Pomiary Automatyka Robotyka, ISSN 1427-9126, R. 24, Nr 1/2020, 35–40, DOI: 10.14313/PAR_235/35

Poprawa wydajności komunikacji sterownika przemysłowego z panelem operatorskim HMI w środowisku inżynierskim CPDev

Dariusz Rzońca
Politechnika Rzeszowska, Katedra Informatyki i Automatyki, ul. W. Pola 2, 35-959 Rzeszów

Streszczenie: Panele operatorskie HMI są powszechnie stosowane w systemach automatyki przemysłowej. Pozwalają na wizualizację sterowania procesem przemysłowym, jak również na zmianę parametrów. Wartości prezentowane operatorowi na ekranie procesowym pochodzą ze sterownika, bądź sterowników, połączonych łączem komunikacyjnym. Wydajna komunikacja między HMI a sterownikiem jest niezbędna dla prawidłowego działania systemu automatyki. W artykule przeanalizowano kilka aspektów takiej komunikacji i zaproponowano rozwiązania pozwalające na skrócenie czasu cyklu komunikacyjnego. Opisane metody zostały zaimplementowane w pakiecie inżynierskim CPDev.

Słowa kluczowe: sterownik przemysłowy, PLC, panel operatorski, HMI, komunikacja

1. Wprowadzenie

Środowisko inżynierskie CPDev [1, 2] jest opracowanym w Katedrze Informatyki i Automatyki Politechniki Rzeszowskiej zestawem narzędzi pozwalającym na programowanie sterowników PLC (ang. *Programmable Logic Controller*) i PAC (ang. *Programmable Automation Controller*) w językach normy IEC 61131-3, przyjętej w Polsce jako PN-EN 61131-3 [3]. Norma ta definiuje pięć języków programowania, dwa tekstowe: ST (ang. *Structured Text*) i IL (ang. *Instruction List*), dwa graficzne: FBD (ang. *Function Block Diagram*) i LD (ang. *Ladder Diagram*) oraz mieszany język SFC (ang. *Sequential Function Chart*). Środowisko CPDev pozwala na programowanie w każdym z wymienionych języków. Pakiet CPDev został wdrożony w kilku rozwiązaniach technicznych, przygotowanych przez polskich i zagranicznych producentów urządzeń automatyki przemysłowej.

Pakiet CPDev zawiera narzędzie CPVis pozwalające na przygotowanie ekranów procesowych dla panelu operatorskiego HMI (ang. *Human-Machine Interface*). Wizualizacja przebiegu procesu sterowania na panelu HMI jest rozwiązaniem często stosowanym w systemach automatyki. Charakterystyczną cechą CPDev jest przygotowywanie wizualizacji w powiązaniu z programem sterowania. Wysokopoziomowe mechanizmy wizualizacyjne są niezależne od sprzętu, mogą być wdrożone na docelowej platformie sprzętowej, po opracowaniu zestawu niskopoziomowych funkcji graficznych. Wąskim gardłem może być jednak interfejs komunikacyjny. Wymagania wydajnościowe były motywacją do przeprowadzenia opisanych w artykule badań i przetestowania proponowanych rozwiązań w praktyce.

W kolejnym rozdziale przedstawiono krótki przegląd literatury powiązanej z tematyką artykułu. Trzeci rozdział zwięźle przedstawia środowisko inżynierskie CPDev, zwłaszcza w kontekście przygotowywania wizualizacji dla paneli HMI. Kolejny z rozdziałów poświęcony jest problematyce wymiany danych między sterownikiem a HMI i możliwym optymalizacjom. Ostatni z rozdziałów podsumowuje artykuł.

2. Przegląd literatury

Wymianę danych między sterownikiem a pozostałymi urządzeniami automatyki, w tym panelami HMI, zazwyczaj zapewnia dedykowana przemysłowa sieć komunikacyjna [4]. Typowo bazuje ona na magistrali polowej [5] i jednym z przemysłowych protokołów komunikacyjnych zdefiniowanych w normie IEC 61158 [6]. W literaturze opisywano także inne, niestandardowe rozwiązania, na przykład korzystające z protokołu HTTP (ang. *Hypertext Transfer Protocol*) zamiast specjalizowanego protokołu przemysłowego [7, 8], ale nie są one zazwyczaj stosowane w praktyce.

