

Wireless Sensor Network protocol developed for microcontroller based Wireless Sensor Units, and data processing with visualization by LabVIEW

Imre Farkas*, Peter Dukan**, Jozsef Katona***, Attila Kovari****

*,***,**** College of Dunaújváros/Institute of Informatics, Dunaújváros, Hungary

** ADMAXON Ltd, Dunaújváros, Hungary

*farkasi@mail.duf.hu, **dukanp@mail.duf.hu, ***katonaj@mail.duf.hu, ****kovari@mail.duf.hu

Abstract – The sensor networks are established by the connection of numerous sensors which are located at different places. In our paper we introduce the general structure of Wireless Sensor Networks as well as different wireless standards and network components. Furthermore, we give an overview about the construction of the Wireless Sensor Unit and the operation of LabVIEW as a web-server and its network setup options. Finally, we highlight the developed Sensor Network Protocol.

I. INTRODUCTION

Sensors are intended for measuring some physical or chemical quantity, but while earlier the results of the measurements were typically detectable by the naked eye (e.g. mercury thermometer), now it is typical that electrical properties change with the quantity measured in practice (current, voltage, or resistance change).

So, sensors are devices that can collect the data from our environment, making the changes observable to us. The main tasks of sensors are controlling a variety of industrial equipment, monitoring of some processes, giving feedback and collecting different data for specific purposes.

Sensor networks are created by linking a large number of sensors which are spatially located in different locations and their communications with each other gives a range of new application areas without using any special hardware components.

In today's rapidly evolving competitive and information technology based-world there is a growing demand for

faster and more effective decision-making in real-time, capturing, interpreting, evaluating, and decision making based on that data. The age of data collection using wired sensors draws to a close, as the low mobility inhibits widespread applicability. High installation and maintenance costs as well as the difficulties related to cabling also make them outdated. Most of these disadvantageous features are avoidable using wireless sensors.

The use of wireless sensors – despite their prevalence – is mostly complicated and difficult. An attempt was made to build a network of easily accessible tools to ease implementation. For the reduction of development time, the real-time data acquisition and processing was done by using LabVIEW blocks. The developed system is easy to upgrade.

A. The general structure of sensor networks

The sensor networks are largely formed in order to analyze an event, phenomenon or device parameters to help influence its course and operation. The components of the network are:

- detector (sensor): its task was explained at the beginning of this chapter;
- converter (in this case a microcontroller): converts an analog signal supplied by the sensor to digital and transmits it to the wireless unit;
- wireless communication unit: the data provided for this unit will be sent in accordance with the data transfer protocol (protocol-kit) via ether, and the sent

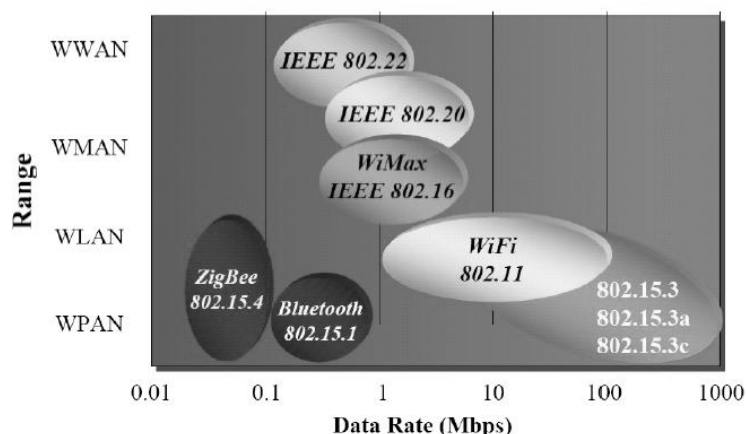


Figure 1. IEEE wireless communication standards [1]

TABLE I.
IEEE 802.11 STANDARDS [3]

Standard	Release (year)	Frequency (GHz)	Data rate (Mbit/s)	Indoor range (m)	Outdoor range (m)
IEEE 802.11	1997	2,4	2	20	100
IEEE 802.11a	1999	5	54	35	120
IEEE 802.11b	1999	2,4	11	38	140
IEEE 802.11g	2003	2,4	54	38	140
IEEE 802.11n	2008	2,4 / 5	248	70	250
IEEE 802.11i	protocol (recommendation) stack to enhance the current and future wireless network security				
IEEE 802.11y	2008	3,7	54	50	5000

data is received by a communicating unit (in our case these features are supplied by WiFi devices);

- the received data is interpreted by a processing unit, and if necessary, it influences the observed event through feedback.

