Optymalizacja mieszanego odczytu zmiennych binarnych i rejestrowych w protokole Modbus ze sterownika PLC implementującego CPDev

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

Streszczenie: Wydajna komunikacja jest kluczowa dla poprawnej pracy rozproszonych systemów automatyki. W artykule skupiono się na jednym z aspektów takiej komunikacji, związanym z mieszanym odczytem zmiennych binarnych i rejestrowych ze sterownika PLC w protokole Modbus RTU. W badaniach przyjęto architekturę pamięci ze wspólną adresacją zmiennych różnych typów, pozwalającą na użycie do odczytu zmiennych binarnych oprócz dedykowanych funkcji bitowych (np. FC1), także funkcji rejestrowych Modbus (np. FC3). Architektura taka występuje m.in. w sterownikach implementujących środowisko inżynierskie CPDev. W artykule zaproponowano metodę grupowania zmiennych, prowadzącą do zmniejszenia łącznego czasu cyklu komunikacyjnego. Zaimplementowano model optymalizacyjny pozwalający na automatyczne znalezienie optymalnego grupowania. Przeprowadzono eksperymenty, a następnie omówiono uzyskane wyniki. Rezultaty przeprowadzonych badań zostaną wykorzystane przy rozwoju środowiska inżynierskiego CPDev.

Słowa kluczowe: sterownik przemysłowy, PLC, komunikacja, Modbus, CPDev

1. Wprowadzenie

Wymiana danych w rozproszonych systemach sterowania nierzadko stanowi wąskie gardło ograniczające wydajność całego systemu. Niniejszy artykuł poświęcony jest problematyce optymalizacji odczytu zmiennych różnych typów (binarnych i rejestrowych) ze sterownika PLC (ang. *Programmable Logic Controller*) lub PAC (ang. *Programmable Automation Controller*) pracującego pod kontrolą pakietu inżynierskiego CPDev. Specyficzna architektura pamięci maszyny wirtualnej CPDev i typowe funkcje zaimplementowane w protokole Modbus niejednokrotnie pozwalają na odczyt żądanych zmiennych na różne sposoby. Wybór optymalnego rozwiązania należy do projektanta, w podjęciu decyzji może go wspierać oprogramowanie obliczające łączny czas cyklu komunikacyjnego w konkretnym przypadku. Oprogramowanie takie może bazować na rezultatach badań przedstawionych w artykule. Bezpośrednią motywacją do przeanalizowania tych zagadnień była praca nad rozwojem środowiska inżynierskiego CPDev, jednakże przedstawione wyniki mogą znaleźć zastosowanie także w innych systemach.

Kolejny rozdział zawiera zwięzły przegląd literatury tematycznie związanej z artykułem. W trzecim rozdziale przedstawiono komunikację w protokole Modbus i zaproponowano metodę pozwalającą na poprawę parametrów czasowych przy mieszanym odczycie zmiennych binarnych i rejestrowych. Kolejny rozdział poświęcony jest dedykowanemu modelowi optymalizacyjnemu programowania z ograniczeniami. W piątym rozdziale opisano przeprowadzone eksperymenty obliczeniowe i przeanalizowano uzyskane wyniki. Ostatni z rozdziałów podsumowuje artykuł.

Autor korespondujący:
Dariusz Rzońca, drzonca@kia.prz.edu.pl

Artykuł recenzowany
nadesłany 02.12.2023 r., przyjęty do druku 04.04.2024 r.

2. Przegląd literatury

Współczesne sterowniki przemysłowe PLC i PAC zazwyczaj programowane są w językach normy IEC 61131-3 [1]. Standard ten, przyjęty w Polsce jako PN-EN 61131-3, obejmuje pięć języków programowania, zarówno tekstowych (ST (ang. *Structured Text*), IL (ang. *Instruction List*)), graficznych (FBD (ang. *Function Block Diagram*), LD (ang. *Ladder Diagram*)), oraz mieszanych (SFC (ang. *Sequential Function Chart*)). Szczegółową charakterystykę poszczególnych języków, poza normą, można znaleźć w licznych publikacjach, spośród których godna polecenia jest książka [2].

Języki normy IEC 61131-3 implementowane są w wielu pakietach inżynierskich. Niniejszy artykuł koncentruje się na opracowanym w Katedrze Informatyki i Automatyki Politech-

niki Rzeszowskiej środowisku CPDev (ang. *Control Program Developer*) [3, 4]. Pakiet ten obecnie jest wdrożony przez kilku producentów systemów automatyki, krajowych i zagranicznych (Hiszpania, Holandia). Wśród ciekawszych zastosowań można wyróżnić systemy automatyki okrętowej, jak np. autopilot [5, 6].

Budowa pakietu inżynierskiego CPDev pozwala na jego łatwe wdrożenie na różnych platformach sprzętowych. Edytory graficzne [7] i kompilator działające na PC tworzą kod pośredni VMASM (ang. *Virtual Machine Assembler*), wykonywany na docelowej platformie sprzętowej przez dedykowany interpreter, wchodzący w skład tzw. maszyny wirtualnej CPDev [8]. Przygotowano oprogramowanie wbudowane (firmware) maszyny wirtualnej dla różnych architektur procesorów, od niewielkich ośmiobitowych AVR (jak w sterowniku SMC pokazanym na Rys. 1) do wielordzeniowych ARM [9]. We wszystkich implementacjach architektura pamięci maszyny wirtualnej CPDev jest podobna, zawiera sąsiadujące ze sobą zmienne różnych typów (np. binarne i szesnastobitowe), co pozwala na wdrożenie mechanizmów poprawy efektywności transmisji przedstawionych w niniejszym artykule.

Efektywna wymiana danych w rozproszonych systemach automatyki stanowi jeden z kluczowych elementów wpływających na wydajność systemu. Ostatnio wydana monografia [10] zawiera bardzo dobry opis wielu zagadnień związanych z komunikacją w przemysłowych systemach komputerowych. Sieci komunikacyjne stosowane w przemyśle [11] zazwyczaj bazują na sieciach polowych [12], a w nowszych rozwiązaniach często wykorzystywany jest przemysłowy Ethernet [13]. W przypadku sieci polowych zazwyczaj stosowane są rozwiązania opisane w normie IEC 61158 [14]. Współcześnie można zaobserwować tendencję do stosowania architektur heterogenicznych, łączących różne typy sieci, zwłaszcza w dużych systemach [15]. Niewielkie rozproszone systemy automatyki wciąż jednak bazują na komunikacji szeregowej (np. RS-485) wykorzystującej przemysłowe protokoły komunikacyjne, jak Modbus czy Profibus. Zwłaszcza protokół Modbus jest bardzo popularny dzięki swojej prostocie.

Problematyka grupowania rejestrów podczas transmisji za pomocą protokołu Modbus była kilkukrotnie poruszana w literaturze, choć z pewnymi ograniczeniami. Zazwyczaj artykuły koncentrowały się na wybranej funkcji Modbus lub konkretnym zastosowaniu (np. wymiana danych między sterownikiem PLC a panelem HMI). Niniejszy artykuł można traktować jako kontynuację poprzednich prac [16, 17]. W artykułach tych rozważano problem grupowania rejestrów w poleceniach FC3, FC6 i FC16 protokołu Modbus RTU, tj. skupiono się na transmisji szesnastobitowych rejestrów. Pominięto możliwość skorzystania z funkcji dedykowanych do transmisji zmiennych binarnych, jak np. FC1, co uzupełniono w niniejszym artykule.

