Transactions of the VŠB – Technical University of Ostrava, Mechanical Series
No. 2, 2015, vol. LXI
article No. 2002

Karla SLADKÁ[*], Marek BABIUCH[**]

.NET GADGETEER APPLICATION DEVELOPMENT USING WIRELESS COMMUNICATIONS

VÝVOJ APLIKACÍ NA PLATFORMĚ .NET GADGETEER S VYUŽITÍM BEZDRÁTOVÉ KOMUNIKACE

Abstract

The article describes the application development on Microsoft platform .NET Gadgeteer, which is part of the .NET Micro Framework. Application development is introduced to applications using ethernet network communication and wireless communication between developments boards connected with the sensor modules.

Abstrakt

Článek popisuje vývoj aplikací na platformě .NET Gadgeteer společnosti Microsoft, která je součástí technologie .NET Micro Framework. Vývoj aplikací je představen na aplikacích využívající síťovou ethernet komunikaci a také bezdrátovou komunikaci mezi vývojovými deskami s připojenými senzorovými moduly.

Keywords

.NET, application, development, device, Gadgeteer, Micro Framework

1 INTRODUCTION

.NET Gadgeteer is technology, which is based on the term gadgeteering. This term indicates the close connection between hardware and software. The idea of .NET Gadgeteer is design the technology of electronic devices for easy implementation and handling especially for smart home automation applications. .NET Gadgeteer was created with support of Microsoft Company. In these collaboration libraries for developer an environment Visual Studio .NET, drivers and graphics library has been developed. Software platform of .NET Gadgeteer is based on .NET Micro Framework (NETMF). It allows create an application on level of objective oriented programing in one of the .NET programming languages. NETMF library are freely available code and are therefore seen as open-source. Hardware part of .NET Gadgeteer technology has two main parts. First part is development mainboard with CPU, memory, power management and I/O sockets. The second part is represented by modules, which are connected by standardized interface. The modules could be sensors, communication devices, displays, etc. The idea of the hardware is the same like with software to be able to modify and add its own components. .NET Gadgeteer Technology has its advantages and one of them is wide support to easy apply new applications source code at C#

[*] Ing., VŠB – Technical University of Ostrava, Faculty of Mechanical Engineering, Department of Control Systems and Instrumentation, 17. listopadu 15, Ostrava - Poruba, 708 33, Czech Republic, tel. (+420) 59 732 4380, e-mail: karla.sladka@vsb.cz

[**] Ing., Ph.D., VŠB – Technical University of Ostrava, Faculty of Mechanical Engineering, Department of Control Systems and Instrumentation, 17. listopadu 15, Ostrava - Poruba, 708 33, Czech Republic, tel. (+420) 59 732 4173, e-mail: marek.babiuch@vsb.cz

language for all mainboard and modules. Other advantages are also high compatibilities with web and desktop applications, communication through web services with mobile devices and embedded systems.

2 .NET MICRO FRAMEWORK TECHNOLOGY AND FEZ PLATFORM

.NET Micro Framework technology was developed for applications with embedded systems, which are usually with small RAM and program memory. .NET Micro Framework (next just NETMF) is small and efficient technology, which is used to develop application in visual studio .NET environment, where is possible to develop application for microprocessor with the same tools and high-level programming languages as well as developing desktop, mobile and smart phone application. Applications of NETMF technology are working with two important facilities PAL and HAL, which are enabled to communicate with drivers of hardware components. Uploading the application from development environment to the device is easy and automatized. Hardware emulator, which enables to communicate and debugging the application is included in the development environment called visual studio and the application is send to the device during compilations.

Architecture of NETMF consists of four layers. First layer is called hardware layer and includes microprocessor and other electronics components. Second layer is runtime component layer, where drivers, PAL, HAL and others facilities are included. The layer provides applications system management of memory, threads and running processes. Superior layer of libraries contain NETMF objective oriented collections of function and objects. This layer allows encryption, graphics components and contains basic libraries for communication standards. The highest layer is application layer, which take care about created application to run and debug them.

