http://dx.doi.org/10.7494/miag.2017.2.530.7

MARIAN HYLA

Universal server for monitoring industrial devices using web browser

This paper presents the implementation of a universal intermediate server for the monitoring of industrial devices using the Modbus TCP protocol. The application provides selected information using the HTTP protocol accepted by web browsers. The software configuration and sample pages generated by the server are described. The results of the Modbus TCP interface performance tests are presented.

Key words: *monitoring, remote access, Modbus TCP, HTTP, WWW*

1. INTRODUCTION

Modern IT technologies provide a lot of possibilities for data transmission using local computer networks as well as the global Internet. In the age of widespread access to the Internet, more and more industrial devices are equipped with network interfaces that are used for remote control and the monitoring of operating conditions. Practically, the location of the monitored devices and the location of the user are no longer relevant.

The Modbus TCP protocol is one of the most-commonly-implemented protocol for industrial devices with Ethernet/Internet connectivity [13]. On the user's side, common and generally accessible network access tools are web browsers that use the HTTP protocol (*Hypertext Transfer Protocol*) [11].

Both Modbus TCP and HTTP are application-layer protocols of the TCP/IP protocol stack and use the TCP (*Transmission Control Protocol*) layer and lower layers for data transport [3–7, 11, 13, 18, 19]. To display the data from Modbus TCP devices via a web browser, it is necessary to use the appropriate tool (application) that is required to allow data conversion between these two protocols.

2. MODBUS TCP PROTOCOL

The Modbus protocol [12, 14] was developed in 1979 by the Modicon company, acquired by the Schneider Electric in the mid-90s of the last century. Despite the rapid development of new communication techniques and the development of new transmission protocols, it is still one of the most-popular industrial protocols and can be used successfully for local communication with devices equipped with serial asynchronous data communication interfaces in a master-slave communication model.

The widespread use of the Internet and the possibility of the TCP/IP stack implementation [3–6, 18, 19] in industrial devices have resulted in the development of a new version of the Modbus protocol for packet networks (named Modbus TCP or Modbus TCP/IP) [13]. This protocol is based on the Modbus RTU protocol variant and is a Modbus implementation for the IP network. A comparison of the frame structure of the Modbus RTU and Modbus TCP is shown in Figure 1.

The Modbus TCP protocol data frame (*Application Data Unit*) contains the Modbus Application Header and the Protocol Data Unit.

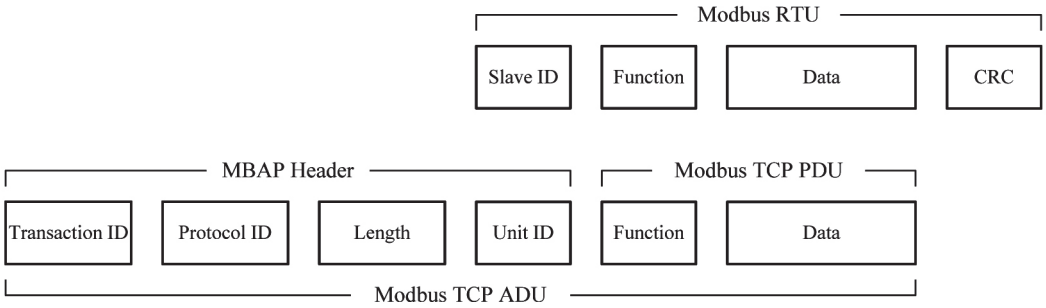

Fig. 1. Modbus RTU and Modbus TCP frame structure

Data transmission over a datagram network results in the fact that the receiving device may receive data in a different order than was sent by the transmitting device. Therefore, in the Modbus TCP header, a two-byte transaction identification number field *Transaction ID* containing the transmitted frame number has been added. It allows for the identification of the master device command for responses received from the slave device.

The two-byte field *Protocol ID* contains the protocol identifier (two bytes equal to 0).

Two-byte field *Length* contains information about the length of the remaining part of the frame and allows for the storage of separate individual frames in the device's receive buffer.

The address of slave device *Slave ID* was replaced by device identifier *Unit ID*.