Rys. 1. Sterownik SMC z modułami I/O
Fig. 1. SMC controller with I/O modules

W pracy [18] przedstawiono metodę przyspieszenia wymiany danych w protokole Modbus TCP, podczas transmisji między sterownikiem PLC a panelem HMI. Zaproponowano grupowanie transmitowanych zmiennych w bloki o optymalnej długości. Przedstawiono także wyniki badań eksperymentalnych.

Artykuły [19, 20] omawiają wydajność komunikacji w protokole Modbus przy różnych scenariuszach wymiany danych. Przeanalizowano tam wpływ sąsiadującego lub osobnego rozmieszczenia rejestrów na osiągane parametry czasowe. Zaproponowano także rozszerzenie dla protokołu Modbus. Zostało ono przebadane eksperymentalnie w pracy [21].

W artykule [22] przedstawiono algorytm komparatywnej przewagi dla protokołu Modbus. Grupowanie jest tam jedną z metod terapeutycznych przywracających zdatność systemu komunikacyjnego podczas przeciążenia magistrali komunikacyjnej.

W pracy [23] przeanalizowano wpływ grupowania na osiągane parametry czasowe podczas transmisji rejestrowej w dowolnym protokole komunikacyjnym, nie zawężając problematyki do protokołu Modbus. Nie rozważano tam jednak funkcji bitowych, a wyłącznie transmisję rejestrową.

3. Wymiana danych w protokole Modbus

Protokół Modbus definiuje szereg funkcji służących do wymiany danych. Najczęściej implementowane w sterownikach PLC są następujące funkcje:

- 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.

Można zauważyć, że wspierane są tu jedynie dwa typy danych: jednobitowe wartości binarne i szesnastobitowe rejestry. Obsługa innych typów zmiennych niż szesnastobitowe (np. zmiennych trzydziestodwubitowych) także prowadzona jest za pomocą funkcji rejestrowych (zmienna trzydziestodwubitowa będzie zajmowała podczas transmisji dwa rejestry).

W niektórych sterownikach wyodrębniono osobne przestrzenie adresowe dla danych binarnych i dla zmiennych wielobitowych. W takim przypadku odczyt czy zapis zmiennych binarnych może być prowadzony wyłącznie za pomocą dedykowanych funkcji (FC1, FC5, FC15). Sterowniki implementujące środowisko inżynierskie CPDev charakteryzują się innym rozwiązaniem. Zmienne różnych typów są przechowywane w tej samej pamięci i są dostępne w ramach wspólnej adresacji. Co więcej, często zmienne różnych typów sąsiadują ze sobą, zwłaszcza przy użyciu typów złożonych jak struktury, które mogą w kolejnych polach zawierać np. zmienne binarne i szesnastobitowe. W pamięci CPDev zmienna binarna zajmuje jeden bajt, w którym siedem najstarszych bitów jest wyzerowanych, a najmłodszy zawiera właściwą informację. Oznacza to, że wyborem projektanta jest czy do odczytu pewnych zmiennych binarnych zastosuje funkcję bitową (FC1) czy rejestrową

(FC3). Oczywiście transmisja zmiennych binarnych funkcją rejestrową powoduje, że w polu danych przesyłane są w większości zera, a jedynie nieliczne bity stanowią dane użyteczne. Mimo wszystko, w pewnych wypadkach wspólna transmisja rejestrów i zmiennych binarnych w jednej ramce pozwala na efektywniejsze wykorzystanie łącza w porównaniu z transmisją osobnych ramek binarnych i rejestrowych, co zostanie pokazane w dalszej części artykułu.

Przeanalizujmy ramki poleceń i odpowiedzi dla funkcji FC1 i FC3 Modbus RTU (Rys. 2–5).

W ramce odpowiedzi FC1 zmienne binarne przesyłane są jako pojedyncze bity upakowane w kolejnych bajtach. Oznacza to, że wielkość (liczba bajtów) pola danych zależy od liczby n przesyłanych zmiennych, podzielonej przez osiem (na jednym bajcie przesyłane są wartości ośmiu zmiennych w kolejnych bitach) i zaokrąglonej do góry $\left\lceil \frac{n}{8} \right\rceil$ (pozostałe bity uzupełniane są do pełnego bajtu zerami).

Podczas transmisji rejestrów ramką FC3 możliwe jest przesłanie także zmiennych binarnych, sąsiadujących z transmitowanymi rejestrami. W takim przypadku w jednym szesnastobitowym rejestrze mogą być przesłane maksymalnie dwie zmienne binarne, na najmłodszych bitach każdego składowego bajtu, co wynika z organizacji pamięci maszyny wirtualnej CPDev.

Celowe jest rozważenie, który ze wspomnianych sposobów komunikacji jest efektywniejszy dla transmisji zmiennych binarnych. Intuicyjnie można zauważyć, że przypuszczalnie podczas przesyłania znacznej liczby zmiennych binarnych (Modbus pozwala na transmisję do 2000 zmiennych binarnych w pojedynczej ramce) dedykowana funkcja FC1 pozwoli na skrócenie czasu transmisji dzięki większemu upakowaniu danych (osiem zmiennych w jednym bajcie). Jeżeli jednak wymagane jest przesłanie jedynie kilku zmiennych binarnych sąsiadujących z już transmitowanymi rejestrami, to można to osiągnąć niewielkim wydłużeniem obecnych komunikatów rejestrowych, bez potrzeby osobnych zapytań binarnych, co pozwoli zmniejszyć łączny czas transmisji.

Przyjmijmy następujące oznaczenia:
- t_b – czas transmisji jednego znaku (zależny od ustalonej prędkości łącza komunikacyjnego),
- t_m – czas potrzebny na przygotowanie polecenia przez urządzenie nadrzędne (master),
- t_s – czas potrzebny na przygotowanie odpowiedzi przez urządzenie podrzędne (slave),
- t_{cycke_i} – łączny czas cyklu komunikacyjnego w i-tym przypadku.

Załóżmy, że chcemy przesłać jeden rejestr szesnastobitowy i n zmiennych binarnych. Grupy te rozdzielone są w pamięci sterownika przez k bajtów. Rozważmy dwa przypadki:
1. transmisja dwoma osobnymi komunikatami, przy użyciu funkcji bitowej FC1 i rejestrowej FC3,
2. transmisja łączna przy użyciu pojedynczej funkcji FC3.

