Real-time Data Acquisition Based on Common Use Interfaces at Matlab and Embedded System

Krzysztof Sieczkowski, and Tadeusz Sondej

Abstract—**The article presents the tests of method for real-time data acquisition from embedded systems using Matlab software. Data transmission is performed using several common use interfaces: UART/USB, Ethernet, Bluetooth and WiFi. The article includes a description of a protocol, measuring station based on two types of embedded system, implementing the proposed protocol, as well as a description of the algorithm of test programs. Experimental studies were performed using a STM32F4 microcontroller and Raspberry PI-3 single-board computer. Executed tests related: (1) the average transmission time, (2) the effective throughput of a full cycle of data exchange, (3), the required working time of Matlab to handle the transmission, and (4) the stability of the program timer used for periodic data transmission calls. Experimental studies have shown that it is possible efficient data exchange between the embedded system and Matlab while maintaining the real-time requirements.**

Index Terms—**data acquisition; real-time; Matlab; serial communications; data processing**

I. Introduction

DATA acquisition is widely used in measurement systems. Such systems read data stored in appropriate buffers, or read data in real time. Almost every modern measuring system consists of a sensor, analog-front-end input circuit, A/C converter, a microprocessor and software for data processing and visualization. We can distinguish 3 groups of measuring systems: (1) autonomous, (2) autonomous with the possibility of cooperation with a computer, (3) systems requiring the use of a computer. The last of these groups is called virtual instruments (VI). Due to the prevalence of PCs and decreasing prices of measuring instruments, they are being used very often. VI instruments require a computer with the appropriate software. It can be done using dedicated solutions or popular computer software can be used, for example Matlab or LabView. Matlab software [1] is readily used because of the enormous processing power and possibilities of displaying data, creating graphical user interfaces (GUI) and because of the support of multiple communication interfaces and measurement cards. Matlab has a comprehensive set of tools optimized for specific types of calculations. For example, using the Matlab programming environment, it is possible design fully functional digital filters. This environment includes tools to help design these blocks for example *FDAtool*. Matlab environment is also commonly used due to the high degree of optimization, advanced numerical computation, where the most commonly used calculations are Fast Fourier Transform (FFT), Discrete Wavelet Decomposition (DWT). Matlab is also equipped with tools used for statistical calculations. The basic functionality of Matlab can be expanded at any time by installing additional software modules. In addition, you can design a controller with an ergonomic GUI, together with the required algorithm for data processing. The time to implement such a system is much shorter than the design of a new original program in another development environment.

K. Sieczkowski and T. Sondej are from Electronic Department, Military University of Technology, Warsaw, Poland (e-mails: krzysztof.sieczkowski@wat.edu.pl, tadeusz.sondej@wat.edu.pl).

An important factor when using Matlab software is the data acquisition method. These data can be read from a file or from a virtual instrument (e.g. card, USB adapter), off-line (after recording data) or on-line (on an ongoing basis - in real time). An example of the use of a popular measurement card NI USB6251 is presented in [2]. The use of a serial port is shown in [3], while article [4] shows how to use a Bluetooth wireless interface.

Data acquisition in real-time requires special techniques for data transmission. One of the important tasks is to provide data synchronization, especially in the case of simultaneous acquisition of data from multiple devices. An example of a distributed measurement system with GPRS is presented in [5]. However, in the case of such a system, delay in data transmission is very high and can amount to tens of seconds. Solutions based on local microprocessor systems with dedicated measurement cards are also used. An example with the use of the free Linux operating system is presented in [6]. Data acquisition in real-time is especially difficult in systems with multiple wireless devices. In this case, TDMA (Time Division Multiple Access) techniques are the most commonly used. Examples of solutions are presented in [7] and [8].

Using Matlab software significantly accelerates and simplifies the design of measurement systems. But this software does not run under the control of the operating system's real time. Therefore, to ensure data acquisition in real time (on-line), special data exchange protocols are the most frequently used, for example, protocols based on sending commands and receiving responses.

This article presents a method of communication between Matlab software and an external measuring device. The proposed method involves the use of a wired serial interface (e.g. serial COM - communication port) or wireless serial interface (e.g. Bluetooth, WiFi). In addition, the article presents the results of experimental tests of a system consisting of Matlab software, STM32 microcontroller [9] and various communication interfaces.

