Pomiary Automatyka Robotyka, ISSN 1427-9126, R. 26, Nr 4/2022, 79–84, DOI: 10.14313/PAR_246/79

Architektura niskoenergetycznego uniwersalnego sterownika programowalnego

Marcin Hubacz, Bartosz Pawłowicz, Bartosz Trybus
Politechnika Rzeszowska, Wydział Elektrotechniki i Informatyki, ul. Wincentego Pola 2, 35-021 Rzeszów

Streszczenie: Artykuł przedstawia koncepcję budowy architektury niskoenergetycznego sterownika programowalnego opracowanego zgodnie z normą IEC 61131-3. Uniwersalność dotyczy swobody wyboru jednostki centralnej oraz możliwości łatwej wymiany algorytmu sterującego. Do proponowanego rozwiązania wybrana zostaje jedna z omówionych trzech metod opracowywania i dystrybucji oprogramowania. W ramach prac przygotowany został prototypowy sterownik w oparciu o elementy ewaluacyjne, z przeznaczeniem do pracy w środowisku rozproszonym. W artykule przedstawiono również porównanie wydajności energetycznej wybranych układów STM32 z kilku odmiennych serii.

Słowa kluczowe: PLC, Internet Rzeczy, Low Power, IEC 61131-3

1. Wprowadzenie

Komputerowe systemy sterowania zrewolucjonizowały nasz system funkcjonowania i wykonywania różnego rodzaju operacji, wspierając w wykonywaniu codziennych działań i automatyzując procesy produkcyjne. Programowalne sterowniki logiczne PLC zazwyczaj kojarzone są z zastosowaniem w przemyśle, gdzie ich głównym zadaniem jest sterowanie procesami. Przy bardzo mocnym uproszczeniu, wygenerowane sterowanie jest efektem zmiany wejść. Różnorodne czujniki dostarczają potrzebnych danych do poprawnego rozpoznania zachodzących procesów.

Niewielkie systemy sterowania, często mieszczące się w jednej szafie sterowniczej, mogą wykorzystywać do swojego działania dane pochodzące z rozproszonej sieci czujników. Budowa takich systemów wymaga nakładów finansowych przeznaczonych na moduły stosujące przemysłowe protokoły komunikacyjne. Podczas próby rozszerzenia potencjalnych granic wykorzystania sterowników PLC można napotkać na ograniczenia związane z obsługiwanymi technologiami, jak komunikacja bezprzewodowa lub wsparcie nowoczesnych rozwiązań transmisji danych. Brak na rynku dedykowanych rozwiązań lub ich koszt zmusza projektantów do łączenia sterowników PLC z zewnętrznymi, dedykowanymi systemami.

Naturalnym następstwem eksploatacji każdego systemu sterowania jest jego modyfikacja lub modernizacja. Języki normy IEC 61131-3 upraszczają i standaryzują wprowadzanie stosownych zmian. Z kolei wykorzystanie w systemie urządzeń spoza automatyki przemysłowej, zamiast dedykowanych modułów może skutecznie uniemożliwić wprowadzenie modyfikacji dostosowanych do nowych wymogów.

W 1999 r. Kevin Ashoton zaprezentował termin Internet Rzeczy IoT (ang. *Internet of Things*), który przez wiele lat ewoluował, stale zwiększając popularność, a obecnie pozwolił zaistnieć 14,4 mld różnych urządzeń [1]. Zakres potencjału stosowania IoT jest bardzo szeroki, obejmuje przemysł (*Industrial IoT*), zarządzanie miastami (*smart city*), usługi zdrowotne, urządzenia gospodarstwa domowego (*smart home*) a nawet wykorzystany jest na urządzeniach noszonych (*wearables*).

Wydaje się, że można skutecznie doposażyć nowoczesne rozwiązania IoT w zasady określone w standardach automatyki, w szczególności w normie IEC 61131-3, opracowując hybrydę przemysłowych i przenaszalnych rozwiązań dedykowanych do szerokiego grona odbiorców. Dzięki takiemu zabiegowi, przez wykorzystanie wieloletniego doświadczenia branży automatyki, poprawiona zostanie spójność opracowywania systemów wykorzystujących do swojego działania zróżnicowane elementy.

