice.Maintenance();
    await Task.Delay(1000);
}

1 odwołanie
private async Task<MethodResponse> MaintenanceHandler(MethodRequest methodRequest, object userContext)
{
    Console.WriteLine($"{methodRequest.Name}");

    await Maintenance();

    return new MethodResponse(0);
}

1 odwołanie
private static async Task<MethodResponse> DefaultServiceHandler(MethodRequest methodRequest, object userContext)
{
    Console.WriteLine($"{methodRequest.Name}");

    await Task.Delay(1000);

    return new MethodResponse(0);
}

```

Bezpośrednie metody możliwe do wykonania w IoT Explorer:

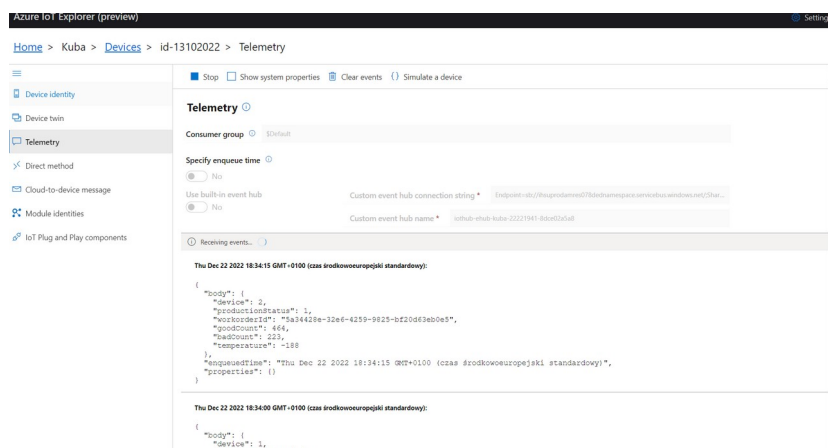
„EmergencyStop”,

„ResetErrorStatus”,

„Maintenance”.

Metody są wywoływane poprzez IoT Explorer lub Microsoft Azure. Poniżej znajduje się zdjęcie przedstawiające działanie i wywołanie metody EmergencyStop.

-Screenshot's:



```
C:\Users\rewiz\Documents\GitHub\IoTProjekt\IoTProjekt\Agent\Agent\bin\Debug\net6.0\Agent.exe
5a34428e-32e6-4259-9825-bf20d63eb0e5
1934
980
-714
Device errors =
Production rate = 50
22.12.2022 18:55:57
1
b850e820-59bc-41ea-96c9-c2093b43368b
1519
754
-602
Device errors =
Production rate = 40
1
5a34428e-32e6-4259-9825-bf20d63eb0e5
1937
981
-611
Device errors =
Production rate = 50
22.12.2022 18:55:59
1
b850e820-59bc-41ea-96c9-c2093b43368b
1526
757
-356
Device errors =
Production rate = 40
```

```
METHOD EXECUTED: EmergencyStop
1
5a34428e-32e6-4259-9825-bf20d63eb0e5
2078
1066
```

Device identity

Device twin

Telemetry

Direct method

Cloud-to-device message

Module identities

IoT Plug and Play components

Invoke method


Direct method ⓘ

Method name *
EmergencyStop

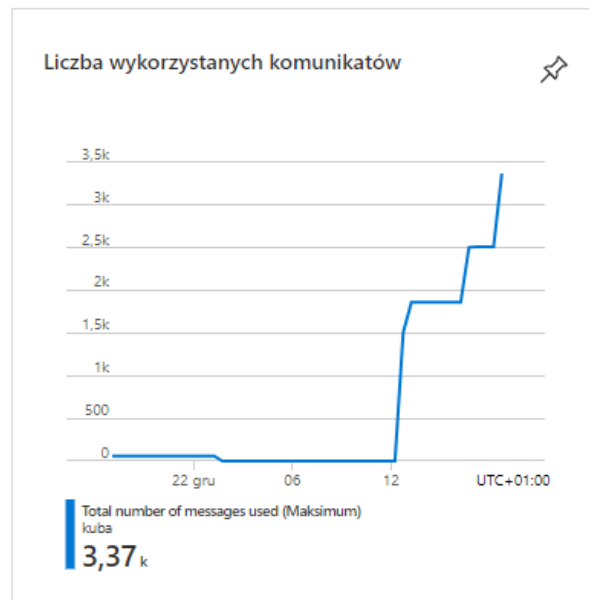
Payload ⓘ
{"deviceId": 1}

Connection timeout in seconds ⓘ
10

Response timeout in seconds ⓘ
10

 Użycie usługi IoT Hub

- Komunikaty wykorzystane dzisiaj: 3388
- Dzienny limit przydziału komunikatów : 8000 ⓘ
- Urządzenia IoT: 1



Links.txt — Notatnik

Plik Edycja Format Widok Pomoc

opc Connection Link:
opc.tcp://localhost:4840/
Primary Connection Parameters:
HostName=Kuba.azure-devices.net;DeviceId=id-13102022;SharedAccessKey=HLYq2gXeTtAC0xSK7Ded4c13n9tCuPvzy8QNqmeP+zY=

Azure IoT Explorer (preview)

File Edit View Window Help

Azure IoT Explorer (preview)

Settings

[Home](#) > Kuba > Devices

+ New

↺ Refresh

🗑 Delete

Query by device ID... 🔍 ➔

🔗 Add query parameter

Device ID	Status	Connection st...	Authenticatio...	Last status up...	IoT Plug and ...	Edge device
id-13102022	Enabled	Disconnected	Sas	--		