Obliczając łączny czas transmisji w obu przypadkach należy uwzględnić, że protokół Modbus RTU wymaga odstępu między kolejnymi komunikatami na magistrali trwającego co najmniej $3{,}5t_b$. Rozważmy czas potrzebny na osobny odczyt (pierwszy przypadek):

$$t_{cycle_1} = t_m + 8t_b + 3{,}5t_b + t_s + 7t_b + 3{,}5t_b + t_m + 8t_b + 3{,}5t_b + \\ + t_s + \left(5 + \left\lceil \frac{n}{8} \right\rceil\right) t_b + 3{,}5t_b = \left(42 + \left\lceil \frac{n}{8} \right\rceil\right) t_b + 2t_m + 2t_s$$

Przy odczycie łącznym (drugi przypadek) potrzebujemy:

$$t_{cycle_2} = t_m + 8t_b + 3{,}5t_b + t_s + \left(7 + 2\left\lceil \frac{n+k}{2} \right\rceil\right) t_b + 3{,}5t_b = \\ = \left(22 + 2\left\lceil \frac{n+k}{2} \right\rceil\right) t_b + t_m + t_s$$

Adres *slave*	Kod funkcji	Adres początkowy	Liczba zmiennych	Suma kontrolna
1B	1B (0x01)	2B	2B	2B

Rys. 2. Ramka polecenia FC1 Modbus RTU
Fig. 2. Frame of Modbus RTU FC1 function

Adres *slave*	Kod funkcji	Liczba bajtów	Dane	Suma kontrolna
1B	1B (0x01)	1B	$\left\lceil \frac{n}{8} \right\rceil$ 2B	2B

Rys. 3. Ramka odpowiedzi FC1 Modbus RTU
Fig. 3. Frame of reply to Modbus RTU FC1 function

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

Rys. 4. Ramka polecenia FC3 Modbus RTU
Fig. 4. Frame of Modbus RTU FC3 function

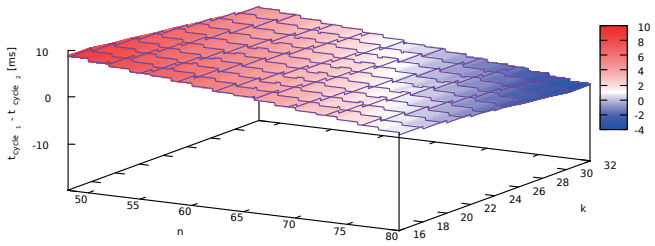
Adres *slave*	Kod funkcji	Liczba bajtów	Dane	Suma kontrolna
1B	1B (0x01)	1B	$2n$ B	2B

Rys. 5. Ramka odpowiedzi FC3 Modbus RTU
Fig. 5. Frame of reply to Modbus RTU FC3 function

Obliczmy różnicę tych czasów:

$$t_{cycle_1} - t_{cycle_2} = \left(20 + \left\lceil \frac{n}{8} \right\rceil - 2\left\lceil \frac{n+k}{2} \right\rceil\right) t_b + t_m + t_s$$

Jeżeli różnica $t_{cycle_1} - t_{cycle_2}$ jest dodatnia, to efektywniejszy jest przypadek drugi, jeżeli ujemna to pierwszy. Różnicę tę zilustrowano na wykresie (Rys. 6), przyjmując transmisję z prędkością 38 400 bps w formacie 8E1 i czasy $t_m = t_s = 10$ ms.



Rys. 6. Różnica $t_{cycle1} - t_{cycle2}$ dla prędkości 38 400 bps w zależności od n i k
Fig. 6. Difference $t_{cycle1} - t_{cycle2}$ for 38 400 bps baudrate, depending on n and k

Odcieniami niebieskiego oznaczono zakres w którym efektywniejszy jest pierwszy przypadek, a czerwonego – drugi, na biało gdy wynikowa różnica $t_{cycle_1} - t_{cycle_2}$ jest w pobliżu zera. Widoczny schodkowy charakter powierzchni wynika z obecnych we wzorze zaokrągleń.

4. Zadanie optymalizacji rejestrowo-binarnej

Charakterystyka mieszanego odczytu danych rejestrowych i binarnych omówiona w poprzedniej sekcji może zostać uogólniona do definicji zadania optymalizacji.

Definicja. Optymalizacja rejestrowo-binarna polega na wyznaczeniu na podstawie danych instancji

$$\mathcal{P} = \left(R, L, H, \alpha_{\mathrm{R}}, \beta_{\mathrm{R}}, \omega_{\mathrm{R}}, \alpha_{\mathrm{B}}, \beta_{\mathrm{B}}, \omega_{\mathrm{B}}\right) \tag{1}$$

gdzie $R, L, H \subset \mathbb{Z}_{\geq 0}$, $R \cap \left(L \cup H\right) = \varnothing$, $\omega_{\mathrm{R}}, \omega_{\mathrm{B}} \in \mathbb{Z}_{\geq 1}$, $\alpha_{\mathrm{R}}, \beta_{\mathrm{R}}, \alpha_{\mathrm{B}}, \beta_{\mathrm{B}} \in \mathbb{R}_{\geq 0}$, rozwiązania $\mathcal{S} = (F_{\mathrm{R}}, F_{\mathrm{B}})$ w formie dwóch zbiorów przedziałów

$$\begin{gathered} F_{\mathrm{R}} = \bigcup_{i=0}^{n_{\mathrm{R}}} \left\{\left[x_i, y_i\right]\right\}, \quad n_{\mathrm{R}} \in \left\{0, \ldots, \left|R\right|\right\}, \\ x_i, y_i \in R \cup L \cup H, \quad y_i \geq x_i, \end{gathered} \tag{2}$$

$$\begin{gathered} F_{\mathrm{B}} = \bigcup_{i=0}^{n_{\mathrm{B}}} \left\{\left[x_i, y_i\right]\right\}, \quad n_{\mathrm{B}} \in \left\{0, \ldots, \left|L\right| + \left|H\right|\right\}, \\ x_i, y_i \in \left\{2v \middle| v \in L\right\} \cup \left\{2v+1 \middle| v \in H\right\}, \quad y_i \geq x_i, \end{gathered} \tag{3}$$

minimalizującego funkcję celu

$$\begin{aligned} t_{cycle} = &\sum_{\left[x,y\right] \in F_{\mathrm{R}}} \left(\alpha_{\mathrm{R}}\left(y - x + 1\right) + \beta_{\mathrm{R}}\right) + \\ &+ \sum_{\left[x,y\right] \in F_{\mathrm{B}}} \left(\alpha_{\mathrm{B}} \left\lceil \frac{y - x + 1}{8} \right\rceil + \beta_{\mathrm{B}}\right), \end{aligned} \tag{4}$$

przy spełnieniu ograniczeń

$$R \subset \bigcup_{\left[x,y\right] \in F_{\mathrm{R}}} \left[x, y\right], \tag{5}$$

$$\left\{2v \middle| v \in L\right\} \cup \left\{2v+1 \middle| v \in H\right\} \subset \bigcup_{\left[x,y\right] \in F_{\mathrm{R}}} \left[2x, 2y+1\right] \cup \bigcup_{\left[x,y\right] \in F_{\mathrm{B}}} \left[x, y\right], \tag{6}$$

$$y - x + 1 \leq \omega_{\mathrm{R}} \quad \forall \left[x, y\right] \in F_{\mathrm{R}}, \tag{7}$$

$$y - x + 1 \leq \omega_{\mathrm{B}} \quad \forall \left[x, y\right] \in F_{\mathrm{B}}. \tag{8}$$