B. Wireless communication facilities

Today, a wide variety of standards are available in the data transfer between components in wireless sensor networks. If we know the conditions of the tasks at hand and the terrain conditions, it is easy to find a sufficient IEEE declared transmission mode [3]. In Figure 1, we see the relationship between these possibilities together with the point of view of the coverage and the data rate.

WiFi stands for Wireless Fidelity, which is one of the most popular technologies built today on the 802.11 standard. To an ordinary observer, WiFi means that the cables disappear around the computer, but in fact it involves more than that. First, we must understand the concept of ISM band (Industrial, Scientific and Medical band) which is throughout most of the world (e.g. in Hungary, but not in Italy) and is a usable frequency band, which is freely available for industrial, scientific and medical purposes. For example, similarly the 2.4-2.5GHz and 5.725 to 5.875GHz ranges are used for data traffic in WiFi technology. The WiFi-enabled devices communicate by radio to other WiFi devices.

II. THE WIRELESS SENSORS UNIT (WSU)

Under the designation of WSU (Wireless Sensor Unit) we refer to the complex unit, which via a wireless connection, can provide the measurement results. The unit consists of three modules: 1) the actual sensor, 2) wireless communication module and 3) a microcontroller performing the conversion between the two (Fig. 2). Analog signals emitted by the sensor to a microcontroller via its analog / digital converter (hereinafter referred to as ADC) are converted into digital values, and then they are sent on the channel provided by the wireless communication module via electromagnetic waves.

A. The construction of the WSU

One of the requirements of the construction was cost feasibility, since a sensor network can consist of several tens or hundreds of sensors. For the prototype system, the

temperature sensing construction was chosen, but the system is adaptable to any sensor type, which outputs an analog voltage. For the wireless connectivity a WiFi connection was selected according to the IEEE 802.11 standard because it can be easily matched to the computer. Since we have found a development kit which can fulfil all these requirements, the WSU was achieved by an Explorer 16 programmable development kit manufactured by Microchip.

The main aspect of the kit is a development board, which includes an integrated TC1047A-type temperature / voltage converter. This is fixedly connected to the card (PIM Installation Processor Module) interface, where a PIC24FJ128GA010 is connected. The board provides an interface to the basic I / O functions (buttons, LEDs, SPI, UART, LCD, EEPROM) and contains the important PICtail™ Plus interface, which is capable of accommodating ZeroG's ZG2100 WiFi module. [5]

The family includes the PIC24FJ 16bit (this applies to the data bus, the address bus is 24-bit) and 31kHz-8MHz microcontrollers. The model we use includes 128KB program memory and has 100 pins. The PIM is to provide 100 feet for these 100 pins, which makes easier to insert the developer board. During the developing process, the ability of ADC was also of great importance, which provided 16 channels at 10bit resolution. The conversion is done by 500kS/s, which can be useful for tasks where you need to respond to perceived changes in the sensor's output in 2 millionth of a second, but the bottlenecks of wireless sensor network transmission rates are 120 kS/s and it is difficult to achieve more than that. [6]

The PICtail™ Plus interface SPI (Serial Peripheral Interface) provides a connection to the module to communicate with the microcontroller. The ZG2100 can provide the specified speed in the IEEE 802.11b standard (~ 2Mbps). If necessary, all these can be provided by an encrypted connection, which can be WEP, WPA or WPA2 as well. Each one has hardware acceleration support, so it does not lag additionally the microcontroller with tasks. [7]

B. Programming the WSU

The programs were prepared separately for the two units connected to the microcontroller. The first is the heat sensor, which is responsible for reading the first ADC

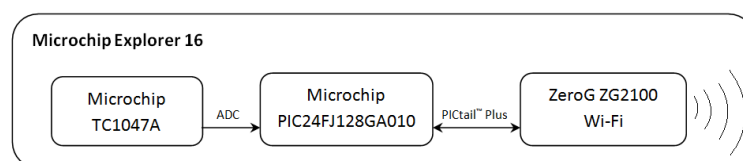


Figure 2. The architecture of the WSU

channel (contains the temperature value). The second unit writes that value to the LCD and sends via UART connectivity to the terminal. The resulting temperature has a one decimal place preciseness, which is not passed forward in this floating format, but as an integer.

The design of the wireless connection was done in two steps. The first step was the configuration of the wireless connection, where the SSID and encryption of the connection had to be set up. The second step was to write the software that is connected as a client using the TCP / IP stack to a PC running the server. Therefore, we need to implement the next three functions: connection to the server, sending the data and disconnecting. During the prototype development phase, these features can be activated by pressing buttons on the developer board.

1) The Sensor Network Protocol

The protocol defines the communication between the PC and WSU in every detail. The protocol relies on the 192.168.75.0/24 IP range from which the IP address of the PC is the first address. On the PC running the server we used the TCP port 0xC001, the port used for data transmission by the WSU's is the TCP port 0xC002 (but the latter can be different).

The data set consists of 4 bytes, of which the first byte is the type of the sensor (in this case 0x01 meant the heat sensor), where the top bit is reserved for future use. The second byte is ID of the sensor, which should be equal to the sensor IP's last byte. The uppermost bit is also reserved for future use, in such way a maximum of 126 sensors can be used simultaneously. The last two bytes contain the value of the measurement; this (sub-divided by 160 at present) gives the measured temperature to one decimal place preciseness.

The communication should ideally take place in the following manner: first, the TCP / IP Stack attempts with an ARP request to resolve the IP address and then establishes the connection to the server. Because it is important that the sent data reach the destination as fast as possible, the Push mark was set in each data packet, which immediately sends the packet without buffering. When a connection ends, both parties indicate the communication has finished.

2) The problems and their solutions

Although problems have occurred from time to time during the communication buildup, our attention was

shifted more towards them when the same errors have occurred during sending the packages – namely, as if the connection between the sensor and the PC had been lost.

During the first tests the connection was set up with a push of a button, a data packet was immediately sent, then the device disconnected. After programming and starting the WSU, this operation executed properly for quite a while, but after a while – at random – during connection phase a time-out happened. We experienced the same chaotic occurrence during echo requests, where we could restore the wireless connection by breaking it and reconnecting.

Then we started to monitor the 'seen' packets by the PC to find out what is generating the error. It was observed that the WSU always needed to resolve an IP address before the connection was established (Fig. 3) and that the sent SYN packets were not always detected in time. When it sensed the request as not accepted – as if it had lost the package – it re-posted it. As the PC got the same packet twice, it expressed its acceptance twice. This terminology in TCP is called Duplicated ACK. This problem could be solved by using separate buttons for connection establishment, for the data sending and for the connection termination option. As a result of this, a new ARP request isn't generated again after disconnection.

In case of data packets, the problem still occurred. The cause of this phenomenon was the same, but at the data packet level. The WSU thought the sent packet was lost; therefore, it resent it (TCP retransmission), to which the server responded in another ACK message (Duplicated ACK). Consequently, there was more time for the WSU's TCP/IP stack to process the received ACK messages.

III. LABVIEW, THE PROCESSING UNIT

National Instruments LabVIEW development environment has been assisting in testing, measurement, and control applications developments for over 20 years. The engineers and developers – regardless of experience – can create interfaces for measurement and control hardware in a short time, in a cost-effective way. They can analyze the measured data, share results, and distribute systems. For these reasons, we also chose the LabVIEW as the processing unit. Its task was to show the values on the PC that came through the WiFi router and was sent by the WSU.

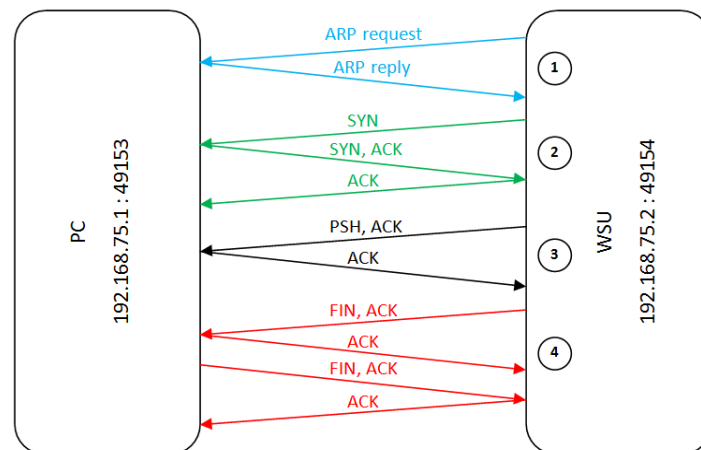


Figure 3. TCP communication between the PC and the WSU

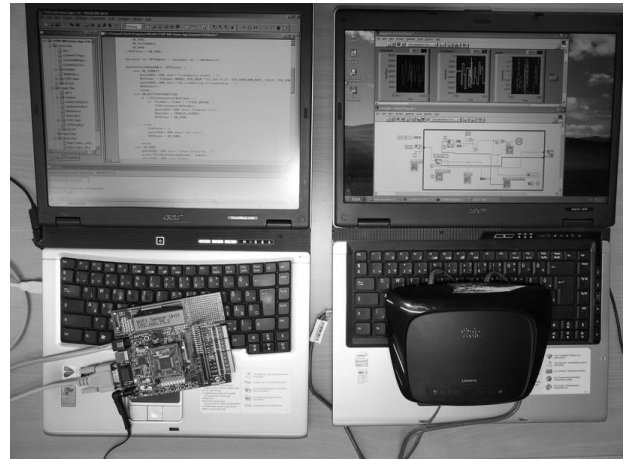
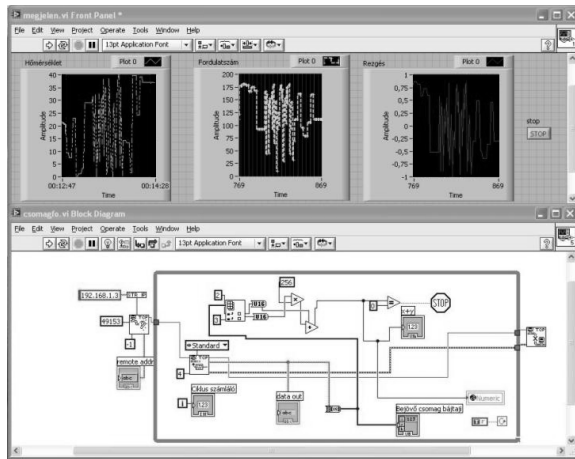


Figure 4. LabVIEW VI and the final system

The first step was to receive the TCP packets sent by the WSU. To do this, we needed a 'TCP Listen.vi' device, for which the input was the desired IP address and port number. This device accepted the connection initiation requests sent on TCP. Following this, we could receive incoming packets sent by the 'TCP Read.vi'. When the processing unit was completed, the connection should have automatically terminated (in case of any error also), so the 'TCP Close Connection.vi' tool had to be used. This is the basic structure of the processing unit communication.

Further tasks were shaping the bytes of post-reading data into decimal numbers and showed the measured values by indicators or graphical display (Fig. 4). However, to make TCP connection continuous and avoid interruption after receiving the first packet, the stage between the connection and disconnection of the device was placed in an infinite while-loop. As a result, the projection became continuous and prorata.

However, another problem occurred until the receipt of the next incoming packet. The value was picked up regardless how much time had passed until the next data was received, the display time between the two values was always constant. To solve this problem we defined a global variable, which stored the values to be displayed, making the values available for all running LabVIEW applications by creating a tool which is responsible for displaying data in the displays (Fig. 4 left upper window).

The task was to run three processes: 1) the packet receiver / processor, 2) the display, and 3) global variable provider. Using these three processes we could solve the task flawlessly in LabVIEW.

IV. THE DEVELOPED SENSOR NETWORK

The development was initially performed separately. One of us was developing the prototype of the WSU, while the other was working on the VI's. Of course, neither can be completed without the other, so it was necessary for both sides to develop a simulation application. To do this, we developed C++ based client to implement the protocol (WSU simulator) and server (LabVIEW simulation). During the development the test-bed was perfectly capable of performing its task, and during the stage of matching the two systems together, there were no recorded problems.

A. The assembled system

The components of the system can be seen on the right side of Fig. 4. One laptop runs the microcontroller development environment, from which the launch of the WSU was done. On another laptop, the LabVIEW processing unit monitors the data received through the WiFi router.

B. Our experience and further development opportunities

As a result, we can say that LabVIEW is perfectly suited for displaying and processing signals provided by sensor networks in wireless configuration. In our further research, we try to answer the question whether LabVIEW can be used for controlling besides processing and visualization.

The PIC microcontroller is only suitable for sending small amounts of data with low sampling rate; therefore, it was sufficient in our work, but cannot be recommended for any other usage.

With the further development of the Sensor Network Protocol the transport layer can be replaced from TCP to UDP, so quasi real-time communication can be realized according to faster data transmission. The transition between two transport protocols is the improved, lossless UDP data communication, which as an option, we would like to implement in our system [2].

REFERENCES

- [1] Á. Ballagi, "Zigbee: vezeték nélküli komplex szenzorhálózatok gyorsan, olcsón, hatékonyan", *Folyamatirányító Rendszerek XIII. Találkozója*, 2007.
- [2] T. Szakáll, P. Dukán, B. Odadžić, P. Odry, "Realization of reliable high speed data transfer over UDP with continuous storage", *11th IEEE International Symposium on Computational Intelligence and Informatics*, 2010.
- [3] J. Reed, J. Neel, "Future Wireless Standards and Emergence of WiMAX", 2007.
- [4] E. Suhayl, "iBurst® - its position in provision of low cost infrastructure solutions", *Public & Private Sector Partnership Forum*, 2007.
- [5] Microchip, "Explorer 16 Development Board User's Guide (DS1589A)", *Microchip Technology Incorporated*, 2005.
- [6] Microchip, "PIC24FJ128GA Family Data Sheet (DS39747C)", *Microchip Technology Incorporated*, 2006.
- [7] ZeroG, "ZG2100 Product Overview (071309)", *ZeroG Wireless*, 2009.