ZESZYTY NAUKOWE POLITECHNIKI RZESZOWSKIEJ 292, Elektrotechnika 34
RUTJEE, z. 34 (2/2015), kwiecień-czerwiec 2015, s. 311-322

Mateusz WIETECHA[1]
Bartosz TRYBUS[2]

STEROWNIK PLC NA PLATFORMIE RASPBERRY PI PROGRAMOWANY W ŚRODOWISKU CPDEV

W pracy przedstawiono sposób wykorzystania popularnej platformy sprzętowej Raspberry Pi do stworzenia niedrogiego sterownika PLC. Tworzenie programów sterujących odbywa się za pomocą środowiska programistycznego CPDev, opracowanego w Katedrze Informatyki i Automatyki Politechniki Rzeszowskiej. Zaprezentowano sposób implementacji na Raspberry Pi maszyny wirtualnej CPDev, tworzącej środowisko wykonawcze dla oprogramowania sterującego, oraz obsługę sygnałów wejściowych i wyjściowych z wykorzystaniem modułu sprzętowego PiFace. Jako przykład aplikacji został zaprezentowany system sterowania tzw. inteligentnym domem, obejmujący sterowanie roletami okiennymi, oświetleniem i wentylacją oraz zamkiem drzwi. Na potrzeby badań zbudowano testowe stanowisko laboratoryjne z makietą domu jednorodzinnego. Algorytm sterowania został przygotowany z wykorzystaniem języka LD normy PN/EN 61131-3 [11]. Prototyp sterownika poddano trzem testom. Pierwszy test sprawdzał, czy logika sterująca działa zgodnie z wymaganiami i przy akceptowalnym czasie cyklu. Drugi test dotyczył obciążenia CPU podczas pracy sterownika PLC, zaś trzeci pokazywał zmianę temperatury urządzenia podczas pracy algorytmu sterowania. Wyniki badań potwierdzają możliwość zastosowania Raspberry Pi oraz środowiska CPDev do sterowania instalacją niewielkiego budynku, np. domu jednorodzinnego lub niewielkiego biura. Dzięki zastosowaniu środowiska CPDev istnieje możliwość stosunkowo prostej rekonfiguracji algorytmów sterowania, w przypadku instalacji innych urządzeń w domu. Oprócz realizacji sterowania, prezentowany system posiada wystarczające zasoby, aby wykonywać współbieżnie inne zadania, na przykład obsługę zdalnego dostępu przez WWW.

Słowa kluczowe: sterownik PLC, Raspberry Pi, PN/EN 61131-3, CPDev, inteligentny dom

[1] Autor do korespondencji: Mateusz Wietecha, Politechnika Rzeszowska, Warzyce 457, 38-200 Jasło, tel. 507 153 363, mateuszwietecha@gmail.com
[2] Bartosz Trybus, Politechnika Rzeszowska, Katedra Informatyki i Automatyki, al. Powst. Warszawy 12, 35-959 Rzeszów, tel. 17 8651685, btrybus@prz.edu.pl

1. Wprowadzenie

Rynek automatyki przemysłowej ciągle zaskakuje nowymi rozwiązaniami z zakresu systemów sterowania. Najwięksi producenci branży jak SIEMENS czy Allen-Bradley stale wypuszczają kolejne modele programowalnych sterowników PLC *(Programmable Logic Controller*) [14]. Mniejsze firmy również mają po kilka flagowych produktów sprzedawanych na całym świecie [1]. Przy tak dużej liczbie dostępnych zasobów trudno wybrać ten najbardziej pasujący. Jednym z kryteriów przy wyborze sterownika PLC jest oczywiście jego planowane zastosowanie, które także ma wpływ na cenę jaką jesteśmy w stanie przeznaczyć w ramach projektu na sam sterownik.

W niniejszej pracy przyjęte zostało, że obszar zastosowania wiąże się z zamiarem przystosowania niewielkiego budynku (np. własnego domu jednorodzinnego lub biura) do wdrożenia instalacji inteligentnego sterowania urządzeniami elektrycznymi. Przy projekcie tego typu istotną rolę odgrywa budżet jakim dysponujemy. Obecnie rynek takich usług zaskakuje stosunkowo wysokimi cenami, które wahają się między 150pln a 300pln za metr powierzchni użytkowej budynku, pomijając już rozbudowane funkcjonalności, które mogą tą kwotę znacznie zwiększyć [12]. Rozważając sprzęt do instalacji takich systemów należy przeglądnąć oferty centralnych sterowników, których ceny mogą sięgnąć nawet kilku tysięcy złotych. Z tego właśnie względu uzasadnione było opracowanie własnego, niedrogiego sterownika PLC, realizującego sterowanie m. in. roletami w oknach, oświetleniem, wentylacją czy zamkiem w drzwiach wejściowych.

