MARIAN HYLA

Uniwersalny serwer do monitorowania urządzeń przemysłowych za pomocą przeglądarki internetowej

W artykule przedstawiono realizację uniwersalnego serwera pośredniczącego w procesie monitorowania urządzeń przemysłowych wykorzystujących protokół Modbus TCP. Aplikacja udostępnia wybrane informacje za pomocą protokołu HTTP akceptowanego przez przeglądarki internetowe. Zaprezentowano możliwość konfiguracji oprogramowania oraz przykładowe strony generowane przez serwer. Przedstawiono wyniki testów wydajnościowych połączenia Modbus TCP.

Słowa kluczowe: *monitorowanie, zdalny dostęp, Modbus TCP, HTTP, WWW*

1. WSTĘP

Nowoczesne technologie informatyczne dają szerokie możliwości transmisji danych z wykorzystaniem komputerowych sieci lokalnych, jak i globalnej sieci Internet. W dobie powszechnego dostępu do Internetu coraz więcej urządzeń przemysłowych wyposaża się w interfejsy sieciowe, wykorzystywane do zdalnego sterowania i monitorowania stanu pracy. Praktycznie przestaje mieć znaczenie zarówno lokalizacja monitorowanych urządzeń, jak i lokalizacja użytkownika.

Jednym z najczęściej implementowanych protokołów urządzeń przemysłowych z możliwością łączności przez sieć Ethernet/Internet jest protokół Modbus TCP [13]. Od strony użytkownika powszechnym i ogólnie dostępnym narzędziem dostępu do sieci są przeglądarki internetowe wykorzystujące protokół HTTP (*Hypertext Transfer Protocol*) [11].

Zarówno Modbus TCP, jak i HTTP są protokołami warstwy aplikacji stosu protokołów TCP/IP i do transportu wykorzystują protokół TCP (*Transmission Control Protocol*) oraz warstwy niższe [3–7, 11, 13, 18, 19]. Aby umożliwić prezentację danych z urządzeń Modbus TCP za pomocą przeglądarki internetowej, konieczne jest jednak zastosowanie odpowiedniego narzędzia (aplikacji) umożliwiającego konwersję danych pomiędzy obydwoma protokołami.

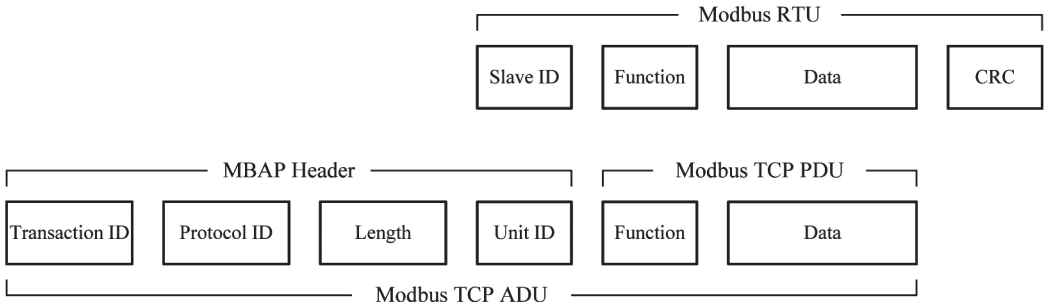
2. PROTOKÓŁ MODBUS TCP

Protokół Modbus [12, 14] został opracowany w 1979 r. przez firmę Modicon, przejętą w połowie lat 90. ubiegłego wieku przez Schneider Electric. Pomimo gwałtownego rozwoju nowych technik komunikacji i opracowania nowych protokołów transmisyjnych jest on w dalszym ciągu jednym z najpopularniejszych protokołów przemysłowych i z powodzeniem stosowany jest do lokalnej komunikacji z urządzeniami wyposażonymi w szeregowe asynchroniczne interfejsy transmisji danych w modelu komunikacji typu Master-Slave.

Powszechny dostęp do Internetu oraz możliwość implementacji stosu protokołów TCP/IP [3–6, 18, 19] w urządzeniach przemysłowych przyczyniły się do opracowania nowej wersji protokołu Modbus dla sieci pakietowych nazwanej Modbus TCP lub Modbus TCP/IP [13]. Protokół ten bazuje na odmianie protokołu Modbus RTU i jest jego implementacją dla sieci IP. Porównanie struktury ramki Modbus RTU oraz Modbus TCP przedstawiono na rysunku 1.

Ramka protokołu Modbus TCP (*Application Data Unit*) składa się z nagłówka (*Modbus Application Header*) oraz właściwej części danych (*Protocol Data Unit*).

Transmisja danych przez sieć datagramową wiąże się z możliwością odebrania danych przez urządzenie



Rys. 1. Struktura ramki Modbus RTU i Modbus TCP

odbiorcze w innej kolejności, niż zostały wysłane przez urządzenie nadawcze. Z tego względu w nagłówku protokołu Modbus TCP wprowadzono dwubajtowe pole identyfikatora transakcji *Transaction ID* zawierające numer transmitowanej ramki. Pozwala ono na identyfikację polecenia urządzenia Master, na które odebrano odpowiedź od urządzenia Slave.

Dwubajtowe pole *Protocol ID* zawiera identyfikator protokołu (dwa bajty o wartości 0).

Dwubajtowe pole *Length* zawierające informację o długości pozostałej części ramki pozwala na rozdzielenie poszczególnych ramek zgromadzonych w buforze odbiorczym urządzenia.

Adres urządzenia podrzędnego *Slave ID* zastąpiono identyfikatorem urządzenia *Unit ID*.

Jednostka danych protokołu *Protocol Data Unit* Modbus TCP jest zgodna z formatem PDU protokołu Modbus RTU [12–14].

Ramka Modbus TCP nie zawiera sumy kontrolnej, gdyż prawidłowość dostarczenia danych gwarantują protokoły warstw niższych [3–6, 18, 19].

3. PROTOKÓŁ HTTP

Protokół HTTP wykorzystywany jest do przesyłania dokumentów hipertekstowych pomiędzy przeglądarkami internetowymi a serwerami sieci WWW (*World Wide Web*) [7, 11]. Zasoby HTTP znajdujące się w sieci identyfikowane są przez adresy zasobów sieciowych w konwencji URI (*Universal Resource Identifier*). Adresy URI mogą wskazywać nie tylko na pliki utworzone w formacie HTML (*HyperText Markup Language*) [2, 15, 17, 20], lecz na dowolne zasoby sieci. Do identyfikacji przesyłanych danych używany jest podzbiór MIME (*Multipurpose Internet Mail Extensions*). Protokołem HTTP można więc przesyłać pliki o dowolnym formacie i zawartości, jeżeli tylko serwer będzie w stanie je udostępnić [7, 11].

Pobierane mogą być dane przechowywane na serwerze lub wygenerowane dynamicznie na żądanie użytkownika. Dynamiczne tworzenie witryn może być realizowane po stronie serwera, np. skrypty CGI (*Common Gateway Interface*), interpretery PHP (*Personal Home Page*), servlety, ASP (*Active Server Pages*) lub po stronie klienta, np. JavaScript, applety, DHTML (*Dynamic HTML*), technologia Flash itp. [2, 8, 15–17, 20]. Zwykle łączy się wybrane technologie po stronie serwera z tymi po stronie klienta. Możliwość dynamicznego generowania zawartości strony wyświetlanej w przeglądarce internetowej może być wykorzystana w systemach monitorowania. Aktualizacja zawartości okna przeglądarki zgodnie ze stanem monitorowanego urządzenia może być realizowana bez konieczności odświeżania strony przez użytkownika. Żądanie przesłania aktualnych danych może być generowane przez skrypt po stronie klienta.

Komunikacja pomiędzy przeglądarką internetową a serwerem WWW odbywa się w modelu klient-serwer. Klient otwiera połączenie, wysyła żądanie do serwera, a serwer przetwarza je, zwraca odpowiedź i zazwyczaj zamyka połączenie. Komunikaty przesyłane przez HTTP składają się z dwóch elementów: nagłówka (*header*) i ciała (*body*) [7, 11]. Żądanie klienta zawarte jest w nagłówku HTTP. W odpowiedzi serwer zwraca swój nagłówek i, jeżeli jest to wymagane, ciało komunikatu. Na rysunkach 2 i 3 przedstawiono przykładową komunikację pomiędzy przeglądarką internetową a serwerem WWW.

Protokół HTTP używa ośmiu metod (GET, POST, HEAD, PUT, DELETE, OPTIONS, TRACE, CONNECT) przesyłanych w nagłówkach i określających żądane akcje [7, 11]. Aplikacja uniwersalnego serwera monitorowania wykorzystuje metodę GET, a dane związane ze sterowaniem przez użytkownika informacjami wyświetlanymi w oknie przeglądarki realizowane jest przez przesyłanie parametrów z ukrytych formularzy zawartych na stronie.

```
GET /index.html HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:50.0) Gecko/20100101 Firefox/50.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: pl,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
```

Rys. 2. Przykładowe zapytanie HTTP klienta

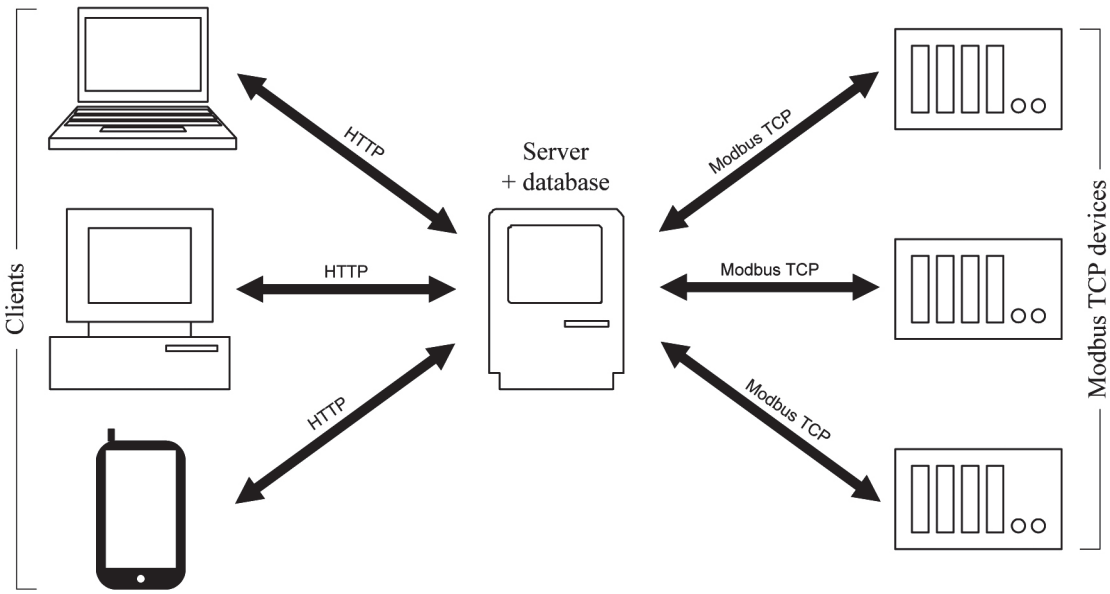
```
HTTP/1.1 200 OK
Pragma: no-cache
Cache-Control: post-check=0, pre-check=0, no-store, max-age=0
Content-Type: text/html
Connection: Keep-Alive
Keep-Alive: timeout=30, max=93
Content-Length: 1631


<html>
    <head>
        <meta charset=”iso-8859-2" />
        <title>Modbus Serwer</title>
        <link href=”style.css” type=”text/css” rel=”stylesheet”>
    </head>
    <body>
        <div class=”title”>Serwer testowy</div>
...
...
        <div class=”footer”>
           <a href=”http://www.kener.elektr.polsl.pl/”><img src=”stopka.gif”/></a>
           &copy; 2016-2017 <a href=”http://www.kener.elektr.polsl.pl/”>
           KENER - Politechnika &#346;l&#261;ska</a>
        </div>
    </body>
</html>
```

Rys. 3. Fragment przykładowej odpowiedzi HTTP serwera

Protokół HTTP jest protokołem bezstanowym, tzn. po zamknięciu połączenia serwer nie przechowuje żadnych informacji o poprzednich transakcjach [7, 11]. Z tego powodu każde zapytanie traktowane jest przez serwer jako nowe i niemożliwe do powiązania z informacjami o poprzednio przesłanych danych. W celu zapamiętania stanów związanych z poprzednim połączeniem najczęściej wykorzystuje się mechanizm ciasteczek (*cookies*). Informacje o ciasteczkach mogą być zawarte w nagłówkach przesyłanych komunikatów. Inną metodą jest przesyłanie ukrytych parametrów z formularzy zawartych na stronie lub umieszczenie parametrów w adresie URL zapytania. Aplikacja uniwersalnego serwera monitorowania wykorzystuje ciasteczka do przesyłania informacji o statusie zalogowania użytkownika oraz elementach wybranych przez użytkownika do wyświetlenia w oknie przeglądarki. Ciasteczka wykorzystywane są także w zapytaniach generowanych automatycznie przez skrypty JavaScript [8, 15–17] aktualizujące zawartość strony monitorowanego urządzenia.

4. KONCEPCJA SYSTEMU

Na rysunku 4 przedstawiono koncepcję systemu łączności.



Rys. 4. Koncepcja systemu łączności

Elementem pośredniczącym pomiędzy urządzeniami pracującymi z protokołem Modbus TCP [13] a urządzeniami wykorzystującymi przeglądarkę internetową jest stworzona w tym celu aplikacja serwerowa współpracująca z bazą danych SQL Firebird [1]. Aplikacja serwerowa pracuje ze stałym, globalnym adresem IP oraz otwiera dwa porty nasłuchujące: jeden do obsługi połączeń realizowanych za pomocą protokołu HTTP, a drugi do komunikacji z urządzeniami Modbus TCP.