.NET Gadgeteer hardware components are electronics devices mainly designed by FEZ Technology. Main idea of the technology had created electronics components without external power supply, which are easy assembled and has possibility of the same programming ways in C# language. FEZ technology is based on reliability and compatibility with robust object programming methods and event handlers.

Technology research FEZ is connected with NETMF and software facilities of TinyCRL for application running. Other idea is allowed to fast and easy developing of application for smart home automation projects. It results in creation of drivers for every components of FEZ Technology, which are situated in SDK of NETMF and are part of facility HAL. The drivers ensure better compatibilities of FEZ components with software, when application is applied to hardware mainboard and modules.

Platform of .NET Gadgeteer is based on fast and simple collaboration between hardware components. The components are connected on plug and play interface and application are uploaded without used programmer or reset device. Project is consisted of hardware part and software part. Hardware part contains developed mainboard, integrated modules and possible external modules. Software part comprises application, which is based on .NET Micro Framework, libraries and designer.

Technology of .NET Gadgeteer has a diverse range of external modules, which is extends and renews. It is giving more opportunities for creating electronics projects and their developing in software and hardware parts. Kinds of the external modules are sensors, communication modules, control modules, memories modules, modules for memory card, RFID modules, optical modules, displays, drivers and other.

On the market are several kind of mainboard of .NET Gadgeteer, which are different in integrated components, controls modules, memories, slots for external modules and integrated modules. For the application in our contribution was chosen three kind of mainboard, which are Cerbuino Net, FEZ Spider and FEZ Raptor.

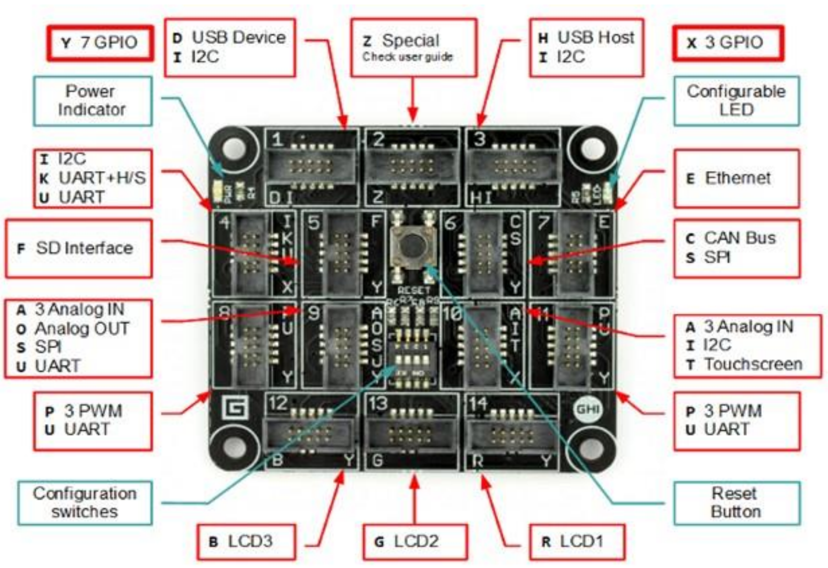


Fig. 1 FEZ Spider development board [3]

FEZ Spider is based on the EMX System on Module with 72MHz 32-bit ARM7 processor, RAM memory is 11 MB and on the board are 14 sockets for external modules. This board is included in the FEZ Spider Starter Kit. The mainboard support network interface, files system and graphics interface. Currently most powerful FEZ mainboard is FEZ Raptor development board. This board contain 18 slots for external modules. FEZ Raptor has control system with processor 400MHz 32-bit ARM9 and RAM memory 92 MB. The mainboard supports network interface, file systems and also graphics interface. [3].

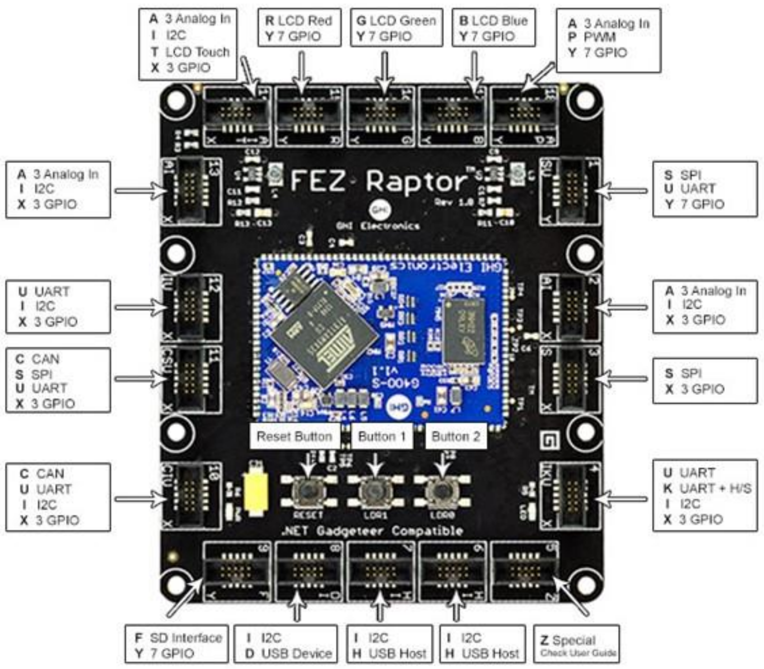


Fig. 2 FEZ Raptor development board [3]

The Third used board is FEZ Cerbuino Net with integrated Ethernet module, which include extended libraries of network protocol TCP/IP and SSL. These libraries allow to easy developing application, which is working with network connection. FEZ Cerbuino Net is based on the family Cerberus control system with processor 168MHz 32-bit Cortex-M4. Mainboard has RAM memory for system 119 KB and for Ethernet 104 KB. Flash memory has 384 KB. [3] Mainboard has slot for SD card, USB host, client and 3 slots for external modules. The advantage of Cerbuino boards is compatibility with Arduino system. This board is used in the following chapter and displayed on Figure 3 below.

3 .NET GADGETEER NETWORK APPLICATIONS

This chapter deals with applications with .Net Gadgeteer mainboards described above, which are connected with gadgeteer sensor modules and units for wired and wireless network communication.

3.1 Application with Ethernet Communication

The figure 3 shows a block diagram of Cerbuino Net boards interconnected with measurement and other modules: barometer (except pressure is also capable of measuring the temperature), light sensor and oled display. Network interface is implemented directly to the development board. The circuit diagram is created within Visual workplace, adequate libraries that include support for the development of C # functions and event handlers are automatically generated by the project.

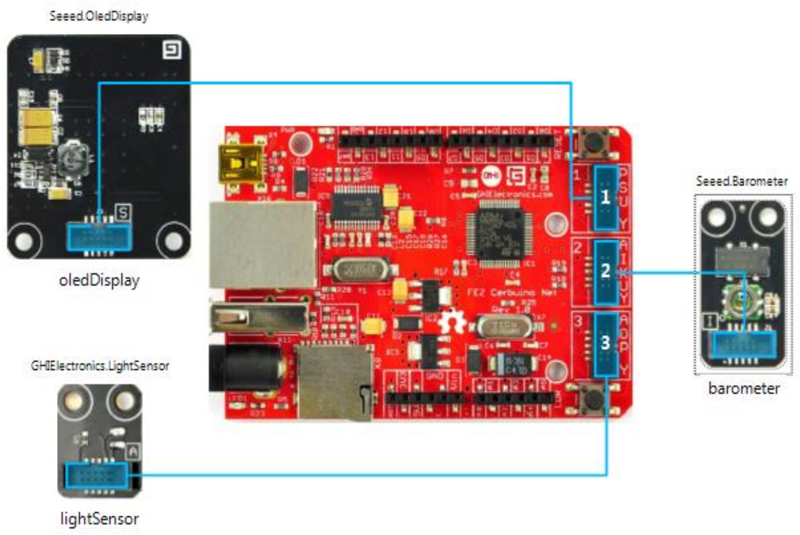


Fig. 3 Block scheme of measurement application with network support

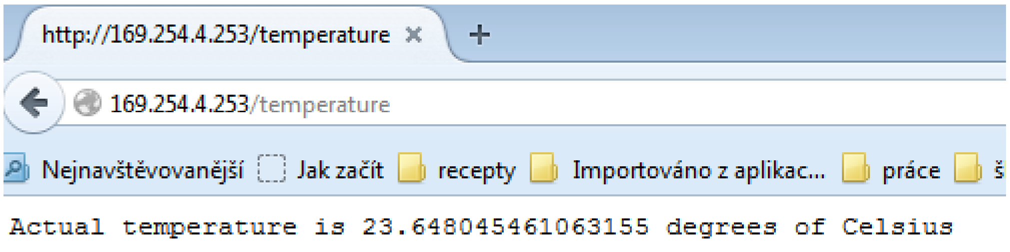
First, we must ensure the declaration of the network interface and define event of network activity, IP address assignment and start the Web server to the relevant IP address and port. We see these functionalities in an abbreviated version of the source code.

```
void ProgramStarted()
        {
            NetworkInterface neti = NetworkInterface.GetAllNetworkInterfaces()[0];
            //EthernetENC28J60 netif;
            NetworkChange.NetworkAvailabilityChanged += new
NetworkAvailabilityChangedEventHandler(NetworkChange_NetworkAvailabilityChanged);
            NetworkChange.NetworkAddressChanged += new
NetworkAddressChangedEventHandler(NetworkChange_NetworkAddressChanged);

            neti.EnableStaticIP("169.254.4.253", "255.255.0.0", "");
            Debug.Print("Program Started");

            WebServer.StartLocalServer("169.254.4.253", 80);
```

The following fragment of source code defines a system timer, which ensures that Gadgeteer sensor module will be measured environments temperature at intervals of two seconds. There is also defined an event handler for Web server request that ensures return of the measured value.

```
            GT.Timer timer = new GT.Timer(2000);
            barometer.MeasurementComplete += new
  Barometer.MeasurementCompleteEventHandler(barometer_MeasurementComplete);
            barometer.StartContinuousMeasurements();

            sendTeplota = WebServer.SetupWebEvent("temperature");
            sendTeplota.WebEventReceived += new
  WebEvent.ReceivedWebEventHandler(sendTeplota_WebEventReceived);
```

Figure 4 shows the measured value from gadgeteer module in web browser of remote connected workstation. We made http request at the internet browser to obtain the value of the ambient environment temperature to a specific IP address of the web server running on the development board.



```
Actual temperature is 23.648045461063155 degrees of Celsius
```

Fig. 4 Web browser window shows measured value from web server on the development board

3.2 Wi-Fi Communication between development boards

Server application creates Wi-Fi network based on IEEE 802.11b/g. Web server, measurement application and data acquisition is implemented on FEZ Spider mainboard, which is using Wi-Fi module to create wireless network. Client applications is represented by powerful FEZ Raptor mainboard, which has implemented source code for displaying information from server sensors on the basis of http request. Block scheme of design of network application is shown on figure 5.

Communication is running through communication layer, where are used functions for http protocol from libraries of Wi-Fi gadgeteer module. The temperature and pressure are measured by barometer module and light intensity is measured by lightSense module. There is used WiFi RS21 gadgeeter module is, for wireless communication between both mainboards. WiFi network is created by Wi-Fi module and network libraries contain function for network configuration. There is necessary to open network interface, set information about network and set IP address of the network in static way or dynamic with DHCP server. Connection to the network is used ad-hoc communication interface for temporary connection between devices, which is used for sharing network connection or files. Data are shown on the client site on the display module.

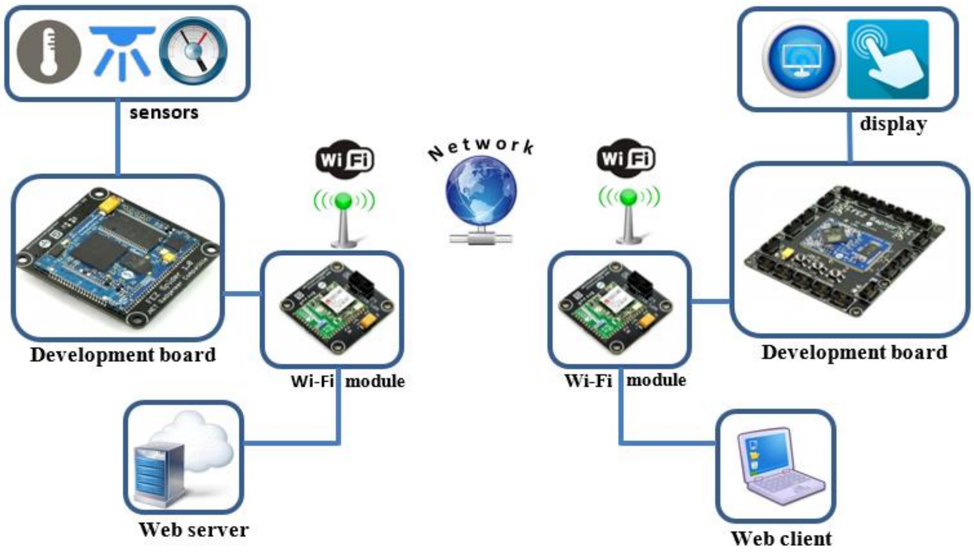


Fig. 5 Wi-Fi Communication between development boards

The solution is created by two applications according web server and client model. Both part contain mainboard, external gadgeteer modules and application with C# source code which is briefly explained in following paragraphs.

- **Creation network and measurement application on web server part**

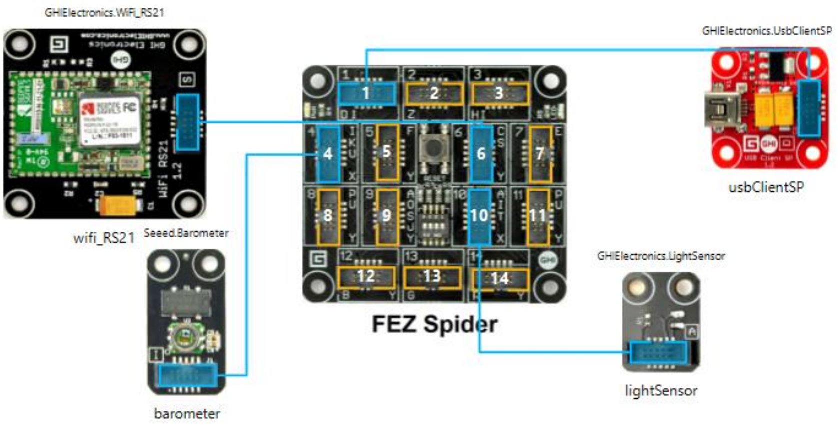


Fig. 6 Develompent board with sensors and gadgeteer wi-fi module

Block scheme from visual Studio.Net workplace is shown on figure 6. Except USB client for power consumption and sensors we also use Wi-Fi gadgeteer module, because development boards have no implementation of on-board wireless network interface.

First easy step is creation of new project in develop environment, where is created hardware connection with mainboard and modules for implementation libraries. Next step is setting timer for calling event, which collect data from measurement sensors.

```
            GT.Timer timer = new GT.Timer(2000);
            barometer.MeasurementComplete +=new
Barometer.MeasurementCompleteEventHandler(barometer_MeasurementComplete);
            barometer.StartContinuousMeasurements();
```

Then follow creation of network connections. Network interface have to be check if is open. We need to ensure settings of network configuration with IP address, network mask, connection and definite SSID, security, password and interface channel.

```
    GHI.Premium.Net.WiFiNetworkInfo info = new GHI.Premium.Net.WiFiNetworkInfo();
                info.SSID = "MyNetwork";

                if (!wifi_RS21.Interface.IsOpen)
                {
                    wifi_RS21.Interface.Open();
                }
                wifi_RS21.UseStaticIP("169.254.4.253", "255.255.0.0", "");
                wifi_RS21.Interface.StartAdHocHost("MyNetwork",
    GHI.Premium.Net.SecurityMode.Open, "12345", 1);

                wifi_RS21.Interface.WirelessConnectivityChanged += new
    WiFiRS9110.WirelessConnectivityChangedEventHandler
    (Interface_WirelessConnectivityChanged);
```

Measured data from sensor are stored to the variables, which are sent after request on web server through the network to web client. For sending these data is necessary make event for running web server and also make event handler for measured data request.

```
            getSensorPersentage = WebServer.SetupWebEvent("SensorPersentage");
            getSensorPersentage.WebEventReceived += new
 WebEvent.ReceivedWebEventHandler(getSensorPersentage_WebEventReceived);

            void Interface_WirelessConnectivityChanged(object sender,
 WiFiRS9110.WirelessConnectivityEventArgs e)
        {
            if (e.IsConnected)
            {
                WebServer.StartLocalServer("169.254.4.253", 8080);
```

- **Web client and display**

Web client is represented by Mainboard FEZ Raptor with external Wi-Fi module for connection to the network and gadgeteer display module for obtaining measured data. Block scheme of developed mainboard and external modules is shown on picture 7.

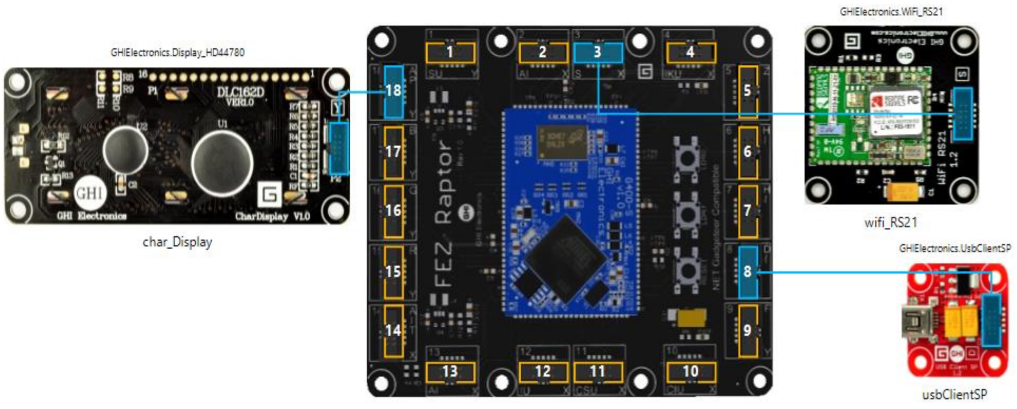


Fig. 7 Raptor mainboard with modules represented client site

First step of the client application is scanning the network and making collection of information about available surrounding networks with a SSID, number of channel, type of network, RSSI and security. Information are stored in object at the array variable and they are shown for debugging in output window of Visual Studio.

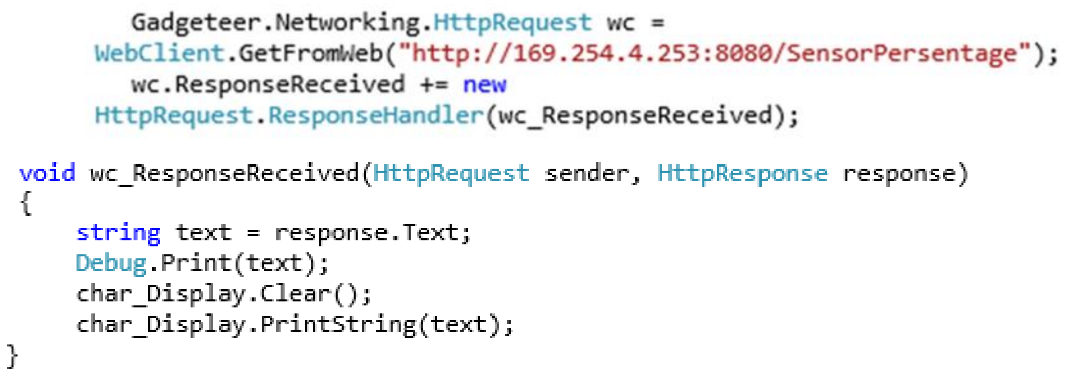
```
            GHI.Premium.Net.WiFiNetworkInfo[] info = null;
            GHI.Premium.Net.WiFiNetworkInfo[] scanResults =
wifi_RS21.Interface.Scan();

            foreach (GHI.Premium.Net.WiFiNetworkInfo result in scanResults)
            {
                Debug.Print("****" + result.SSID + "****");
                Debug.Print("ChannelNumber = " + result.ChannelNumber);
                Debug.Print("networkType = " + result.networkType);
                Debug.Print("RSSI = " + result.RSSI);
                Debug.Print("SecMode = " + result.SecMode);

            }
```

Important action is searching the network and check its availability. If the network is available, mainboard connect through Wi-Fi module the network and set configuration of Wi-Fi module. Configuration of network connection is represented by IP address, network mask and gateway address, which are also printed to output debug window to confirm successful setting.

```
                info = wifi_RS21.Interface.Scan("MyNetwork");
                if (info != null)
                {
                    wifi_RS21.Interface.Join(info[0], "12345");
                    Debug.Print("Network joined");


        wifi_RS21.UseStaticIP("169.254.4.254", "255.255.255.0", "");

         Debug.Print("IP Address: " + wifi_RS21.NetworkSettings.IPAddress);
         Debug.Print(wifi_RS21.Interface.NetworkInterface.GatewayAddress);
```

Now mainboard is connected to the network of the other mainboard. Communication is created by web client, which creates http request to get data from web server by GET functionality.

Request contains IP address of server, port or directory, where data can be stored. Http response with web server sends measured data, which are shown on display module at client application.



```
    Gadgeteer.Networking.HttpRequest wc =
WebClient.GetFromWeb("http://169.254.4.253:8080/SensorPersentage");
    wc.ResponseReceived += new
HttpRequest.ResponseHandler(wc_ResponseReceived);

void wc_ResponseReceived(HttpRequest sender, HttpResponse response)
{
    string text = response.Text;
    Debug.Print(text);
    char_Display.Clear();
    char_Display.PrintString(text);
}
```

4 CONCLUSIONS

The goal of the contribution was introducing .NET Gadgeteer platform as a technology, with their advantages of C# programming, description of the related hardware platform and description of the Micro Framework technology, which .Net Gadgeteer is based on. We describe in this paper 3 types of developed mainboard, which we used at Gadgeteer network applications and internal and external modules. Network applications with the Gadgeteer boards and sensor modules are explained with short details of C# source codes and block schemas at visual studio .NET environment. Contribution describes in detail two applications oriented on the network interface, specifically the application uses the built-in Ethernet module as well as communication between two development boards over the wireless network. In these applications it is also used measuring sensor modules and display units. Contribution demonstrates the principle of the development of .NET Gadgeteer applications on chosen development boards and modules. Among described modules, we use a wide range of Gadgeteer boards and modules as I/O devices such as buttons, joystick, various displays including OLED and touchscreen, RFID devices, SD card storage, various network modules and other sensor modules in the field of smart home automation. This result has been elaborated in the framework of the project No: SP 2015/83 - “New approaches to Machine and Process Control” supported by the Ministry of Education, Youth and Sports.

REFERENCES

[1] Microsoft Developer Network. MICROSOFT. [online]. Available from: http://msdn.microsoft.com/

[2] THOMSON, D. a R. MILES. *Embedded Programming with the Microsoft .NET Micro Framework*. 4. vyd. Microsoft Press, 2007. ISBN 978-0735623651.

[3] GHI Tutorials. GHI ELECTRONICS LLC. *GHI Electronics* [online]. [cit. 2014-07-04]. Available from: www.ghielectronics.com/docs/37/netmf-and-gadgeteer-tutorial-index

[4] MICROSOFT. Microsoft Visual Studio. [online]. Available from: www.microsoft.com /cze/msdn/vstudio/

[5] MONK, Simon. *Getting Started with .NET Gadgeteer*. 1. vyd. O´Reilly Media, 2012. ISBN 978-1-449-32823-8.

[6] KUHNER, Jens. *Expert .NET Micro Framework*. Apress, 2008. ISBN 978-1-59059-973-0. Available from: http://it-ebooks.info/book/2053/

[7] MICROSOFT Micro Framework. [online]. Available from: http://www.netmf.com/