The Modbus TCP *Protocol Data Unit* is compatible with the Modbus RTU protocol PDU [12, 13, 14].

The Modbus TCP frame does not contain a checksum, as the delivery of the data is guaranteed by the lower layer protocols [3, 6, 18, 19].

3. HTTP PROTOCOL

The HTTP protocol is used to send hypertext documents between web browsers and WWW (*World Wide Web*) servers [7, 11]. HTTP resources on the network are identified by the network resource addresses in the URI (*Universal Resource Identifier*) convention. URIs can point to not only the http (*HyperText Markup Language*) files [2, 15, 17, 20] but to any network resource. To identify the transferred data, a subset of MIME (*Multipurpose Internet Mail Extensions*) is used. Therefore, the HTTP can transfer files of any format and content as long as the server is able to share them [7, 11].

Downloaded data may be stored on the server or dynamically generated at the request of the user. Dynamic page creation can be done on the server side (e.g., CGI [*Common Gateway Interface*] scripts, PHP [*Personal Home Page*] interpreters, servlets, ASP [*Active Server Pages*]) or on the client side (e.g., JavaScript, applets, DHTML [*Dynamic HTML*], Flash technology, etc.). [2, 8, 15–17, 20]. Typically, a combination of server-side and client-side technologies are used. The ability to dynamically generate the page content displayed on a web browser can be used in the monitoring systems. Updating the content of the browser window according to the status of the monitored device can be realized without the user having to refresh the page. A request to upload current data may be generated by a client-side script.

Communication between the web browser and web server takes place in the client-server model. The client opens the connection and sends the request to the server, and the server processes it, returns the response, and usually closes the connection. Messages sent via HTTP consist of two elements: the header and body [7, 11]. The client request is included in the HTTP header. In the response, the server returns its header and (if required) the body of the message. Figures 2 and 3 show an example of the communication between a web browser and a web server.

HTTP uses 8 methods (GET, POST, HEAD, PUT, DELETE, OPTIONS, TRACE, CONNECT) placed in the headers and specifying the requested actions [7, 11]. The universal monitoring server application uses the GET method. The data displayed in the browser window, related to the user action, is processed by passing the parameters from the hidden forms contained on the page.

HTTP is a stateless protocol; i.e., the server does not store any information about previous transactions after the connection is closed [7, 11]. Therefore, each query is treated as new by the server and cannot be

```
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
```

Fig. 2. Example of HTTP client request

```
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>
```

Fig. 3. Fragment of example of HTTP server response

associated with information about previously transmitted data. In order to remember the states associated with the previous connection, the *cookies* mechanism is most-commonly used. Cookies information can be included in the HTTP message headers. Another method is to upload the hidden parameters from the forms included on the page or to place the parameters in the query URL. The universal monitoring server application uses cookies to send user login status information and user selectable items for displaying in a browser window.

Cookies are also used in the queries generated automatically by JavaScript [8, 15–16] that update the contents of a monitored device page.

4. SYSTEM IDEA

Figure 4 illustrates the idea of a communications system.

For this purpose, the server application was developed. The server application, cooperating with the Firebird SQL database [1], is an interface between devices using the Modbus TCP protocol [13] and web browsers. The server application operates with a fixed global IP address and opens two listening ports: one for handling HTTP connections and the other for communication with Modbus TCP devices.

