Remote Control and Monitoring in the Simulink

Martin Sysel, Michal Vaclavsky

***Abstract*—**This paper describes a created application SimWebLink.NET, which allows remote control and monitoring of technological processes using Simulink. SimWebLink.NET application enables remote monitoring and control processes using an common Web browser supporting JavaScript. Remote monitoring and control can be just a simulation or control of a real system. The TCP/IP block sends out data from MATLAB/Simulink model using the TCP/IP. The new developed SimWebLink.NET and instructions for building this system are described here. This new version has been completely rewritten and now uses modern techniques on Microsoft platform as Window Communication Foundation and MS SQL server instead of Linux and open source.

***Keywords*—** ASP.NET, Simulink, , Socket, thread, WCF.

I. Introduction

AN emerging strategy for application software is to provide web access to software previously distributed as local applications. Depending on the type of application, it may require the development of an entirely different browser-based interface, or merely adapting an existing application to use different presentation technology. In software engineering, a web application is an application that is accessed via web browser over a network such as the Internet or an intranet. It is also a computer software application that is coded in a browser-supported language (such as HTML, JavaScript, etc.) and reliant on a common web browser to render the application executable. Web applications are popular due to the ubiquity of a client. The ability to update and maintain web applications without distributing and installing software on potentially thousands of client computers is a key reason for their popularity. One of the possible application which offers advantage of high performance tools for calculation and simulation is program MATLAB/Simulink.

The aim of this work is to create a web interface for remote control labs created in Matlab/Simulink. This work builds on earlier work [1]-[6].

A. MATLAB Web Server

MATLAB Web Server was the first application taking significant advantage as web applications. The MATLAB Web Server enables the creation MATLAB applications that use the capabilities of the World Wide Web to send data to MATLAB for computation and to display the results in a Web browser. The MATLAB Web Server depends upon TCP/IP networking for transmission of data between the client system and MATLAB. The required networking software and hardware must be installed on the system prior to using the MATLAB Web Server. This toolbox contains tools to connect MATLAB programs and HTML pages. The MATLAB Web Server requires MATLAB Release 11 or later. Unfortunately as of Release 2006b, this toolbox was discontinued and is no longer available for purchase. It is necessary continue using existing MATLAB Web Servers with MATLAB R2006a.

The application development process for MATLAB Web Server requires a small number of simple steps defined by Mathworks [7]:

1. Create the HTML documents for collection of the input data from users and display of output. It can be used a text editor to input HTML directly, or it can be used one of the commercially available HTML authoring systems.
2. List the application name and associated configuration data in the configuration file matweb.conf.
3. Write a MATLAB m-file that:

- Receives the data entered in the HTML input form.
- Analyzes the data and generates any requested graphics.
- Places the output data into a MATLAB structure.
- Calls htmlrep to place the output data into an HTML output document template. The maximum amount of HTML received data from MATLAB is 256 kB.

MATLAB Web Server applications are a combination of M files, Hypertext Markup Language (HTML), and graphics. Knowledge of MATLAB programming and basic HTML are the only requirements, Mathworks [7].

B. Web Deployment of MATLAB Applications

Going forward, there are several options for making MATLAB applications available via the Web. Each of the cases below contains brief describe the approach to create the web application.

It is possible to choose to create Web applications that access MATLAB directly (hosting MATLAB on the application server), or by building executables or components using the MATLAB deployment products (MATLAB Compiler, MATLAB Builder JA and MATLAB Builder NE). Deployed applications do not require MATLAB, and may be deployed royalty-free to as many servers as user wish [8].

The table 1 lists recommended deployment options for each

Martin Sysel works at the Tomas Bata University in Zlín, Faculty of Applied Informatics, Department of Computer and Communication Systems. Address: nám. T. G. Masaryka 5555, 760 01 Zlín, Czech Republic, (corresponding author to provide phone: 420-57-603-5180; e-mail: Sysel@fai.utb.cz).

Michal Vaclavsky collaborates with the the Tomas Bata University in Zlín, Department of Computer and Communication Systems.

platform, but other options are possible.