Rolą aplikacji jest odczyt danych z urządzeń Modbus TCP oraz udostępnianie ich klientom w formie akceptowanej przez przeglądarki internetowe.

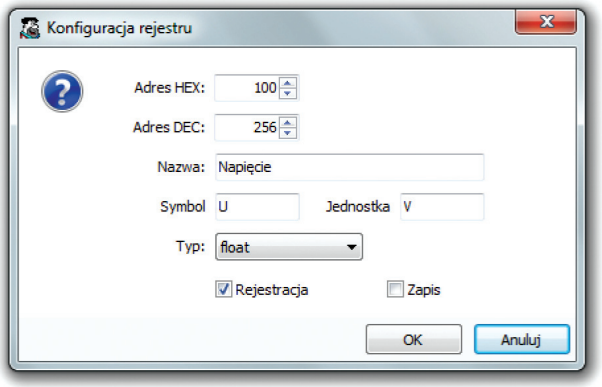
Współpraca z bazą danych pozwala na archiwizowanie danych odebranych z urządzeń oraz udostępnianie danych archiwalnych urządzeniom klienckim.

5. APLIKACJA SERWERA

Aplikacja serwerowa została stworzona dla systemu Windows w postaci pojedynczego pliku wykonywalnego z możliwością uruchomienia jako usługa systemowa. Jest aplikacją autonomiczną, niewymagającą instalacji innych serwerów WWW, takich jak np. Apache, Nginx czy ISS.

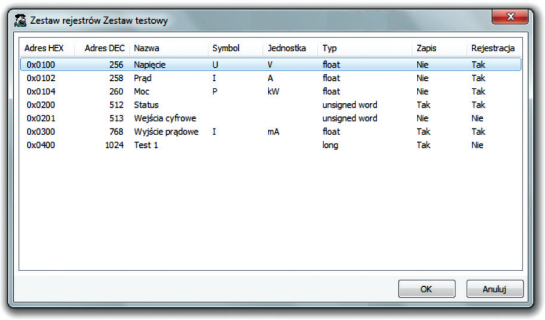
Założeniem przy opracowywaniu aplikacji było stworzenie uniwersalnego serwera współpracującego z dowolnego typu urządzeniami Modbus TCP. Protokół Modbus TCP [13] nie narzuca powiązania informacji zawartej w rejestrze z adresem rejestru. Dopuszcza także różny sposób kodowania wartości zawartych w rejestrach (liczby stałoprzecinkowe ze znakiem lub bez znaku, liczby zmiennoprzecinkowe) oraz rozmiar kodowanych liczb (liczby o zakresie szerszym niż 16 bitów mogą być umieszczane w kilku rejestrach). Dodatkowo w ramach jednego urządzenia możliwy jest różny sposób kodowania zawartości poszczególnych rejestrów.

Z tego względu konieczne jest udostępnienie możliwości określenia zakresu rejestrów i formatu danych dla każdego z wprowadzanych do systemu urządzeń. Na rysunku 5 przedstawiono okno aplikacji serwerowej umożliwiające konfigurację rejestru urządzenia Modbus TCP. Można wprowadzić adres rejestru, nazwę, symbol i jednostkę, typ kodowania danych, informację, czy odczyt rejestru ma być archiwizowany w bazie danych oraz czy możliwa będzie edycja zawartości rejestru.



Rys. 5. Konfiguracja rejestru urządzenia Modbus TCP

Rejestry grupowane są w zestawy. Zdefiniowane zestawy rejestrów można przypisać do urządzeń Modbus TCP wprowadzanych do systemu. Na rysunku 6 przedstawiono przykładowy zestaw rejestrów.

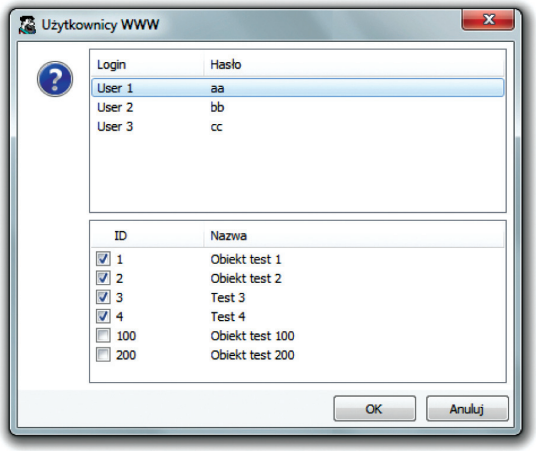


Rys. 6. Definiowanie zestawu rejestrów

Kolejnym etapem jest zdefiniowanie urządzeń Modbus TCP polegające na wprowadzeniu identyfikatora urządzenia wykorzystywanego w polu *Unit ID* protokołu Modbus TCP, przypisanie nazwy urządzenia, zestawu rejestrów oraz określenia częstotliwości odpytywania urządzenia przez aplikację serwerową.