Niesłabnąca od kilku lat popularność minikomputera Raspberry Pi [5] oraz jego niska cena umotywowała podjęcie próby wykorzystania tej platformy do stworzenia funkcjonalnego systemu sterowania inteligentnym domem. Do utworzenia oprogramowania sterującego wybrano środowisko inżynierskie CPDev [3], głównie ze względu na jego niezależność sprzętową oraz możliwość zastosowania języków normy PN/EN 61131-3. Logika sterująca opisana językiem LD może być łatwo modyfikowana w zależności od potrzeb, także wynikłych w przyszłości.

Praca jest podzielona następująco. W punkcie 2 przedstawiono podstawową platformę sprzętowo-programową, na której zbudowano sterownik PLC, tj. miniaturowy komputer Raspberry Pi oraz środowisko inżynierskie CPDev służące do tworzenia i uruchamiania programów sterowania. W kolejnym punkcie omówiono sposób realizacji i strukturę oprogramowania sterownika, w tym obsługę wejść i wyjść oraz interakcję z programem sterowania. Dalej (p. 4.1) zaprezentowano prototyp sterownika wraz z testowym stanowiskiem laboratoryjnym dla sterowania inteligentnym domem, programem sterującym w języku LD oraz aplikacją internetową do zdalnego monitorowania pracy sterownika i stanu urządzeń w domu. W kolejnych punktach przedstawiono sposób przeprowadza-

nia oraz wyniki testów prototypu, tj. testu logiki sterującej (p. 4.2), testu wykorzystania procesora (4.3) oraz testu temperatury układu (4.4). W podsumowaniu zawarto końcowe wnioski oraz podano obszary dalszych prac i możliwości rozbudowy.

2. Platforma sprzętowo-programowa

Zaproponowany w niniejszej pracy sterownik PLC opracowano w oparciu o niedrogie i dostępne rozwiązania sprzętowe i programowe. Platformą sprzętową jest w tym wypadku miniaturowy komputer Raspberry Pi, zaś platformę programową stanowi pakiet CPDev. Jego możliwości wydają się być porównywalne w stosunku do istniejących, lecz droższych rozwiązań dostępnych na rynku. Przykładem może być sterownik Sterbox WPTC48T, posiadający podobną liczbę wejść/wyjść i umożliwiający budowę kompletnego układu sterowania [17].

2.1. Raspberry Pi

Raspberry Pi stworzył David Braben z myślą o tanim komputerze służącym początkowo dzieciom do nauki. Szybko jednak urządzenie stało się pożądane przez rzesze użytkowników zwykłych komputerów PC [2]. Mimo, że podobne rozwiązania były dostępne już wcześniej, Raspberry jest stosunkowo tani oraz działa w oparciu o typowe systemy operacyjne, w szczególności system Linux (w odmianie o nazwie Raspbian), dzięki czemu jego programowanie nie różni się znacząco od programowania typowych komputerów osobistych. Miniaturowe rozmiary sprawiają, że może być stosowany w systemach wbudowanych (*embedded*).

Urządzenie bazuje na 32-bitowym dwurdzeniowym procesorze ARM11 typu RISC (*Reduced Instruction Set Computing*), używanym także m.in. w telefonach komórkowych. Jednostka centralna taktowana jest zegarem 700MHz. Oprócz niej w urządzeniu znajduje się 512MB pamięci RAM, z której cześć wykorzystywana jest przez układ graficzny z obsługą technologii OpenGL ES 2.0 [6]. Pamięć zewnętrzną uzyskuje się poprzez zainstalowanie karty SD. Dodatkowo Raspberry Pi posiada 2 lub 4 porty USB (w zależności od modelu), 26 pinów szyny GPIO oraz port HDMI wyświetlający obraz w jakości FullHD. Nowsza wersja Rasperry Pi, oznaczona jako model B+ posiada 40 pinów GPIO i kosztuje około 140zł [16]. Obecnie wprowadzana jest do sprzedaży wersja Rasperry Pi 2, której moc obliczeniowa została znacznie zwiększona dzięki zastosowaniu 4-rdzeniowego procesora 900MHz ARM Cortex-A7 oraz zwiększeniu dostępnej pamięci RAM do 1GB. Dodatkowo będzie ona mogła działać pod kontrolą systemu operacyjnego Windows 10 [15].