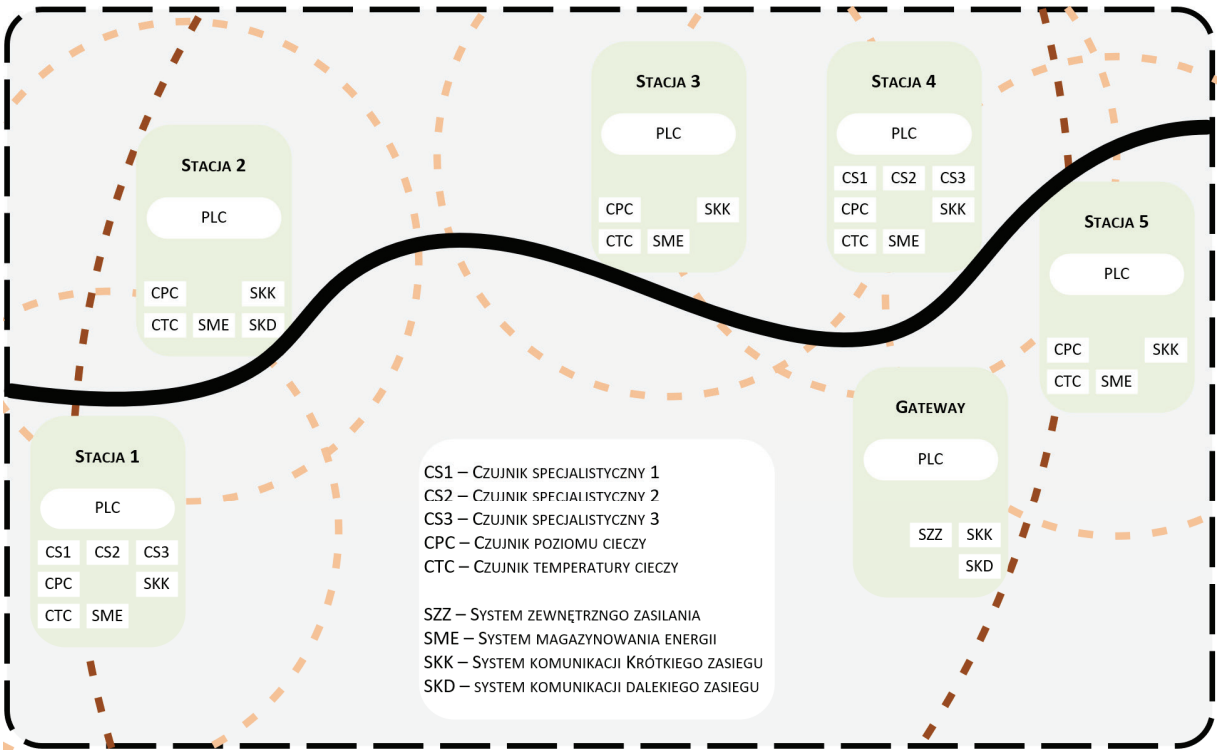
Ważnym aspektem systemów Internetu Rzeczy jest zapotrzebowanie na energię. Producenci elektroniki systematycznie poprawiają wydajność energetyczną swoich urządzeń. Zasilanie bateryjne jest tu podstawowym sposobem dostarczenia energii do urządzenia. Podstawowym parametrem branym pod uwagę podczas wyboru konkretnego rozwiązania będzie interwał wymiany źródła zasilania. Sposobem na uniknięcie zbyt częstego serwisowania mogą być alternatywne źródła energii jak ogniwa fotowoltaiczne.

2. Struktura systemu rozproszonego

Rozproszony system sterowania wymaga rozwinięcia kwestii projektowania, dystrybucji oraz obsługi oprogramowania. Ustalenie odpowiednich zasad i poznanie możliwych ograniczeń jest konieczne do poprawnej pracy uniwersalnego systemu PLC z IoT.

Autor korespondujący:
Marcin Hubacz, m.hubacz@prz.edu.pl

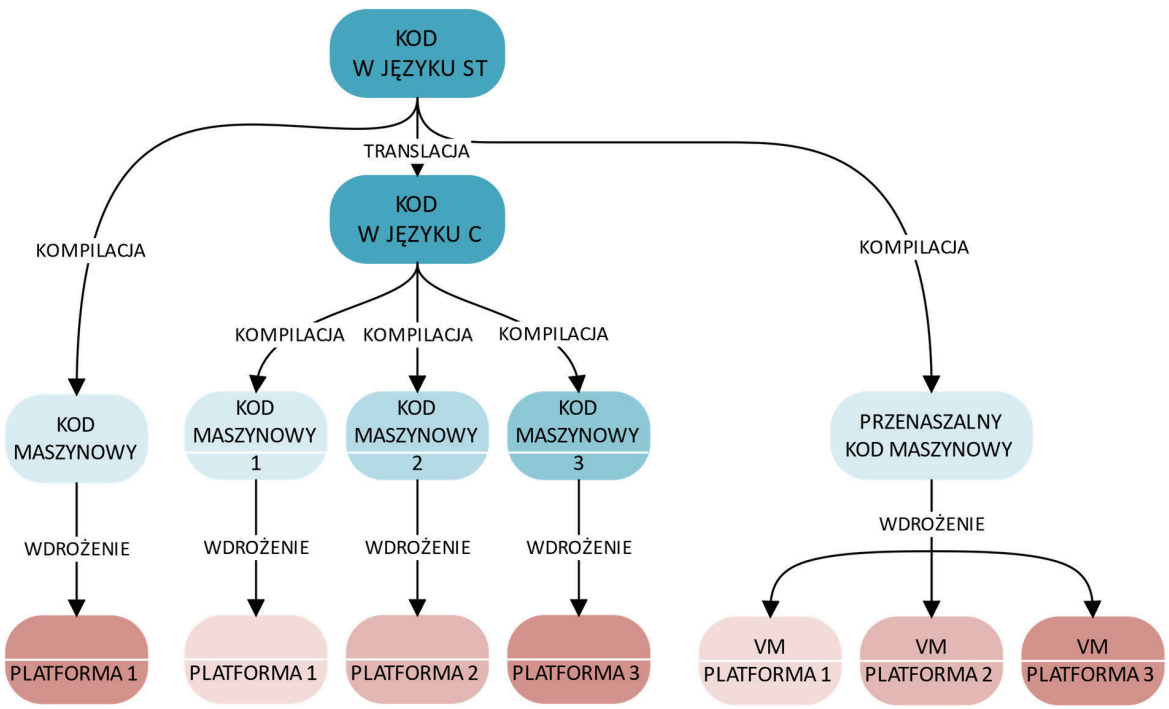
Artykuł recenzowany
nadesłany 23.10.2022 r., przyjęty do druku 16.11.2022 r.



Rys. 1. Struktura systemu rozproszonego monitorowania parametrów rzeki
Fig. 1. Structure of a distributed monitoring system for river parameters

Zaprojektowanie systemu sterowania zaczyna się od ustalenia badanych zmiennych fizycznych wiążąc to z wymaganym sterowaniem. Ciekawym przykładem może być system monitorowania i reagowania dla hydrologii, badający zmiany jakości i poziomu wód np. w rzece. Zróżnicowane warunki oraz duże rozproszenie systemu wymagać będzie skonstruowania rozmaitych stacji badawczych. Dodatkowo, w planie takiego systemu należy przewidzieć rozszerzenie możliwości pomiarowych systemu oraz jego rozbudowę o kolejne stacje monitorująco-sterujące.

Przykładową strukturę takiego modelu zamieszczono na rysunku 1. Zróżnicowane stacje pracują w jednym, spójnym systemie. Wykorzystując komunikację krótkiego i dalekiego zasięgu możliwe jest zbudowanie sieci stacji monitorujących i sterujących. Liczba badanych parametrów może zostać rozbudowana ponad kilka przykładowych elementów przestawionych na rysunku. Urządzenia pozbawione możliwości stałego źródła zasilania wyposażone są w alternatywny systemu pozyskiwania i magazynowania energii. Tym samym stają się samowystar-



Rys. 2. Trzy sposoby opracowywania oprogramowania dla systemów w normie IEC 61131-3
Fig. 2. Three ways to develop software for systems in the IEC 61131-3 standard

czalnymi stacjami rozmieszczonymi np. w trudnym rozległym terenie. Urządzenie typu *gateway* umożliwia natomiast scalenie rozproszonego systemu i dostęp do rozwiązań chmurowych.

Realizacja powyższej koncepcji systemu monitorująco-sterującego w oparciu o typowe sterowniki PLC wygeneruje znaczący koszt pojedynczego punktu obserwacji i prawdopodobnie ograniczy możliwość znacznego zwielokrotnienia i zagęszczenia czujników. Wymóg stałego źródła zasilania dodatkowo zawęzi możliwość montażu stacji badawczych do niewielkiego obszaru. Dedykowane systemy monitorowania będą uzależniać projekt wyłącznie od kompatybilnych elementów, blokując innowacyjność i utrudniając rozszerzanie i uzależniając je od producenta.

Wychodząc na przekór pojawiającym się ograniczeniom, podczas projektowania systemu monitowania możliwe jest zastosowanie uniwersalnych rozwiązań, w którym część sterująca jest niezależna od konkretnej platformy sprzętowej i łatwo reprogramowalna. W takim podejściu uzyskamy idealne dopasowanie co do zastosowanych technologii, czujników, sterowań oraz źródła zasilania.

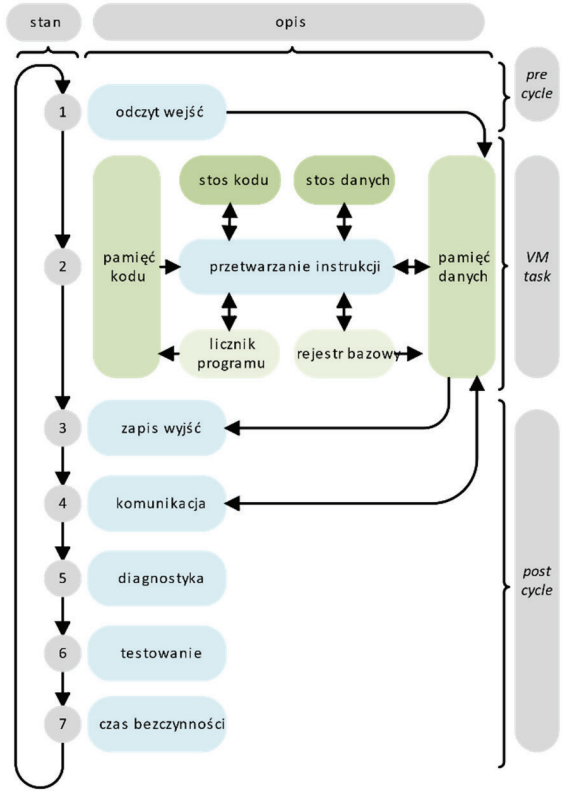
3. Programowanie wieloplatformowe

Przy projektowaniu rozbudowanych systemów sterowania zastosowanie zunifikowanego mechanizmu tworzenia oprogramowania i jego dystrybucji jest pożądaną cechą. W systemach sterowania standaryzacja języków programowania jest osiągana przede wszystkim dzięki normie IEC 61131-3. W narzędziach implementujących tę normę oprogramowanie opracowywane jest obecnie na trzy różne sposoby.

W pierwszym z nich, przedstawionym z lewej strony rysunku 2, programy IEC (na przykład w języku ST) kompilowane są do kodu natywnego platformy docelowej. Stosunkowo nieskomplikowane środowisko wykonawcze pozwala na efektywne wykonywanie wdrożonego oprogramowania. Podejście to stosowane jest przez dużych producentów, którzy produkują urządzenia w długich seriach oparte o zunifikowane komponenty sprzętowe. Jest to rozwiązanie dla określonego procesora, ponieważ zmiana platformy wymaga nowego kompilatora.

Drugie podejście, przedstawione w części środkowej rysunku 2 polega na przetłumaczeniu programów IEC na kod w języku C/C++, a następnie wygenerowaniu natywnego kodu binarnego za pomocą kompilatorów specyficznych dla każdej z platform [2-4]. Pierwszy krok umożliwia tworzenie kodu wieloplatformowego, jednak drugi etap utrudnia dystrybucję programów na różne platformy, gdyż niezbędne jest wykorzystanie wielu kompilatorów C/C++. Przygotowany kod binarny jest jednak dobrze dostosowany do danej platformy, co oznacza dobrą wydajność.

Istnieje również trzecie podejście pokazane na rysunku 2 z prawej strony. Jego celem jest uniezależnienie dystrybuowanego kodu od platformy docelowej. Rozwiązanie opiera się na koncepcji maszyny wirtualnej [5, 6]. Wygenerowany przenaszalny wieloplatformowy kod jest interpretowany przez odpowiednie środowisko wykonawcze na platformie docelowej [IsaGraf 7, Straton 8]. Wspominane środowisko wykonawcze określane jest często jako maszyna wirtualna VM (ang. *Virtual Machine*), ponieważ emuluje działanie procesora definiowanego programowo [9]. Rozwiązanie to umożliwia łatwą wymianę oprogramowania w tracie pracy urządzenia oraz przenoszenie oprogramowania między zróżnicowanymi platformami. Ponadto wdrożenie nie wymaga dodatkowego kroku natywnej kompilacji. Wadą jest natomiast mniejsza wydajność wykonywania oprogramowania. Mimo to, podejście wykorzystujące VM jest interesujące dla małych i średnich przedsiębiorstw, które produkują urządzenia w krótkich seriach, wykorzystując przy tym różnorodne procesory i mikrokontrolery.



Rys. 3. Składowe oprogramowania sterownika PLC opartego na maszynie wirtualnej
Fig. 3. Components of PLC software based on a virtual machine

W omawianej koncepcji sterownika programowalnego jako narzędzie do tworzenia oprogramowania zastosowano pakiet CPDev [10]. Maszyna wirtualna CPDev jest przenośna, co oznacza, że można dostosować do różnych platform sprzętowych i procesorowych [11]. W swojej idei oparta jest na architekturze harwardzkiej, a więc wykorzystuje do pracy dwa osobne obszary pamięci czyli kodu i danych (rys. 3). Pierwszy przechowuje interpretowany kod w postaci binarnej, drugi – wartości zmiennych prostych i złożonych (tablic, struktur). W zależności od potrzeb i sprzętu, maszyna wirtualna może korzystać z różnych metod dostępu do pamięci [12].