W interpretacji praktycznej zbiory R, L i H zawierają adresy odpowiednio zmiennych rejestrowych oraz binarnych L i H. W przypadku każdego z tych zbiorów, wartości są numerami słów 16-bitowych, przy czym dla R oznacza to zajętość całego słowa, a dla L i H zajętość odpowiednio dolnej lub górnej połówki słowa przez zmienną binarną. W rozwiązaniu $\mathcal{S}$ elementy zbioru F_{R} reprezentują wyznaczone w drodze optymalizacji ramki rejestrowe, a elementy zbioru F_{B} ramki binarne. W przypadku zbioru F_{B}, adresy rejestrowe zmiennych binarnych są mapowane funkcjami $f_{\mathrm{L}}(n) = 2n$ i $f_{\mathrm{H}}(n) = 2n+1$, co wprowadza rozróżnialność lokalizacji L i H w ramach jednego słowa, niezbędną do sformułowania funkcji celu (4) oraz w ograniczeniu (6). Liczba ramek rejestrowych (2) nie przekroczy $|R|$ w rozwiązaniu optymalnym, ponieważ niezbędna będzie co najwyżej jedna ramka dla każdej zmiennej rejestrowej z osobna. Zmienne binarne również mogą zostać przydzielone do tych ramek, ale rozwiązanie optymalne nie będzie zawierać ramek rejestrowych przekazujących wyłącznie zmienne binarne, bo taki ich zbiór zawsze jest szybciej transmitowany w ramce binarnej. Analogicznie, w rozwiązaniu optymalnym potrzeba co najwyżej $|L| + |H|$ ramek binarnych (3), ponieważ mogą one przekazywać tylko zmienne binarne i w granicznym przypadku każda będzie przesyłana w odrębnej ramce. Ograniczenie (5) zapewnia, że każda zmienna rejestrowa zostanie przesłana w pewnej ramce rejestrowej, natomiast ograniczenie (6) wymusza przydział każdej zmiennej binarnej do pewnej ramki binarnej lub rejestrowej. Ograniczenia (7) i (8) narzucają limity długości ramek zgodnie z wartościami parametrów ω_{R} i ω_{B}, odpowiednio dla ramek rejestrowych i binarnych. Sformułowanie funkcji celu (4) wskazuje, że α_{R} i β_{R} są współczynnikami zależności liniowej łączącej rozmiar ramki rejestrowej z czasem jej transmisji, analogicznie parametry α_{B} i β_{B} dotyczą ramek binarnych, z zastrzeżeniem nieciągłości charakterystycznej dla rozmiarów ramek binarnych, która jest uwzględniona w (4).

Utworzona definicja problemu może posłużyć do zaprojektowania algorytmu optymalizacji lub deklaratywnego modelu optymalizacyjnego w formie programowania liniowego mieszanego MILP (ang. *Mixed-Integer Linear Programming*) lub programowania z ograniczeniami CP (ang. *Constraint Programming*). W niniejszej pracy wykorzystano solwer Constraint Programming Optimizer (CPO) z pakietu IBM CPLEX [24]. Model CP przedstawiono w Dodatku.

5. Eksperymenty obliczeniowe

Zestawienie specyfikacji i konkretnej konfiguracji protokołu Modbus RTU z rozdziału 3 z modelem optymalizacyjnym omówionym w rozdziale 4 pozwala określić wartości parametrów

$$\alpha_{\mathrm{R}} = 2t_b = 22000 / v, \quad \alpha_{\mathrm{B}} = t_b = 11000 / v,$$

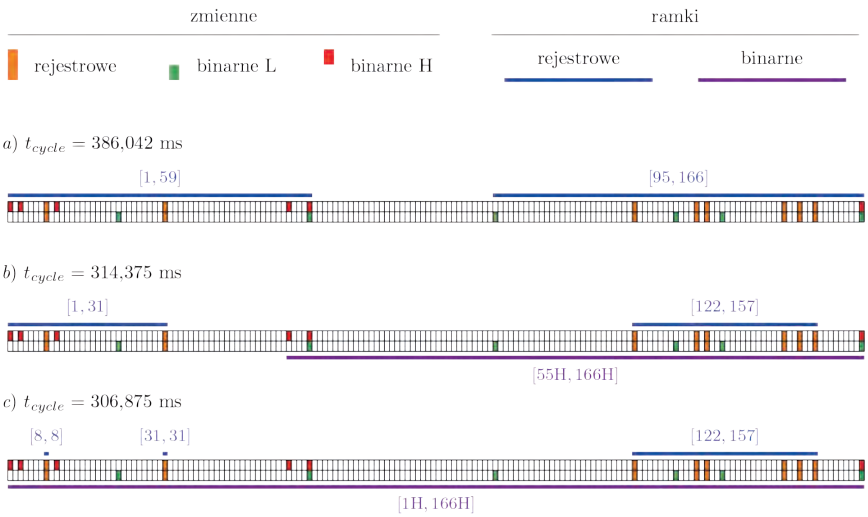
$$\beta_{\mathrm{R}} = \beta_{\mathrm{B}} = 20t_b + t_m + t_s = 220000 / v + 20,$$

reprezentujących czas w milisekundach, gdzie v jest szybkością transmisji, oraz ograniczenia długości ramek

$$\omega_{\mathrm{R}} = 125, \quad \omega_{\mathrm{B}} = 2000.$$

Tabela 1: Liczba i zakres zmiennych w instancjach testowych
Table 1: Number and range of variables in benchmark instances

Rejestrowe	Binarne L	Binarne H	Suma	Zakres
8	6 (w tym 2 LH)	6 (w tym 2 LH)	20	0–256
24	18 (w tym 6 LH)	18 (w tym 6 LH)	60	0–512
72	54 (w tym 18 LH)	54 (w tym 18 LH)	180	0–1024



Rys. 7. Przykładowe rozwiązania instancji problemu z 20 zmiennymi
Fig. 7. Exemplary solutions to a problem instance with 20 variables