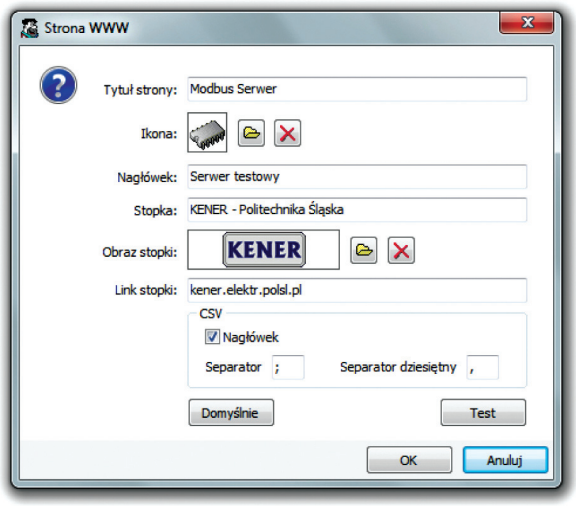
Po nawiązaniu przez urządzenie połączenia z serwerem aplikacja serwerowa automatycznie co zadany czas komunikuje się z urządzeniem, odczytując zawartość rejestrów zdefiniowanych w przypisanym zestawie, łącząc w miarę możliwości grupy rejestrów w pojedyncze ramki w celu optymalizacji czasu wymaganego na przesłanie wszystkich danych [9, 10], dekoduje ich wartość zgodnie z wprowadzonym typem oraz rejestruje odczyt w bazie danych. Umożliwia także zapis wartości do rejestrów, dla których ustalono taką możliwość.

Konfiguracja serwera od strony połączeń przeglądarki internetowej wymaga wprowadzenia listy użytkowników, haseł dostępu oraz przypisania urządzeń, do których dany użytkownik może uzyskać dostęp. Na rysunku 7 przedstawiono okno konfiguracji listy użytkowników.



Rys. 7. Okno konfiguracji listy użytkowników

W obecnej wersji aplikacja serwera generuje strony WWW według jednego, wbudowanego szablonu. Umożliwia jednak konfigurację sposobu wyświetlania wybranych elementów strony. Na rysunku 8 przedstawiono okno konfiguracji elementów wizualnych strony WWW generowanej przez serwer oraz formatu pliku CSV (*Comma-separated values*) zawierającego archiwalne dane udostępniane przez serwer.



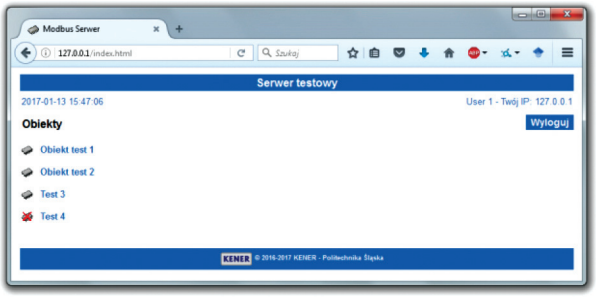
Rys. 8. Okno konfiguracji elementów strony WWW generowanej przez serwer

Po skonfigurowaniu wymaganych opcji aplikacja serwera nie wymaga reakcji użytkownika i może być uruchomiona w trybie usługi systemowej.

6. PRZEGLĄDARKA INTERNETOWA

Monitorowanie stanu urządzeń Modbus TCP oraz przeglądanie danych archiwalnych możliwe jest za pomocą przeglądarki internetowej. Dostęp do informacji udostępnianych przez aplikację serwera wymaga logowania użytkownika. Proces logowania zrealizowano za pomocą algorytmu kryptograficznego MD5 (*Message-Digest algorithm 5*), a dane dotyczące autoryzacji sesji kontrolowane są za pomocą mechanizmu ciasteczek przesyłanych w nagłówkach protokołu HTTP [7, 11].

Po weryfikacji danych logowania przesyłana jest strona zawierająca listę urządzeń, do których przydzielono dostęp zalogowanemu użytkownikowi wraz z symbolem stanu połączenia (rys. 9). Stan połączenia urządzeń Modbus TCP z serwerem aktualizowany jest automatycznie w odstępach 10-sekundowych.