II. Characteristics of Common Use Interfaces

To test the proposed method of data acquisition from external devices, three popular interfaces have been used: COM serial port, Ethernet, Bluetooth and WiFi. Depending on the type of interface and its method of implementation, they offer a different throughput.

A COM serial port was initially implemented as an RS232C interface and now it appears in the form of different types of adapters, e.g. USB<->RS232C or USB<->UART. However, in the operating system and in Matlab software, it is recognised as a COM port. Solution implemented by FTDI [10] are used very often, which offer full compatibility with the COM interface, while providing much higher data rates than in RS232C.

Another very frequently used solution is the SPP (Serial Port Profile), which uses the Bluetooth wireless interface. Bluetooth is commonly used in personal communication for the transmission of multimedia data, but also in distributed measurement systems. The use of Bluetooth communication has become widespread due to the availability of multiple integrated modules. Differences between the modules usually come down to the implemented profiles. There are two basic groups of Bluetooth modules: those supporting different Bluetooth profiles and those with HCI (Host Controller Interface) interfaces. The use of modules with HCI interface requires the use of additional procedures that support the Bluetooth protocol stack. On the other hand, configuring the module as a device operating in the SPP makes it possible to use the module as a wireless equivalent of the RS232C interface. Usually before the first use of a Bluetooth module, its initial configuration is required, e.g. using AT commands.

Communication via the WiFi interface (IEEE 802.11) has found wide use in computer networks. It is the wireless equivalent of the Ethernet interface. The WiFi standard allows for wireless and faster transmission of data between devices which are at a much greater distance than previously described Bluetooth. Depending on the version, WiFi offers a variety of data transmission rates, ranging from 11Mbps (802.11b) to 1Gbps (802.11ac). In embedded systems and off-line measuring devices, modules dedicated to WiFi connectivity are used most often. Such modules are available with various implementations of protocol stacks or without the stack. Lack of support for the stack requires the use of an additional microprocessor. In contrast, others offer support for TCP/IP, UDP/IP as well as network control mechanisms, e.g. ping or traceroute. There are also modules containing optional additional higher layer protocols. The modules supporting specific protocols are often equipped with additional (embedded) microcontroller, providing support for these protocols. Modules with protocol support can be set up to work automatically. New to the market are WiFi modules working as a wireless RS232C or UART interface. For a WiFi module to work as a WiFi - RS232C/UART bridge, the network address and password to which the module will connect must be pre-configured. In addition, it is necessary to set the number of ports required for TCP or UDP. Modern WiFi modules also offer the opportunity to work in 'SoftAP' mode, allowing you to create a wireless network from the WiFi module. In any case, the wireless transmission must take into account the time required for connection.

III. Method for Transmitting Data in Real Time to Matlab Software

A protocol with command-response architecture has been designed for error-free data transmission. The master system sends certain commands, which must always be answered by the slave device. This protocol is based on a basic frame, the structure of which is shown in Fig. 1.

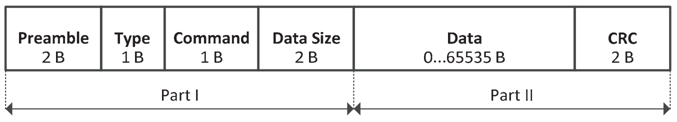

Fig. 1. The structure of the basic frame of the communication protocol.

The frame consists of 6 fields. The "Preamble" field contains the constant value synchronizing the slave device. The "Type" field determines the frame type: command, response, error when receiving the last frame, and the direction of transmission. The "Command" field contains the command code. "Data Size" specifies the size of the "Data" field. The "Data" field contains the transferred data. The field with the "CRC" checksum allows to detect an error in the frame structure. The protocol allows to send up to 64 kB of data in one frame. The proposed protocol is conventionally divided into two parts (Part I and Part II). The first part consists of the fields: „Preamble”, „Type”, „Command”, „Data Size”. The second part is: the "Date" and "CRC" fields. This division allows to accelerate the process of handling the frame by the receiver device, because the first part always has a constant structure and determines the size of the rest of the frame. The protocol has been designed to enable the detection of data error or loss.

In this study, the transmission always takes place on a command-response basis. The master device (computer and Matlab software), performing calculations and fulfilling an administrative function, sends a query to the device. A new query is sent only if a response has been received from the slave device, or when there was no response (timeout). To enable efficient communication in the command-response mode, an appropriate function that supports this type of transmission has been designed. The algorithm of this function is shown in Figure 2.

As an argument, the function adopts a data vector whose structure maps the structure of the frame. When a command is sent, a frame is transmitted containing data specifying that it is a request to the device. After each call, the "fTransmit" flag is checked. When the function is called at the beginning of the command-response cycle, the checked flag has a zero value. This allows the implementation of the first part of the cycle, i.e. transmission. Therefore, the value of the timeout counter is set next, the command is sent and the "fTransmit" flag is set. Recalling the function (with the same argument as before) makes the function check if there is already an active transmission cycle.

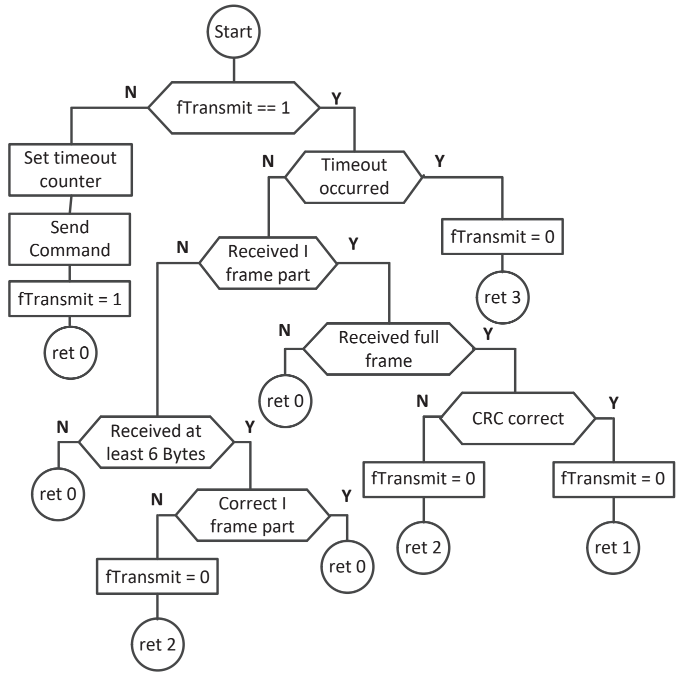

Fig. 2. Algorithm of the function sending and receiving data.

Since the last function call sets the active transmission flag, the function performs procedures to correctly receive a response from the device. Therefore, it checks whether the maximum time from sending the command has been reached. If so, it returns a value signaling a timeout. If the timeout has not occurred, one of two conventionally accepted parts of the frame is checked. After receiving any part, the correctness of data is verified.

If the data in the first part or in the entire frame are invalid, the function returns a value indicating a CRC error. Therefore, to send a command to the slave device and receive a confirmation, it is necessary to call the function that supports the transmission at least 3 times. The first call is made to the command transmission and two to the receipt of a response - respectively to the first part of the frame (Part I - 6 bytes) and to the second part (Part II). The function returns 0 when the internal procedures will be performed correctly and it does not require the activity of other procedures. The value of 1 returned by the function supporting transmission signals that the received frame is an integral whole and can be passed to the function that supports a higher layer of the protocol.

In order to achieve the optimum time occupied by data transmission (maintaining the required throughput) in any data acquisition and processing system, it is necessary to correctly select the frequency of sending commands (and verification whether a set of data has already been received) to the slave device. In addition, it is necessary to select the proper size of a data packet that will be sent as a reply in one transmission.

IV. Organization of the Measuring System

A. Hardware and software setup

To test the operation of the interfaces mentioned in Matlab, system has been compiled according to the following topology. The role of the master device is played by a computer running 64-bit Windows 7 operating system with Intel Core i7-3770K, 3.50 GHz and 16 GB of RAM. The computer has a SSD with 128 GB.

In order to further testing of test, two external measurement systems have been made. The first external measurement system was constructed based on the STM32F4 discovery kit (32-bit microcontroller with Cortex-M4 - STM32F407 core, 168 MHz clock). The assembled system is shown in Fig. 3.

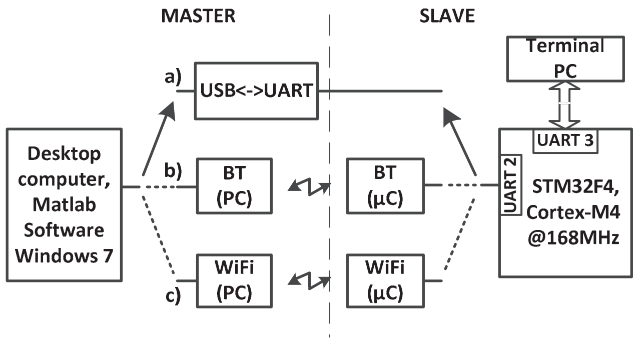

Fig. 3. Block diagram of the first measurement system.

The master side, for the first system used in the study features: a) USB<->UART adapter type FT232RL made by FTDI, b) Bluetooth module, c) WiFi card. The FT232RL system after installing the drivers is reported as a COM serial port, which can work with any configured Baud Rate (BR). In the tests, the BR is set to 1 Mb/s. Bluetooth module - BT (PC) is a module designed by Pentagram. This module operates using the Bluetooth 2.1 standard. It allows configuration in the serial port profile (SPP). As the WiFi module, an external network adapter TP-LINK TL-WN722N has been used, allowing work in the network using the 802.11n standard.

The slave of the first measurement system with the STM32F4 microcontroller uses two UART interfaces (2 and 3). UART2 handles communication with the master system using: a) USB<->UART adapter, b) Bluetooth module, c) WiFi module. The second interface (UART3) was connected permanently to the serial port terminal in a computer. This terminal is used to display messages relating to system operation, and to present measurement results. For each UART interface, the BR was set to 1 Mb/s. For Bluetooth - BT (μC) connections, the RN-41 module from Roving Networks [11] was used. It makes it possible to work in the SPP allowing you to create a wireless serial link. In the study, the module was configured to operate automatically as a master system in a private Bluetooth network. From the side of the microcontroller, this module also worked at a speed of 1 Mbps. After the one-time configuration, software written using Matlab was able to establish a connection with the pre-configured RN-41 module. For connections via WiFi (μC), AMW106 module designed by Ackman Networks [12] was used. This system's structure features a Cortex-M3 microcontroller, in which the IP protocol stack has been implemented. In addition, this system has been configured to run in "Wireless Serial Port" mode, taking advantage of UDP datagrams. With the help of the computer WiFi network adapter and the ACK106 module, a wireless computer network was launched in which role of an AccessPoint device was played by the ACK106 module while the network card was the client. After powering up the microprocessor system, the computer was able to connect to the associated network (known SSID). Previously configured UDP transmission allows for data exchange between two devices. Radio equipment, both Bluetooth and Wi-Fi, were located at a distance of about 1 m from each other during testing. In addition, each of the devices had implemented support for hardware-based data flow control (implemented using the RTS/CTS line in the UART interface).

Organization of the second measuring system is very similar to the previously presented. The main difference between the systems is occurring type of evaluation board used in the system. In the second system uses single board computer such as Raspberry PI-3 Model B [13]. Raspberry PI is running Linux, the distribution Raspbian. The Raspberry PI computer provides SoC Broadcom BCM2837 type. The BCM2837 chip includes a 4-core, 64-bit ARM Cortex-A53 processor. Each of the cores work with a maximum frequency of 1.2 GHz. The BCM2837 SoC also include GPU unit, type VideoCore IV, offering computing power at the level of 28.8 GFLOPS. RAM 1GB can be shared between the CPU and GPU processors. Schematic diagram of the assembled second measurement system shown in Fig. 4.

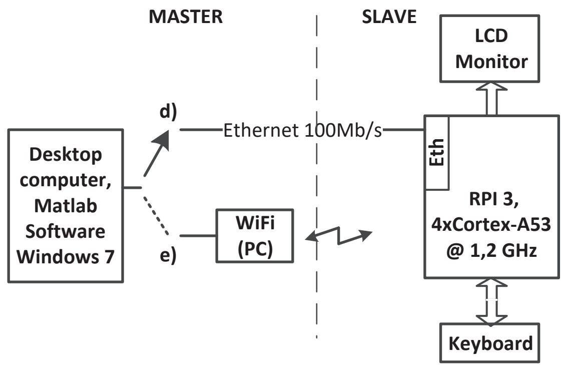
MASTER
SLAVE
LCD Monitor
d)
Ethernet 100Mb/s
Eth
Desktop computer, Matlab Software Windows 7
RPI 3, 4xCortex-A53 @ 1,2 GHz
WiFi (PC)
e)
Keyboard

Fig. 4. Block diagram of the second measurement system.

At the Master side, for the second measurement system is the same PC as the first system. It occurs also the same WiFi (e) network card. A novelty in the second measurement system to replace Bluetooth and USB-UART bridge through the Ethernet hardware interface (d). Uses Ethernet adapter type Realtek PCIe GBE Family Controller 1Gbit.

At the slave side of the second measurement system used in the factory integrated Ethernet controller and WiFi. Raspberry PI computer contains 100Mb Ethernet controller type, and WiFi network can operate in 802.11n standard. Controlling the test program at the slave side, was done by a local keyboard and monitor connected directly to the computer Raspberry PI 3.

Testing software on the master side was designed in Matlab software, version R2013a. Only standard tools available in Matlab were used. The testing program was designed in the form of a GUI window. The algorithm of the testing program is shown in Fig. 5.

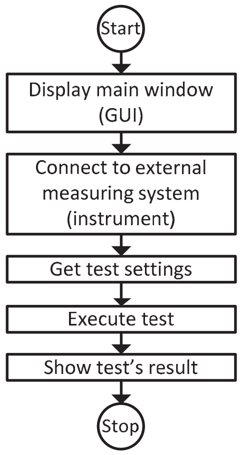
Start
Display main window (GUI)
Connect to external measuring system (instrument)
Get test settings
Execute test
Show test's result
Stop

Fig. 5. Matlab's algorithm of the testing program.

The GUI is displayed after running the script. The user can enter measurement parameters and selects one of the three communication interfaces tested. Once this is done, it is possible to run the measurement. After the end of the test, the result is displayed in the "Command Window" field. Access to the equipment from the Matlab environment, whether it is a serial port, Bluetooth, or UDP datagrams in a WiFi network, is very similar. In Matlab software, the interface through which the program connects to an external device is called the instrument. The method of supporting the communication interface in each case amounts to creating an object that represents the given instrument. All the data defining the connection parameters are stored in this object. After the object is created, it is possible to open the connection (occupation of a resource by Matlab in Windows). When the connection is established, it is possible to transfer data to/from an external device using one of the data transmission functions. Any communication interface can be used.

Data are sent from the master device without delay. Only a handle for the opened and configured instrument is required. The data received by the selected interface are stored in the FIFO buffer of the given instrument. The number of bytes contained in the receive buffer is stored in a variable associated with the given communication interface. Therefore, it is important to set the size of the receive buffer to a size that will not allow a buffer overflow or the data must be collected at a speed greater than the speed of sending data by an external device.

Software for the microprocessor (first measurement system) was written in C using the Keil MDK ARM development environment. The main algorithm of the program assumes independent receipt and immediate verification of received frames. After correct decoding, the slave sends a response to the received command. The microprocessor system has a 32-bit counter, operating at a frequency of

1 MHz. The counter makes it possible to obtain a resolution of 1 μs and to maintain a sufficiently long measurement time. The counter is used to retrieve timestamps for the beginning and end of "command" frames and the beginning and end of "response" frames. Moments of retrieving timestamps are shown in Fig. 6. Retrieved timestamps are then processed and displayed in the computer terminal's console for further analysis.

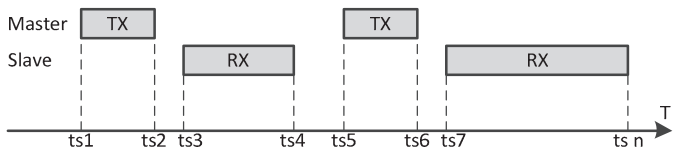

Fig. 6. Moments of retrieving timestamps in the slave device.

Measurements consisted of sending 1 MB of data from the microprocessor device to a computer in packets of fixed size for the three studied communication interfaces. This was implemented in the following manner. In the first cycle, the master system sends a query command to the slave system, regarding the amount of available data. The slave device sends the response (1 MB of data available). The next command-response cycle comes down to sending requests for a certain amount of data in order to receive the full set of data.