Pracujący sterownik PLC w oparciu o maszynę wirtualną wykonuje trzy fazy pracy (rys. 3):

— *Pre cycle*: odczyt wartości zmiennych z wejść fizycznych i źródeł zewnętrznych;
— *VM task*: przetwarzanie kodu maszynowego i interpretacja danych;
— *Post cycle*: ustawianie wyjść na podstawie obliczonych wartości zmiennych, komunikacja, diagnostyka, testowanie itp.

Moduł przetwarzania instrukcji jest kluczowym elementem całej architektury. Zestaw instrukcji dla maszyny wirtualnej składa się z bezpośrednich odpowiedników normy IEC 61131-3, skoków, instrukcji kopiowania pamięci oraz wywołań podprogramów [11,13].

Przedstawiona na rys. 3 struktura zakłada, że sterownik wykonuje pojedyncze zadanie składające się z programów i innych jednostek organizacyjnych [14]. Aby uruchomić więcej zadań jednocześnie można utworzyć dodatkowe instancje maszyn wirtualnych. W przypadku procesora wielordzeniowego można to osiągnąć poprzez przypisanie każdej instancji maszyny wirtualnej do konkretnego rdzenia. W przypadku wielozadaniowego systemu operacyjnego, zadania VM wykonywane są jako wątki lub procesy [15].

4. Sterownik PLC Low Power

Konwencjonalny sterownik PLC nie zawiera systemów oszczędzania energii. Ze względu na charakterystykę jego pracy, w szczególności potrzebę utrzymania stałego czasu cyklu, część fazy *Post cycle* z rys. 3 pozostaje zwykle niewykorzystana. Z tego powodu, po wykonaniu programu wraz z niezbędnymi elementami komunikacji oraz diagnostyki, możliwe jest bezpieczne uśpienie urządzenia na czas oczekiwania na następny cykl. W zależności od interwału wykonywania cyklu zastosowanie takiego rozwiązania może przynieść znaczące efekty w obszarze zapotrzebowania na energię. Układy licznikowe będą odpowiedzialne za regularne wybudzanie urządzenia dla uzyskania stałego cyklu.

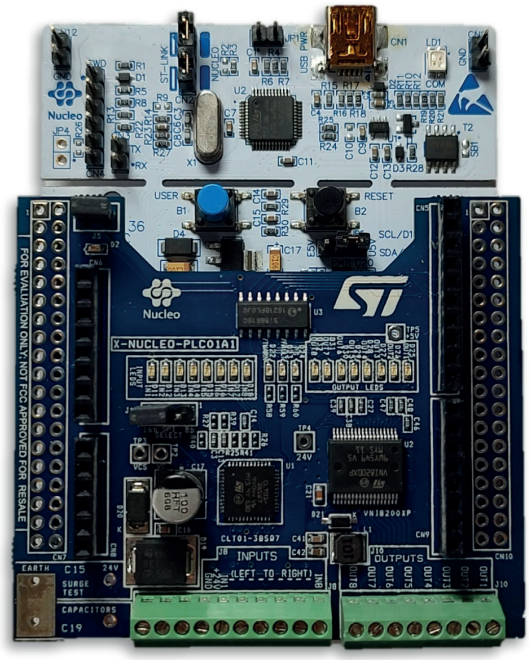
Poziomy oszczędzania energii, do których można wprowadzić dany układ elektroniczny są mocno zróżnicowane, pozwalając na dostosowanie ich głębokości do potrzeb. Do budowy prototypów rozważano m.in. mikrokontrolery serii STM32, NRF, ESP32. Przykładowo, w układzie STM32L476 można wyróżnić następujące tryby pracy [16]:

- *Run mode* – normalny tryb pracy mikrokontrolera;
- *Sleep* – CPU zostaje zatrzymany, wszystkie peryferia zostają włączone i mogą wybudzić CPU, gdy wystąpi przerwanie lub zdarzenie;
- *Low-power run* – zmniejszona częstotliwość procesora do 2 MHz, program może być wykonywany z pamięci SRAM lub FLASH, zasilanie procesora pochodzi z regulatora małej mocy, peryferia taktowane mogą być z HSI16;
- *Low-power sleep* – włączenie tego trybu wymaga uprzedniego uruchomienia trybu *Low-power run*, zatrzymywany jest tylko zegar procesora, po wybudzeniu przez zdarzenie lub przerwanie system powraca do trybu *Low-power run;*
- *Stop* 0*, Stop* 1*, Stop* 2 – tryb zatrzymania osiągający najniższe zużycie energii przy zachowaniu zawartości pamięci SRAM i rejestrów. Większość zegarów zostaje zatrzymana, podtrzymywane są wyłącznie LSE i LSI;
- Różnica między poziomami zatrzymania, a więc i ilością możliwych działających peryferii mogących wybudzić układ ze stanu uśpienia;
- *Stand by mode* – tryb czuwania z którego wybudzenie może nastąpić po resecie za pomocą dedykowanego pinu lub watchdoga lub specjalnych konfigurowalnych zdarzeń od pinów WKUP oraz RTC. Pamięć zostaje utracona z wyjątkiem rejestrów w domenie kopii zapasowej;
- *Shutdown* – tryb wyłączenia, najniższy poziom zużycia energii, sposoby wybudzenia układu zbliżone do trybu *Stand by mode.*