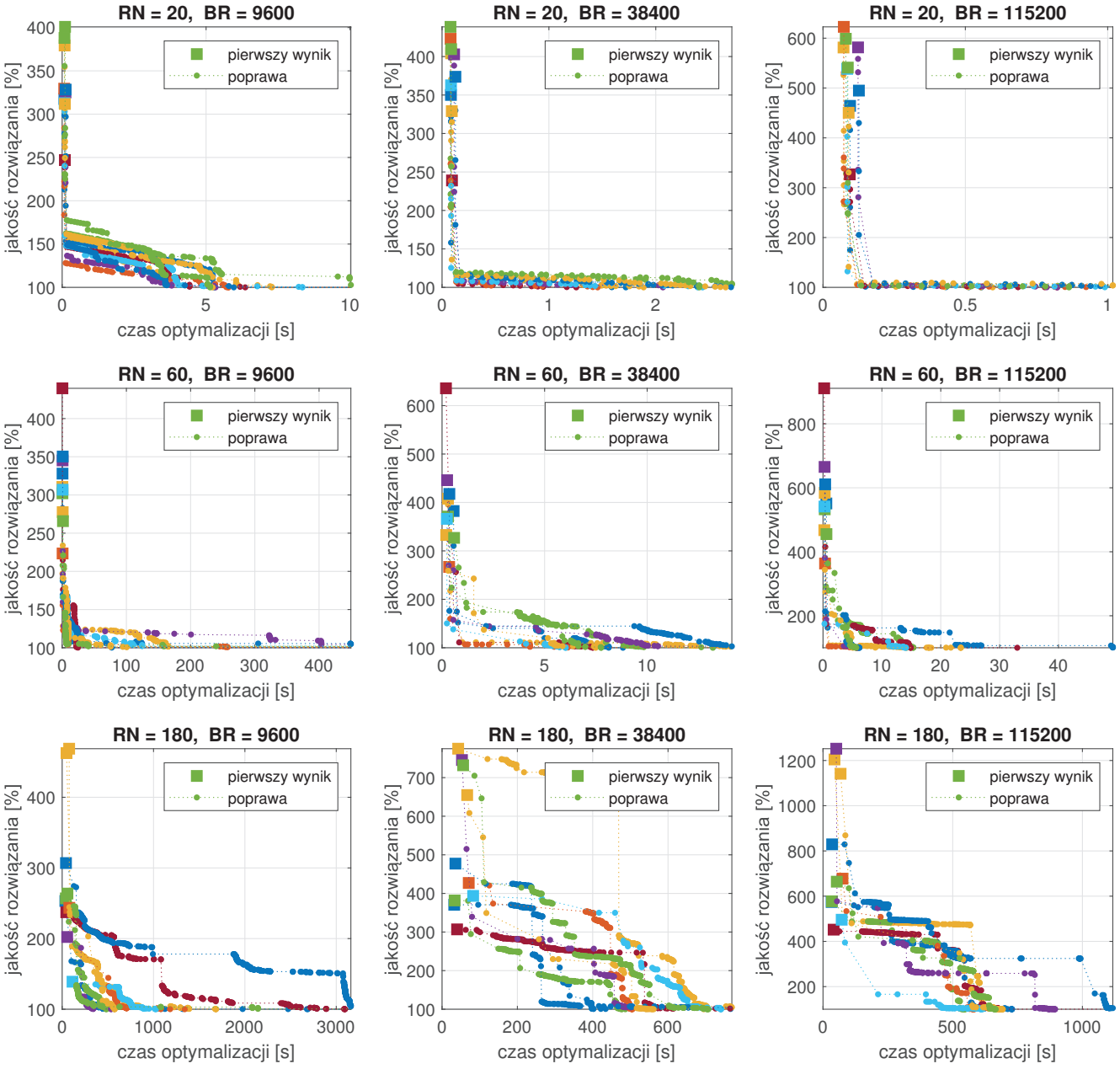
Do badań obliczeniowych wygenerowano trzy zestawy instancji testowych z losowym rozmieszczeniem zmiennych. Zestawy zawierają kolejno 20, 60 i 180 zmiennych, przy czym 40 % wszystkich zmiennych w każdej instancji stanowią zmienne rejestrowe, a 60 % binarne, po 30 % L i H, w tym 20 % to pary LH zajmujące to samo słowo 16-bitowe. Wraz ze wzrostem liczby zmiennych przyjęto również powiększanie zakresu adresowego, z którego były losowane. Parametry instancji testowych zostały zebrane w Tab. 1. Dla każdej instancji wykonano obliczenia z uwzględnieniem trzech popularnych prędkości transmisji $v \in \{9600, 38\,400, 115\,200\}$. Łącznie wykonano zatem optymalizację dla 90 wariantów testowych. W każdym przypadku przyjęto limit czasu obliczeń równy jednej godzinie.

Na Rys. 7 przedstawiono wynik uzyskany dla jednej przykładowej instancji zawierającej 20 zmiennych przy prędkości transmisji 9600 b/s. Zaprezentowano wyznaczone ramki i odpowiadające im wartości funkcji celu t_{cycle} dla wybranych rozwiązań pośrednich z toku optymalizacji (a) i (b), a także dla końcowego najlepszego rozwiązania (c). W każdym z kolejnych rozwiązań można zaobserwować zasadne decyzje prowadzące do skrócenia czasu komunikacji. Najpierw (a) wyodrębnione zostają dwie ramki rejestrowe pomijające obszar największej przerwy między zmiennymi, którego transmitowanie zwiększyłoby czas komunikacji. Następnie (b) algorytm wykrył, że korzystniej będzie skrócić ramki rejestrowe i wprowadzić ramkę binarną obejmującą większość zmiennych w środkowej części zakresu adresowego. W końcu (c) pierwsza ramka rejestrowa została zupełnie rozbita na komunikaty zawierające pojedyncze zmienne, a ramka binarna rozszerzona na pełny zakres adresowy, co okazuje się jeszcze bardziej skracać czas komunikacji.

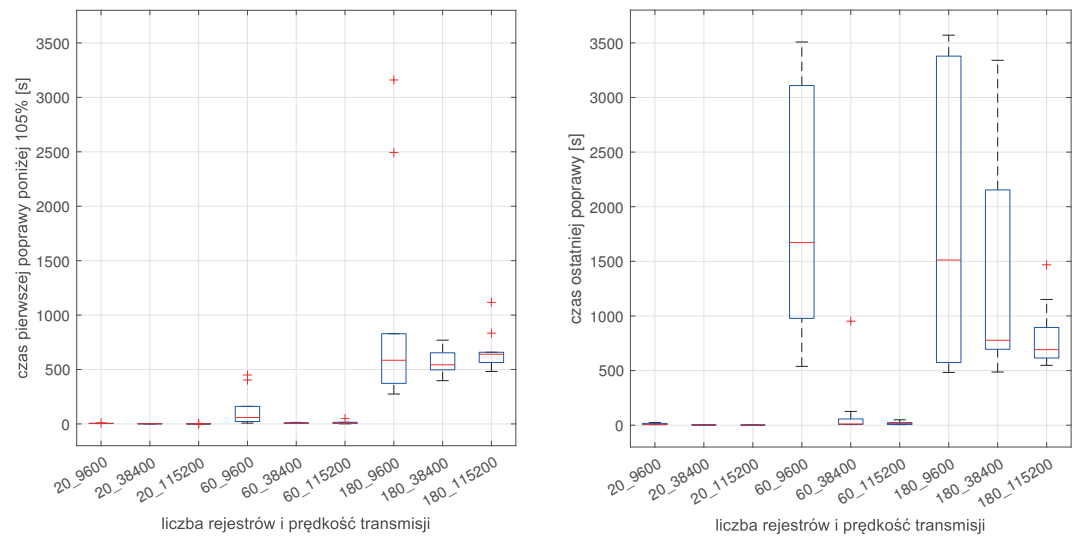
Przebieg procesu optymalizacji zbiorczo dla wszystkich wariantów testowych został przedstawiony na wykresach z Rys. 8. Ukazują one zmiany wartości funkcji celu w czasie, odrębnie dla poszczególnych 10-elementowych zestawów instancji testowych z określoną liczbą rejestrów (RN) i prędkością transmisji (BR). Z uwagi na cel porównawczy, wartości funkcji celu podane są w formie znormalizowanej względem ostatniego (najlepszego) wyniku uzyskanego dla każdego testu, co można uznać za wskaźnik bieżącej jakości rozwiązania. Rozwiązania początkowe są zawsze od kilku do kilkunastu razy gorsze od finalnych, co ogólnie potwierdza, że zły plan transmisji może znacząco wydłużyć jej czas i optymalizacja jest ważna. Jak należało oczekiwać, szybkość zbieżności algorytmu wyraźnie spada ze wzrostem liczby zmiennych. Można zauważyć, że ta szybkość zależy również od prędkości transmisji przy stałej liczbie zmiennych i jest wyraźnie najmniejsza dla BR = 9600, natomiast większa i mniej zróżnicowana dla dwóch pozostałych prędkości. Wykresy wskazują najszybszą zbieżność dla przypadku 20 zmiennych i BR = 38 400 lub BR = 115 200. Rzeczywiście, dla tych 20 wariantów testowych solwer zakończył obliczenia przed limitem czasu i uzyskał potwierdzenie optymalności rozwiązania. We wszystkich pozostałych wariantach obliczenia trwały pełną godzinę i optymalność najlepszego rozwiązania nie została potwierdzona.

