Wyniki testów i wnioski

Obecny przemysł zyskuje coraz więcej na wprowadzanej sukcesywnie od dawna cyfryzacji. Daje ona mnogość możliwości choćby w zbieraniu różnorodnych danych, które można wykorzystać np. Przy optymalizacji czy modernizacji urządzeń. Jak jednak powszechnie wiadomo nie ma nic za darmo. Wdrożenie takich usprawnień niesie ze sobą duże zapotrzebowanie na przesył danych a to oznacza jedno: wyższe koszty, gdyż wdrożenie podobnych protokołów na własną rękę budując je od zera wymaga ogromnego nakładu pracy i dodatkowych pracowników. Poza tym należy również liczyć się z ograniczeniami płynącymi z możliwości łączy. Całe szczęście, pod koniec XX wieku opracowano protokół MQTT, który skutecznie zniwelował częściowo owe bolączki.

Rozwinięcie skrótu to Message Queue Telemetry Transport ale podaję to tylko jako ciekawostkę, gdyż dużo więcej mówi nam o nim jego definicja, która brzmi: lekki protokół transmisji danych oparty o wzorzec: publikacja/subskrypcja. Jego zasada działanie jest stosunkowo prosta (schemat działania mqtt) : Jego główny element to broker, który można porównać do serwera. Jest on swego rodzaju dyspozytorem wiadomości przesyłanych pomiędzy połączonymi z nim urządzeniami. Urządzenia te dzielą się w zależności od tego czy wysyłają czy oczekują wiadomości. Kolejno są to publisher i subscriber. Tak więc cały proces polega na tym, że urządzenie 1 subskrybuje dany topic zdefiniowany w brokerze, a broker gdy otrzyma wiadomość od urządzenia 2 na ten sam topic, podaje ją subskrybentowi.

Zacznijmy może od rdzenia protokołu czyli naszego brokera. Na rynku dostępnych jest wiele opcji takich jak Mosca, HiveMQ, RabbitMQ, lecz w naszym projekcie użyliśmy brokera mosquitto. Wybraliśmy go głównie z trzech powodów: Jest on po pierwsze jednym z najpopularniejszych brokerów. Po drugie jest darmowy, a po trzecie jest wykorzystywany w showroomie Bosch rexroth.

Drugą bardzo istotną kwestią są biblioteki za pomocą których można wykorzystać nasz protokół w systemach i programach funkcjonalnych. Jak Państwo wiecie w naszym projekcie łączymy zarówno programowanie w języku Python jaki i programowanie sterowników PLC w języku Structure text. W pierwszym z nich posługiwaliśmy się biblioteką o nazwie Paho MQTT , która umożliwia ustawienie połączenia i komunikację z brokerem. Poza tym dodaje kilka użytecznych funkcji i metod, dzięki którym możemy wprowadzać do protokołu różne parametry takie jak QOS, retain oraz keep alive, o których opowiemy w swoim czasie. Natomiast w przypadku programowania sterownika (biblioteka Stefana Barona, biblioteka Janztec i biblioteka Bosch Rexroth UK). Po przeprowadzeniu testów wybór padł na te ostatnią, bo cechowała ją względna przejrzystość, i naszym zdaniem jedna z prostszych składni. Warto jednak wspomnieć, że biblioteka Stefana Barona mimo, że według nas najmniej przystępna dla początkujących, była obsługiwana bezpośrednio przez sterownik w serwonapędzie, a dwie pozostałe wymagały użycia pośredniego sterownika.

Przy okazji opisywania bibliotek warto zapoznać się z kilkoma pojęciami, obecnymi wewnątrz nich. Pierwszym jest: QOS czyli quality of sevice. Odnosi się ono do klasy w jakiej wiadomość została przeslana, co w uproszczeniu odnosi się do stopnia zagwarantowania dostarczenia wiadomości. Rozróżniamy trzy takie stopnie: 0, 1 i 2. Pierwszy z nich najniższy cechuje się tym, że nie ma żadnego potwierdzenia czy wiadomość dotarła do subskrybenta. Drugi z nich gwarantuje za to, że wiadomość dotrze do adresata co najmniej raz a nadawca przechowuje wysłaną wiadomość do momentu aż nie otrzyma potwierdzenia odbioru od subskrybenta. Jednakże w razie gdy np. Z powodu przeciążenia sieci, wiadomość nie zdąży dotrzeć przed czasem przewidzianym na potwierdzenie przez subskrybenta, zostanie wysłana jeszcze raz. Może to powodować jej zdublowanie. Natomiast klasa druga oznacza już, że wiadomość została wysłana dokładnie raz i podlega podwójnemu potwierdzeniu aby nie została zdublowana.