Podstawowym wymogiem dla sterownika przemysłowego jest nieprzerwane i niezawodne działanie. Stąd wyłącznie niezagospodarowany czas może zostać wykorzystywany na zmniejszenie poboru energii urządzenia. Sterownik powinien być regularnie wybudzany – zgodnie z zadanym cyklem, za pomocą wewnętrznych liczników generujących przerwania. Dostępne tryby usypiania układu pozwalają na opracowanie dwóch odmiennych metod oszczędzania energii.

Pierwszym podstawowym sposobem jest usypianie układu do poziomu, w którym pamięć tymczasowa oraz rejestry są podtrzymywane. Dzięki takiemu zabiegowi urządzenie pracuje w sposób zbliżony do układów niewykorzystujących funkcji oszczędzania energii. Uzyskane rozwiązanie pozwala w bardzo prosty sposób zaimplementować tryb energooszczędny.

Drugie rozwiązanie polega na wykorzystaniu najbardziej radykalnych systemów zarządzania energią i zmniejszenie poboru prądu poprzez wyłączenie wszystkich zbędnych peryferii, w tym systemu potrzymania pamięci SRAM. Takie podejście pozwala na uzyskanie najniższych możliwych poziomów zapotrzebowania



Rys. 4. Prototypowy sterownik PLC oparty na płycie ewaluacyjnej Nulceo-64 wyposażonej w mikrokontroler STM32L476 i nakładkę PLC01A1

Fig. 4. Prototype PLC controller based on the Nucleo-64 evaluation board equipped with the STM32L476 microcontroller and the PLC01A1 module

na energię, ale pociąga za sobą konsekwencje w postaci utraty pamięci tymczasowej. Ze względu na potrzebę podtrzymania pamięci ulotnej w proponowanym sterowniku, możliwe jest wykonanie zrzutu pamięci ulotnej do alternatywnej nieulotnej, która przy wznowieniu pracy urządzenia zostanie przywrócona. Przy bardzo dużym okresie czasu uśpienia sterownika względem długości pojedynczego cyklu sterownika, tryb ten może przynieść bardzo wymierne efekty.

Prototyp sterownika PLC przedstawiony został na rysunku 4. Do budowy wykorzystana została płyta ewaluacyjna Nucelo-64 z mikrokontrolerem STM32L476. Za dopasowanie wejść i wyjść do standardu automatyki odpowiada płyta rozszerzeń PLC01A1. Zbudowany model posiada wbudowany programator i debuger w wersji ST-LINK/V2-1. Płyta ewaluacyjna umożliwia pomiar prądu wbudowanego mikrokontrolera oraz doprowadzenie zewnętrznego zasilania.

5. Efektywność energetyczna

Do porównania sprawności energetycznej zgodnie z wytycznymi [17] wybrano serię mikrokontrolerów STM32 firmy STMicroelectronics. Każdy układ można scharakteryzować właściwościami takimi jak wykorzystana architektura, możliwa maksymalna częstotliwość taktowania, średni pobór prądu na MHz oraz wydajność obliczeniowa. W większości broszur informacyjnych mikrokontrolerów, szczególnie nakierowanych na zasilanie bateryjne, można znaleźć informacje o wydajności układu względem poboru prądu.

Producenci deklarują wydajność przy pomocy testu CoreMark, który w odróżnieniu od Dhrystone lepiej pasuje do systemów wbudowanych. Specyficzny sposób działania oprogramowania wykorzystuje uproszczony benchmark do określenia maksymalnej wydajności maszyny wirtualnej. Dlatego też, do porównania zużycia energii średni pobór prądu w trybie pracy (RUN) maszyny jest mierzony tylko przy maksymalnej częstotliwości chipa, a wszystkie zewnętrzne peryferia są wyłączone. Sygnał zegarowy do procesora podawany jest z HSE z wykorzystaniem