Uzupełnienie wykresów zbieżności algorytmu stanowią wykresy pudełkowe z Rys. 9 przedstawiające rozkład dwóch wybranych czasów charakterystycznych dla przebiegu obliczeń: (a) czas do pierwszej poprawy poniżej 105 % wartości końcowej i (b) czas do ostatniej poprawy wartości funkcji celu. Wszystkie te czasy są zaniedbywalnie małe dla wariantów z 20 zmiennymi, ale są znaczące dla wszystkich wariantów z instancjami zawierającymi 180 zmiennych. W przypadku 60 zmiennych sytuacja jest zróżnicowana i charakteryzuje się długimi czasami tylko dla BR = 9600. Wykresy wskazują, że w wielu przypadkach czasy charakterystyczne sięgają jednogodzinnego limitu czasu obliczeń, zatem wydłużenie czasu optymalizacji dałoby w tych przypadkach dużą szansę na dalszą poprawę wyników.

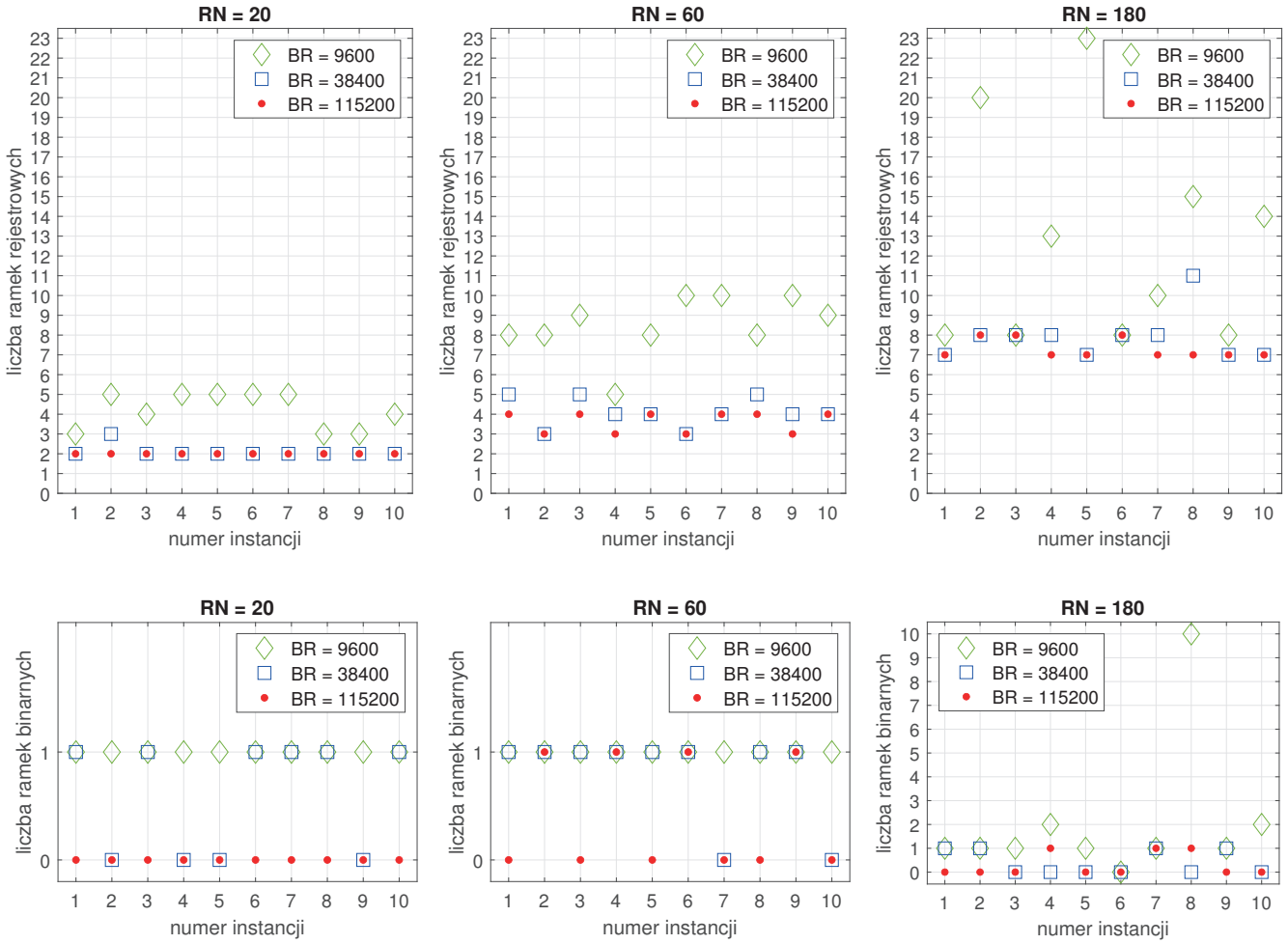
Na wykresach z Rys. 10 przedstawiono podstawowe cechy strukturalne najlepszych uzyskanych rozwiązań w postaci liczby ramek rejestrowych i binarnych, na jakie został podzielony proces transmisji danych. Liczba ramek rejestrowych rośnie w systematyczny sposób wraz ze wzrostem liczby zmiennych,



Rys. 8. Zmiana funkcji celu w czasie optymalizacji
Fig. 8. Objective function value as a function of optimization time



Rys. 9. Czasy charakterystyczne dla przebiegu optymalizacji
Fig. 9. Characteristic times of optimization process



Rys. 10. Liczby ramek rejestrowych i binarnych w zoptymalizowanych planach odczytu
Fig. 10. Numbers of register and binary frames in optimized reading plans

przy czym wynosi ona przynajmniej 2, 3 lub 7 odpowiednio dla 20, 60 lub 180 zmiennych. W żadnym przypadku wszystkie zmienne nie zostały spakowane do jednej ramki rejestrowej, ale uniemożliwia to jej limit długości $\omega_R = 125$, przy jego braku być może powstałyby takie rozwiązania optymalne. Dla ustalonej liczby zmiennych liczba ramek rejestrowych rośnie wraz ze zmniejszaniem prędkości transmisji i uzyskuje duże wartości szczególnie dla BR = 9600. Tę zależność można łatwo wyjaśnić jakościowo. Dla relatywnie małej prędkości transmisji stały wkład czasu przetwarzania $t_p = t_m + t_s$ przypadający na każdą ramkę staje się relatywnie mniejszy względem czasu transmisji znaku t_b, przez co zwielokrotnianie t_p przy wyodrębnianiu dodatkowych ramek może być lepsze od zwielokrotniania t_b w długich ramkach z adresami nieobsadzonymi przez zmienne. Dla relatywnie dużej prędkości transmisji zależności odwracają się. Duża liczba ramek dla BR = 9600 pozwala też lepiej zrozumieć wydłużenie czasów optymalizacji dla takich przypadków obserwowane na Rys. 8. Jest to najprawdopodobniej spowodowane znaczną liczbą rozwiązań suboptymalnych z dużą liczbą ramek, które tworzą większą przestrzeń przeszukiwań. Liczba ramek binarnych w rozwiązaniach najlepszych wynosi zwykle 0 lub 1. W pierwszym przypadku wykorzystywana jest możliwość przesłania zmiennej binarnej w ramce rejestrowej. W drugim przypadku, jeśli już zastosowanie ramki binarnej jest korzystne, zwykle jej podział na mniejsze ramki byłby nieprzydatny z uwagi na znaczną proporcję czasu tp względem czasu transmisji właściwych danych. Wyjątkowo, w przypadku jednej instancji dla RN = 180 i BR = 9600, rozwiązanie ma aż 10 ramek binarnych.