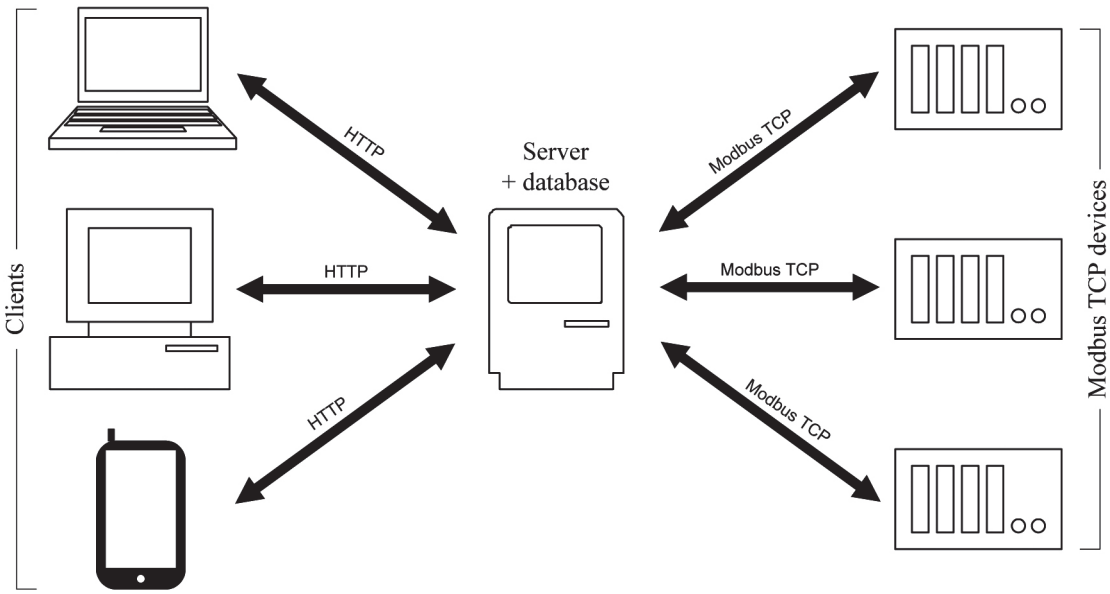

Fig. 4. Idea of communications system

The purpose of the application is to read data from the Modbus TCP devices and make it available to clients in the form accepted by the Internet browsers.

Cooperation with the database allows for the archiving of data received from the devices and sharing of the archived data to the client devices.

5. SERVER APPLICATION

The server application was created for the Windows operating system as a single executable file that can also be run as a system service. It is a standalone application that does not require the installation of other web servers like Apache, Nginx, or ISS, for example.

The assumption when developing the application was the creation of a universal server working with any type of Modbus TCP device. The Modbus TCP protocol [13] does not dictate the association of the information contained in the registry with the registry address. It also allows for the different coding of values contained in the registers (fixed point number with or without a sign, floating point numbers, etc.) and the size of the encoded numbers (those greater than 16 bits can be placed in several registers). In addition, a different method of coding the contents of individual registers can be used in single device.

For this reason, it is necessary to make it possible to define the configuration of the registers and format of the data for each of the devices entered into the system. Figure 5 shows a server application window that allows us to configure the Modbus TCP registry. The user can enter the address of the registry, name, symbol and unit, type of data encoding, information about whether the read register value can be archived in the database, and whether the contents of the register can be edited.

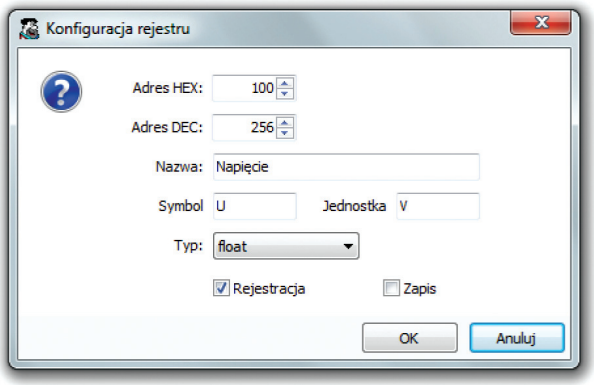

Fig. 5. Modbus TCP device registry configuration window

Registers are grouped in sets. Defined register sets can be assigned to the Modbus TCP devices declared in the system. Figure 6 shows an example set of registers.

The next step is to define the Modbus TCP devices by entering the device identifier used in the Modbus TCP Unit ID field, assigning a device name and a set of registers as well as determining the device communication frequency by the server application.