W omawianym rozwiązaniu zastosowano dodatkowy moduł rozszerzający o nazwie PiFace Digital (w cenie ok. 120pln), zawierającemu m.in. przekaźniki, mikroprzełączniki oraz śrubowe zaciski, dzięki któremu możliwe jest bezpo-

średnie sterowanie pracą różnych urządzeń, w tym źródeł światła, silników rolet okiennych itp. Urządzenie wraz z modułem PiFace obsługuje 8 wejść oraz 8 wyjść cyfrowych, które wystarczą do budowy prostego sterownika PLC.

2.2. Środowisko programistyczne CPDev

Środowisko programistyczne CPDev (Control Program Developer) [9] jest rozwijane na Politechnice Rzeszowskiej i służy do programowania sterowników PLC i niewielkich systemów rozproszonych, tzw. mini-DCS (*Distributed Control Systems*) [8][13]. Tworzenie oprogramowania sterującego odbywa się za pomocą języków normy PN-EN 61131-3 [11], tj. ST (*Structured Text*), LD (*Ladder Diagram*), IL (*Structured Text*), FBD (*Function Block Diagram*) oraz SFC (*Sequential Function Chart*). LD, FBD I SFC to języki graficzne, pozostałe zaś są tekstowe. Środowisko wykonawcze CPDev zostało zrealizowane w oparciu o wieloplatformową maszynę wirtualną [18], dzięki czemu programy sterujące mogą być uruchamiane na platformach sprzętowych wyposażonych w procesory ARM, AVR oraz x86 [4], czy w układach FPGA [9]. Takie rozwiązanie umożliwiło również implementację maszyny wirtualnej wraz z programem sterującym inteligentnym domem na platformie sprzętowej Raspberry Pi. W tym przypadku diagramy LD opisujące sterowanie są kompilowane do kodu wykonywalnego dla maszyny wirtualnej.

Warto wspomnieć, że nowe rozwiązania wprowadzone w ostatnim czasie do CPDev pozwalają także tworzyć graficzny panel operatorski HMI (*Human-Machine Interface*) oraz modelować problem za pomocą diagramów SysML [9].

3. Realizacja sterownika PLC na platformie Raspberry Pi

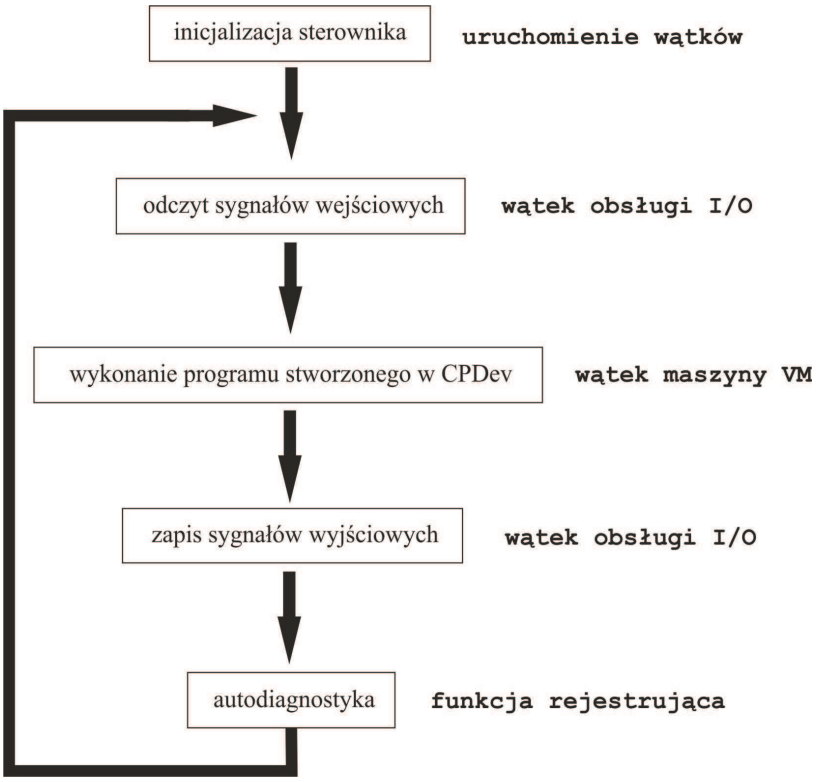
Oprogramowanie systemowe sterownika PLC zostało przygotowane w języku C. Ogólny schemat działania został przedstawiony na rysunku 1. Jak widać, oprogramowanie działa jako cykliczna maszyna stanowa. Jej sekwencyjny charakter w powiązaniu z systemem operacyjnym został jednak rozszerzony o operacje współbieżne, aby wykorzystać możliwości programowania wielozadaniowego [10]. W tym celu poszczególne części programu są realizowane przez osobne wątki.

Działanie rozpoczyna się od inicjalizacji sterownika, obejmującej przygotowanie środowiska pracy maszyny wirtualnej CPDev, w tym obszarów roboczych w pamięci RAM oraz załadowania programu sterującego w kodzie wykonywalnym maszyny wirtualnej.

