Politechnika Śląska w Gliwicach Wydział Automatyki, Elektroniki i Informatyki

Projektowanie Systemów Cybernetyczno-Fizycznych

Przetwarzanie danych ze sterownika PLC w chmurze

Autorzy	Krzysztof Ból, Dawid Suchy, Łukasz Latusik
Prowadzący	mgr inż. Ireneusz Smołka
Rok akademicki	2019/2020
Kierunek	Informatyka
Rodzaj studiów	SSI
Semestr	6
Specjalność	ISMiP
Sekcja	11
Data oddania raportu	2020-05-25

1. Wprowadzenie

Przetwarzanie informacji w chmurze jest zagadnieniem, które już na stałe zagościło w wielu dziedzinach informatyki i stało się jej nieodłączną częścią. Większość tworzonych dzisiaj aplikacji o zastosowaniu komercyjnym jest wytwarzane z myślą o uruchomieniu ich w architekturze chmurowej. Głównymi zaletami przetwarzania chmurowego są m.in. duża moc obliczeniowa czy brak konieczności obsługi infrastruktury informatycznej, np. serwerów, na których działa aplikacja. Obecnie na rynku popularne są rozwiązania PaaS, SaaS oraz IaaS.

W przypadku informatyki przemysłowej wykorzystanie technologii chmurowych jest dość nowym rozwiązaniem. Fabryki, zakłady przemysłowe oraz inne miejsca wykorzystujące sterowniki PLC lub inne urządzenia tego typu nie są miejscami, które korzystają z najnowszych technologii. Liczy się tam niezawodność oraz brak awaryjności zastosowanego rozwiązania, dlatego często decyduje się na wykorzystanie starszych, ale sprawdzonych technologii. Należy jednak zauważyć, że technologia przetwarzania chmurowego zaczyna się wkradać w świat informatyki przemysłowej oraz urządzeń IoT, co pokazuje jak istotne jest to zagadnienie.

Przykładem wykorzystania możliwości obróbki danych pochodzących z czujników oraz zarządzania systemem przemysłowym może być przygotowany scenariusz obsługi stacji uzdatniania wody z wykorzystaniem urządzenia firmy Beckhoff EK9160, które umożliwia przesył danych z czujników systemu przemysłowego do dostawcy usług chmurowych, czy też przesył sygnałów sterujących z powrotem do systemu przemysłowego. Daje to możliwość wytworzenia informacji sterującej na podstawie otrzymanych i przetworzonych danych wprowadzając połączenie pomiędzy światem chmurowym oraz sterownikami PLC.

1.1. Treść zadania do wykonania

Na podstawie zebranych pomiarów dobierana ma być reakcja urządzeń dozujących środki chemiczne. W chmurze dane powinny być przetwarzane w taki sposób, aby możliwe było jak najlepsze wysterowanie urządzeń. Na rzeczywistym obiekcie takie wysterowanie nie zawsze jest od razu możliwe ze względu na odległość pomiędzy urządzeniami dozującymi, a miejscem pomiaru. W związku z tym do wyliczeń powinny być brane również takie wartości jak odległość dozowania od pomiaru oraz parametry danej partii dozowanego preparatu. System PLC domyślnie jest połączony z systemem nadrzędnym DCS.

Mierzone parametry:

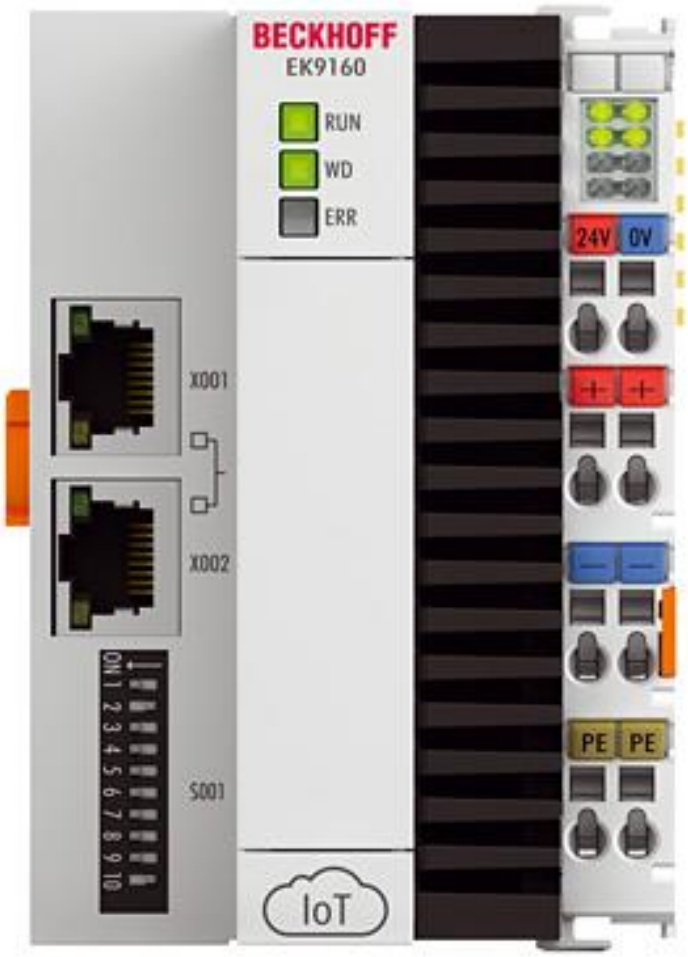
- pomiar pH,
- pomiar temperatury,
- pomiar zawartości wprowadzanych środków chemicznych.

Monitorowane elementy:

- stan filtrów (węglowy, żwirowy),
- stan pracy urządzeń dodatkowych (np. odwróconej osmozy),
- stan pracy pomp z falownikami.

2. EK9160 i Microsoft Azure

Beckhoff EK9160 – urządzenie typu „IoT Bus Coupler", które zapewnia bezpośrednie połączenie pomiędzy urządzeniami sterującymi pobierając dane z czujników oraz wysyłając sygnały sterujące bez konieczności pisania programu. Dane mogą być pobierane za pomocą magistrali EtherCat, CAN lub PROFIBUS. Urządzenie komunikuje się z umieszczonym po stronie dostawcy chmurowego brokerem obsługującym protokół MQTT. Dane dostarczane są do urządzenia za pomocą przystawek I/O takich jak EL2009 lub EL1819. Urządzenie posiada certyfikat Microsoft Azure oraz Amazon AWS gwarantujący kompatybilność.



