1. Komunikacja z urządzeniem (serwerem OPC UA)

1.1 Sposób uruchomienia aplikacji

Uruchomienie aplikacji jest możliwe poprzez wykonanie pliku main.py będącego głównym projektem. Należy upewnić się, że wszystkie niezbędne biblioteki są zainstalowane przed uruchomieniem aplikacji.

1.Jak można połączyć się z serwerem OPC UA.

Biblioteka asyncua jest używana do nawiązania połączenia z serwerem OPC UA. Aplikacja używa adresu opcua_endpoint i wywołuje metodę connect() na obiekcie klienta.

1.Metoda odczytu i zapisu danych oraz wywoływania węzłów powinna być określona przez sposób i częstotliwość.

Metoda update_data() jest wywoływana na obiekcie klasy Machine w celu odczytania i aktualizacji danych z serwera OPC UA. Te metody są wywoływane cyklicznie w głównej pętli programu.

_____

2. Konfiguracja agenta

2.Jak agent jest skonfigurowany do komunikacji z określonym serwerem OPC UA oraz instancją IoT Hub.

Agent jest skonfigurowany poprzez ustawienie parametrów, takich jak opcua_endpoint, CONNECTION_STRING do połączenia z IoT Hub oraz STORAGE_CONNECTION_STRING dostępu do usługi blob storage. Te parametry można zdefiniować w pliku konfiguracyjnym lub jako zmienne środowiskowe.

2.Jakie są rodzaj, format i częstotliwość wiadomości (messages) wysyłanych przez agenta do IoT Hub?

Dane D2C są przesyłane w formacie JSON i zawierają informacje o stanie urządzenia oraz ewentualnych błędach. Do przesyłania wiadomości asynchronicznie używamy metody send_message() na obiekcie klienta IoT Hub.

2.Jakie są rodzaje i formaty danych przechowywanych w Device Twin?

Informacje przechowywane w Device Twin obejmują dane dotyczące stanu produkcji oraz błędów urządzenia. To są cechy, takie jak ProductionRate i Errors.

_____

3. Dokumentacja Direct Methods

3.1 Sygnatury metod

emergency_stop. Metoda służy do aktywowania zatrzymania awaryjnego na określonym urządzeniu.

reset_err_status. Metoda służy do wyzerowania stanu błędów na określonym urządzeniu.

_____

4. Obliczenia i logika biznesowa

4.1Funkcje i obliczenia dostępne.

Dane urządzenia są aktualizowane z serwera OPC UA.

Porównywanie danych z urządzenia z danymi pożądanymi bliźniaka i podejmowanie działań w przypadku różnic.

Obsługa żądań Direct Method.

Powiadamianie poprzez wysyłanie e-maili w przypadku wystąpienia błędów na urządzeniu.

Czyszczenie magazynu twin oraz blob storage.

4.2 Sposób implementacji

Asynchroniczna obsługa zdarzeń jest wykorzystywana do implementacji logiki biznesowej. Do wykorzystania są biblioteki, takie jak asyncio, asyncua oraz azure-sdk-for-python. Logika biznesowa zawiera odczyt, aktualizację danych, obsługę żądań Direct Method oraz wysyłanie powiadomień e-mail.

_____

5. Obsługa błędów

5.1 Obsługa błędów połączenia

Jeśli próba połączenia z serwerem OPC UA lub IoT Hub się nie uda, aplikacja wyświetli odpowiednie komunikaty błędów i zakończy działanie. Trzeba sprawdzić, czy adresy serwerów są poprawne i upewnij się, że połączenie internetowe działa.

5.Obsługa błędów przy aktualizacji danych.

Jeśli aktualizacja danych z serwera OPC UA nie powiedzie się, aplikacja może pokazać komunikat błędu lub podjąć działania naprawcze, takie jak ponowna próba aktualizacji danych.

5.Obsługa błędów podczas wysyłania wiadomości D2C.

Jeśli wysłanie wiadomości D2C do IoT Hub nie powiedzie się, aplikacja może ponowić próbę wysłania lub zapisać wiadomość w lokalnym buforze celem późniejszego przesłania.

---

6. Zabezpieczenia

6.1 Zabezpieczenia połączenia

Dzięki wykorzystaniu połączeń zabezpieczonych z serwerem OPC UA i IoT Hub, aplikacja gwarantuje poufność i integralność przesyłanych danych. Podczas transmisji danych są szyfrowane, a autoryzacja odbywa się za pomocą kluczy uwierzytelniających.

6.2 Zabezpieczenia Direct Method

Do autentykacji żądań metody bezpośredniej wykorzystuje się klucze dostępu, co uniemożliwia nieupraw niony dostęp do funkcji sterujących urządzeniami.

---

7. Monitorowanie działania aplikacji

7.1 Logi aplikacyjne pozwalają monitorować działania podejmowane przez aplikację oraz rejestrować ewentualne błędy.

7.2 Raportowanie błędów

W razie pojawienia się błędów aplikacji, informacje o tych błędach mogą być przekazywane do ustalonego systemu raportowania w celu szybkiego rozwiązania problemów.

---

8. Wymagania systemowe

8.1 Wymagania sprzętowe

Aplikacja może działać na każdym komputerze lub urządzeniu spełniającym wymagania systemowe bibliotek i usług wykorzystywanych w projekcie.

8.2 Wymagania software'owe

Aby uruchomić aplikację, należy zainstalować odpowiednie biblioteki* i zależności wymienione w pliku wymagań projektu oraz mieć dostęp do internetu w celu komunikacji z serwerami OPC UA i IoT Hub.

*Biblioteki:
aiofiles==23.2.1
aiosqlite==0.19.0
anyio==4.2.0
asyncua==1.0.6
azure-common==1.1.28
azure-communication-email==1.0.0
azure-core==1.29.6
azure-iot-device==2.12.0
azure-iot-hub==2.6.1
azure-mgmt-core==1.4.0
azure-storage-blob==12.19.0
certifi==2023.11.17
cffi==1.16.0
charset-normalizer==3.3.2
cryptography==41.0.7
deprecation==2.1.0
idna==3.6
isodate==0.6.1
janus==1.0.0
msrest==0.7.1
oauthlib==3.2.2
packaging==23.2
paho-mqtt==1.6.1
pycparser==2.21
pyOpenSSL==23.3.0
PySocks==1.7.1
python-dateutil==2.8.2
pytz==2023.3.post1
requests==2.31.0
requests-oauthlib==1.3.1
requests-unixsocket==0.3.0
six==1.16.0
sniffio==1.3.0
sortedcontainers==2.4.0
typing_extensions==4.9.0
uamqp==1.6.6
urllib3==1.26.18