Po inicjalizacji rozpoczyna się właściwa praca programu jako sterownika PLC. Pierwszą część cyklu stanowi odczyt sygnałów wejściowych. Wykorzystywane są sygnały z modułu PiFace, zaś jego obsługa programowa odbywa się za pomocą funkcji z biblioteki *wiringPi* [7].

Najważniejszym etapem pracy sterownika jest wykonanie programu sterującego przez maszynę wirtualną CPDev. W każdym cyklu maszyna przetwarza sygnały wejściowe na odpowiednie wartości zmiennych wyjściowych, zgodnie z logiką sterującą określoną w środowisku CPDev. Implementacja maszyny wirtualnej w systemie operacyjnym Raspbian wymagała dostosowania funkcji maszyny wirtualnej zależnych od platformy. Skorzystano tutaj z doświadczeń z implementacji tej maszyny w systemie operacyjnym QNX [4], bowiem oba systemy są w znacznej mierze zgodne pod względem programistycznym.

W kolejnej fazie wyjścia modułu PiFace są ustawiane zgodnie z wartościami zmiennych programu CPDev. Warto zauważyć (Rys.1), że odczyt wejść i zapis wyjść są realizowane przez jeden wspólny wątek. W innym wątku działa maszyna wirtualna, dzięki czemu możliwe jest rozdzielenie sterowania od obsługi I/O. Oprócz tego sterownik wyposażono w funkcję autodiagnostyki, czuwającą nad całością i rejestrującą pracę sterownika.



Rys. 1. Schemat ogólny sterownika PLC na platformie Raspberry Pi

Fig. 1. General PLC schema on Raspberry Pi platform

4. Testy prototypu

Opracowany sterownik przeznaczony do ciągłej pracy jako centralny moduł inteligentnego domu został poddany trzem zasadniczym badaniom. Pierwsze z nich dotyczyło sprawdzenia logiki sterującej. Kolejne sprawdzały obciążenie CPU podczas pracy oraz temperaturę urządzenia. Testy przeprowadzano na specjalnie skonstruowanym stanowisku doświadczalnym.

4.1. Stanowisko doświadczalne

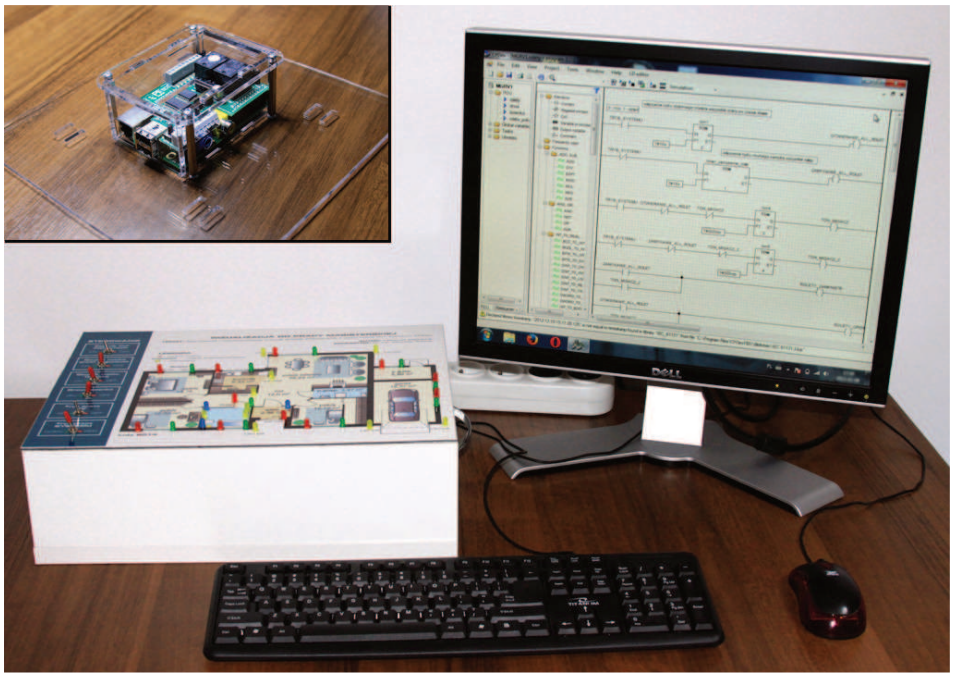
Prototyp sterownika zrealizowano w oparciu o urządzenie Raspberry Pi model B w wersji 2.0 z układem Broadcom BCM2835 (częstotliwość procesora 700MHz, 512MB pamięci SDRAM taktowanej zegarem 400MHz), wyposażone w moduł PiFace.