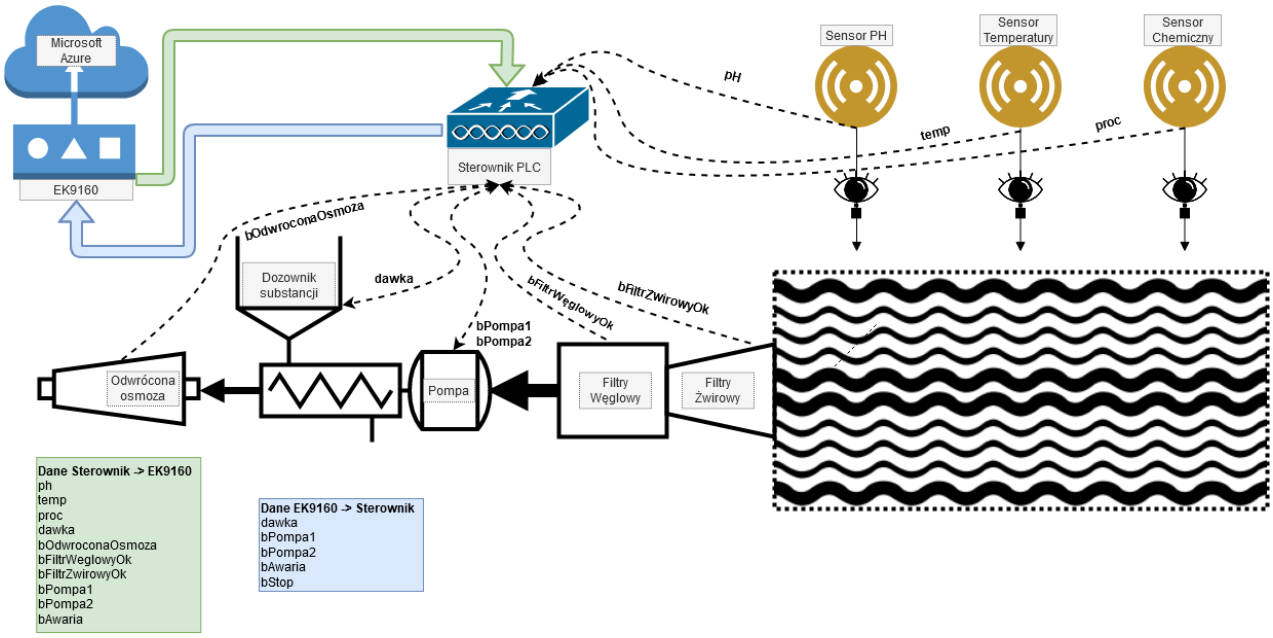
Rys. 1. EK9160

Microsoft Azure – platforma chmurowa firmy Microsoft oparta o model PaaS. Dostarcza ona wielu rozbudowanych funkcji, które umożliwiają rozwijanie aplikacji chmurowych. Do przykładowych zastosowań należą:

- Tworzenie skonteneryzowanych aplikacji, wykorzystujących skalowanie za pomocą Azure Kubernetes Service.
- Wykorzystywanie mocy obliczeniowej serwerów platformy Azure do wykonywania obliczeń związanych z uczeniem maszynowym lub sztuczną inteligencją np. Azure Machine Learning.
- Wykorzystywanie mocy obliczeniowej platformy Azure za pośrednictwem utworzonych maszyn wirtualnych.
- Tworzenie rozwiązań z dziedziny IoT np. Azure IoT Central.

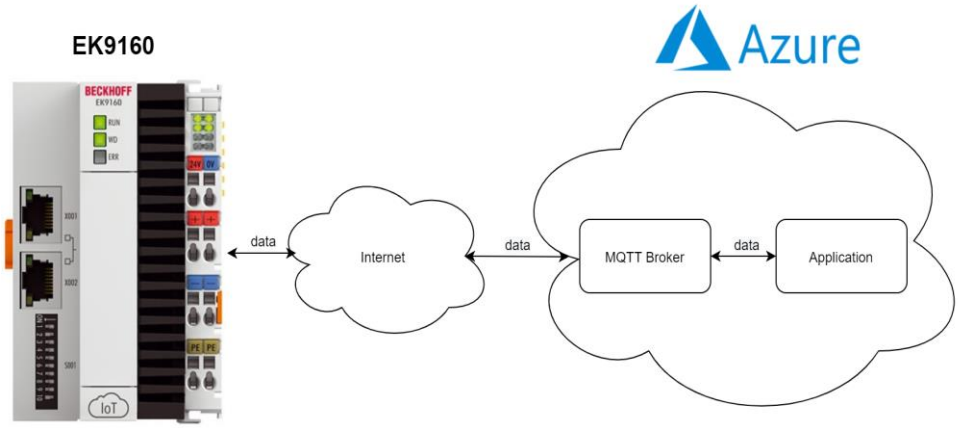
3. Przykładowa wizualizacja systemu.

Do połączenia systemu z EK9160 mogą posłużyć moduły I/O kompatybilne z systemem EtherCat np. EL1819. Dane będą dostarczane z jednego lub wielu sterowników PLC kontrolujących odpowiednie części systemu oraz dostarczające dane pomiarowe. Zmiana parametrów może odbywać się po przeanalizowaniu danych wejściowych za pomocą wytworzonych w aplikacji chmurowej sygnałów sterujących.