Tab 1. Porównanie układów pod względem maksymalnej wydajności, energochłonności oraz wydajności w proponowanej aplikacji
Tab 1. Comparison of systems in terms of maximum efficiency, energy consumption and efficiency in the proposed application

mikrokontroler	max częstotliwość (MHz)	deklarowana wydajność (CoreMark)	konsumpcja prądu (µA/MHz)	czas cyklu (µs)	wynik (µA/cykl)
STM32F072	48	106	270	241 630	65,23
STM32F303	72	245	380	144 511	54,91
STM32F446	180	608	178	122 922	21,88
STM32L476	80	273,5	128,75	123 012	15,82
STM32F746	216	1082	500	82 469	41,23

pętli PLL a oprogramowanie jest ładowane z pamięci FLASH. Wyniki czasu pojedynczego cyklu są średnią przygotowanych dla trzech metod dostępu do pamięci przez maszynę wirtualną.

Tabela 1 obrazuje mocne zróżnicowanie między seriami układów. Skupiając się na różnicach między zastosowanymi przykładami architektury, chipy z serii Cortex-M4 mają najlepszą wydajność energetyczną. Najniższa seria Cortex-M0 jest najmniej wydajna spośród zestawienia. W razie zapotrzebowania na największą możliwą wydajność spośród gamy tego producenta, najlepiej sprawdzi się najwyższa seria 7 w wersji zwykłej F lub usprawnionej H.

Przedstawione wyniki zostały uzyskane dla programu sterującego wykonywanego przez maszynę wirtualną CPDev. Maszyna interpretuje instrukcje w przenośnym kodzie binarnym i wykonuje je jako bloki kodu maszynowego dla docelowego procesora. Każda instrukcja VM wymaga więc kilku instrukcji procesora. Dodatkowy narzut czasowy jest wynikiem ochrony obszarów pamięci danych używanych przez zmienne, co zabezpiecza przed poważnymi awariami w przypadku wystąpienia błędu programowania. Zwiększona złożoność obliczeniowa jest kompensowana przez przenośność i kompatybilność kodu pomiędzy różnymi platformami.

6. Podsumowanie

W ramach prac opracowany został rozwojowy energooszczędny sterownik PLC oparty o mikrokontroler STM32 z serii L o niskim poborze prądu (*Ultra Low Power*) firmy STMicroelectronics. Wykorzystanie układu o nazwie kodowej STM32L476 jest kompromisem między wydajnością i niskim poborem prądu w badanym zestawieniu. Podsumowująca kolumna poboru prądu względem czasu cyklu (Tab. 1) prowadzi do wniosku, iż w zastosowaniu energooszczędnym układ tego typu dobrze sprawdzi się w roli elementu wykonawczego. Mniejsze zapotrzebowanie na energię można będzie osiągnąć przede wszystkim w systemach pracujących z długim okresem cyklu oraz wzbudzanych do działania zewnętrznymi zdarzeniami. W zastosowaniach, w których można spotkać tego rodzaju pracę oprogramowania sterującego zasilanie jest często zapewniane przez ogniwa bateryjne lub akumulatorowe, niekiedy wspomagane przy pomocy ogniw fotowoltaicznych. Proponowane rozwiązanie może więc usprawnić funkcjonowanie rozwiązań IoT, a szczególnie część związaną z rozproszonymi urządzeniami sterującymi i monitorującymi.

Zaproponowana architektura niskoenergetycznego sterownika logicznego uwzględnia korzyści płynące ze standaryzacji oprogramowania. Dzięki wykorzystaniu maszyny wirtualnej możliwe jest zastosowanie pojedynczego zestawu narzędzi (kompilatora) obsługującego różne mikrokontrolery. Ułatwia to utrzymanie jednolitej struktury oprogramowania i dystrybucji w niejednorodnym pod względem sprzętowym systemie rozproszonym. Zastosowaną w projekcie maszynę wirtualną można umieścić na dowolnej platformie, od zaprezentowanych mikrokontrolerów przez klasyczne systemy operacyjne, jak również rozwiązania chmurowe.

Podziękowania

Projekt finansowany w ramach programu Ministra Edukacji i Nauki pod nazwą „Regionalna Inicjatywa Doskonałości" w latach 2019–2023 nr projektu 027/RID/2018/19 kwota finansowania 11 999 900 zł.