Do testów przygotowano stanowisko laboratoryjne umożliwiające symulowanie pracy jako sterownika inteligentnego domu, przedstawione na rys. 2. W celu zbliżenia go do warunków rzeczywistych opracowano makietę domu jednorodzinnego z wizualizacją stanu poszczególnych urządzeń (u dołu, z lewej strony rys. 2). U góry po lewej widoczny jest moduł sterownika zrealizowany w oparciu o Raspberry Pi, zaś po prawej widać środowisko programistyczne CP-Dev z programem sterowania.

Oprogramowanie sterujące opracowane w języku LD realizuje następujące funkcjonalności systemu:

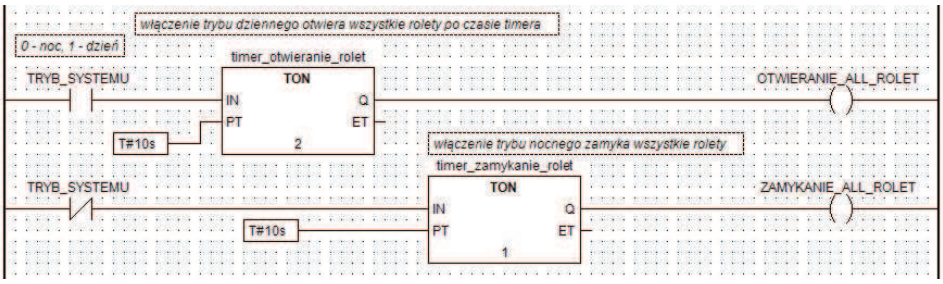
- sterowanie otwieraniem i zamykaniem rolet okiennych,
- sterowanie roletą w pokoju z telewizorem,
- sterowanie oświetleniem w łazience,
- sterowanie wentylacją,
- sterowanie zamkiem drzwi wejściowych.

Na rysunku 3 przedstawiono dla przykładu fragment sterowania roletami opracowany w języku LD. Na podstawie wartości zmiennej TRYB_SYSTEMU (dzienny lub nocny) aktywowane są bloki funkcjonalne TON (timer on), które po upływie 10 sekund (T#10s) ustawiają wyjścia Q służące do otwierania i zamykania rolet.



Rys. 2. Stanowisko laboratoryjne do badania systemu sterowania

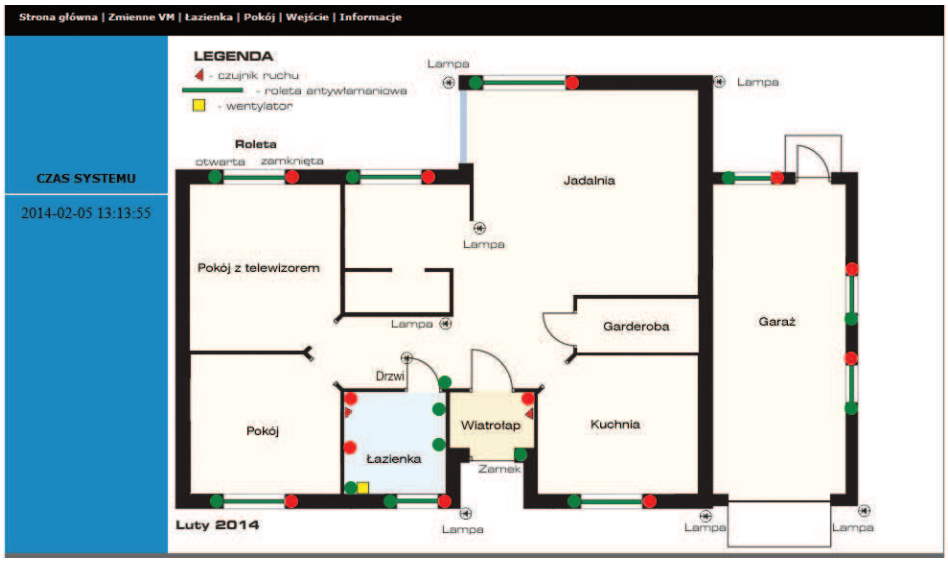
Fig. 2. Laboratory stand for control system testing



Rys. 3. Fragment programu w języku LD utworzony w edytorze CPDev

Fig. 3. Part of LD program developed in the CPDev editor

Prawidłowa praca sterownika może być także weryfikowana poprzez specjalną aplikację internetową. Monitorowanie odbywa się zdalnie, zaś wizualizacja na stronie WWW prezentuje plan pomieszczeń i stan poszczególnych urządzeń (Rys. 4). Należy podkreślić, że obsługa aplikacji internetowej jest realizowana przez Raspberry Pi równolegle z programem sterownika.