Rys. 2. Wizualizacja systemu

4. System pracujący w chmurze



Rys. 3. Schemat połączenia EK9160 z platformą Azure

4.1. Protokół MQTT

Najczęściej stosowanym rozwiązaniem umożliwiającym komunikację z systemami chmurowymi jest protokół MQTT. Jest to bardzo prosty i lekki protokół transmisji danych oparty o wzorzec publikacja/subskrypcja. Sprawdza się on idealnie w Internecie rzeczy oraz przy połączeniach między maszynami, a także tam, gdzie wymagana jest oszczędność energii. Zgodnie z nim klienci nie komunikują się bezpośrednio, lecz w oparciu o element pośredniczący, który jest nazywany brokerem. Pełni on rolę serwera, z którym klienci łączą się aby za jego pośrednictwem publikować informacje. Dzięki temu możliwe jest udostępnianie danych innym klientom bez znajomości ich adresu IP. Dane publikowane przy użyciu protokołu MQTT są danymi tekstowymi, więc z łatwością można przekształcić je do formatu JSON. Na potrzeby projektu wybraliśmy właśnie ten protokół, ze wszystkich obsługiwanych przez koncentrator danych IoT EK9160, ze względu na ilość dostępnej dokumentacji.

4.2. Broker MQTT w chmurze

Aby komunikacja była możliwa, broker MQTT musi zostać umieszczony na platformie chmurowej Azure. Jest na to wiele sposobów. Do komunikacji przy użyciu protokołu MQTT Microsoft proponuje usługę IoT Hub. Usługa ta ma jednak wiele wad, między innymi nie jest w pełni funkcjonalnym brokerem MQTT i nie obsługuje standardu MQTT 3.1.1. Według nas lepszym rozwiązaniem jest skorzystanie z usługi Azure Virtual Machines. Na maszynie wirtualnej z systemem Windows 10 bez problemu można przeprowadzić instalację popularnego brokera Eclipse Mosquitto, który nie ma problemu z obsługą nowszych wersji protokołu. Rozwiązanie to jest proste w użyciu, jednak nie jest idealne – jest ono z pewnością bardziej zawodne od usługi IoT Hub. System Windows jest przeznaczony głównie do użytku codziennego i czasami posiada błędy powodujące tzw. Niebieski Ekran śmierci. W związku z tym rozwiązanie to nie jest często stosowane w przemyśle, jednak na potrzebny tego projektu jest wystarczające.

W celu podłączenia EK9160 do brokera należy skorzystać z menedżera urządzeń Beckhoff dostępnego z poziomu przeglądarki internetowej. W zakładce IoT należy ustawić odpowiedni typ połączenia (General MQTT), typ danych (JSON) oraz adres IP brokera. Następnie należy przejść do zakładki I/O i wybrać, z którego modułu oraz jakie dane chcemy udostępniać.

5. Aplikacja chmurowa

Jako platformę naszej aplikacji wybraliśmy znaną nam szkielet tworzenia aplikacji Spring Boot. Bazuje on na języku programowania Java i posiada wiele rozwiązań ułatwiających tworzenie aplikacji chmurowych. Jego głównymi zaletami są łatwość uruchamiania, automatyczna konfiguracja oraz szybkość procesu tworzenia.

Platforma ta posiada możliwość integracji z protokołem MQTT dzięki wykorzystaniu biblioteki Eclipse Paho MQTT Client. Aby odbierać i wysyłać wiadomości należy zaimplementować dostępny interfejs IMqttClient, w którym zawarte są wszystkie metody potrzebne do nawiązania połączenia oraz komunikacji z brokerem MQTT. Można także skorzystać z proponowanych przez twórców biblioteki gotowych implementacji: synchronicznej lub asynchronicznej.

5.1. Opis przesyłanych danych

Dane są przesyłane w formacie JSON. Zawierają wszystkie mierzone parametry oraz stan monitorowanych elementów. Przykładowa wiadomość wysłana do brokera MQTT:

```
{
	"pH": 10,
	"temp": 40,
	"proc": 7,
	"dawka": 2.3,
	"bOdwroconaOsmoza": "false",
	"bFiltrWeglowyOk": "true",
	"bFiltrZwirowyOk": "true",
	"bPompa1": "true",
	"bPompa2": "false",
	"bAwaria": "false"
}
```

Szczegółowy opis poszczególnych pól:

1. *pH* – zmierzona wartość stopnia kwasowości wody.
2. *temp* – zmierzona wartość temperatury wody. Pomiar wykonywany jest w skali Celsjusza.
3. *proc* – pomiar procentowej zawartości wprowadzanych środków chemicznych.