Platform	*Application Server Does Not have MATLAB*	*Application Server Does have MATLAB*
Windows	MATLAB Builder NE MATLAB Builder JA	MATLAB COM Automation
Linux Unix	MATLAB Builder JA	CGI with MATLAB as an Executable
MAC	CGI with an Executable Built by MATLAB Compiler	CGI with MATLAB as an Executable

Table 1. Web deployment options

1) MATLAB Builder NE

This part describes using MATLAB Builder NE to create a server-side COM or .NET component. Builder NE requires MATLAB, the MATLAB Compiler. The code using .NET or COM components from Builder NE with ASP.NET requires Internet Information Server (IIS).

Here are some pointers for ASP.NET deployment, particularly for transitioning from MATLAB Web Server [8]:

• In M-code, it needs to use the print command to save the figure to a JPEG format as opposed to wsprintjpeg.

• It should be ensured that IIS points to the correct version of ASP.NET (as selected during compilation). After publishing the ASP.NET application to the web using the Publish to Web option in Visual Studio. If this step is not performed, the application displays an error pointing to the web.config file (depends on application).

• MATLAB Web Server HTML tags needed modification for ASP.NET/design mode compatibility: tags caused errors for design mode, and the page would not display in this mode. Errors were very helpful in providing guidance on needed changes. It is required to convert HTML controls to ASP control format (drop-down menus, text fields, etc.).

• If prohibiting client write access for the virtual directory containing the ASP.NET web application, pre-extraction of the CTF-archive file is necessary. Component Technology File (CTF). This file is independent of the final target type (standalone application or library) and specific to each operating system platform. This file, which is named with a .ctf suffix, contains the MATLAB functions and data that define the application or library and is embedded in the binaries of shared libraries and standalones by default.

2) MATLAB Builder JA

MATLAB Builder JA allows integrate MATLAB applications into the organization's Java programs by creating MATLAB based Java classes that can be deployed royalty free on desktop machines or Web servers [8]. The MATLAB Builder JA product creates the Java classes by encrypting MATLAB functions and generating a Java wrapper around them.

It can be referenced MATLAB based Java classes the same way as any other Java class, for easy integration into servlets or JavaServer Pages (JSP). The Java classes created by the builder run against the MATLAB Compiler Runtime (MCR), which is the full set of shared libraries that support MATLAB. The MCR is provided with MATLAB Compiler for distribution with the Java classes. Both the Java classes created in MATLAB and the MCR can be deployed royalty free.

For Web-based applications, the builder provides AJAX based zoom, pan, and rotate controls for figures created in MATLAB. The builder also provides an API for automatically converting between Java and MATLAB data types. Using the builder's Remote Method Invocation (RMI) interface, the class can also be run as a persistent service, or spread processing across multiple processes [8]. Java classes created with the builder can be distributed at no additional charge.

3) Calling MATLAB functions via CGI

MATLAB can be called directly as an executable via CGI, or executables created with the MATLAB Compiler can be created and deployed. Using MATLAB Compiler it can be created a shared library which can be called from an executable written in C. This executable can be called by web server using the Common Gateway Interface (CGI). Using this solution requires the following steps [8].

• Compile the M-file into a shared library. (mcc B csharedlib:mylib myfile.m)

• Compile the C-file into an EXE-file which links to the created library. (mbuild mycfile.c mylib.lib)

• Copy the DLL, EXE, and CTF files to the web server.

• Install the MCR on the web server.

4) Direct access from the Web, using COM automation

It is possible to call MATLAB through its COM automation interface, similarly to how would be called a COM object from MATLAB Builder NE. Then it is necessary wrap the MATLAB COM automation interface as a Web service.

Note that the MathWorks Software License Agreement specifically forbids exposing the MATLAB command line, i.e. allowing the end users of the Web site to define the XML string to execute. Instead, this approach must be used in the context of an application, executing specific user created M functions.

C. Problems and disadvantages

The problem of the above mentioned solutions lies in executing Simulink schema, because only final results are provided. The problem with running results is very serious, because the simulation of technological processes or real measurements can take a lot of time, especially in chemical processes. To avoid the problem with no access to running results, the development of a new application called SimWebLink can help to solve this disadvantage.

II. APPLICATION STRUCTURE

The whole system consists of four parts: Simulink Block, Server, Client and web interface. Server part was originally written in C + + and operated under Linux. Originally used database was MySQL. The web part was in the spirit of open source written in PHP. Client part was intended for end users and thus worked under Windows; this was created in the programming language C++.

These parts have been redesigned with the use of technologies .NET from Microsoft [9] and by using modern techniques such as WCF (Windows Communication Foundation) [9], [10]. As a database server MSSQL server was used. The web interface uses ASP.NET.

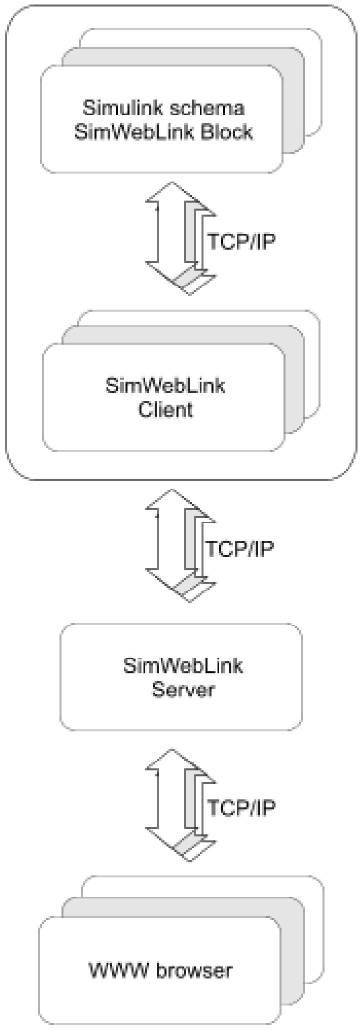


Fig. 1: application structure

A. Application Structure

The entire application is divided into logical units. SimWebLink block embedded in the Simulink block diagram sends data to the client that runs as a service on the same computer. Client connected to the Matlab engine controls Simulink scheme and provide communication with the server. Server listens on TCP port, and if any of the clients is connected, receive from the client the configuration and waits for further instructions. Control of the clients is provided via the web interface via server part. Data obtained from clients is stored in a database and can be plotted in real time in the web interface in the form of graphs. Scheme of application structure is shown in the figure 1. The whole conception is based on the idea that it is possible to observe running simulation or real measurement results in the web browser.

B. SimWebLink Block

This chapter contains simplified description of the source code of the developed Simulink block. The TCP/IP client block sends out data from model using the TCP/IP protocol. This data is sent at fixed intervals during a simulation. The TCP/IP client block has one input port. The size of the input port is dynamic, and is inherited from the driving block. This block has no output ports. The developed TCP/IP output block is shown in the figure 2.

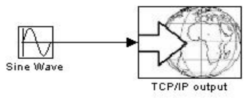


Fig. 2. The TCP/IP Client Block.

The Sink Block Parameters dialog box can be used for selecting communication parameters (Figure 3).

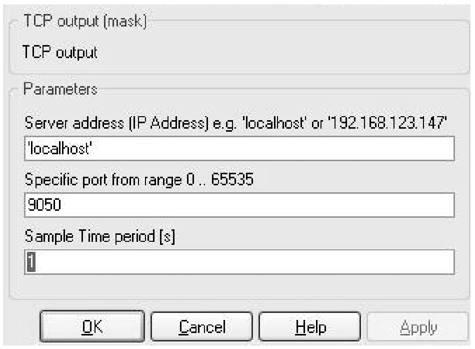

TCP output (mask)
TCP output
Parameters
Server address (IP Address) e.g. 'localhost' or '192.168.123.147'
'localhost'
Specific port from range 0 .. 65535
9050
Sample Time period [s]
1

Fig. 3. The TCP/IP Client Block Parameters dialog box.

It is possible to specify a remote server address, port and sample time period. The sample time period is the rate at which the block send the data to specified port on the server during the simulation.

The base element of the block is S-function block, which use C MEX file. Finally, it is necessary compile source code. Compiling the MEX-Files is similar to compiling with gcc or any other command line compiler.

S-functions (system-functions) provide a powerful mechanism for extending the capabilities of the Simulink environment. An S-function is a computer language description of a Simulink block written in MATLAB, C, C++, Ada, or Fortran. S-functions are dynamically linked subroutines that the MATLAB interpreter can automatically load and execute. S-functions use a special calling syntax called the S-function API that enables to interact with the Simulink engine. By following a set of simple rules, it can be implemented an algorithm in an S-function and used the S-Function block to add it to a Simulink model.

Execution of a Simulink model proceeds in stages [12], [13]. First comes the initialization phase. In this phase, the

Simulink engine incorporates library blocks into the model, propagates signal widths, data types, and sample times, evaluates block parameters, determines block execution order, and allocates memory. The engine then enters a simulation loop, where each pass through the loop is referred to as a simulation step. During each simulation step, the engine executes each block in the model in the order determined during initialization. For each block, the engine invokes functions that compute the block states, derivatives, and outputs for the current sample time. The entire simulation loop then continues until the simulation is complete.

A MEX S-function consists of a set of callback methods that the Simulink engine invokes to perform various block related tasks during a simulation. Because the engine invokes the functions directly, MEX S-functions must follow standard naming conventions specified by the S-function API.

MEX S-functions provide many sample time options, which allow for a high degree of flexibility in specifying when an S-function executes. If the behavior of S-function is a function of discrete time intervals, it can be defined a sample time to control when the Simulink engine calls the S-function mdlOutput and mdlUpdate.

Traditional network programming implemented in Windows environment uses Windows Sockets API (Winsock API - WSA). WSA is similar to Linux Sockets programming with a few exception such as header files, that provided to suit Windows environment and enhances the functionalities. Windows Sockets 2 (Winsock) enables programmers to create advanced Internet, intranet, and other network capable applications to transmit application data across the wire, independent of the network protocol being used. With Winsock, programmers are provided access to advanced Microsoft Windows networking capabilities. Winsock programming previously centered around TCP/IP. [14], [15].

There are two distinct types of socket network applications: Server and Client. Servers and Clients have different behaviors; therefore, the process of creating them is different. The developed Simulink block is a client.

Following MATLAB command links the object code together with the library WS2_32.lib (or it is possible to use older library wsock32.lib) [14], [15]. Win32 architecture is supposed.

```
>> mex -O client.cpp WS2_32.lib  -DWIN32
```

1) Defines and Includes

The S-function code starts with the define and include statements. It is necessary to define statement which specifies the name of the S function.

After defining these two items, the code includes simstruc.h, which is a header file that gives access to the SimStruct data structure and the MATLAB Application Program Interface (API) functions. The simstruc.h file defines a data structure, called the SimStruct, which the Simulink engine uses to maintain information about the S-function. The simstruc.h file also defines macros that enable MEX-file to set values in and get values (such as the input and output signal to the block) from the SimStruct. The winsock2.h and ws2tcpip.h should be added to access sockets under Microsoft Windows. Next parts describe callback method implementations.

2) mdlInitializeSizes

The Simulink engine calls mdlInitializeSizes to inquire about the number of input and output ports, sizes of the ports, and any other information (such as the number of states) needed by the S-function.

The client implementation of mdlInitializeSizes specifies the following size information:

```
ssSetNumSFcnParams(S, 3);
```

It defines three input parameters:

- Address – (String input parameter) Name address of the server (IP address) - An Internet Protocol (IP) address is a numerical identification that is assigned to devices participating in a computer network.
- Port – (Integer input parameter) - In computer networking, a port is an application-specific or process-specific software construct serving as a communications endpoint used by Transport Layer protocols of the Internet Protocol Suite such as Transmission Control Protocol (TCP) or User Datagram Protocol (UDP). A specific port is identified by its number associated with the IP and the protocol used for communication.
- Sample time period – Sampling period of the signal output.

```
if (!ssSetNumInputPorts(S, 1)) return;
ssSetInputPortWidth(S,0,DYNAMICALLY_SIZED);
```

It defines one dynamically sized input port, that’s why TCP output is in the special format which is easy modifiable.

```
ssSetOptions(S,
SS_OPTION_WORKS_WITH_CODE_REUSE          |
SS_OPTION_EXCEPTION_FREE_CODE            |
SS_OPTION_USE_TLC_WITH_ACCELERATOR);
```

Specifying these options together with exception-free code speeds up execution of S-function.

It defines one dynamically sized input port, that’s why TCP output is in the special format which is easy modifiable.

3) mdlInitializeSizes

The Simulink engine calls mdlInitializeSampleTimes to set the sample times of the S-function. A client block executes in specified period (the third input parameter).

```
ssSetSampleTime(S,0,       mxGetScalar(ssGetSFcnParam(S,
2)));
```

4) *mdlStart*

Simulink invokes this optional method at the beginning of a simulation. It should initialize and connect the windows socket. Input parameters Address and Port are used, TCP communication is used here.

```
WSAStartup(MAKEWORD(2,2), &wsaData);
ConnectSocket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
getaddrinfo(myHost, port, &hints, &result);
connect( ConnectSocket, result->ai_addr, (int)result->ai_addrlen);
```

5) *mdlOutputs*

The engine calls mdlOutputs at each time step to calculate the block outputs. The client implementation of mdlOutputs takes the input signal and writes the data to the created output socket.

```
send(mySocket, data, strlen(data), 0);
```

6) *mdlTerminate*

The engine calls mdlTerminate to provide the S function with an opportunity to perform tasks at the end of the simulation. This is a mandatory S function routine. The client S-function terminate created socket.

```
shutdown(ConnectSocket, SD_SEND);
closesocket(ConnectSocket);
WSACleanup();
```

C. *SimWebLink Client*

Part SimWebLink Client have to be run on the computer where is accessible the program Matlab/Simulink and laboratory task is attached. The main objectives of this part of the application can be described in four points:

- Contact SimWebLink Server after start and announce readiness to work.
- Expect control commands from SimWebLink Server.
- In the event that the command comes, SimWebLink Client runs Simulink schema, configure parameters and runs the Matlab engine.
- Expect the incoming data from the SimWebLink Block and then immediately forward them to the SimWebLink Server.

This part of the application is written as a multithreaded program. This layer of the entire application is required just only for remote control and has the benefit of placing measuring computer in the private network; the public IP address is not required. In the case of monitoring, TCP/IP Output Block can send data directly to the SimWebLink Server or to another application (or device). Client part is written by .NET technology [9], [11].

D. *SimWebLink Server*

Server part was originally programmed in C++ and provided on the Linux platform. The current solution uses the Microsoft .NET framework and is provided on Windows platform. Server part was compiled as a WinForm application to debug an application and reference dump. In real traffic can be run as a Windows service. Communication between server and client part is realized by WCF.

1) *Server Functionality Description*

- After the Server starts, the application sets the maximum possible number of connected clients using server configuration file. The initial counter number of connected clients is set to 0
- Server socket starts and listens on port settings and waits for client connection.
- The client connects to the server socket and is stored in the list of connected clients. Client identification is realized by UID. Increments the counter value of the connected clients. It also sends data to a WCF service, which maintains a list of clients to use in part SimWebLinkWeb.
- Connected client sends configuration of the task to the server. After the correct reception of the XML configuration, the configuration is forwarded to a SimWebLinkServerCore where are available for part of the application SimWebLinkWeb.
- The server waits for further commands from the SimWebLinkWeb or receives data from SimWebLinkClient, which subsequently saves the database.

Sample of XML SimWebLink Server configuration:

```
<userSettings>
<SimWebLinkServer.Properties.Settings>
<setting name=”MaxClientSockets” serializeAs=”String”>
<value>10</value>
</setting>
<setting name=”DbServerName” serializeAs=”String”>
<value>MY-PC\SQLEXPRESS</value>
</setting>
</SimWebLinkServer.Properties.Settings>
</userSettings>
```

2) *Client/Server Data Transferring*

Transferring data between client and server is implemented using XML messages. This method was already used in the previous version and was maintained. The advantage of this method of communication is that it is used by all parts of the application (client, server, web and WCF service) without problems and also it is suitable for debugging, because this format is easy to read.

3) *SimWebLinkCore*

WCF service The previous solution used the direct use of sockets from the web interface. WCF service allows dividing the application into separate logical units. As already described, this service may be linked directly into an application and can be run as a system service, or calls via IIS. Selection of the best solution depends on the location of application components (the network topology and location of each client). In this case, for debugging purposes, the service is encapsulated into the console application. For real-life is preferable running as Windows service or linking DLL directly into the server application.

4) SimWebLinkServerCore Class Methods

This class is the basis of the WCF service, which provides an interface between individual parts of the application. The methods are accessible from the server and the Web Part.

- AddClient() - method adds an item of type Task to the list of Tasks. Accepts parameters - IP address, port, and client identification (CID). Before adding the client is verified that the list of clients has clients with this CID does not exist. If the client exists, the method returns false and no record is added. The return value is a logical (true - the client is added, false - the client is not added - probably exists in the list)
- DeleteClient() - method delete an item from the list Tasks. Accepts parameters - CID and compare items from the list by CID. The method has no return value.
- Echo() - method for testing of service. Accepts parameter of type string. The return value is a string and returns the same value as was given in the input parameter.
- FindClient() - private (non-public) method, which looks for the client in the list. The input parameter is a string (CID - client identification). The return value is of type Task if the client exists; if the client is not found, method returns a null value (null).
- GetClientCount() - method for detecting the number of connected clients. It has no input parameters. The return value is of type integer.
- GetClientList() - method for returning a list of connected clients. The return value is of type String array. It returns list of the individual connected clients with CID.
- GetConfiguration() - method for returning the selected client configuration. The input parameter is a string (CID client identification). The method uses private methods FindClient(). The return value is of type string. Returns the XML configuration obtained from client. Server then forwards the XML to the WCF service.
- GetData() - method for returning the measured data, which sent klient. Input parameter is a string (CID client identification). The method uses private methods FindClient(). The return value is of type string. Returns the XML data, which are obtained from the client. Server then forwards the value to the WCF service.
- GetIdentifier() - method for returning identifiers (names) of clients . Input parameter is a string (CID client identification). The return value is of type string.
- SetConfiguration() - method for task configuration setting up of the selected client. Input parameters are of type string, the first parameter specifies the client identification (CID) and the second parameter is the XML configuration. The method has no return value.
- SetData() - method for setting up the value of the XML data of the selected client. Input parameters are of type string, the first parameter specifies the client identification (CID) and the second parameter is the XML data. The procedure has no return value.
- StartMeasure() - method for starting up the task of the selected client. The input parameter is a string (CID client identification). The return value is a logical (true - the task was started, false - the task is not running).

5) Task class description

This class provides passing parameters between individual modules. Each client has its own instance of this class. Recognition of individual clients is done via a unique identifier (CID).

- ClientID - type of string, represents the unique identification of the connected client. The value is generated by SimWebLinkServer.
- Identifier – type of string – task name obtained from the connected client.
- IPAddress – type of string - IP address of connected client.
- Port - type of string - port of connected client.

To operate the commands sent by a web application is used a queue. It was designed as a simple structure that contains the client's CID and XML message.

E. Database

The previous database was implemented in the MySQL database engine. For compatibility reasons, consistency, and system tuning options was chosen database engine from Microsoft [8]. For the operation of system will suffice SQL Server Express edition.

Connecting to a database uses more application modules. It was therefore necessary to resolve how to set the parameters. The changes should take effect for all modules, which have used. The connection information is set up by using SimWebLinkServer and hand over to the WCF service that is part of the application SimWebLinkServerCore. This information is loaded into the web application part (SimWebLinkWeb).

F. SimWebLink Web Interface

Web interface was realized using Microsoft ASP.NET [9]. Data communication between ASP.NET and server part is done via WCF. The content of each page is dynamically generated from XML data, which provides a WCF service. Sample of XML configuration (shortened):

```
<?xml version=”1.0” ?>
<SimWeblinkWeb>
 <Simulink id=” schema_socketout ”>
   <block id=”WWWout”>
     <name>Perioda vzorkovani</name>
     < variable >parameters</variable>
     <value>10</value>
   </ block>

 <block>
   <name>Doba mereni</name>
   <variable>stoptime</variable>
   <value>20</ value>
```

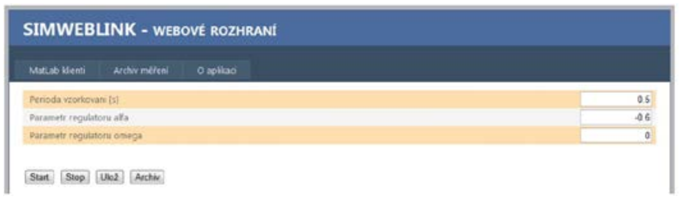
```
  </ block>

  <commands>
    <block>
      <variable>SimulationCommand</variable>
      <value>START</value>
    </block>
  </commands>
</SimWeblinkWeb>
```

1) Web Part Description

After launching a web browser and entered the correct address is displayed Title page of ASP.NET web application. Pages are generated dynamically according to the data retrieved from a WCF service. If the WCF service is not running or is not available for any other reason, the application detects the number of connected clients as a value of 0.