Rys. 4. Strona internetowa do zdalnego monitorowania

Fig. 4. WWW page for remote monitoring

4.2. Test logiki sterującej

Sprawdzenie poprawności działania logiki sterującej zaprogramowanej w języku LD było najdłużej trwającym testem. Odbywało się w kilku etapach. Na początku oprogramowanie sterujące było sprawdzane z wykorzystaniem symulatora CPSim wchodzącego w skład środowiska CPDev [13]. Symulator umożliwia uruchamianie i monitorowanie pracy programu na PC u bez konieczności jego ładowania do sterownika. Możliwe jest rejestrowanie wartości zmiennych programu i ich późniejsza analiza. Wartości zmiennych wejściowych ustawiane są ręcznie lub automatycznie. W trybie automatycznym symulator sam ustawia wejścia w odpowiednich chwilach czasowych na podstawie wcześniej przygotowanego przepisu.

Środowisko CPDev generuje tzw. raport kompilacji, który w powiązaniu z raportem symulacji pozwala określić zasoby wymagane przez program sterujący oraz jego złożoność. W omawianym prototypie program sterujący używa ok. 30 zmiennych globalnych oraz cztery jednostki organizacyjne oprogramowania (*Program Organization Units*), z których każda dotyczy jednego obsługiwanego urządzenia (np. rolet). Łącznie zużywanych jest ok. 5% zasobów pamięciowych maszyny wirtualnej, co sprawia, że sterowaniem można by objąć kilkukrotnie większy obiekt.

W kolejnym etapie testy przeprowadzano z użyciem prototypu sterownika, lecz bez obsługi wejść i wyjść modułu PiFace. Pozwoliło to stwierdzić, czy czas

cyklu sterownika nie jest zbyt długi, tj. czy moduł wykonawczy w odpowiednim czasie ustawia sygnały wyjściowe. Testy wykazały, że czas cyklu nie przekraczał 100ms, co jest wartością akceptowalną dla tego typu zastosowań.

Ostatnią fazą było sprawdzenie działania każdego z wejść oraz wyjść obsługiwanych przez moduł PiFace. Do tego celu wykorzystano makietę domu jednorodzinnego połączonego z wejściami i wyjściami sterownika.

4.3. Test wykorzystania zasobów CPU

Kolejne badanie obejmowało sprawdzenie użycia procesora Raspberry Pi podczas pracy jako sterownik PLC. W ramach testu uruchomiono program sterujący na 10 minut, wykonując każdą zaprogramowaną czynność oraz symulując wszystkie możliwe tryby pracy systemu sterowania. Parametry pracy procesora były w tym czasie pod ciągłym monitoringiem. Wyniki testu pokazały, że użycie procesora przez sterownik PLC utrzymywało się na poziomie 67% przy odchyleniu 0,5%. Obrazuje to rys. 5, gdzie przedstawiono listę procesów działających w systemie. Pierwszy proces na liście odpowiada za funkcjonalność sterownika, pozostałe to procesy systemowe.

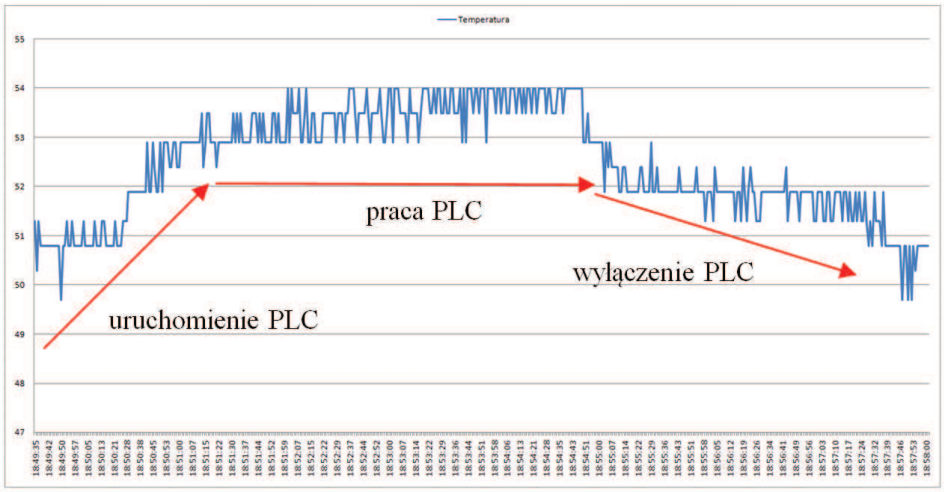
```
top - 17:27:28 up 34 min,  4 users,  load average: 2.34, 2.41, 1.86
Tasks:  74 total,   2 running,  72 sleeping,   0 stopped,   0 zombie
%Cpu(s): 76.1 us, 23.9 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:    448736 total,    84456 used,   364280 free,    11804 buffers
KiB Swap:   102396 total,        0 used,   102396 free,    41772 cached

  PID USER      PR  NI  VIRT  RES  SHR S  %CPU %MEM    TIME+  COMMAND
 2452 root      20   0 27188  788  648 R  67.1  0.2  10:18.47 mgr_main
 2490 root      20   0     0    0    0 S  31.1  0.0   0:29.84 kworker/u:2
 2478 root      20   0  4664 1360 1024 R   1.0  0.3   0:07.01 top
   37 root      20   0     0    0    0 S   0.3  0.0   0:06.81 mmcqd/0
 2463 root      20   0  9804 3252 2620 S   0.3  0.7   0:01.99 sshd
    1 root      20   0  2140  712  608 S   0.0  0.2   0:01.71 init
```

Rys. 5. Parametry procesów w sterowniku PLC

Fig. 5. Process parameters in the PLC controller

4.4. Test temperatury procesora

Kolejne badanie miało na celu sprawdzenie, czy urządzenie podczas pracy i wykonywania zaprogramowanych funkcji sterujących nie będzie się nadmiernie nagrzewało i sprosta wymaganiom ciągłej pracy, jakie stawia się przed sterownikami systemów inteligentnego domu. Do przeprowadzenia testu przygotowano skrypt, który zbierający wartości temperatury i zapisujący je co sekundę do pliku.



Rys. 6. Wykres temperatury procesora podczas pracy sterownika

Fig. 6. CPU temperature graph during PLC activity

Zebrane próbki dla testu trwającego prawie 8 minut przedstawiono na rys. 6. Początkowo był uruchomiony tylko system operacyjny i temperatura wynosiła około 51 stopni Celsjusza. Następnie został uruchomiony program sterownika PLC oraz przeprowadzono symulację wszystkich możliwych trybów pracy systemu sterowania inteligentnym domem. Podczas tej fazy temperatura wzrosła do około 54ºC. Sterownik pracował 5 minut, a następnie został wyłączony. Temperatura CPU spadła do poprzedniego poziomu około 51ºC. Na rys. 5 wyraźnie widać moment włączenia, gdy temperatura wzrosła oraz moment przejścia w stan bezczynności powodujący powrót do poprzednich wartości temperatury. Jak widać, nawet intensywna praca sterownika PLC powoduje nieznaczny wzrost temperatury układu.

5. Podsumowanie

Uzyskane wyniki pozwalają na sformułowanie kilku wniosków dotyczących obiektu badań. Raspberry Pi może posłużyć jako baza do stworzenia prostego, niedrogiego sterownika PLC do zastosowań niewielkiej skali. Urządzenie, mimo że nie zostało wyprodukowane jako sprzęt przemysłowy, sprawdzi się w roli sterownika prostej instalacji inteligentnego domu. Możliwość programowania w języku LD ułatwia adaptację do określonych potrzeb oraz późniejszą rekonfigurację, w przypadku gdyby zainstalowano kolejne urządzenia, którymi należałoby sterować. Oprócz urządzeń uwzględnionych w prototypie mogłyby to być gniazda sieciowe, odbiorniki zasilane dowolnym napięciem, brama garażowa, oświetlenie RGB itp. Wielozadaniowy system operacyjny pozwala na rów-

noległe wykonywanie dodatkowych funkcji, jak obsługę strony internetowej do zdalnego monitorowania pracy.

Korzystając z platformy Raspberry Pi podczas tworzenia instalacji inteligentnego domu należy pamiętać o jego ograniczeniach w ilości wejść/wyjść. W przypadku większej ilości sterowanych urządzeń może okazać się ona niewystarczająca. Rozwiązaniem byłoby wyposażenie urządzenia w dodatkowy interfejs, tzw. *port expander* oferujący kolejne kanały I/O. Nowe perspektywy wiążą się także z wprowadzaną do sprzedaży wersją Rasperry Pi 2 wyposażoną w szybszy procesor i większą pamięcią RAM. W przyszłości planowane jest opracowanie i uruchomienie na tej platformie graficznego panelu operatorskiego z możliwością włączania i wyłączania urządzeń oraz zmiany ustawień sterownika za pomocą panelu dotykowego.

Literatura

[1] BECKHOFF Automation GmbH & Co. KG, Embedded PC CX: http://www.beckhoff.com/english.asp?embedded_pc/cx.htm?id=15987759973374

[2] Chrobot M.: Pecet wielkości pendrive. Musisz to zobaczyć!, http://spokogadzet.komputerswiat.pl/pecet-wielkosci-pendrive-musisz-to-zobaczyc.