Rys. 9. Lista urządzeń dostępnych dla zalogowanego użytkownika wygenerowana przez aplikację serwera

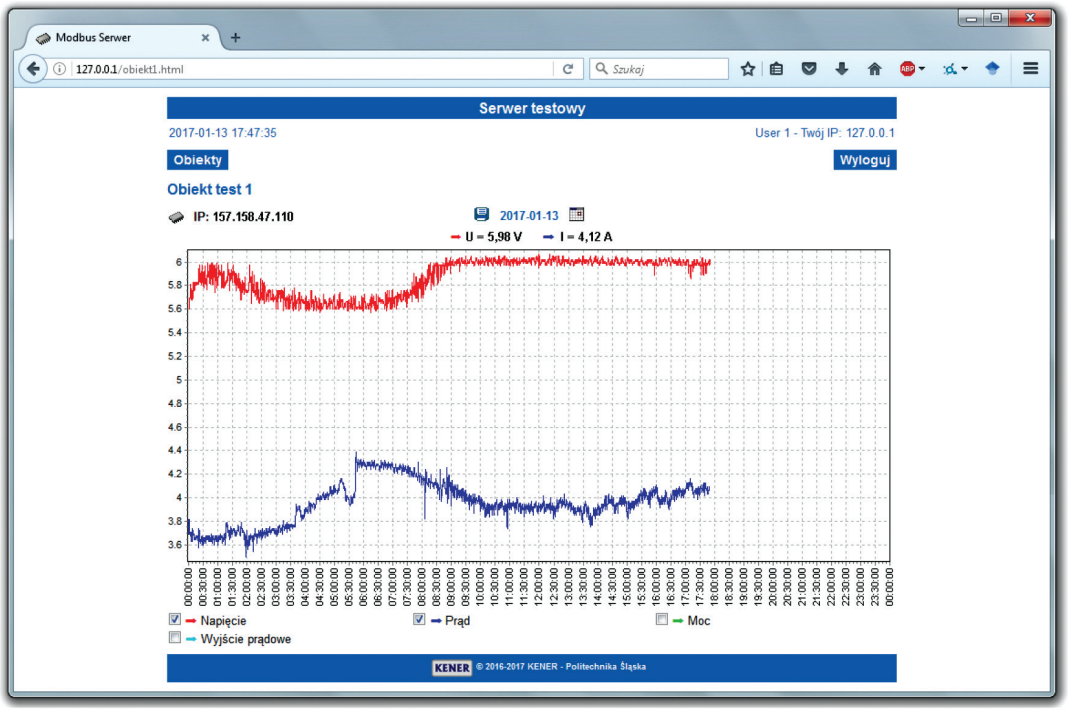
Wybór urządzenia przez użytkownika powoduje wygenerowanie strony z przebiegami pomiarowymi zarejestrowanymi przez serwer. Na podstawie danych zawartych w bazie aplikacja serwera tworzy plik graficzny w formacie PNG (*Portable Network Graphics*) z przebiegami czasowymi dla wybranego dnia i osadza jego URI w treści HTML strony przesyłanej do klienta [2, 20]. Generowany jest także skrypt JavaScript [8, 15–17] realizujący automatyczną aktualizację pliku PNG wyświetlanego w przeglądarce internetowej klienta, daty, czasu oraz wartości chwilowych wielkości pomiarowych podczas ostatniej aktualizacji danych, a także statusu połączenia urządzenia z serwerem. Operacje te odbywają się bez konieczności podejmowania dodatkowych działań przez użytkownika. Na rysunku 10 przedstawiono przykładowy widok strony monitorowania pomiarów odczytanych z urządzenia.

Ikona kalendarza umieszczona w górnej części strony pozwala na uruchomienie skryptu wyświetlającego kalendarz i umożliwia pobranie z serwera danych dla innego dnia. Pola wyboru umieszczone pod wykresem pozwalają na uwzględnienie w generowanym przez serwer pliku PNG wybranych wielkości pomiarowych.

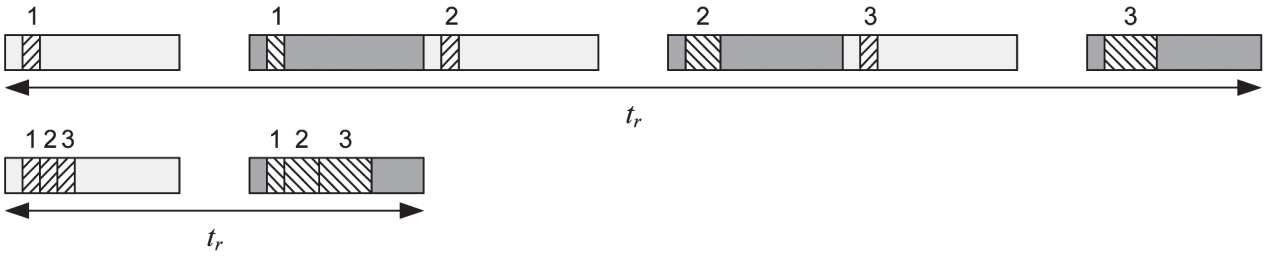
Ikona dyskietki umieszczona w górnej części strony pozwala na wysłanie do serwera żądania pobrania danych pomiarowych dla wybranego dnia. Aplikacja serwera na podstawie żądania klienta generuje odpowiednie dane w formie pliku SCV i przesyła je do klienta. Przeglądarka internetowa umożliwia zapis pobranych danych do pliku na komputerze klienta w celu dalszej ich analizy z wykorzystaniem powszechnie dostępnych narzędzi, np. oprogramowania Microsoft Excel.

