Dokumentacja projektowa – Internet of Things w chmurze Azure

1. Połączenie z urządzeniem.

Przed uruchomieniem agenta należy upewnić się, że nasz plik konfiguracyjny – appsettings.json, zawiera poprawne ustawienia dla serwerów OPC Ua, IoT Hub, Azure Storage, Communication Service, ServiceBus oraz ustawiony email do wysyłania wiadomości:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznieObraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie  
  
Urządzenie domyślnie skonfigurowane jest dla dwóch urządzeń. Jeśli chcielibyśmy aby tych urządzeń było więcej, należy dodać kolejne urządzenia, a następnie dodać w sekcji IoTHubConnectionStrings dodatkowe klucze odpowiadające danym urządzeniom.  
Po udanym skonfigurowaniu naszych urządzeń. Możemy odpalić nasz symulator IoT i stworzyć w nim odpowiednią liczbę urządzeń zgodną z naszym plikiem konfiguracyjnym.  
Po uruchomieniu naszej aplikacji nastąpi automatyczne nawiązanie połączenia z serwerem. W przypadku niepowodzenia połączenia, aplikacja wyświetli komunikat o błędzie i zakończy działanie. Ponadto aplikacja zapytana nas o wybranie na jakiej maszynie mamy pracować:

Obraz zawierający tekst, zrzut ekranu, Czcionka, menu

Opis wygenerowany automatycznie  
2. Sposób i częstotliwość odczytu, zapisu danych oraz wywoływania węzłów-metod**.**Dane odczytywane są co wartość TelemetryInterval (w ms) zapisaną w pliku appsettings. (domyślnie wynosi ona 10s):

1. Rodzaj, format i częstotliwość wiadomości wysyłanych przez agenta do IoTHub.  
   Są dwa rodzaje danych wysyłanych do IoTHub:
2. Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

   Opis wygenerowany automatycznieDane telemetryczne odświeżane oraz wysyłane są co 10s. w postaci wiadomości D2C (przykładowe dane):
3. Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

   Opis wygenerowany automatycznieDane dotyczące błędów urządzenia wysyłane jednorazowo za każdym razem gdy nastąpi zmiana w błędach :
4. Rodzaj i format danych przechowywanych w Device Twin

W device twin przechowywane są dane desired oraz reported:  
a) Desired:

b) Reported:



1. Dokumentacja Direct Methods zaimplementowanych w agencie

W Direct Methods możemy wywołać metody EmergencyStop (służącą do zatrzymania urządzenia), ResetErrorStatus (służącą do wyczyszczenia raportu błędów) oraz funkcję SetProductionRate która zmienia wartość ProductionRate na podaną przez nas.

1. EmergencyStop:  
   Obraz zawierający tekst, zrzut ekranu, linia, oprogramowanie

   Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, numer, Czcionka

Opis wygenerowany automatycznieObraz zawierający tekst, zrzut ekranu, Czcionka, czarne

Opis wygenerowany automatycznieStatus: 200 oznacza, że operacja się udała, a tak wygląda to w konsoli:

1. ResetErrorStatus:

Obraz zawierający tekst, zrzut ekranu, linia, numer

Opis wygenerowany automatycznieObraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, wyświetlacz, oprogramowanie

Opis wygenerowany automatycznie

1. Obraz zawierający tekst, zrzut ekranu, wyświetlacz, numer

   Opis wygenerowany automatycznieObraz zawierający tekst, zrzut ekranu, Czcionka

   Opis wygenerowany automatycznieObraz zawierający zrzut ekranu, tekst, linia, numer

   Opis wygenerowany automatycznieSetProductionRate:
2. Kalkulacje i logika biznesowa.

Przy kalkulacjach oraz logice biznesowej użyte zostały takie funkcjonalności platformy Azure jak Azure Stream Analytics, ServiceBus oraz Storage Service.

W kalkulacjach biznesowych mamy za zadanie:

1. Obliczyć % dobrej produkcji, grupowany w 5 – minutowych oknach
2. Co minutę mamy otrzymywać dane na temat średniej, minimalnej oraz maksymalnej temperatury przez ostatnie 5 minut
3. Mamy zapisywać informację kiedy urządzenie doświadczy większej niż 3 ilości błędów poniżej 1 minuty

6.1.1 Obliczanie % dobrej produkcji:  
Zapytanie odpowiedzialne za to w Azure Stream Analytics:  
SELECT

    DeviceId,((SUM(GoodCount) \* 1.0 / (SUM(GoodCount) + SUM(BadCount))) \* 100) AS percentage\_good\_production,

    System.Timestamp AS window\_end

INTO

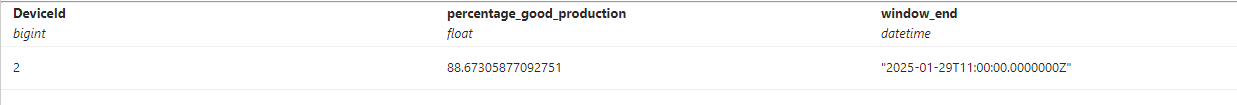
    [productionkpi]