6. Podsumowanie

W artykule przeanalizowano wpływ różnych sposobów odczytu zmiennych binarnych za pomocą protokołu Modbus na łączny czas cyklu komunikacyjnego. Pokazano sytuację, gdy odczyt zmiennych binarnych wraz ze zmiennymi rejestrowymi może prowadzić do poprawy parametrów czasowych, jak też przypadki gdy korzystniejsze jest użycie dedykowanych funkcji bitowych.

Dobór optymalnego grupowania zmiennych do osobnych zadań komunikacyjnych jest złożonym zagadnieniem, zwłaszcza w praktycznych zastosowaniach, gdy liczba zmiennych jest znaczna. W artykule zaproponowano model optymalizacyjny pozwalający na automatyczne znalezienie najlepszego rozwiązania za pomocą uniwersalnego oprogramowania optymalizacyjnego. Przedstawiono przeprowadzone eksperymenty i omówiono uzyskane wyniki. Jak pokazano, zaproponowany model może być użyteczny dla projektanta systemu.

Uzyskane wyniki eksperymentów wskazują, że celowy będzie dalszy rozwój modeli i algorytmów optymalizacji dla problemu transmisji rejestrowobinarnej. Nowe podejścia mogą się okazać szybsze i wydajniejsze od zaproponowanego, a także pozwolą lepiej zweryfikować jakość uzyskanych rozwiązań przez ich porównanie z dodatkowymi wynikami.

Zaobserwowane cechy zoptymalizowanych rozwiązań, zwłaszcza widoczne na Rys. 7 i 10, wskazują, że najlepsze plany transmisji charakteryzują się pewnymi łatwo identyfikowalnymi cechami, takimi jak np. minimalna liczba ramek binarnych, czy podział ramek rejestrowych w miejscach, w których

odstęp między zmiennymi przekracza pewną wartość. Sugeruje to możliwość opracowania szybkich algorytmów heurystycznych, możliwych do zaimplementowania bezpośrednio w urządzeniach PLC, które na podstawie wyodrębnionych prostych reguł mogłyby tworzyć przynajmniej suboptymalne plany komunikacji. Rozwiązania zoptymalizowane metodami zaawansowanymi, jak użyte w niniejszej pracy programowanie z ograniczeniami, mogą stanowić dane referencyjne do oceny algorytmów heurystycznych.

Dodatek

W dodatku przedstawiono główne elementy modelu programowania z ograniczeniami, który wykorzystano do rozwiązania zadania optymalizacji omówionego w rozdziale 4 z pomocą oprogramowania IBM CPO.

Na podstawie danych zadania $\mathcal{P}(1)$ zdefiniowano: $c_{\mathrm{R}} = |R|$,

$R = \{r_1, r_2, ..., r_{c_{\mathrm{R}}}\}$, $B = \left\{2v \middle| v \in L\right\} \cup \left\{2v+1 \middle| v \in H\right\}$, $c_{\mathrm{B}} = |B|$,

$B = \{b_1, b_2, ..., b_{c_{\mathrm{B}}}\}$, $m_{\mathrm{B}} = \max(B)$, $m_{\mathrm{RB}} = \max\left(R \cup L \cup H\right)$.

W modelu optymalizacyjnym CP przyjęto następujące zmienne decyzyjne:

- $x_i^{\mathrm{R}}, y_i^{\mathrm{R}} \in \left\{0, ..., m_{\mathrm{RB}}\right\}$ dla $i \in \left\{0, ..., c_{\mathrm{R}}\right\}$ – reprezentują dolne i górne ograniczenia i-tego przedziału w F_{R},
- $x_i^{\mathrm{B}}, y_i^{\mathrm{B}} \in \left\{0, ..., m_{\mathrm{B}}\right\}$ dla $i \in \left\{0, ..., c_{\mathrm{B}}\right\}$ – reprezentują dolne i górne ograniczenia i-tego przedziału w F_{B},
- $v_{ij}^{\mathrm{R}} \in \left\{0, 1\right\}$ dla $i, j \in \left\{1, ..., c_{\mathrm{R}}\right\}$ – równa 1 wtedy i tylko wtedy, gdy $r_i \in \left[x_j^{\mathrm{R}}, y_j^{\mathrm{R}}\right]$,
- $v_{ij}^{\mathrm{B}} \in \left\{0, 1\right\}$ dla $i, j \in \left\{1, ..., c_{\mathrm{B}}\right\}$ – równa 1 wtedy i tylko wtedy, gdy $b_i \in \left[x_j^{\mathrm{B}}, y_j^{\mathrm{B}}\right]$,
- $v_{ij}^{\mathrm{BR}} \in \left\{0, 1\right\}$ dla $i \in \left\{1, ..., c_{\mathrm{B}}\right\}$, $j \in \left\{1, ..., c_{\mathrm{R}}\right\}$ – równa 1 wtedy i tylko wtedy, gdy $\left\lfloor b_i / 2 \right\rfloor \in \left[x_j^{\mathrm{R}}, y_j^{\mathrm{R}}\right]$.

Dostosowanie przedziałów do przydzielonych zmiennych zapewniają ograniczenia warunkowe

$$v_{ij}^{\mathrm{R}} = 1 \Rightarrow x_j^{\mathrm{R}} \le r_i, \quad v_{ij}^{\mathrm{R}} = 1 \Rightarrow y_j^{\mathrm{R}} \ge r_i, \text{ dla } i, j \in \left\{1, ..., c_{\mathrm{R}}\right\},$$

$$v_{ij}^{\mathrm{B}} = 1 \Rightarrow x_j^{\mathrm{B}} \le b_i, \quad v_{ij}^{\mathrm{B}} = 1 \Rightarrow y_j^{\mathrm{B}} \ge b_i, \text{ dla } i, j \in \left\{1, ..., c_{\mathrm{B}}\right\},$$

$$v_{ij}^{\mathrm{BR}} = 1 \Rightarrow x_j^{\mathrm{R}} \le \left\lfloor b_i / 2 \right\rfloor, \quad v_{ij}^{\mathrm{BR}} = 1 \Rightarrow y_j^{\mathrm{R}} \le \left\lfloor b_i / 2 \right\rfloor,$$

$$\text{dla } i \in \left\{1, ..., c_{\mathrm{B}}\right\}, \quad j \in \left\{1, ..., c_{\mathrm{R}}\right\}.$$

Przy tak zdefiniowanych i powiązanych zmiennych decyzyjnych warunek (5) sprowadza się do ograniczeń

$$\sum_{j=0}^{c_{\mathrm{B}}} v_{ij}^{\mathrm{R}} \ge 1 \quad \text{dla} \quad i \in \left\{1, ..., c_{\mathrm{R}}\right\},$$

natomiast warunek (6) do ograniczeń

$$\sum_{j=0}^{c_{\mathrm{B}}} v_{ij}^{\mathrm{B}} + \sum_{j=0}^{c_{\mathrm{R}}} v_{ij}^{\mathrm{BR}} \ge 1 \quad \text{dla} \quad i \in \left\{1, ..., c_{\mathrm{B}}\right\}.$$

Ograniczenia (7) i (8) bez zmian przenoszą się do modelu CP. Jeżeli do jakiegoś przedziału w F_{R} lub F_{B} nie zostanie przydzielona żadna transmitowana zmienna, to możliwe staje się spełnienie ograniczeń odpowiednio $x_i^{\mathrm{R}} > y_i^{\mathrm{R}}$ lub $x_i^{\mathrm{B}} > y_i^{\mathrm{B}}$, które w modelu potraktowano jako warunki wykluczenia wkładu danego przedziału do wartości funkcji celu. Dzięki temu model działa poprawnie przy zmiennej, nieznanej z góry, liczbie ramek. W implementacji funkcji celu (4) pomocne były działania dzielenia całkowitoliczbowego i reszty z dzielenia wspierane przez CPO, które ułatwiły odwzorowanie występującej w niej nieciągłości.