7. TESTY WYDAJNOŚCI POŁĄCZENIA MODBUS TCP

W celu weryfikacji koncepcji łączenia odczytywanych rejestrów urządzeń Modbus TCP w ramach jednego pakietu warstwy ethernetowej przeprowadzono testy porównawcze wydajności systemu. Zestaw rejestrów zdefiniowano w taki sposób, aby aplikacja serwera wysyłała do urządzenia trzy rozkazy odczytujące odpowiednio 1 rejestr, 16 rejestrów i 64 rejestry – w pierwszym przypadku wysyłając każde z zapytań osobno, a w drugim łącząc je w jeden pakiet.



Rys. 10. Strona monitorowania wielkości pomiarowych urządzenia wygenerowana przez aplikację serwera



Rys. 11. Ilustracja sposobu łączenia ramek oraz pomiaru czasu odpowiedzi t_r

Na rysunku 11 przedstawiono ilustrację sposobu łączenia ramek protokołu Modbus TCP oraz pomiaru czasu odpowiedzi. W przypadku transmisji osobnych pakietów dla każdej z ramek następne zapytanie wysyłane było niezwłocznie po otrzymaniu odpowiedzi na poprzedni rozkaz. Rejestrowano czas od chwili zainicjowania transmisji do chwili odebrania kompletnej odpowiedzi zawierającej dane wszystkich rejestrów. Sekwencja testowa była powtarzana co 200 ms. Testy prowadzone były w ramach sieci składającej się zarówno z urządzeń warstwy Ethernet zbudowanej z wykorzystaniem przewodów miedzianych, jak i urządzeń warstwy internetowej wykorzystującej światłowody. Trasa pakietów wiodła przez 11 węzłów sieciowych identyfikowanych programem *tracert*. Pomiary testowe wykonano kilkanaście razy w różnych okresach doby, a czas pojedynczego testu wynosił 5 min. Testowanym urządzeniem Modbus TCP był sterownik przemysłowy typu STPC-09 firmy ENEL-PC.

Tabela 1

Wyniki testów wydajnościowych

Tryb testu	Maksymalny czas odpowiedzi [ms]	Średni czas odpowiedzi [ms]
Trzy ramki w osobnych pakietach	132	88,6
Trzy ramki we wspólnym pakiecie	28,4	13,8

Jak można zauważyć na podstawie tabeli 1 średni czas odpowiedzi przy wysyłaniu trzech osobnych pakietów jest dłuższy niż trzykrotna wartość czasu zmierzonego podczas wysyłania jednego pakietu zawierającego trzy ramki protokołu Modbus TCP. Różnica ta wynika ze sposobu odbierania, identyfikowania zapytania i zwracania odpowiedzi przez testowane urządzenie Modbus. Należy również wziąć pod uwagę, że obsługa łączności interfejsu Modbus TCP nie jest głównym zadaniem testowanego urządzenia i prawdopodobnie realizacja tego zadania nie jest traktowana priorytetowo.

Na podstawie przeprowadzonych testów można stwierdzić, że idea łączenia ramek protokołu Modbus TCP we wspólnym pakiecie warstwy Ethernet zwiększa wydajność systemu w wyniku skrócenia czasów uzyskania odpowiedzi na wysyłane rozkazy oraz pozwala na zmniejszenie natężenia ruchu sieciowego. Ma to szczególne znaczenie w przypadku, gdy aplikacja serwerowa monitoruje stan wielu urządzeń. Niezbędna jest jednak możliwość odbioru i prawidłowej identyfikacji tego typu zapytań w urządzeniu Modbus TCP.

8. PODSUMOWANIE

Celem prac przedstawionych w artykule było sprawdzenie koncepcji oraz wydajności uniwersalnego serwera pośredniczącego pomiędzy urządzeniami realizującymi transmisję danych z wykorzystaniem protokołu Modbus TCP a przeglądarką internetową.

