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.

Obraz zawierający tekst

Opis wygenerowany automatycznie

Obraz zawierający tekst

Opis wygenerowany automatycznie

**Obraz zawierający tekst

Opis wygenerowany automatycznie**

**Obraz zawierający tekst

Opis wygenerowany automatycznie**

**Konfiguracja agenta**

****

**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:

Obraz zawierający tekst

Opis wygenerowany automatycznie

Obraz zawierający tekst

Opis wygenerowany automatycznie

Obraz zawierający tekst

Opis wygenerowany automatycznie

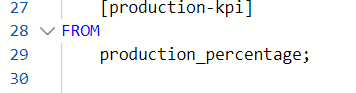


Obraz zawierający tekst

Opis wygenerowany automatycznie

Obraz zawierający tekst

Opis wygenerowany automatycznie



Obraz zawierający tekst

Opis wygenerowany automatycznie

Obraz zawierający tekst

Opis wygenerowany automatycznie

Obraz zawierający tekst

Opis wygenerowany automatycznie

Obraz zawierający tekst

Opis wygenerowany automatycznie

Obraz zawierający tekst

Opis wygenerowany automatycznie

Obraz zawierający tekst

Opis wygenerowany automatycznie

Obraz zawierający tekst

Opis wygenerowany automatycznie

Storagefactory:

Obraz zawierający tekst

Opis wygenerowany automatycznie

Przykłady:

Obraz zawierający stół

Opis wygenerowany automatycznie

Obraz zawierający stół

Opis wygenerowany automatycznie

**Device Twin**

Device Twin, jest używany do aktualizacji zgłoszonych właściwości maszyny.

**Obraz zawierający tekst

Opis wygenerowany automatycznie**

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.

**Obraz zawierający tekst

Opis wygenerowany automatycznie**

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.

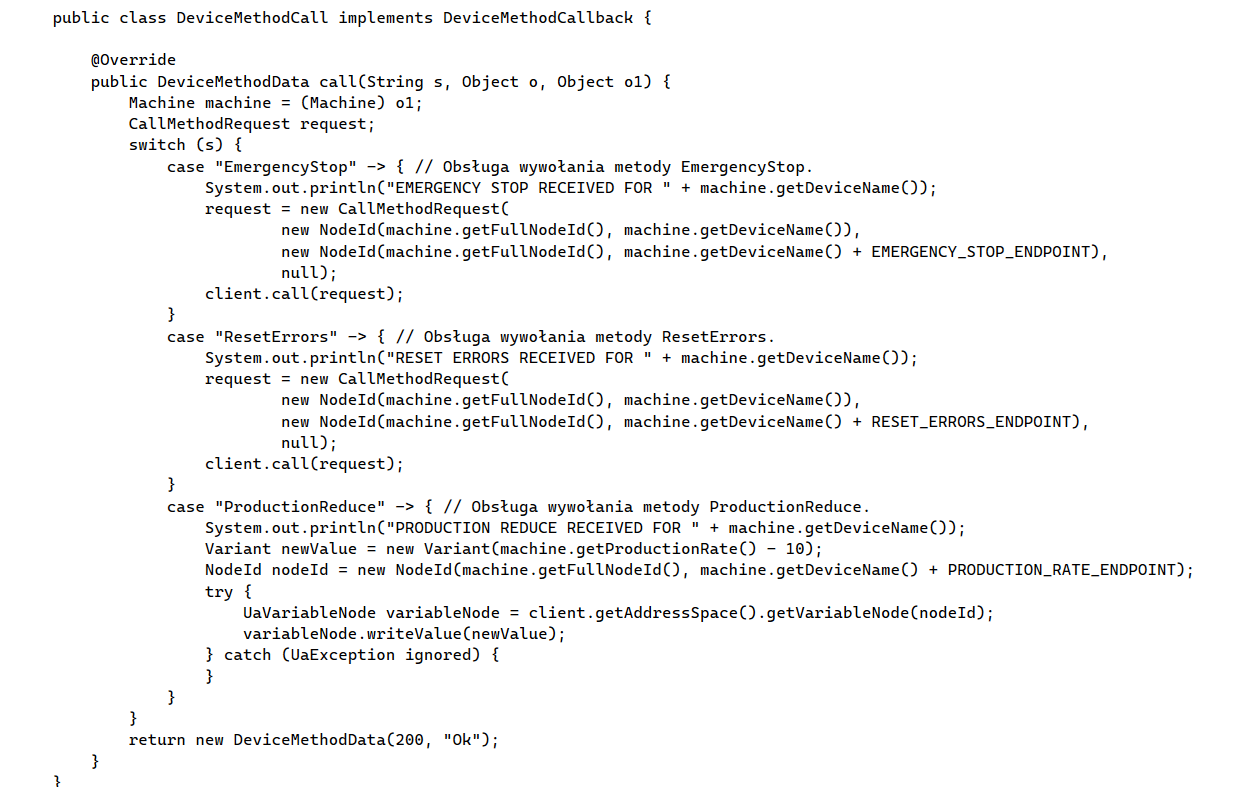
Przykład:

Obraz zawierający tekst

Opis wygenerowany automatycznie

**Metoda bezpośrednia**

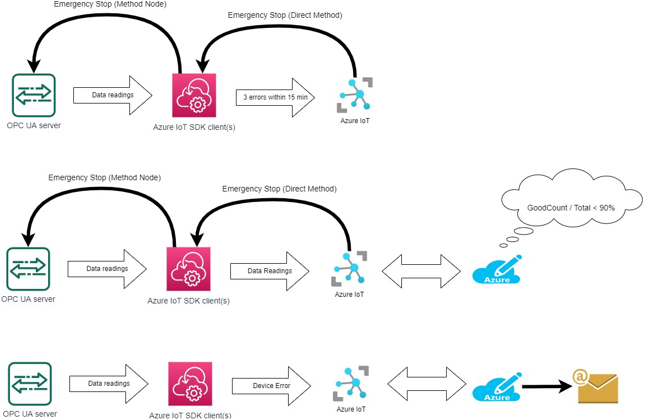
Zaimplementowane są trzy metody bezpośrednie:



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:



**Obraz zawierający tekst

Opis wygenerowany automatycznie**

**Obraz zawierający tekst

Opis wygenerowany automatycznie**

**Obraz zawierający tekst

Opis wygenerowany automatycznie**

**Obraz zawierający tekst

Opis wygenerowany automatycznie**