Bibliografia

1. IEC, *IEC 61131-3 – Programmable controllers – Part 3: Programming languages*, 2003, 2013.
2. Kasprzyk J., *Programowanie sterowników przemysłowych*. Wydawnictwo Naukowe PWN, 2006.
3. Rzońca D., Sadolewski J., Stec A., Świder Z., Trybus B., Trybus L., *Implementacja środowiska inżynierskiego na przykładzie pakietu CPDev*, „Pomiary Automatyka Robotyka", Vol. 24, Nr. 1, 2020, 21–28, DOI: 10.14313/PAR_235/21.
4. Rzońca D., Sadolewski J., Stec A., Świder Z., Trybus B., Trybus L., *Developing a Multiplatform Control Environment*, "Journal of Automation, Mobile Robotics and Intelligent Systems", Vol. 13, No. 4, 2019, 73–84, DOI: 10.14313/JAMRIS/4-2019/40.
5. Świder Z., *Prototyp kaskadowego autopilota okrętowego zaimplementowany w środowisku CPDev*, „Pomiary Automatyka Robotyka", Vol. 27, Nr 1, 2023, 61–66, DOI: 10.14313/PAR_247/61.
6. Stec A., Świder Z., Trybus L., *Jednolite projektowanie regulatorów kursu i ścieżki dla autopilota statku*, "Pomiary Automatyka Robotyka", Vol. 27, Nr 1, 2023, 45–50, DOI: 10.14313/PAR_247/45.
7. Świder Z., *Edytory graficzne języków LD i FBD w pakiecie CPDev*, „Pomiary Automatyka Robotyka", Vol. 24, Nr 1, 2020, 29–34, DOI: 10.14313/PAR_235/29.
8. Trybus B., *Development and Implementation of IEC 61131-3 Virtual Machine*, "Theoretical and Applied Informatics", Vol. 23, No. 1, 2011, 21–35, DOI: 10.2478/v10179-011-0002-z.
9. Hubacz M., Trybus B., *Dual-Core PLC for Cooperating Projects with Software Implementation*, "Electronics", Vol. 12, No. 23, 2023, DOI: 10.3390/electronics12234730.
10. Stój J., *Wybrane zagadnienia sieci komunikacyjnych w przemysłowych systemach komputerowych*. Wydawnictwo Politechniki Śląskiej, 2023.
11. 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 (Flores P., Viadero F., eds.), (Cham), Springer International Publishing, 2015, 933–940, DOI: 10.1007/978-3-319-09411-3_97.
12. Thomesse J., *Fieldbus Technology in Industrial Automation*, "Proceedings of the IEEE", Vol. 93, No. 6, 2005, 1073–1101, DOI: 10.1109/JPROC.2005.849724.
13. 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.
14. IEC, *IEC 61158 – Industrial Communication Networks – Fieldbus Specifications*, 2007.
15. Scanzio S., Wisniewski L., Gaj P., *Heterogeneous and dependable networks in industry – A survey*, "Computers in Industry", Vol. 125, 2021, DOI: 10.1016/j.compind.2020.103388.

16. Rzońca D., *Poprawa wydajności komunikacji sterownika przemysłowego z panelem operatorskim HMI w środowisku inżynierskim CPDev*, „Pomiary Automatyka Robotyka", Vol. 24, Nr 1, 2020, 35–40, DOI: 10.14313/PAR_235/35.
17. Rzońca D., *Przyspieszenie wymiany danych w protokole Modbus między PLC a HMI wykorzystującymi pakiet inżynierski CPDev*, Pomiary Automatyka Robotyka, Vol. 26, Nr 4, 2022, 85–89, DOI: 10.14313/PAR_246/85.
18. Titaev A., *Reducing update data time for exchange via MODBUS TCP protocol by controlling a frame length*, "Automatic Control and Computer Sciences", Vol. 51, 2017, 357–365, DOI: 10.3103/S014641161705008X.
19. Gãitan V.G., Zagan I., *Modbus Protocol Performance Analysis in a Variable Configuration of the Physical Fieldbus Architecture*, "IEEE Access", Vol. 10, 2022, 123942–123955, DOI: 10.1109/ACCESS.2022.3224720.
20. Zagan I., Gãitan V.G., *Enhancing the Modbus Communication Protocol to Minimize Acquisition Times Based on an STM32-Embedded Device*, "Mathematics", Vol. 10, No. 24, 2022, DOI: 10.3390/math10244686.
21. Gãitan V.G., Zagan I., *Experimental Implementation and Performance Evaluation of an IoT Access Gateway for the Modbus Extension*, "Sensors", Vol. 21, No. 1, 2021, DOI: 10.3390/s21010246.
22. Bednarek M., Będkowski L., Dąbrowski T., *Wybrane funkcje systemu dozorująco-terapeutycznego w układzie komunikacji*, „Diagnostyka", Vol. 34, 2005, 31–36.
23. Bożek A., Rzońca D., *Communication Time Optimization of Register-Based Data Transfer*, "Electronics", Vol. 12, No. 24, 2023, DOI: 10.3390/electronics12244917.
24. Laborie P., Rogerie J., Shaw P., Vilím P., *IBM ILOG CP optimizer for scheduling*, "Constraints", Vol. 23, 2018, 210–250, DOI: 10.1007/s10601-018-9281-x.

Optimization of Mixed Reading of Binary and Register Variables in Modbus Protocol from PLC Implementing CPDev

Abstract: Efficient communication is crucial for the proper operation of distributed automation systems. The article focuses on one of the aspects of such communication, related to the mixed reading of binary and register variables from the PLC controller in the Modbus RTU protocol. The research assumed a memory architecture with common addressing of variables of different types, allowing reading binary variables not only using dedicated bit functions (such as FC1), but also Modbus register functions (such as FC3). Such an architecture occurs, e.g., in controllers implementing the CPDev engineering environment. The article proposes a method of appropriate grouping of variables, leading to a reduction of the total communication cycle time. An optimization model has been implemented to automatically find the optimal grouping. Experiments were carried out and the obtained results were discussed. The results of the conducted research will be used in the development of the CPDev engineering environment.

Keywords: industrial controller, PLC, communication, Modbus, CPDev

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). Od 2004 roku asystent, a od 2013 adiunkt w Katedrze Informatyki i Automatyki Politechniki Rzeszowskiej. Jego zainteresowania naukowe koncentrują się na kolorowanych sieciach Petriego oraz zagadnieniach związanych z komunikacją w systemach automatyki.

dr inż. Andrzej Bożek
abozek@prz.edu.pl
ORCID: 0000-0003-3015-7474

Absolwent Wydziału Elektrotechniki i Informatyki Politechniki Rzeszowskiej (2008). Stopień doktora nauk technicznych w dyscyplinie informatyka uzyskał w 2015 r. Pracuje jako adiunkt w Katedrze Informatyki i Automatyki Politechniki Rzeszowskiej. Jego zainteresowania naukowe dotyczą optymalizacji dyskretnej, harmonogramowania zadań oraz projektowania układów sterowania.