Dzięki możliwości konfiguracji w aplikacji serwera adresów rejestrów i sposobu kodowania danych niezależnie dla każdego ze zdefiniowanych urządzeń Modbus TCP uzyskano uniwersalność systemu. Możliwa jest realizacja łączności i interpretacja danych dla dowolnego urządzenia pracującego z tym protokołem. Wymagana jest jedynie możliwość zainicjowania połączenia do serwera przez monitorowane urządzenie.

Realizowane przez aplikację automatyczne grupowanie rejestrów przesyłanych w pojedynczej ramce transmisyjnej pozwala na ograniczenie ilości przesyłanych danych związanych z nagłówkami niższych warstw stosu protokołów sieciowych i zwiększa wydajność systemu.

Opracowana niezależna aplikacja serwera ma postać pojedynczego pliku wykonywalnego, a generowanie stron dla przeglądarek internetowych nie wymaga instalacji zewnętrznych serwerów WWW.

Kolejnym etapem prac będzie wprowadzenie możliwości definiowania komunikatów tekstowych powiązanych ze stanem wybranych rejestrów. Umożliwi to wyświetlenie na stronach WWW generowanych przez serwer słownego opisu stanu pracy monitorowanych urządzeń.

Trwają także prace nad włączeniem do systemu starszych urządzeń z tradycyjnym protokołem Modbus, nieobsługujących połączeń w sieci IP. Transmisja danych w sieci IP możliwa jest w takim przypadku w wyniku zastosowania sprzętowego konwertera RS-232/RS-485 – TCP/IP, np. urządzeń z serii NPort firmy Moxa.

Motywacją do podjęcia tematu były liczne zapytania z ośrodków przemysłu dotyczące możliwości monitorowania stanu urządzeń za pomocą przeglądarki internetowej, a dotychczasowe wdrożenia potwierdzają prawidłowe działanie przyjętej koncepcji.

Literatura

[1] Borrie H.: *The Firebird Book. A Reference for Database Developers*, Apress 2004.
[2] Castro E.: *HTML 4*, Helion, Gliwice 2003.
[3] Comer D.E.: *Sieci komputerowe TCP/IP. Tom 1. Zasady, protokoły i architektura*, WNT, Warszawa 1997.
[4] Comer D.E., Stevens D.L.: *Sieci komputerowe TCP/IP. Tom 2. Projektowanie i realizacja protokołów*, WNT, Warszawa 1997.
[5] Comer D.E., Stevens D.L.: *Sieci komputerowe TCP/IP. Tom 3. Programowanie w trybie klient–serwer*, WNT, Warszawa 1997.
[6] Fall K.R., Stevens W.R.: *TCP/IP od środka: protokoły*, Helion, Gliwice 2013.
[7] Gourley D., Totty B., Sayer M.: *HTTP: The Definitive Guide*, O'Reilly Media 2002.
[8] Jakut T.: *JavaScript: programowanie zaawansowane*, Helion, Gliwice 2006.
[9] Jestratjew A., Kwiecień A.: *Performance of HTTP Protocol in Networked Control Systems*, IEEE Transactions on Industrial Informatics, 2013, 9, 1: 271–276.
[10] Joelianto E., Hosana H.: *Performance of an industrial data communication protocol on Ethernet network*, 5th IFIP International Conference on Wireless and Optical Communications Networks (WOCN '08), Surabaya 2008: 1–5.
[11] Krishnamurthy B., Rexford J.: *Web protocols and practice: HTTP/1.1, networking protocols, caching, and traffic measurement*, Addison-Wesley Professional, Boston 2001.
[12] *MODBUS application protocol specification V1.1b*, Modbus-IDA, 2006.
[13] *MODBUS Messaging on TCP/IP Implementation Guide V1.0b*, Modbus-IDA, 2006.
[14] *Modicon MODBUS Protocol Reference Guide*, Modicon Inc., USA 1993.
[15] Radziszewski B.: *HTML, JavaScript i Java od podstaw*, Wyd. Politechniki Świętokrzyskiej, Kielce 2002.
[16] Resig J., Ferguson R., Paxton J.: *Zaawansowane techniki języka JavaScript*, Helion, Gliwice 2016.
[17] Romowicz W.: *HTML i JavaScript*, Helion, Gliwice 1998.
[18] Scrimger R., LaSalle P., Leitzke C., Parihar M., Gupta M.: *TCP/IP: biblia*, Helion, Gliwice 2002.
[19] Siyan K.S., Parker T.: *TCP/IP. Księga eksperta*, Helion, Gliwice 2002.
[20] Strychalski R.: *HTML: tworzenie stron www i programów desktopowych*, Nakom, Poznań 2015.

dr inż. MARIAN HYLA
Wydział Elektryczny
Politechnika Śląska
ul. B. Krzywoustego 2, 44-100 Gliwice
marian.hyla@polsl.pl