FROM

    [IoTProject2025-2]

GROUP BY

    DeviceId, TumblingWindow(minute, 5);

Przykładowe zwracane dane:  


6.1.2 Średnia, minimalna oraz maksymalna wartość temperatury:

Zapytanie:

SELECT

    DeviceId,

    AVG(temperature) AS avg\_temperature,

    MIN(temperature) AS min\_temperature,

    MAX(temperature) AS max\_temperature,

    System.Timestamp AS window\_end

INTO

    [minmaxavgtemp]

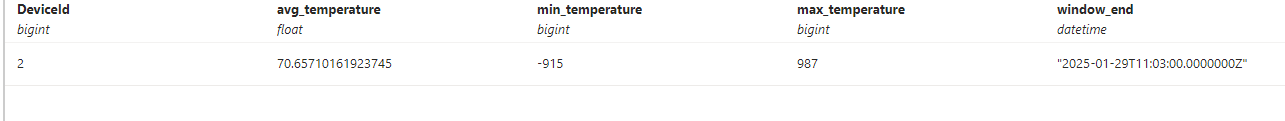
FROM

    [IoTProject2025-2]

GROUP BY

    DeviceId, TumblingWindow(minute, 1);

Przykładowe Dane:



6.1.3 Informacja o ilościach błędów:

Zapytanie:

SELECT

    IoT.DeviceId,

    COUNT(\*) AS total\_error\_count,

    System.Timestamp AS window\_end

INTO

    [errorsamount]

FROM

    [IoTProject2025-2] AS IoT

CROSS APPLY

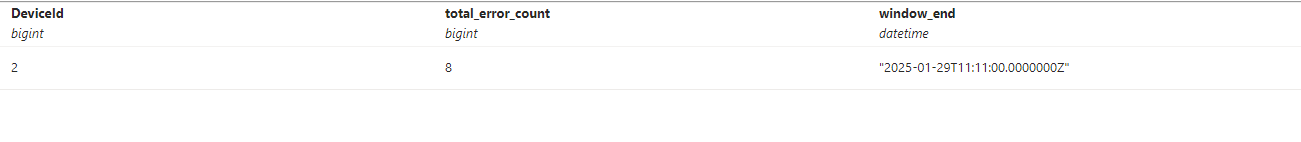
    GetArrayElements(IoT.DeviceErrors) AS Error -- Rozdzielenie tablicy na pojedyncze elementy

GROUP BY

    IoT.DeviceId, TumblingWindow(minute, 1)

HAVING

    COUNT(\*) >= 3;

Przykładowa zawartość:  


W Logice biznesowej mieliśmy za zadanie:

1. Wywoływać funkcję EmergencyStop, gdy maszyna natrafi na więcej niż 3 błędy poniżej minuty
2. Obniżać ProductionRate o 10 punktów, jeśli procent dobrej produkcji będzie wynosił poniżej 90%
3. Wysyłać maila na podany adres, jeśli wystąpi jakikolwiek błąd

Logika biznesowa została wykonana przy pomocy Azure Stream Analytics, ServiceBus oraz Communication Service. W Azure Stream Analytics wywołujemy odpowiednie zapytania, następnie do kolejki trafia informacja o tym, jaką akcję wykonać (obniżyć ProductionRate, wywołać EmergencyStop, czy wysłać maila):

Zapytania dotyczące logiki biznesowej:

1. Obniżanie ProductionRate:

SELECT

    DeviceId,

    'DecreaseProductionRate' AS ActionType

INTO

    [blogicqueueprate]

FROM

    [productionkpi] -- Korzystamy z wyniku pierwszego zapytania

WHERE

    percentage\_good\_production < 90.000; -- Jeśli dobra produkcja spadnie poniżej 90%

Przykładowe działanie zmiany ProductionRate:

Obraz zawierający tekst, Czcionka, zrzut ekranu, czarne

Opis wygenerowany automatycznie

1. Wywoływanie EmergencyStop

SELECT

    DeviceId,

    'EmergencyStop' AS ActionType,

    COUNT(\*) AS total\_error\_count,

    System.Timestamp AS window\_end

INTO

    [blogicqueueerror]

FROM

    [errorsamount]

GROUP BY

    DeviceId, TumblingWindow(minute, 1)

HAVING

    COUNT(\*) >= 3;

1. Wysyłanie maila

SELECT

    DeviceId,

    'SendEmail' AS ActionType,

    COUNT(\*) AS total\_error\_count,

    System.Timestamp AS window\_end

INTO

    [blogicqueue]

FROM

    [errorsamount]

GROUP BY

    DeviceId, TumblingWindow(minute, 1)

HAVING

    COUNT(\*) > 0;

Tak wygląda przykładowy email:

