Politechnika Gdańska

Wydział Elektrotechniki i Automatyki

Gdańsk 2018

doi: 10.32016/1.61.08

STEROWANIE I MONITOROWANIE SIECI WYDZIELONEJ Z ODNAWIALNYMI ŹRÓDŁAMI ENERGII

Piotr KOŁODZIEJEK[1], Robert KACZMAREK[2], Elżbieta BOGALECKA[3]

1. Politechnika Gdańska, Wydział Elektrotechniki i Automatyki
   tel.: 58 348 60 76 e-mail: piokolod@pg.edu.pl
2. Starsoft
   tel.: 792 441 139 e-mail: robkaczm@gmail.com
3. Politechnika Gdańska, Wydział Elektrotechniki i Automatyki
   tel.: 58 348 29 36 e-mail: elzbieta.bogalecka@pg.edu.pl

Streszczenie: W artykule przedstawiono zagadnienia związane z opracowaniem dedykowanego oprogramowania systemu sterowania i monitorowania odnawialnymi źródłami energii w sieci wydzielonej. Do implementacji systemu wykorzystano stanowisko laboratorium Systemów Sterowania w Energetyce Odnawialnej Wydziału Elektrotechniki i Automatyki Politechniki Gdańskiej. Opracowane oprogramowanie wykorzystuje platformę Windows Presentation Foundation oraz wzorzec projektowy Mode-View-ViewModel.

Słowa kluczowe: SCADA, odnawialne źródła energii, sieć wydzielona, Windows Presentation Foundation, Model-View-ViewModel.

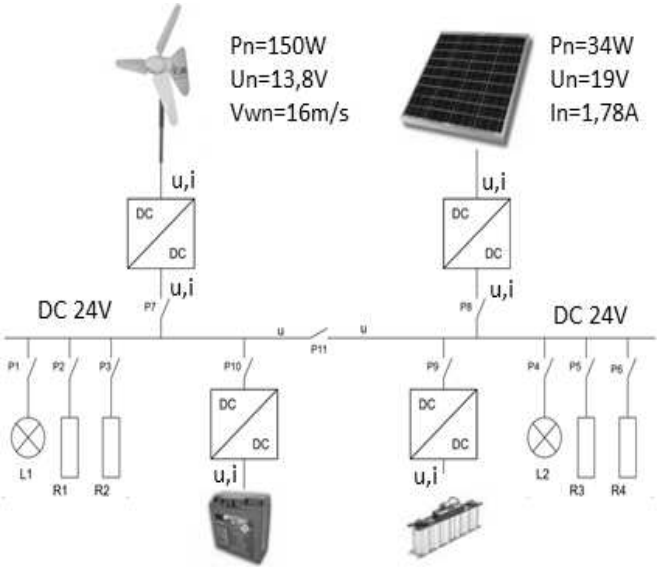
1. WSTĘP

Obecnie w energetyce zawodowej oraz wielu systemach technicznych instalowanych w ubiegłych latach występuje problem modernizacji systemu w zakresie dostosowania pracy urządzeń zaprojektowanych do korzystania z komunikacji analogowej z nowoczesnymi sterownikami, w których zakłada się wykorzystanie komunikacji w oparciu o technologię cyfrową. Od ponad 14 lat podejmowane są działania w kierunku standaryzacji protokołów komunikacyjnych w kontekście normy IEC 61850 określającej m. in. standardy komunikacyjne z wykorzystaniem protokołów MMS, GOOSE, SMV i innych opartych na stosie protokołów TCP/IP, które jednak nie zawsze są uwzględniane przez producentów układów sterowania nadrzędnego. Modernizacji wymaga także oprogramowanie w kontekście wykorzystania nowych technik i narzędzi informatycznych zapewniające nowe funkcjonalności, uniwersalność oraz możliwości rozbudowy. Nowoczesne rozwiązania wymagają zwykle opracowania nowego oprogramowania, do którego wykorzystuje się aktualne narzędzia, języki i paradygmaty programowania. W artykule przedstawiono dedykowane rozwiązanie oprogramowania pozwalające na komunikację cyfrową z wykorzystaniem połączenia bezprzewodowego z układem SH363 w języku C# zapewniające możliwości rozbudowy systemu, zdalnego udostępniania stanowiska, uniezależnienie użytkownika od platformy sprzętowej oraz systemu operacyjnego oraz możliwość rozwoju oprogramowania przez społeczność akademicką.

2. STANOWISKO LABORATORYJNE

Systemy wytwarzania energii ze źródeł odnawialnych poza energetyką zawodową znajdują także zastosowanie w mikroinstalacjach, w których wytwarzana energia wykorzystywana jest na potrzeby własne lub magazynowana. Laboratorium Systemów Sterowania w Energetyce Odnawialnej Wydziału Elektrotechniki i Automatyki Politechniki Gdańskiej dysponuje stanowiskiem do badania układów wytwarzania energii z elektrowni słonecznej i wiatrowej z możliwością magazynowania w akumulatorach żelowych, litowo-jonowych oraz w superkondensatorach. Mikroprocesorowy układ sterowania zapewnia zintegrowane sterowanie przetwornicami z wykorzystaniem dedykowanych algorytmów śledzenia optymalnego punktu pracy odpowiednio elektrowni słonecznej i wiatrowej, a także przetwornicami współpracującymi z magazynami energii stabilizującymi napięcie szyny głównej poprzez oddawanie lub pobieranie energii w zależności od generacji oraz obciążenia mikroinstalacji. Układy sterowania przetwornicami obsługiwane są przez mikrokontroler Sharc SH363. Sterownik SH363 oparty jest na procesorze sygnałowym ADSP 21363 i układzie logiki programowalnej Altera Cyclone II. Układ ponadto posiada pamięć FLASH 8 Mb, pamięć MRAM 4 Mb oraz nieulotną pamięć EEPROM o pojemności 512 kb. Komunikację z urządzeniem zewnętrznym zapewnia JTAG, RS232 oraz izolowany port USB z konwerterem CP2102. Sterownik zapewnia 76 wejść/wyjść cyfrowych, dowolnie konfigurowalnych. Rozwiązanie to pozwala na wykorzystanie urządzenia w wielu konfiguracjach. Jednym z ograniczeń jest długość przewodu komunikacyjnego, zatem odczyt i kontrola danych pomiarowych może odbywać się jedynie w pobliżu układu mikroprocesorowego.

Na rysunku 1 przedstawiono schemat stanowiska laboratoryjnego.



Rys .1. Schemat ideowy stanowiska laboratoryjnego [1]

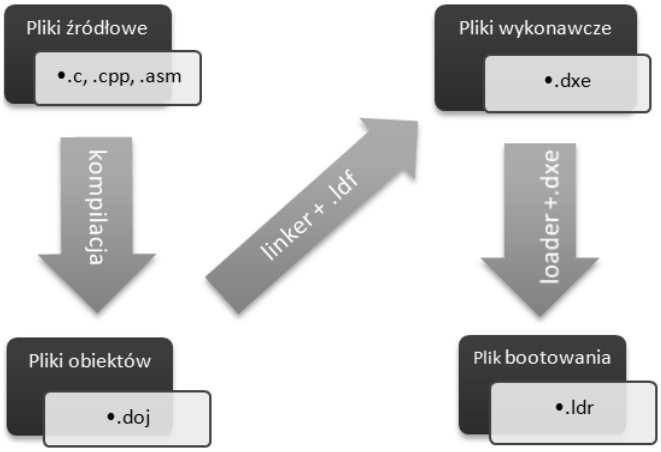
Obecnie dostępne są rozwiązania otwarte zapewniające funkcjonalności systemu nadrzędnego sterowania i monitorowania, jednakże są one ograniczone do z góry przewidzianych przez autorów oprogramowania rozwiązań sprzętowych, komunikacyjnych i wizualizacyjnych. Możliwości opracowania dedykowanego oprogramowania do zdalnego sterowania układami napędowymi opisano w [2]. Na stanowisku z OZE funkcjonalność systemu SCADA zapewniał sterownik PLC B&R Automation X20CP1484 oraz komputer nadrzędny z oprogramowaniem Proficy HMI/SCADA iFIX [3], który wykorzystywany był do sterowania i monitorowania nadrzędnego mikrosieci. Do zacisków listwy zaciskowej wyprowadzono zespół przewodów sygnałowych dostarczających informacje o prądach i napięciach poszczególnych elementów wytwórczych i magazynujących stacji. Wykorzystano powyższe rozwiązanie, ponieważ na tym etapie rozwoju systemu nie istniała możliwość bezpośredniej komunikacji pomiędzy sterownikiem SH363 a sterownikiem PLC. W artykule opisano opracowanie komunikacji, która umożliwia bezprzewodowy odczyt i zapis danych z układu SH363. Eliminuje to konieczność zastosowania komunikacji analogowej, sterownika programowalnego PLC oraz komputera nadrzędnego do komunikacji ze sterownikiem programowalnym. Na stanowisku laboratoryjnym układ procesora sygnałowego stanowi integralną część sterownika mikroprocesorowego SH363. Zastosowany model ADSP 21363 z rodziny Sharc jest procesorem architektury super harwardzkiej, która jest rozwinięciem standardowej architektury harwardzkiej [4].

3. WARSTWA STEROWANIA LOKALNEGO

Do oprogramowania sterownika wykorzystano dedykowane oprogramowanie producentów procesorów, tj. Altera Cyclone II oraz ADSP21363. W pierwszym z programów przygotowano strukturę FPGA definiując połączenia logiczne. Wykorzystano język programowania wysokopoziomowego VHDL. Program procesora sygnałowego realizuje:

- obsługę przetwornic DC/DC dedykowanych dla każdego z urządzenia generującego lub magazynującego energię,
- odczyt i modyfikację wybranych zmiennych,
- pomiar napięć i prądów dla panelu fotowoltaicznego, turbiny wiatrowej oraz zasobników energii,
- obsługę awarii.

Po skompilowaniu projektu dla procesora DSP otrzymuje się plik binarny z rozszerzeniem .ldr oraz plik mapowy. Przebieg budowy potokowej programu zaprezentowano na rysunku 2, gdzie plik .ldf jest opisem konsoidacji zawierającym komendy globalne, opis pamięci oraz komendy linkujące projekt. Plikami wynikowymi pracy konsolidatora jest plik wykonawczy .dxe (binarny)



Rys. 2. Proces budowy plików uruchomieniowych dla procesora ADSP 21363 ++

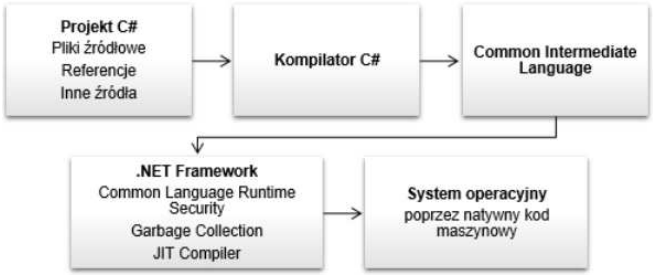
zawierający obraz pamięci oraz plik mapowy pamięci .map (ASCII).

4. WARSTWA STEROWANIA NADRZĘDNEGO

4.1. Wybór środowiska i języka programowania

Kryteria wyboru środowiska do budowy oprogramowania uwzględniały założenie wysokopoziomowego rozwiązania przy dostępności wsparcia jego twórcy oraz społeczności, przy ograniczeniu złożoności projektowania aplikacji oraz umożliwiające testowanie. Do realizacji oprogramowania wybrano język C# oraz środowisko programowania – Visual Studio, zapewniające wymagane możliwości. Wybór nowoczesnego rozwiązania zapewnia uniwersalność języka, zapewniającego możliwość migracji na inne platformy sprzętowe z wykorzystaniem narzędzia Xamarin lub Universal Windows Platform. C# jest nieskomplikowanym, wieloparadygmatowym językiem, zorientowanym głównie na programowanie obiektowe. Twórcy przewidzieli mechanizmy zarówno dla samych klas jak i poszczególnych elementów w nich zawartych, do których zaliczyć można m.in. dziedziczenie, polimorfizm i hermetyzację.

Język C# uważa się za bezpieczny, ponieważ posiada zaimplementowaną funkcję modyfikatorów dostępu, pozwalającą na ograniczenie dostępu z zewnątrz do elementów danej klasy. Na szczególną uwagę zasługuje tzw. „Garbage Collector", który zarządza pamięcią z poziomu środowiska uruchomieniowego i dba o usuwanie zbędnych zasobów po niewykorzystywanych obiektach. Program, który został napisany w języku C# jest kompilowany do tzw. Common Interface Language, czyli wspólnego języka pośredniego. Stanowi on odpowiednik asemblera dla

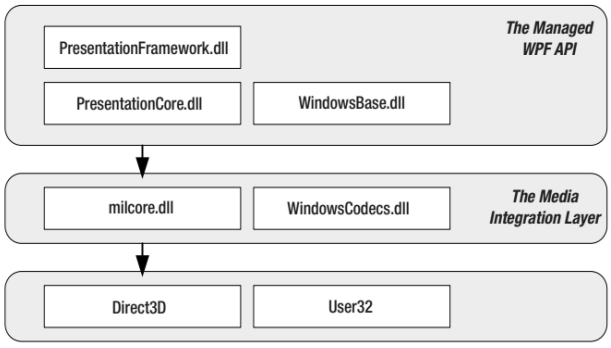


Rys. 3. Kompilacja projektu w języku programowania C#

języków wysokiego poziomu. W dalszych krokach program tłumaczony jest na kod bajtowy, który jest wykonywany za pomocą maszyny wirtualnej. Przebieg kompilacji przedstawiono na rys. 3. Konieczność instalacji platformy .NET w celu skorzystania z aplikacji okienkowej stanowi wadę programów napisanych w języku C#. Jednakże ze względu na prostotę nauki, dużą liczbę nowoczesnych funkcji np. wyrażenia Lambda czy LINQ oraz wszechstronność, język ten zyskuje popularność. Wykorzystanie mechanizmu wspierania pisania programu – IntelliSense pozwala na wygodny i szybki dostęp do nazw parametrów, metod czy pól, a system refaktoryzacji kodu proponuje jego modyfikacje w trybie ciągłym w celu wyeliminowania nadmiarowości.

4.2. Platforma Windows Presentation Foundation

Środowisko Visual Studio 2017 wspiera platformę Windows Presentation Foundation, która razem z .NET Framework umożliwia tworzenie aplikacji okienkowych dla systemu Windows. Platforma WPF wprowadziła XAML, czyli język oparty na XML zoptymalizowany do tworzenia interfejsów graficznych. Obsługa zdarzeń oraz tzw. code-behind odpowiadającego za logikę aplikacji tworzona jest w języku C#. Takie rozwiązanie znacznie ułatwia pracę nad projektem. Jednocześnie stronę wizualną można projektować niezależnie od logiki programu. Należy jedynie zapewnić odpowiednią komunikację pomiędzy obiema warstwami.
Platforma WPF wykorzystuje architekturę składającą się z kilku płaszczyzn, gdzie na szczycie znajdują się usługi wysokiego poziomu, które są zaprojektowane w języku C#.
W następnej warstwie znajduje się część odpowiedzialna za transformację obiektów .NET do komponentów Direct3D wykorzystując w tym celu dynamiczną bibliotekę milcore oraz WindowsCodecs. Przebieg transformacji na poszczególnych płaszczyznach przedstawiono na rys. 4 [5].



Rys. 4. Warstwowa architektura platformy WPF [4]

Do zalet WPF należy tzw. wiązanie danych (z ang. binding), czyli łączenie prawie każdego typu danych z polami znajdującymi się w aplikacji. Wykorzystanie takiego rozwiązania jest bardziej efektywne niż w dotychczasowym rozwiązaniu Windows Forms, jednakże wykorzystuje więcej zasobów pamięci oraz wymaga znajomości języka XAML.

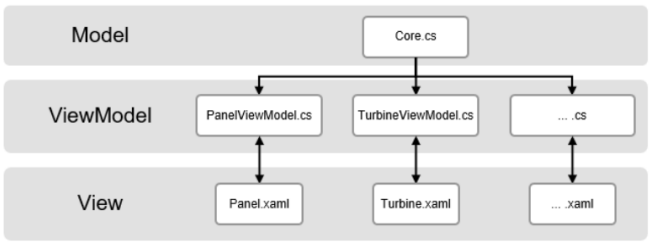
Zaprojektowana aplikacja została stworzona z wykorzystaniem technologii WPF z podziałem kodu na część interfejsu graficznego w języku XAML oraz część logiczną w języku C#.

4.3. Wzorzec projektowy Model-View-ViewModel

Wykorzystanie wzorców projektowych ma na celu napisane uporządkowanego, dobrze zoptymalizowanego oraz łatwego do refaktoryzacji kodu. Sam wzorzec wskazuje sposób budowy szkieletu aplikacji. Aktualnie istnieje wiele wzorców dedykowanych do rozwiązania różnych problemów. Wzorce projektowe podzielić można na cztery główne kategorie:

- kreacyjne – inaczej konstrukcyjne, opisują przebieg tworzenia obiektów,
- behawioralne – inaczej czynnościowe, opisują zachowania i powiązania między obiektami,
- strukturalne – opisują sposób budowania struktur powiązanych obiektów.
- architektoniczne, które z założenia opisują ogólną strukturę aplikacji, typy obiektów z jakich się składa oraz sposób w jaki obiekty wymieniają się informacjami.

Analizując powyżej opisane możliwości zdecydowano na zbudowanie aplikacji zgodnie ze wzorcem MVVM (Model View ViewModel), który umożliwia uniezależnienie kodu części logicznej programu od kodu odpowiedzialnego za wyświetlanie widoków tj. podział na View i ViewModel. Do korzyści należy zaliczyć również łatwość edycji widoków w przypadku gotowych części ViewModel [6].
Zgodnie z rysunkiem 5 aplikacja została podzielona na płaszczyzny, które przekazują między sobą informacje. Warstwa Model jest odpowiedzialna za komunikację ze sterownikiem, przetwarzanie i magazynowanie danych oraz obsługę modułu odpowiedzialnego za zapis danych do pliku. Klasa Core jest inicjalizowana jedynie jeden raz, jednakże dane zapisywane są w polach statycznych, co pozwoliło na dostęp z innych klas warstw ViewModel. Druga warstwa odpowiedzialna jest za obsługę wszelkich akcji, które mają miejsce w widoku. Przekazywanie danych między publicznymi właściwościami w klasie ViewModel a obiektami w widoku odbywa się z wykorzystaniem interfejsu INotifyPropertyChanged. Pliki widoków zostały opracowane w języku opisu interfejsu użytkownika – XAML. Ich rolą jest wyświetlanie interfejsu oraz opracowanych danych pochodzących ze sterownika mikroprocesorowego.



Rys. 5. Struktura zaprojektowanego programu zgodnie ze wzorcem MVVM

Wszystkie widoki połączono za pomocą jednej ramki głównego widoku, którą opracowano w pliku MainWindow.xaml oraz MainWindowViewModel.cs.

Podczas uruchamiania aplikacji tworzona jest instancja klasy głównego okna, która posiada w konstruktorze domyślnym inicjalizację klasy Panels wywołującą poszczególne widoki. Graficzny interfejs użytkownika zaprojektowano z wykorzystaniem biblioteki gotowych elementów Material Design in XAML Toolkit udostępnionej na licencji MIT, której autorem jest James Willock. „Material Design" jest konceptem stworzonym przez firmę Google, który określa styl budowania graficznego interfejsu użytkownika.

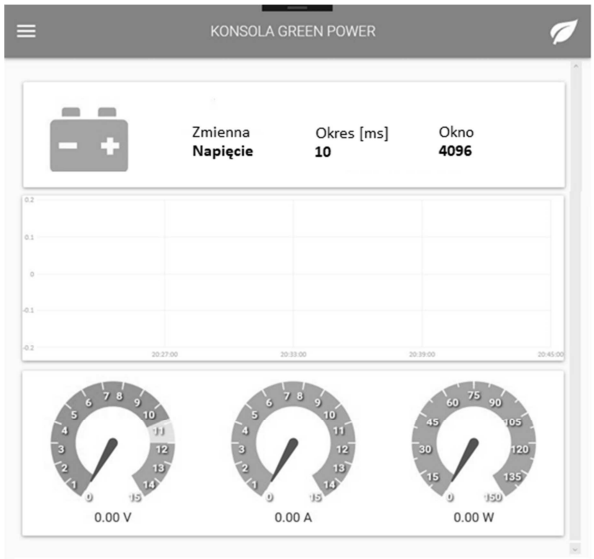
Rys. 6. Część okna interfejsu użytkownika do sterowania i monitorowania elektrowni wiatrowej

Zasady te wprowadzono w celu intuicyjnej obsługi interfejsu użytkownika. Aplikacja podzielona została na okna aplikacji do nawiązania połączenia z układem mikroprocesorowym, uruchamiania i zatrzymywania układu sterowania mikrosiecią oraz do rejestracji pomiarów.

Na rysunkach 6 i 7 przedstawiono część aplikacji do sterowania i monitorowania elektrowni wiatrowej, i magazynów energii. Analogicznie opracowano interfejs dla elektrowni słonecznej. Po nawiązaniu połączenia z układem mikroprocesorowym możliwy jest podgląd zmian wartości chwilowych napięcia, prądu oraz mocy w czasie rzeczywistym.

5. WNIOSKI KOŃCOWE

W artykule przedstawiono opracowane oprogramowanie do sterowania mikrosiecią z OZE. Omówiono kryteria doboru narzędzi informatycznych oraz zagadnienia projektowania dedykowanego wielowarstwowego systemu typu SCADA z wykorzystaniem narzędzi i bibliotek wspomagania projektowania, WPF API oraz wzorca projektowego MVVM. W praktyce migracja oprogramowania do wizualizacji i sterowania nadrzędnego zapewniła możliwość tworzenia dedykowanych interfejsów użytkownika dla układów sterowania opartych na mikrokontrolerze SH363, umożliwia zdalne sterowanie, uniezależnienie od systemu operacyjnego oraz zapewnia szeroki dostęp społeczności akademickiej w celu przyszłej rozbudowy oprogramowania.



Rys. 7. Część okna aplikacji do zarządzania magazynami energii oraz do rejestracji pomiarów

6. BIBLIOGRAFIA

1. MMB Drives – DTR stanowiska "Green Power".
2. Kołodziejek P., Bogalecka E., Guziński J. „Sterowanie układem napędowym w sieci Internet", 3 Krajowa Konferencja Modelowanie i Symulacja MiS-3, Kościelisko 2006.
3. Mielcarek F., Monitoring i wizualizacja pracy układu zasilania z odnawialnymi źródłami energii, Praca dyplomowa inżynierska, Politechnika Gdańska, Gdańsk 2011.
4. Analog Devices, Sharc Processor ADSP-21362, Karta katalogowa.
5. MacDonald M., Pro WPF in C# 2010, Apress, Nowy Jork 2010.
6. Troelsen A., Japikse P., C# 6.0 and .NET 4.6 Framework, Aspress, Nowy Jork 2015.

OFF-GRID MICROGRID SCADA SYSTEM WITH RENEABLE ENERGY SOURCES

The article deals with issues related to the development of dedicated software for the multi-layer supervisory control and monitoring system of renewable energy sources operating in autonomous microgrid with solar and wind power plants, Li-Ion and supercapacitor storage systems with load model. Migration issues of the control panel from C++ application designed in the C++ Builder rapid application development environment to the modern C# with Visual Studio is presented. The criteria for selection of programming tools, languages, paradigms for the design of a dedicated multi-layer SCADA system are discussed including design support libraries, Windows Presentation Foundation API and the selected Model-View-ViewModel design pattern. In practice, the migration of software for visualization and supervisory control ensured the possibility of creating dedicated user interfaces for control systems based on the SH363 microcontroller, enables remote control, independence from the operating system and provides broad access to the academic community for future software development.

Keywords: SCADA, renewable energy microgrid, Windows Presentation Foundation, Model-View-View-Model.