Kolejnym pojęciem które warto znać jest retain, dzięki któremu po odłączeniu brokera od sieci zapisana zostanie ostatnia wiadomość.

Keep alive to

Lastwill (topic/message)

Stworzyliśmy komunikację pomiędzy komputerem PC a sterownikiem PLC:

- publikacja wiadomości przez PC na dany topic do brokera, odebranie tej wiadomości z brokera przez sterownik PLC. Po odebraniu tej wiadomości PLC publikuje wiadomość na inny topic, który to jest subskrybowany – wiadomość jest oczekiwana przez PC

Takie rozwiązanie umożliwiło nam przeprowadzenie dwóch testów - średni czas przesyłu informacji pomiędzy PC a PLC oraz maksymalna długość wysyłanej wiadomości – polecenia, o czym później

2 Sterowaliśmy serwonapędem, podając polecenia w Pythonie:

- w kodzie Pythona łączymy się z brokerem i sprawdzamy czy połączenie z PLC – serwonapędem jest aktywne. Jeśli tak, to umożliwiamy użytkownikowi na podanie różnych komend:

* RUN – aktywacja serwonapędu
* RVliczba – zadanie prędkości i obrotów w prawą stronę
* LVliczba - zadanie prędkości i obrotów w lewą stronę
* STOP – zatrzymanie obrotów serwonapędu
* START – umożliwienie uruchomienia serwonapędu po komendzie STOP
* OFF – odcięcie serwonapędu od zasilania – awaryjny STOP

3 Sterowaliśmy napędem, wybierając napisane wcześniej sekwencje w Pythonie:

Zaletą takiego rozwiązania jest ingerencja tylko w zewnętrzny plik. Kilka linijek kodu zmieniamy i możemy dowolnie zaprogramować nasz serwonapęd. Pisanie kodu w Pythonie jest znacznie prostsze niż w PLC

Pod koniec prezentacji będą mogli Państwo przetestować naszą pracę

W naszym projekcie nie wykorzystywaliśmy stricte linii Showroom, ale korzystaliśmy ze sterownika PLC umieszczonego na linii oraz głównego brokera linii. Pozwoliło to nam na komunikację wyszczególnioną wcześniej PLC – PC.

Odbieraliśmy dane z czujników ()

Testy jakie przeprowadziliśmy dotyczyły czasu przesyłu informacji oraz maksymalnej ilości znaków.

Przesył informacji pomiędzy brokerem a PC był zróżnicowany, jak wspomnieliśmy wcześniej QoS wpływa na czas, jak widać w tabelce różnica pomiędzy QoS 0 a QoS 2 jest 60 krotna

Podobnie próbowaliśmy zmierzyć czas pomiędzy wysłaniem wiadomości z PC do brokera, odebranie jej na sterowniku PLC i przeslanie innej do brokera, z którego odebraliśmy ja na PC. Test się nie powiódł, ponieważ broker z linii był prawdopodobnie zbyt obciążony, a mieliśmy problem, żeby postawić broker lokalny

Innym testem była maksymalna długość ciągu znaków jaką może odebrać PLC z brokera w jednej wiadomości. Teoretyczna maksymalna wartość to 128, ale nam udało się wysłać najdłuższy string o długości 116 znaków, możliwe że PLC zajmuje pozostałe bity na coś innego. Test przeprowadziliśmy na jednej bibliotece PLC – Bosch Rexroth UK

Wnioski:

1. MQTT jest wartym uwagi protokołem komunikacji, gdyż można korzystać z niego za darmo, pozwala wysyłać polecenia znacznie łatwiej dzięki brokerowi.
2. QOS znacząco na czas przesyłu informacji pomiędzy urządzeniem i brokerem.
3. Wprowadzanie modyfikacji w pythonie jest dużo prostsze niż w plc dlatego też.
4. Nie wprowadzaliśmy zabezpieczeń dostępu, mimo, że mqtt daje ku temu narzędzia takie jak hasło czy szyfrowanie.
5. Jedynym problemem dotyczącym mqtt jest znikoma ilość informacji w internecie dotyczących stosowania go w sterownikach PLC.