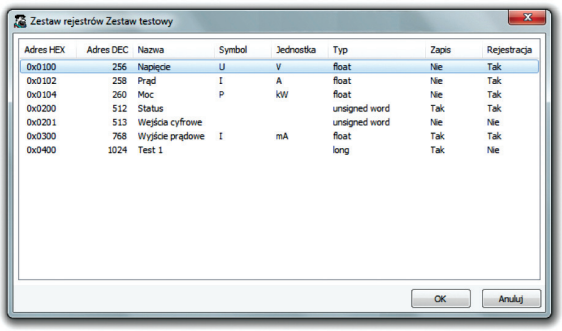

Fig. 6. Defining set of registers

After the device has connected to the server, the server application automatically communicates with the device and sends a request to the device with a set period of time. The server reads the contents of the registers defined in the assigned registry set, combining as many registers as possible into a single frame to optimize the time required to transmit all data, [9, 10]. It then decodes their value according to the entered type and registers the read values in the database. It also allows us to enter values to the registers for which this possibility was set.

Configuration of the server application for web browser connection requires us to provide a user list with access passwords and assign the devices that the user can access. Figure 7 shows the user list configuration window.

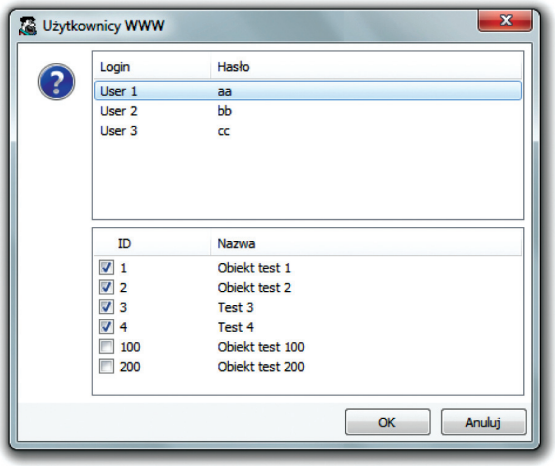

Fig. 7. User list configuration window

The current version of the server application generates web pages according to the single built-in template. However, it allows us to configure the presentation of selected page elements. Figure 8 shows the configuration window for the visual elements of a web page generated by the server and a CSV (*Comma-separated values*) file format that contains archived data provided by the server.

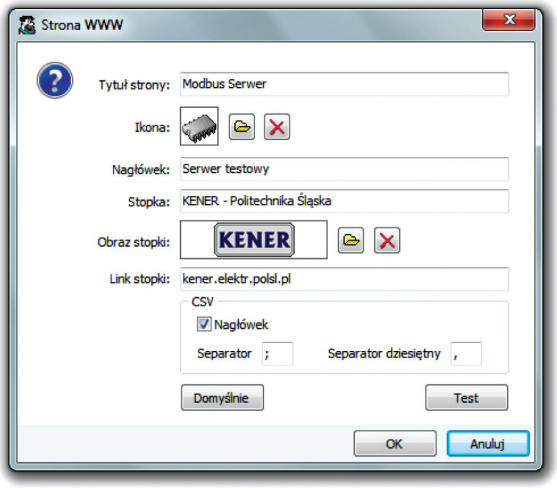

Fig. 8. Window for configuring elements of web page generated by server

After configuring the required options, the server application does not require user interaction and can be run in system service mode.

6. WEB BROWSER

Monitoring of current status of the Modbus TCP devices and archived data review can be accessed by a web browser. Access to the information provided by the server application requires a user login. The logon process was implemented using the cryptographic MD5 (*Message-Digest algorithm 5*) algorithm, and session authorization data is controlled by the HTTP-based cookie mechanism [7, 11].

After verification of the login data, the server application sends a web page containing the list of devices to which the logged-in user has been granted access together with the connection status symbol as shown in Figure 9. The status of the Modbus TCP connections to the server is automatically updated at ten-second intervals.

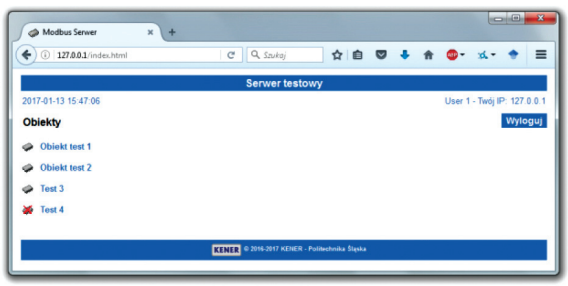

Fig. 9. List of devices available to logged-in user generated by server application

Selection of the device by the user generates a web page containing the measurement values registered by the server. Based on the data contained in the database, the server application generates a PNG (*Portable Network Graphics*) graphic file with the waveforms for the selected day and embeds its URI in the HTML content of the page sent to the client [2, 20]. The server also generates a JavaScript [8, 15–17] that automatically updates the PNG file displayed in the client's web browser, refreshes the date, time, and instantaneous measurement values during the last data update, and shows the status of the Modbus TCP device connection. These operations take place without the need for additional user action. Figure 10 shows an example view of the Modbus TCP device measurement web page.

The calendar icon at the top of the page runs a script that displays the calendar and allows data from another day to be downloaded from the server. The checkboxes below the graph are used to include a selection of the measurements returned by the server in the server-generated PNG file.

The floppy disk icon at the top of the page allows the page to send a request to the server to retrieve the measurement data for the selected day. The server application generates the appropriate data based on the client request in the form of a CSV file and sends it to the client. A web browser allows the page to save the downloaded data to a file on the client computer for further analysis using commonly available spreadsheets such as Microsoft Excel.

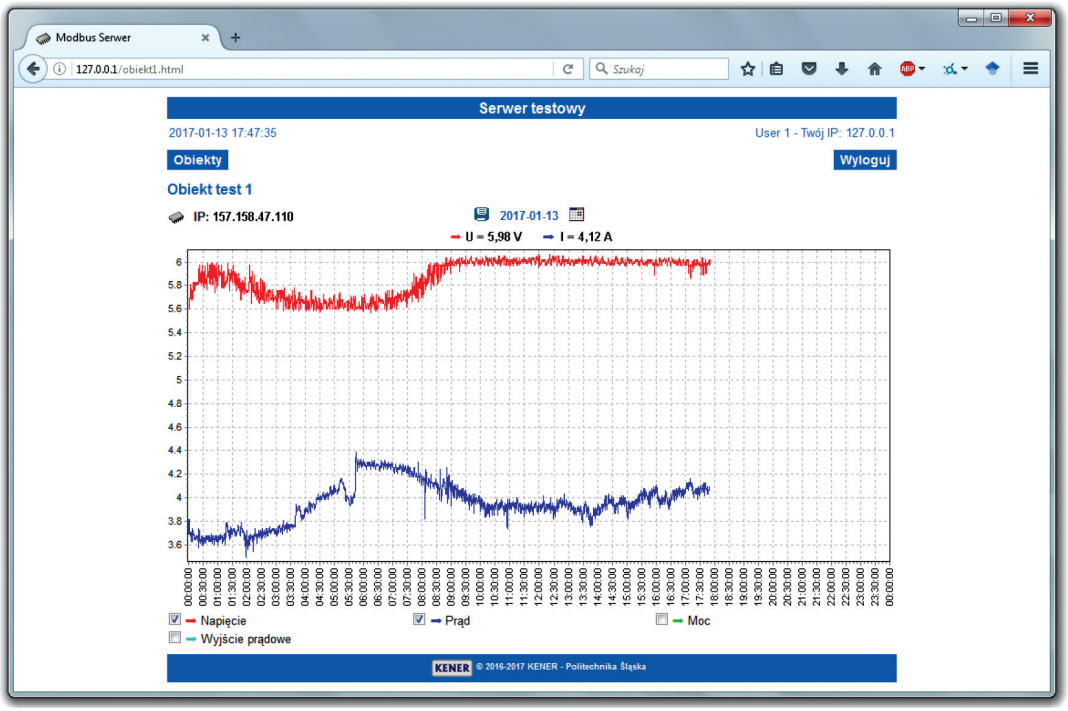

Fig. 10. Web page generated by server application for measured value monitoring

7. EFFICIENCY TESTS OF MODBUS TCP CONNECTIONS

In order to verify the concept of combining the read registers of the Modbus TCP devices within a single Ethernet layer package, system performance tests were performed. The set of registers was defined in such a way that the server application sends 3 commands to the device, reading 1, 16, and 64 registers, respectively – in the first case, by sending each query separately, and in the others by merging them into a single package.