Komunikacja w systemach automatyki przemysłowej może korzystać z różnych paradygmatów dostępu do łącza komunikacyjnego, takich jak na przykład producent-dystrybutor-konsument, przekazywanie znacznika (ang. *token passing*), czy najczęściej stosowany paradygmat *master-slave* [9]. Każdy z nich bazuje na multipleksowaniu z podziałem czasu TDM (ang. *Time Division Multiplexing*) definiując szczeliny czasowe, gdy poszczególne urządzenia mają dostęp do łącza. Przykła-

Autor korespondujący:
Dariusz Rzońca, drzonca@prz-rzeszow.pl

Artykuł recenzowany
nadesłany 16.10.2019 r., przyjęty do druku 27.11.2019 r.

Zezwala się na korzystanie z artykułu na warunkach licencji Creative Commons Uznanie autorstwa 3.0

dowo w paradygmacie *master-slave* jedno z urządzeń podłączonych do wspólnej magistrali jest urządzeniem nadrzędnym, inicjującym transmisję, a pozostałe urządzeniami podrzędnymi, mogącymi przesłać komunikat jedynie w odpowiedzi na polecenie *mastera*. Alternatywne podejście, korzystające z multipleksowania z podziałem częstotliwości FDM (ang. *Frequency Division Multiplexing*) jest rozważane w literaturze [10], ale nie zostało przyjęte w praktyce.

Zasady graficznego projektowania aplikacji sterowania, do których należą także ekrany procesowe HMI, przedstawiono w pracach [11, 12]. Obszaru tego dotyczy także standard VDI/VDE 3699 [13]. Sposoby uwzględnienia czynnika ludzkiego podczas projektowania HMI przedstawiono w pracach [14, 15]. Pierwsze wersje rozszerzeń pakietu inżynierskiego CPDev o możliwość projektowania interfejsów graficznych opisano w [16, 17]. Aktualną funkcjonalność w tym zakresie pokazano w [18].

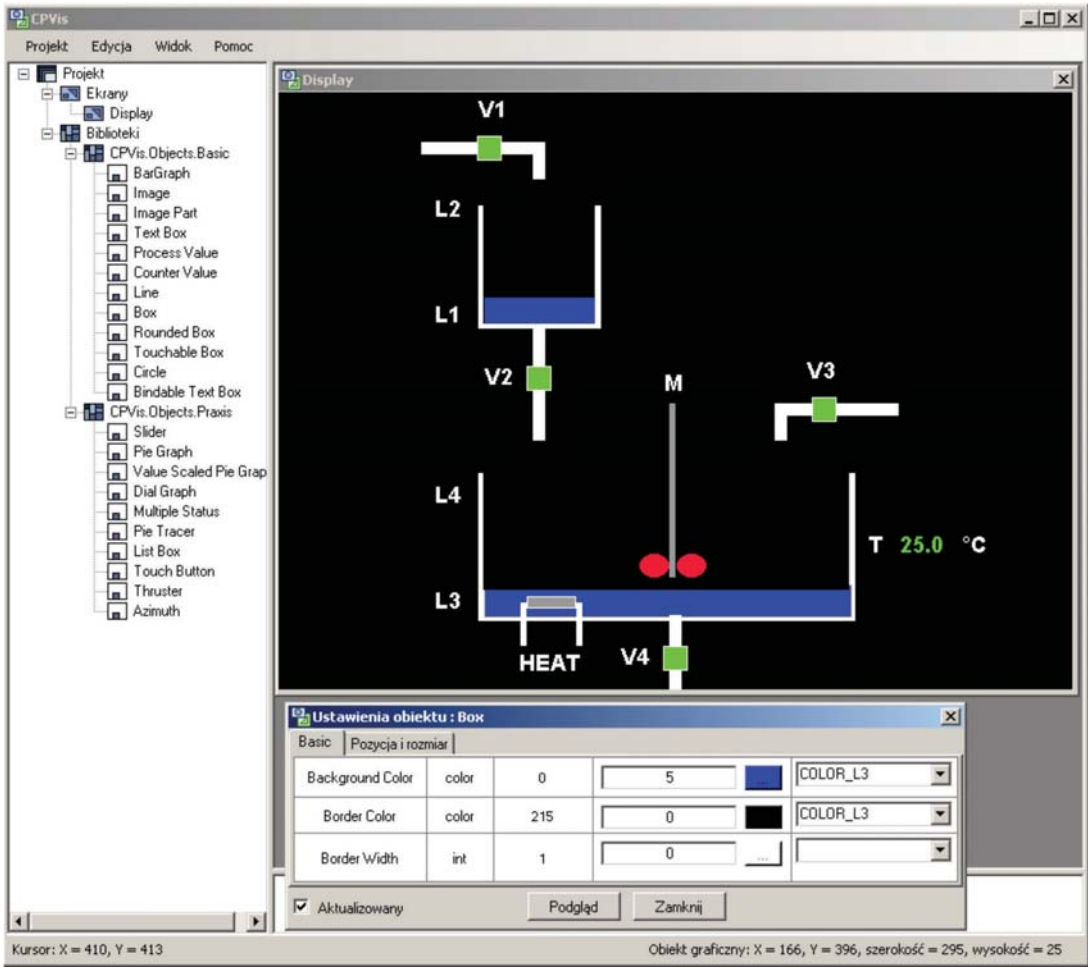
3. Tworzenie wizualizacji HMI w środowisku inżynierskim CPDev

Jak wspomniano w poprzednich rozdziałach środowisko inżynierskie CPDev pozwala na programowanie sterowników przemysłowych w językach normy IEC 61131-3. Jednym z modułów wbudowanych w pakiet CPDev jest CPVis, narzędzie dedykowane do przygotowywania wizualizacji w powiązaniu z implementacją programów sterowania.

CPVis składa się z dwóch części – edytora graficznego uruchamianego na komputerze PC i środowiska wykonawczego implementowanego na docelowym panelu operatorskim HMI [17]. Edytor graficzny pozwala na projektowanie wizualizacji korzystającej z biblioteki obiektów graficznych. Standardowa biblioteka zawiera zarówno proste obiekty (np. prostokąt), jak też złożone (np. bargraf, wykres). Parametry poszczególnych obiektów (rozmiar, kolor itp.) mogą być ustalone jako stałe, bądź powiązane ze zmiennymi z programu sterowania, wizualizując stan procesu. Takie powiązanie może dotyczyć zmiennej wejściowej, bezpośrednio odczytywanej z sensora (np. stan zaworu otwarty/zamknięty), bądź dodatkowej zmiennej, obliczanej wyłącznie na potrzeby wizualizacji (np. kolor obiektu zmienny przedziałami, uzależniony od temperatury). W drugim przypadku jest konieczne przygotowanie wraz z wizualizacją dodatkowych, prostych programów w jednym z języków IEC 61131-3 (typowo w ST), obliczających wartości wizualizowanych zmiennych. Proces ten szerzej opisano w [18]. Należy podkreślić, że takie programy wizualizacyjne mogą być przygotowane w innym języku niż główny program sterowania. Przykładowy wygląd edytora graficznego CPVis pokazano na rys. 1.

Druga część – środowisko wykonawcze – jest uruchamiana na panelu HMI. Wizualizacja zaprojektowana na komputerze PC w programie CPVis jest kompilowana do kodu pośredniego, który jest interpretowany przez środowisko wykonawcze uruchomione na panelu HMI. Kod pośredni zawiera informacje o obiektach z biblioteki użytych na poszczególnych ekranach, ich lokalizacji, wielkości, powiązaniu ze zmiennymi itd. Interpretacja tego kodu pozwala na obliczenie bieżącego wyglądu ekranu procesowego i wyświetlenie go operatorowi. Proces ten ponawiany jest cyklicznie, z zadanym interwałem.

Biblioteka obiektów graficznych bazuje na podstawowych elementach, jak na przykład linia, koło czy tekst. Podczas implementacji środowiska wykonawczego CPVis na docelowym panelu HMI jest konieczne przygotowanie procedur graficznych rysujących te podstawowe elementy, w sposób zależny od sprzętu. Pozostała część środowiska wykonawczego jest niezależna od sprzętu, co ułatwia wdrożenie. Jedno z takich wdrożeń, dla autopilota okrętowego, przedstawiono w [19].



Rys. 1. Edytor graficzny CPVis
Fig. 1. CPVis graphical editor

4. Wymiana danych między sterownikiem a HMI

Panele HMI wyświetlają operatorowi ekrany procesowe, przedstawiające stan obiektu, bieżące wartości sygnałów, sterowania, wartość zadaną itp. Wiele z elementów graficznych, z których zbudowane są ekrany procesowe, to elementy dynamiczne, a nie statyczne, co oznacza, że ich aktualny wygląd zależy od wartości poszczególnych zmiennych. Dynamika może wprowadzać zmianę elementu graficznego w zależności od stanu sygnału (np. odmienne grafiki przedstawiające zawór otwarty lub zamknięty), jak też zmianę jego parametrów (np. wysokość prostokąta ilustrującego poziom cieczy w zbiorniku, czy jego kolor odzwierciedlający temperaturę płynu). By to osiągnąć niezbędne jest cykliczne przekazywanie do panelu HMI bieżących wartości zmiennych, powiązanych z elementami graficznymi. Taka wymiana danych prowadzona jest zazwyczaj w jednym z przemysłowych protokołów komunikacyjnych. Na potrzeby dalszej analizy przyjmijmy, że zastosowano protokół Modbus RTU. Nie zawęża to jednak ogólności rozważań, wiele dalszych kwestii można analogicznie rozpatrywać także dla innych protokołów.

Protokół Modbus RTU bazuje na paradygmacie dostępu do łącza typu *master-slave*. Oznacza to, że jedno z urządzeń jest urządzeniem nadrzędnym, inicjującym transmisję, a pozostałe są urządzeniami podrzędnymi. W przypadku komunikacji panelu HMI ze sterownikiem możliwa jest zarówno konfiguracja, gdzie panel HMI jest urządzeniem typu *master* (nadrzędnym), a sterownik – *slave* (urządzeniem podrzędnym) jak też sytuacja odwrotna (sterownik – *master*, HMI – *slave*). Wybór odpowiedniej konfiguracji dla danego systemu automatyki zależy od możliwości komunikacyjnych sterownika, panelu HMI, jak też od liczby połączonych sterowników i paneli. Jeżeli panel HMI ma wyświetlać obrazy procesowe wymagające odczytu zmiennych z kilku sterowników, to zazwyczaj konfiguruje się je jako *slave*, a panel jako *master*. Jeżeli natomiast do jednego sterownika ma być podłączonych kilka paneli HMI, to często konfiguracja jest odwrotna i to sterownik zapisuje wartości do panelu HMI. Zakładając ustaloną liczbę zmiennych, których wartości należy przesłać, konfiguracja ta nie ma większego wpływu na liczbę transmitowanych bajtów, może natomiast pośrednio wpływać na czas cyklu komunikacyjnego, w zależności od czasu odpowiedzi poszczególnych urządzeń. Dla uproszczenia przyjmijmy w dalszych rozważaniach, że urządzeniem nadrzędnym jest panel HMI.

W protokole Modbus zdefiniowano polecenia (funkcje) pozwalające na wymianę danych. Najczęściej używane wymieniono poniżej:
- FC1 – odczyt zmiennych binarnych,
- FC2 – odczyt wejść binarnych,
- FC3 – odczyt rejestrów,
- FC4 – odczyt rejestrów wejściowych,
- FC5 – zapis pojedynczej zmiennej binarnej,
- FC6 – zapis pojedynczego rejestru,
- FC15 – zapis wielu zmiennych binarnych,
- FC16 – zapis wielu rejestrów.

Zmienne binarne i rejestry mogą być zarówno odczytywane i zapisywane, wejścia binarne i rejestry wejściowe są tylko do odczytu. Podczas korzystania z funkcji bitowych (FC1, FC2, FC5, FC15) stan zmiennej binarnej kodowany jest na pojedynczym bicie, co pozwala na przesłanie informacji o ośmiu zmiennych w jednym bajcie. Rejestry są szesnastobitowe. W praktyce zestaw zaimplementowanych funkcji spośród wymienionych i adresacja wykorzystana w sterowniku zależy od producenta. Spotykane są na przykład rozwiązania implementujące jedynie funkcje dotyczące zmiennych binarnych i rejestrów (bez wejść binarnych i rejestrów wejściowych) i mapujące wejścia pod odpowiednimi adresami pozostałych zmiennych. Niekiedy zmienne binarne mapowane są w przestrzeni adresowej rejestrów, co pozwala na odczyt zmiennych różnego typu (także binarnych) jednym poleceniem, wraz z innymi rejestrami.

Rozważmy polecenie odczytujące rejestry (FC3). Jego ramkę przedstawiono na rysunku 2. W jednym poleceniu urządzenie nadrzędne może zażądać od urządzenia podrzędnego przesłania wartości wielu zmiennych (rejestrów). Liczba rejestrów odczytywanych pojedynczym poleceniem nie może przekroczyć 125, z uwagi na ograniczenie wielkości pojedynczej wiadomości (odpowiedzi) do 256 bajtów. Oznacza to, że starszy bajt w polu liczby rejestrów będzie zawsze równy zero.

Adres *slave*	Kod funkcji	Adres początkowy	Liczba rejestrów	Suma kontrolna
1B	1B (0x03)	2B	2B	2B

Rys. 2. Ramka polecenia FC3 Modbus RTU
Fig. 2. Frame of Modbus RTU FC3 function

Adres *slave*	Kod funkcji	Liczba bajtów	Dane	Suma kontrolna
1B	1B (0x03)	1B	n × 2B	2B

Rys. 3. Ramka odpowiedzi FC3 Modbus RTU
Fig. 3. Frame of reply to Modbus RTU FC3 function

W odpowiedzi urządzenie podrzędne przesyła ramkę (rys. 3). Jej długość jest zmienna i zależy od liczby odczytywanych rejestrów.

Łatwo zauważyć, że przesłanie w jednej transakcji komunikacyjnej (zapytanie-odpowiedź) wartości wielu rejestrów wymaga, by były to kolejne rejestry. Oznacza to, że najkorzystniejsza jest sytuacja, gdy wszystkie zmienne wykorzystywane na danym ekranie sąsiadują ze sobą w pamięci sterownika.

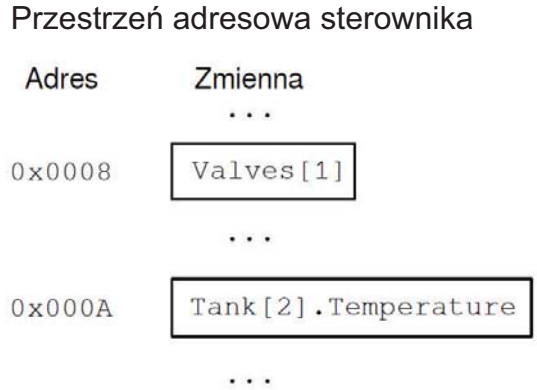
Aby zapewnić korzystną, sąsiednią lokalizację zmiennych, najprostsze jest ich właściwe rozmieszczenie bezpośrednio na etapie implementacji programu sterowania. W przypadku zmiennych globalnych i programu w języku ST inżynier może wpływać na lokalizację zmiennych za pomocą modyfikatora AT. Jest to jednak żmudne rozwiązanie, może także prowadzić do komplikacji podczas dalszych modyfikacji programu. Zadanie to znacznie upraszcza się, jeżeli pakiet inżynierski wspiera inżynierię wahadłową [20] lub przynajmniej udostępnia szablony kodów źródłowych na podstawie tworzonych modeli, na przykład w języku SysML [21]. Niestety, optymalne, z uwagi na wymagania komunikacyjne, rozmieszczenie wszystkich zmiennych niekiedy jest niemożliwe, zwłaszcza dla zmiennych złożonych (tablic, struktur) lub wykorzystywanych na wielu ekranach procesowych.

Innym rozwiązaniem jest utworzenie kopii wszystkich transmitowanych zmiennych, sąsiadująco w kolejnych komórkach pamięci. Pozwala to na odczytanie jednym poleceniem wielu zmiennych będących elementami różnych struktur czy tablic, jak też wykorzystywanych na wielu ekranach (wówczas tworzonych jest wiele kopii, osobna dla każdego ekranu). Niestety podejście to wymaga każdorazowej synchronizacji wartości oryginalnej zmiennej i każdej jej kopii, po dowolnej modyfikacji.