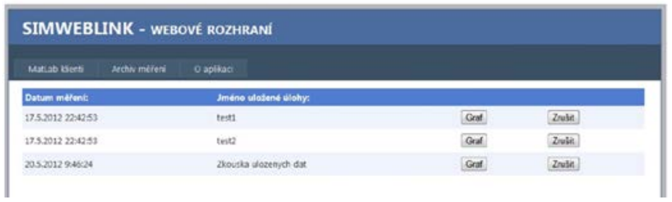
On the Fig. 4 is displayed automatically generated output from configuration. It is possible set-up initial variables.



Perioda vzorkovani [s]	0.5
Parametr regulatoru alfa	-0.6
Parametr regulatoru omega	0

Fig. 4. Setting-up initial parameters.

The measurement can be run by button start. The progress of measurement is continuously updated in the graph. Communication with the Web application part does not work in both directions. This means that the information needs in a certain time interval to reload on the page. Thanks static class SimWebLinkCore in the Web part, application reads the current status of the connected client and displayed information as a web page. Fig. 5 shows a list of archived jobs stored in the database. Fig. 6 shows final graph loaded from the database archive.



Datum měření:	Jméno uložené úlohy:		
17.5.2012 22:42:53	test1	Graf	Zrušit
17.5.2012 22:42:53	test2	Graf	Zrušit
20.5.2012 9:46:24	Zkouska ulozenych dat	Graf	Zrušit

Fig. 5. List of archived jobs in the database

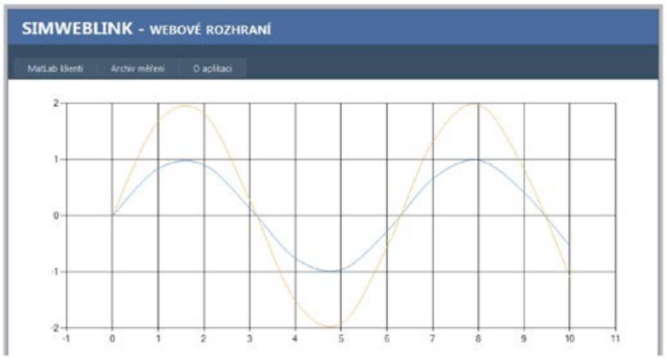


Fig. 6. Final graph loaded from the database

III. Instalation and Running App

Installation pack has been created by Inno Setup [16], which is distributed under the GPL license. The installation was created in version 5.4.3, which can be downloaded from [6]. The installation is divided into 2 separate parts. The first part installs the application SimWebLinkServer and WCF service (SimWebLinkerverCore). To install this applications part is required installed and configured of SQL Server database. Second part of the application then installs the ASP.NET application SimWebLinkWeb for access via web interface. This requires correctly installed Microsoft Internet Information Services (IIS). Described application SimWebLink.Net could be downloaded from [17].

IV. Conclusion

An emerging strategy for application software is to provide web access to software previously distributed as local applications. It requires the development of an entirely different browser-based interface and adapting an existing application to use different presentation technology. This paper describes options for making MATLAB/Simulink applications available via the Web and the new developed web application called SimWebLink. SimWebLink extends computation possibilities of the MATLAB/Simulink. The application SimWebLink is a multithreaded and a multiplatform application.

The whole conception is based on the idea that it is possible to observe running simulation or real measurement results in the web browser. The data presentation lag is minimal and SimWebLink Block does not affect Simulink schema runtime too much. The main advantage is a remote control and monitoring from any web browser in the Internet. The presented data are stored in the database for the later uses.