Figure 11 shows an illustration of how to combine the Modbus TCP frames and the response time measurement method. In the case of separate packets for each frame, the next query was sent immediately after receiving the response to the previous command. The time of the response was measured from the initiation of the transmission until a complete response containing all register data had been received. The test sequence was repeated every 200 ms. The tests were performed in a network consisting of both an Ethernet-based computer network and Internet-based fiber-optic devices. The packet route passed through

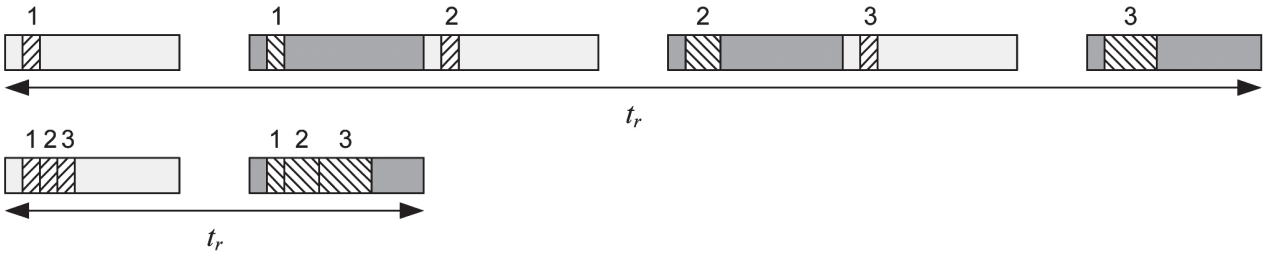

Fig. 11. Illustration of frames combining and response time t_r measuring method

11 network nodes identified by *tracert* traceroute software. Test measurements were done several times at different periods of the day, and the time of a single test was five minutes. The tested Modbus TCP device was an STPC-09-type industrial controller produced by the ENEL-PC company.

Table 1

Results of performance tests

Type of test	Maximum response time [ms]	Average response time [ms]
3 frames in separate packets	132	88.6
3 frames in a common packet	28.4	13.8

As can be seen in Table 1, the average response time in the case of sending three separate packets is more than three times greater than the measured time when sending a single packet containing three Modbus TCP frames. This difference is the result of the method of receiving, identifying a query, and returning a response by a tested Modbus TCP device. Also, it is worth noting that Modbus TCP interface support is not the primary task of the tested device and it is likely to accomplish this task with a relatively low priority.

The performed tests confirmed the idea of combining Modbus TCP frames in a common Ethernet layer packet to increase system performance by reducing the response time to the sent commands and reducing the amount of network traffic. This is especially true when the server application monitors the status of multiple devices. However, it is necessary to receive and correctly identify such queries on a Modbus TCP device.

8. SUMMARY

The purpose of the work presented in this article was to examine the concept and performance of a universal intermediate server application that handles data communication between devices using the Modbus TCP protocol and a web browser.

Thanks to the ability to configure the register addresses and encode the data mode independently to each of the defined Modbus TCP devices into the server application, system universality has been achieved. It is possible to perform communication and interpretation of data for any device working with this protocol. Only the ability to initiate a connection to the server by the monitored device is required.

Thanks to the automated grouping of registries sent in a single transmission packet, the server application can reduce the amount of data transmitted to the lower layers of the network protocol stack and increase system performance.

The developed server application is an independent application in the form of a single executable file, and web page generation does not require the installation of external web servers.

The next step will be the implementation of the ability to define text messages associated with the status of selected registers. This will allow us to display text-based status information of the monitored devices on the web pages generated by the server.

Work is also underway to connect older devices to the system with traditional Modbus protocols that do not support IP network connections. In this case, IP data transmission is possible using an RS-232/RS-485 – TCP/IP hardware converter; e.g., the Moxa NPort series.

Motivation to take up the topic were the numerous inquiries from the industry regarding the ability to monitor the state of devices with the use of a web browser. The implementations realized to this day confirm the proper operation of the proposed concept.

References

[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.

MARIAN HYLA, Ph.D., Eng.
Faculty of Electrical Engineering
Silesian University of Technology
ul. B. Krzywoustego 2, 44-100 Gliwice, Poland
marian.hyla@polsl.pl