Alternatywne podejście, proponowane w tym artykule, nie wymaga modyfikacji programu sterowania. Zauważmy, że niekiedy korzystniejsze czasowo może być odczytanie większego fragmentu pamięci, zawierającego oprócz wymaganych zmiennych także inne rejestry i zignorowanie nadmiarowych informacji, tylko po to by wymiana danych bazowała na pojedynczej transakcji komunikacyjnej, a nie dwóch, bądź nawet kilku, sukcesywnie odczytujących jedynie wymagane obszary. Nie zawsze jednak takie rozwiązanie będzie prowadzić do poprawy para-

metrów czasowych. Przedstawiona dalej analiza teoretyczna pozwala na obliczenie czasu cyklu komunikacyjnego w różnych przypadkach i wybranie korzystniejszego.

Rozważmy sytuację, gdy wymagany jest odczyt dwóch szesnastobitowych zmiennych (dwóch rejestrów Modbus) tak rozmieszczonych w pamięci sterownika, że między nimi leżą także inne rejestry. Przykład takiej sytuacji pokazano na rysunku 4.



Rys. 4. Przykładowe rozmieszczenie zmiennych w pamięci sterownika
Fig. 4. Example location of variables in the controller's memory

Pokazane na rysunku 4 zmienne `Valves[1]` i `Tank[2].Temperature` są wizualizowane na jednym ekranie procesowym i ich wartości muszą być cyklicznie odczytywane przez panel HMI. Nie ma możliwości zmiany adresów tych zmiennych, tak by leżały obok siebie, gdyż są to elementy złożonych zmiennych (tablic i struktury). Zmienne te mogą być odczytywane pojedynczo, w dwóch osobnych transakcjach komunikacyjnych, bądź grupowo, w jednej transakcji, ale wraz z nadmiarowymi rejestrami leżącymi między nimi. Przeanalizujmy te dwie możliwości pod kątem wpływu na czas cyklu komunikacyjnego. Dla zachowania ogólności załóżmy, że interesujące nas zmienne są odseparowane od siebie przez k innych rejestrów.

Przy pojedynczym odczycie zmiennych w ramach jednego cyklu komunikacyjnego przesyłane są dwie pary zapytanie/odpowiedź. Każdorazowo zapytanie wymaga przesłania 8 bajtów, a odpowiedź 7 bajtów (zgodnie z ramkami przedstawionymi na rysunkach 2 i 3). W cyklu komunikacyjnym przesyłane jest więc 30 bajtów. Dodatkowo konieczne jest spełnienie wymagań czasowych protokołu Modbus RTU. Protokół wymaga, by między ramkami przesyłanymi na magistralę występowały okresy ciszy na linii, o minimalnej długości równej czasowi transmisji 3,5 znaku. Dalsze opóźnienia wprowadzane są przez przetwarzanie danych po stronie obu urządzeń.

Oznaczmy jako t_b czas transmisji pojedynczego bajtu (zależny od ustalonej prędkości łącza komunikacyjnego), t_m – czas potrzebny na przygotowanie zapytania przez urządzenie nadrzędne (*master*), t_s – czas potrzebny na przygotowanie odpowiedzi przez urządzenie podrzędne (*slave*), a t_{cycle_1} – łączny czas cyklu komunikacyjnego w i-tym przypadku. Bazując na poprzednio przedstawionych wyliczeniach uzyskujemy:

$$t_{cycle_1} = 2\left(t_m + 8t_b + 3{,}5t_b + t_s + 7t_b + 3{,}5t_b\right) = 44t_b + 2t_m + 2t_s$$

W drugim przypadku, jeżeli zmienne będą transmitowane w ramach jednej transakcji zapytanie/odpowiedź, zapytanie również wymaga przesłania 8 bajtów, ale długość odpowiedzi zależy od liczby k dodatkowych transmitowanych rejestrów i wyraża się wzorem $9 + 2k$. Oczywiście k nie może przekroczyć wartości 123, by spełnić wspomniane wcześniej ograniczenie na łączną długość pojedynczej ramki Modbus. W takim przypadku czas cyklu wynosi:

$$t_{cycle_2} = t_m + 8t_b + 3{,}5t_b + t_s + (9 + 2k)t_b + 3{,}5t_b = (24 + 2k)t_b + t_m + t_s$$

Różnica $t_{cycle_1} - t_{cycle_2}$ określa, o ile krótszy jest czas cyklu komunikacyjnego w drugim przypadku. Wartości dodatnie oznaczają, że drugi przypadek jest korzystniejszy; ujemne, że korzystniejszy jest przypadek pierwszy.