[3] CPDev, strona internetowa: http://www.cpdev.kia.prz.edu.pl/

[4] Dulęba Ł., Kotula W., Trybus B.: Implementacja maszyny wirtualnej CPDev w systemach operacyjnych QNX Neutrino i Windows CE, [w:] Trybus L., Samolej S.: Projektowanie, analiza i implementacja systemów czasu rzeczywistego, ISBN 878-83-206-1822-8, WKŁ. Warszawa 2011, s. 207-216.

[5] Forum Raspberry Pi, Fundacja Raspberry-Pi: http://forum.r-pi.pl/fundacji/fundacja-raspberry-t7.html.

[6] Forum Raspberry Pi, Specyfikacja Techniczna Raspberry-Pi Model B: http://forum.r-pi.pl/specyfikacja-mini-komputera/specyfikacja-techniczna-raspberry-model-t9.html.

[7] Gordon Project, Download and Install: https://projects.drogon.net/raspberry-pi/wiringpi/download-and-install/.

[8] 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: Szewczyk R., Zieliński C., Kaliczyńska M. (Eds.): Advances in Intelligent Systems and Computing vol. 267, Recent Advances in Automation, Robotics and Measuring Techniques, Springer-Verlag Berlin Heidelberg 2014, pp. 81-90.

[9] Jamro M., Rzońca D., Sadolewski J., Stec A., Świder Z., Trybus B., Trybus L.: Środowisko inżynierskie Control Program Developer obecnie, Zeszyty Naukowe Politechniki Rzeszowskiej, Elektrotechnika 33, 2013, pp. 117-132.

[10] Juźwiak P.: Programowanie współbieżne 4. Wątki Pthread w Linuxie, Wydział Elektroniki i Technik Informacyjnych, Politechnika Warszawska.

[11] PN-EN 61131-3:2013-10, Sterowniki programowalne -- Część 3: Języki programowania.

[12] Riley M.: Inteligentny dom, Helion, 2013.
[13] Rzońca D., Sadolewski J., Stec A., Świder Z., Trybus B., Trybus L.: Open Environment for programming small controllers according to IEC 61131-3 standard, Scalable Computing: Practice and Experience, 2009.
[14] SIEMENS, Systemy automatyki SIMATIC, Jednostki centralne S7-1500: http://www.automatyka.siemens.pl/solutionandproducts_ia/12034.htm
[15] Sklep Botland, Raspberry Pi 2 model B 1GB RAM: http://botland.com.pl/moduly-i-zestawy-raspberry-pi-2/3181-raspberry-pi-2-model-b-1gb-ram.html
[16] Sklep Botland, Raspberry Pi Model B+ 512MB RAM: http://botland.com.pl/moduly-i-zestawy-raspberry-pi-2/2543-raspberry-pi-model-bplus-512mb-ram.html
[17] Sterbox, Sterownik internetowy PLC, Sterbox WPTC48T, http://www.sterbox.eu/index.php/sklep/product/view/1/108
[18] Trybus B. „Development and Implementation of IEC 61131-3 Virtual Machine", Theoretical and Applied Informatics. Volume 23, Issue 1, 2011, Pages 21–35.

PLC CONTROLLER BASED ON RASPBERRY PI AND PROGRAMMABLE WITH CPDEV ENVIRONMENT

S u m m a r y

The paper presents a low-cost PLC controller prototype based on the popular Raspberry Pi hardware platform. CPDev programming environment, developed at Rzeszow University of Technology, is used to create control programs. CPDev virtual machine has been implemented in Raspberry Pi to make a runtime environment for control software. Input and output signal handling is achieved via Piface hardware module add-on. A smart home is presented as an example application, controlling window shades, lighting, ventilation and door lock. For testing purposes, a lab installation has been constructed with model of a family home. Control algorithms have been prepared in LD language which complies with IEC 61131-3 standard [11]. Three tests have been performed on this system. The first test verifies whether the control logic meets the requirements and works with acceptable cycle time. The second test involves CPU resources taken by the PLC during work. The third test shows how the temperature changes during operation of the control algorithm. The test results confirm that Raspberry Pi and CPDev environment can be used to control a simple installation of smart home, e.g. single-family home or a small office building. When new devices are installed at home, straightforward reconfiguration is possible by using the CPDev environment. Apart from running control algorithms, the presented system is powerful enough to execute concurrently other tasks, such as handling of remote access via WWW.

Keywords: PLC, Raspberry Pi, PN/EN 61131-3, CPDev, smart home

DOI: 10.7862/re.2015.23

Tekst złożono w redakcji: luty 2015
Przyjęto do druku: kwiecień 2015