Bibliografia

1. Hasan M., *State of IoT 2022: Number of connected IoT devices growing 18% to 14.4 billion globally.* https://iot-analytics.com/number-connected-iot-devices.
2. Beremiz Integrated Development Environment, www.beremiz.org.
3. Tisserant E., Bessard L., Sousa M., *An Open Source IEC 61131-3 Integrated Development Environment.* [In:] IEEE International Conference on Industrial Informatics, 2007, 183–187, DOI: 10.1109/INDIN.2007.4384753.
4. GEB Automation. GEB Automation IDE Guide. www.gebautomation.org.
5. Cavalieri S., Puglisi G., Scroppo M.S., Galvagno L., *Moving IEC 61131-3 applications to a computing frame-work based on CLR virtual machine.* [In:] Proceedings of the IEEE 21st International Conference on Emerging Technolo-gies and Factory Automation, Berlin, Germany, 6–9 September 2016; 1–8.
6. Lee Y., Jeong J., Son Y., *Design and implementation of the secure compiler and virtual machine for developing se-cure IoT services.* "Future Generation Computer Systems", Vol. 76, 2017, 350–357, DOI: 10.1016/j.future.2016.03.014.
7. Rockwell Automation. ISaGRAFWorkbench. www.isagraf.com.
8. COPA-DATA France. STRATON. www.straton-plc.com.
9. Zhang M., Lu Y., Xia T., *The design and implementation of virtual machine system in embedded SoftPLC system.* [In:] Proceedings of the International Conference on Computer Science and Applications, Trento, Italy, 6–8 June 2013; 775–778, DOI: 10.1109/CSA.2013.185.
10. 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.

11. Trybus B., *Development and Implementation of IEC 61131-3 Virtual Machine*. "Theoretical and Applied Informatics" Vol. 23, No. 1, 2011, 21–35.
12. Hubacz M., Sadolewski J., Trybus B., Obsługa typów danych normy PN-EN 61131-3 w architekturze ARM z ograniczeniami dostępu do pamięci. „Pomiary Automatyka Robotyka", R. 26, Nr 1, 2022, 23–31, DOI: 10.14313/PAR_243/23..
13. Sadolewski J., Trybus B., *Compiler and virtual machine of a multiplatform control environment*. "Bulletin of the Polish Academy of Sciences. Technical Sciences", Vol. 70, No. 2, 2022, DOI: 10.24425/bpasts.2022.140554
14. IEC 611313:2013; *Programmable Controllers—Part 3: Programming Languages*. European Committee for Electrotechnical Standardization: Brussels, Belgium, 2013.
15. Wang K.C., *Embedded Real-Time Operating Systems. Embedded and Real-Time Operating Systems*; Springer: Cham, Germany, 2017, 401–475, ISBN 978-3-319-51517-5.
16. Nota katalogowa STM32L476. www.st.com/resource/en/datasheet/stm32l476rg.pdf.
17. Wu H., Chen C., Weng K., *An Energy-Efficient Strategy for Microcontrollers*. "Applied Sciences", Vol. 11, No. 6, 2021, DOI: 10.3390/app11062581.

Architecture of Low Energy Universal Programmable Controller

Abstract: The article presents the concept of building a low-energy programmable controller architecture developed in accord-ance with the IEC 61131-3 standard. The universality concerns the freedom of choice of the central unit and the possibility of easy replacement of the control algorithm. One of the three methods of software development and distribution is selected for the proposed solution. As part of the work, a prototype controller was prepared based on evaluation elements, intended to work in a distributed environment. The article also presents a comparison of the energy efficiency of selected STM32 systems from several different series.

Keywords: PLC, Internet of Things, Low Power, IEC 61131-3

mgr inż. Marcin Hubacz
m.hubacz@prz.edu.pl
ORCID: 0000-0002-2748-11454

W 2019 r. ukończył studia na Wydziale Elektrotechniki i Informatyki Politechniki Rzeszowskiej – kierunek Automatyka i Robotyka oraz Informatyka. Obecnie Asystent w Katedrze Informatyki i Automatyki Politechniki Rzeszowskiej. Jego główne zainteresowania dotyczą robotyki, elektroniki, systemów wbudowanych oraz druku 3D.

dr inż. Bartosz Pawłowicz
barpaw@prz.edu.pl
ORCID: 0000-0001-9469-2754

Adiunkt w Katedrze Systemów Elektronicznych i Telekomunikacyjnych Wydziału Elektrotechniki i Informatyki Politechniki Rzeszowskiej. Doktorat w dyscyplinie telekomunikacja uzyskał na Wydziale Elektrotechniki, Automatyki, Informatyki i Elektroniki AGH w Krakowie w 2012 r. Jego główne badania dotyczą systemów identyfikacji bezstykowej RFID i ich zastosowań.

dr inż. Bartosz Trybus
btrybus@kia.prz.edu.pl
ORCID: 0000-0002-4588-3973

Adiunkt w Katedrze Informatyki i Automatyki Politechniki Rzeszowskiej. Ukończył studia na Wydziale Elektrycznym, Automatyki, Informatyki i Elektroniki AGH w Krakowie. Doktorat z informatyki uzyskał w 2004 r. Jego główne badania dotyczą systemów czasu rzeczywistego i środowisk wykonawczych oprogramowania sterującego.