$$t_{cycle_1} - t_{cycle_2} = \left(20 - 2k\right)t_b + t_m + t_s$$

Łatwo zauważyć, że dla $k \leq 10$ istnieje korzyść z zastosowania drugiego sposobu odczytu, niezależnie od czasów t_m, $t_s > 0$. Dla $k > 10$, wybierając optymalny sposób należy wziąć pod uwagę czasy t_m i t_s zależne od zastosowanych urządzeń. Oczywiście analogiczne obliczenia można przeprowadzić dla wielu grup wizualizowanych zmiennych, rozdzielonych innymi zmiennymi w pamięci sterownika. Takie rozwiązanie, dynamicznie wybierające sposób transmisji w zależności od położenia zmiennych, zaimplementowano w pakiecie inżynierskim CPDev.

5. Podsumowanie

W artykule przedstawiono wpływ rozmieszczenia zmiennych globalnych, wizualizowanych na panelu HMI, na opóźnienia komunikacyjne. Zaproponowano metody poprawy parametrów czasowych, w szczególności bazujące na koncepcji grupowania zmiennych i zbiorczego odczytu. Przedstawiono sposób obliczania czasu cyklu komunikacyjnego w zależności od przyjętej metody odczytu, pozwalający na stwierdzenie czy automatyczne grupowanie poprawi parametry czasowe w konkretnym przypadku. Wspomniany mechanizm został zaimplementowany w pakiecie inżynierskim CPDev.

Podziękowania

Projekt finansowany w ramach programu Ministra Nauki i Szkolnictwa Wyższego pod nazwą „Regionalna Inicjatywa Doskonałości" w latach 2019–2022 nr projektu 027/RID/2018/19 całkowita kwota finansowania 11 999 900 zł.

Bibliografia

1. Rzońca D., Sadolewski J., Stec A., Świder Z., Trybus B., Trybus L., *CPDev engineering environment for control programming*, [in:] *Trends in Advanced Intelligent Control, Optimization and Automation* (W. Mitkowski, J. Kacprzyk, K. Oprzędkiewicz, P. Skruch, eds.), (Cham), Vol. 577, 303–314, Springer International Publishing, 2017, DOI: 10.1007/978-3-319-60699-6_29.
2. Jamro M., Rzońca D., Sadolewski J., Stec A., Świder Z., Trybus B., Trybus L., *CPDev Engineering Environment for Modeling, Implementation, Testing, and Visualization of Control Software*, [in:] *Recent Advances in Automation, Robotics and Measuring Techniques* (R. Szewczyk, C. Zieliński, and M. Kaliczyńska, eds.), (Cham), Vol. 267, 81–90, Springer International Publishing, 2014, DOI: 10.1007/978-3-319-05353-0_9.
3. IEC 61131-3 - Programmable controllers – Part 3: Programming languages, 2003, 2013.
4. Silva M., Pereira F., Soares F., Leão C.P., Machado J., Carvalho V., *An Overview of Industrial Communication Networks*, [in:] *New Trends in Mechanism and Machine Science* (P. Flores and F. Viadero, eds.), (Cham),