SimWebLink is suitable for a remote control and monitoring of the technological process uses MATLAB/Simulink via Internet. The development of the application still continues and new operating functions will be implemented.

Developed Simulink TCP/IP Output Block can be used in Simulink models to communicate with others applications and devices over TCP/IP network.

REFERENCES

[1] M. Sysel, MATLAB/Simulink TCP/IP Communication. In *Proceedings of the 15th WSEAS International Conference on Computers*. Corfu Island, Greece : WSEAS Press, 2011. s. 71-75.

[2] Sysel, M., Remote Control and Monitoring In the Simulink. In *Proceedings of the 21st International DAAAM Symposium "Intelligent Manufacturing & Automation: Focus on Interdisciplinary Solutions"*. Vienna : DAAAM International Vienna, 2010, pp. 205-206.

[3] M. Sysel, SimWebLink.NET - Remote Control and Monitoring in the Simulink. In *Proceedings of the 11th WSEAS International Conference on Data Networks, Communications, Computers*. Malta, Sliema : WSEAS Press, 2012. s. 58-62.

[4] Václavský, M., *Rozhraní pro vzdálenou správu laboratorních úloh*. UTB ve Zlíně, 2012.

[5] M. Matýsek, M. Adámek, P. Neumann, T. Matulík, The mobile ordering system with the PDA. In *Recent Researches in Automatic Control. Montreux* : WSEAS Press, 2011, s. 268-271. ISBN 978-1-61804-004-6.

[6] M. Matýsek, M. Adámek, P. Neumann, T. Karafiát, The mobile monitoring and controlling of real systems via the GSM. In *Proceedings of the 15th WSEAS International conference on Communication (Part of the 15th WSEAS CSCC Multiconference)*. Rhodes : WSEAS Press (GR), 2011, s. 143-146.

[7] J. Vojtesek,P. Dostal, Use of MATLAB environment for simulation and control of CSTR. *International Journal of Mathematics and Computers in Simulation* Volume 5, Issue 6, 2011, Pages 528-535

[8] Mathworks, Matlab [online]. 2012 [cit. 2012-05-17]. Available from: http://www.mathworks.com

[9] Microsoft, MSDN [online], 2012 [cit. 2012-05-17]. Available from: www.msdn.com

[10] WCF, www.vyvojar.cz [online]. 2007 [cit. 2012-05-17]. Available from: http://www.vyvojar.cz/Articles/454-wcf-zakladne-pojmy.aspx

[11] E. Ruffaldi, 1. .2. .3 ways of integrating MATLAB with the .NET. [online]. 2003-11-19 [cit. 2012-02-02]. Available from: http://www.codeproject.com/Articles/5468/1-2-3-ways-of-integrating-MATLAB-with-the-NET

[12] Matlab Inc., MATLAB C and Fortran API reference. The Mathworks Inc., Natick, USA, 2008

[13] Matlab Inc., Writing S-Functions. The Mathworks Inc., Natick, USA, 2008.

[14] J. Nair, Asynchronous Socket Programming in C#: Part I: Client-Server Example with Multiple Simultaneous Clients, Available from: http://www.codeguru.com/csharp/csharp/cs_misc/sampleprograms/article.php/c7695, Accesed: 2009-05-03, 2005.

[15] Nair, J. Asynchronous Socket Programming in C#: Part II : An advanced C# socket program example with a single server and multiple simultaneous clients, Available from: http://www.codeguru.com/csharp/csharp/cs_network/sockets/article.php/, Accesed: 2009-05-03, 2005.

[16] Inno Setup, www.jrsoftware.org [online]. 2011 [cit. 2012-05-17]. Available from: http://www.jrsoftware.org

[17] M. Sysel, TBU in Zlin, SimWebLink.NET [online] Available from: http://terra.utb.cz/SimWebLink

Martin Sysel was born in Zlin, Czech Republic in 1975 and studied at the Brno University of Technology, Facultz of Technology in Zlin. where he got his master degree in Automation and control technology in 1998. He has finished his Ph.D. focused on Technical Cybernetics in 2001. He now works as a associate professor at Department of Computer and Communication Systems, Faculty of Applied Informatics, Tomas Bata University in Zlin. His research interest is remote simulation and control of technological processes.