4. *dawka* – aktualna procentowa dawka wprowadzanych środków chemicznych.
5. *bOdwroconaOsmoza* – zmienna określająca stan pracy urządzeń odpowiedzialnych za proces odwróconej osmozy. Wartość *true* oznacza, że urządzenia pracują, natomiast wartość *false*, że urządzenia nie pracują.
6. *bFiltrWeglowyOk* oraz *bFiltrZwirowyOk* – zmienna określająca stan filtrów, odpowiednio węglowego oraz żwirowego. Wartość *false* oznacza, że dany filtr jest zużyty i należy go wymienić.
7. *bPompa1* oraz *bPompa2* – zmienne określające stan pracy pomp z falownikami. Wartość *true* oznacza, że urządzenia pracują, natomiast wartość *false*, że urządzenia nie pracują.
8. *bAwaria* – zmienna określająca czy wystąpiła awaria któregoś z elementów systemu.

6. Opis i funkcjonalność serwisu

Rolę klienta pełni aplikacja napisana w języku JAVA z użyciem szkieletu tworzenia Spring Framework. Komunikacja MQTT jest implementowana z użyciem bibliotek udostępnionych przez projekt *Eclipse Paho*.

Aplikacja jest subskrybentem, nasłuchuje na brokerze cyklicznie pobierając dane ze sterownika oraz pobiera polecenia od użytkownika i zapisuje do listy. Rozmiar listy jest stosowny do ustalonego przedziału czasowego, z którego dane chcemy przechowywać. Aplikacja dane te przesyła jako polecenie do sterownika gdy lista się zapełni lub wystąpi informacja, którą trzeba obsłużyć natychmiast, na przykład wskazująca na awarię systemu. Po przekazaniu żądań lista jest usuwana.

Dane, które obsługuje aplikacja, opisane są w poprzednim punkcie. Na zmianę wartości poszczególnych pól serwis reaguje inaczej:

1. *pH* – jeżeli wartość tego pola jest zbyt wysoka, aplikacja wysyła sygnał zmniejszenia dawki. Jeżeli jest zbyt niska, wysyła sygnał zwiększenia dawki.
2. *proc* – jeżeli wartość tego pola jest zbyt wysoka, aplikacja wysyła sygnał zmniejszenia dawki. Jeżeli jest zbyt niska, wysyła sygnał zwiększenia dawki.
3. *bOdwroconaOsmoza* – jeżeli wartość tego pola wynosi false, aplikacja wysyła sygnał wyłączenia obu pomp.
4. *bFiltrWeglowyOk* – jeżeli wartość tego pola wynosi false, aplikacja wysyła sygnał bAwaria.
5. *bFiltrZwirowyOk* – jeżeli wartość tego pola wynosi false, aplikacja wysyła sygnał bAwaria.
6. *bPompa1* oraz *bPompa2* – zmienne określające stan pracy pomp z falownikami. Wartość *true* oznacza, że urządzenia pracują, natomiast wartość *false*, że urządzenia nie pracują.

7. *bAwaria* – jeżeli wartość tego pola wynosi true, aplikacja wyłącza pompy.

7. Możliwości korzystania z AZURE na podstawie licencji POLSL.

Dzięki porozumieniu między Politechniką Śląską a firmą Microsoft każdy pracownik i student posiada możliwość pobierania i korzystania z profesjonalnych narzędzi programistycznych i projektowych, oprogramowania oraz usług udostępnionych przez tę firmę. Umowa ta obejmuje opisywaną wcześniej platformę chmurową Azure, która pozwala na uzyskanie 200 USD bezpłatnych środków do wykorzystania na produkty i usługi tej platformy oraz możliwości korzystania z popularnych bezpłatnych usług przez 12 miesięcy.

8. Interfejs użytkownika

Interfejs składa się z trzech głównych komponentów: wizualizacji, monitorowania oraz wprowadzania. Komponent wizualizacji obrazuje zbiornik wodny. Komponent monitorowania wyświetla informacje - wartości mierzonych parametrów wraz z diodami wyświetlającymi stan filtrów, stan pracy urządzeń oraz poziom wody w zbiorniku.