Vol. 24, 933–940, Springer International Publishing, 2015, DOI: 10.1007/978-3-319-09411-3_97.
5. Thomesse J., *Fieldbus Technology in Industrial Automation*, "Proceedings of the IEEE", Vol. 93, No. 6, June 2005, 1073–1101, DOI: 10.1109/JPROC.2005.849724.
6. IEC 61158 - Industrial Communication Networks – Fieldbus Specifications, 2007.
7. Jestratjew A., Kwiecień A., *Using HTTP as Field Network Transfer Protocol*, [in:] *Computer Networks* (A. Kwiecień, P. Gaj, P. Stera, eds.), (Berlin, Heidelberg), Vol. 160, 2011, 306–313, Springer Berlin Heidelberg, DOI: 10.1007/978-3-642-21771-5_33.
8. Jestratjew A., Kwiecień A., *Performance of HTTP Protocol in Networked Control Systems*, "IEEE Transactions on Industrial Informatics", Vol. 9, No. 1, 2013, 271–276, DOI: 10.1109/TII.2012.2183138.
9. Gaj P., Jasperneite J., Felser M., *Computer Communication within Industrial Distributed Environment – a Survey*, "IEEE Transactions on Industrial Informatics", Vol. 9, No. 1, 2013, 182–189, DOI: 10.1109/TII.2012.2209668.
10. Stój J., *Real-Time Communication Network Concept Based on Frequency Division Multiplexing*, [in:] *Computer Networks* (A. Kwiecień, P. Gaj, P. Stera, eds.), (Berlin, Heidelberg), Vol. 291, 2012, 247–260, Springer Berlin Heidelberg, DOI: 10.1007/978-3-642-31217-5_27.
11. Fiset J.-Y., *Human-Machine Interface Design for Process Control Applications*. Instrumentation, Systems and Automation Society, 2009.
12. Zhang P., *Human-machine interfaces*, [in:] *Advanced Industrial Control Technology* (Zhang P., ed.), 527–555, Oxford: William Andrew Publishing, 2010.
13. VDI/VDE 3699 Process control using display screens, 2015.
14. Oshana R., *Human Factors and User Interface Design for Embedded Systems*, in *Software Engineering for Embedded Systems* (R. Oshana, M. Kraeling, eds.), 417–440, Oxford: Newnes, 2013.
15. Nachreiner F., Nickel P., Meyer I., *Human factors in process control systems: The design of human–machine interfaces*, "Safety Science", Vol. 44, No. 1, 2006, 5–26. Safety and Design, DOI: 10.1016/j.ssci.2005.09.003.
16. Jamro M., Trybus B., *Configurable Operator Interface for CPDev Environment*, "Pomiary Automatyka Robotyka", R. 17, Nr 2, 2013, 426–431.
17. Jamro M., Trybus B., *IEC 61131-3 programmable human machine interfaces for control devices*, [in:] 6th International Conference on Human System Interactions (HSI), 2013, 48–55, DOI: 10.1109/HSI.2013.6577801.
18. Rzońca D., Stec A., Trybus B., *Control Program Development in CPDev Using SFC Language, HMI and Runtime Environment*, [in:] *Automation 2018* (R. Szewczyk, C. Zieliński, and M. Kaliczyńska, eds.), (Cham), 223–232, Springer International Publishing, DOI: 10.1007/978-3-319-77179-3_21.
19. Jamro M., Rzońca D., Sadolewski J., Stec A., Świder Z., Trybus B., Trybus L., *Structure and Functionalities of Ship Autopilot Simulator*, [in:] *Challenges in Automation, Robotics and Measurement Techniques* (R. Szewczyk, C. Zieliński, and M. Kaliczyńska, eds.), (Cham), 2016, 223–231, Springer International Publishing, DOI: 10.1007/978-3-319-29357-8_20.
20. Jamro M., Rzońca D., *Agile and hierarchical round-trip engineering of IEC 61131-3 control software*, "Computers in Industry", Vol. 96, 2018, 1–9, DOI: 10.1016/j.compind.2018.01.004.
21. Jamro M., Rzońca D., *SysML-based Optimisation of Global Variables Arrangement for Visualisation in Distributed Control Systems Oriented Towards Communication Performance*, [in:] MATEC Web Conf., Vol. 252, 2019, DOI: 10.1051/matecconf/201925201005.

Performance Improvement of PLC – HMI Communication in CPDev Engineering Environment

Abstract: HMI (Human-Machine Interface) panels are commonly used in industrial automation systems. They facilitate the visualization of industrial process control as well as enable the change of parameters. The values displayed on a HMI are read from a controller or controllers connected by a communication link. Thus efficient PLC – HMI communication is essential for proper functioning of the whole automation system. Several aspects of such communication are analyzed in the paper and some solutions proposed to reduce the communication cycle. The described methods have been implemented in the CPDev engineering environment.

Keywords: industrial controller, PLC, operator panel, HMI, communication

dr inż. Dariusz Rzońca
drzonca@kia.prz.edu.pl
ORCID: 0000-0001-5724-0978

Licencjat matematyki (Uniwersytet Rzeszowski 2002), magister inżynier informatyki (Politechnika Rzeszowska 2004), doktor nauk technicznych w dyscyplinie informatyka, specjalność przemysłowe systemy informatyki (Politechnika Śląska 2012). Asystent w Katedrze Automatyki i Informatyki Politechniki Rzeszowskiej od 2004 r. Jego zainteresowania naukowe koncentrują się na kolorowanych sieciach Petriego oraz zagadnieniach związanych z komunikacją w systemach automatyki.