Dedicated software for Slave side, second measuring system (Raspberry PI 3), it has been designed using the C language and GCC compiler. In this system, is used network connection (Ethernet and WiFi). As a lower layer protocol, it was used connectionless UDP protocol. UDP protocol handling was implemented in a separate thread, where after receiving the UDP datagram was immediately sent to a decoding procedure for an original, designed frame transmission. Information about the current state of the system were sent to the terminal window Linux shell.

B. Measurement tests

The four different data transmission tests were performed. The first three tests were performed using a first measurement system. Test no. 1 determined the transmission time of one data packet and the effective throughput when packets of various sizes are transmitted. Test no. 2 determined the time required to handle the full command-response cycle and the throughput achieved. In this test, the function transmitting data was called with a fixed time interval. As a result, Matlab was able to also perform other tasks than handling the transmission. In addition, the link's effective throughput was determined. The idea of the test boils down to finding the optimum period for calling the function handling transmission. This was to ensure maintaining the best throughput and the smallest usage of Matlab for handling the transmission of packets of varying lengths and the three interfaces tested. If the system is designed in Matlab software, which is to be similar to the RTOS (Real Time Operating System) system, there may be a need to calling the function or procedure transmitting data in a cyclic manner. Therefore, test no. 3 was concerned with determining the stability of data calls using a dedicated "timer" object for this purpose. This test consisted of sending a frame (8 bytes) to the slave device in a fixed period of time, in which the role of the clock signal is executed by the timer object in Matlab software. The timer object was configured to call the function sending the data packet when a specified time interval is reached. If these packets are received in the microprocessor system, timestamps for the beginning and end of the packet are retrieved. These timestamps, after the end of the test, allow to define the moment of receiving the frame transmitted by the slave device. In each of the tests, besides the aforementioned parameters, the correctness of data transfer was verified.

Fourth test was performed using a second measuring system. It is a complement of the first studies of tests focusing on a different type of embedded system. The fourth test consists of two types of tests. In the first type of program Matlab only took away the data (in the same manner as in the first study). In the second part of this test, the received amount of data that has been subjected to processing. Each received amount of data was subjected to Fourier processing (FFT) and inverse Fourier Transform (IFFT). The FFT-IFFT processing cycle was repeated 100 times for each received part of data.

Therefore, the fourth study made using the same software as the first. For the second part of this study was added built-in procedures FFT and IFFT. The second system is based on a different type of processor (Broadcom BCM2837 SoC), so the slave-side software has been adapted to a different architecture. The principle of operation and major algorithm of the program has not been changed.

V. TESTS RESULTS

As stated earlier, three tests were performed. In the first test, the average transmission time of one complete cycle was measured and determined, and the average throughput for packets with different lengths was also determined. These tests were performed for the three communication interfaces analysed. Fig. 7 shows the results.

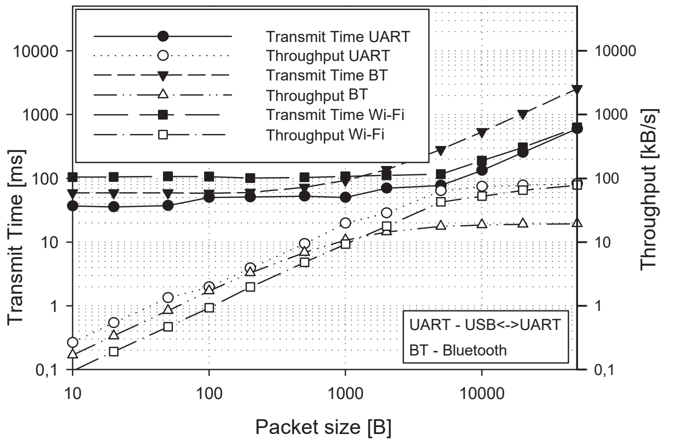

Fig. 7. Transmission time and effective throughput for one packet.

From the obtained results, it can be concluded that the highest throughput was obtained when devices communicate via a wired link. Throughput in case of Bluetooth transmission is greater than that of WiFi transmission for packets with a size of less than 2 kB of data. Above this value, there is a large increase in the duration of packet transmission using Bluetooth. In the case of very large packets (50 kB), the effective throughput for the WiFi module significantly improved, approaching the values achieved in the case of a wired connection. For value of approx. 5 kB, the Bluetooth module is saturated reaching a throughput of 20 kB/s. The transmission times achieved have averaged values, in which the value of the standard deviation is shown in Fig. 8.

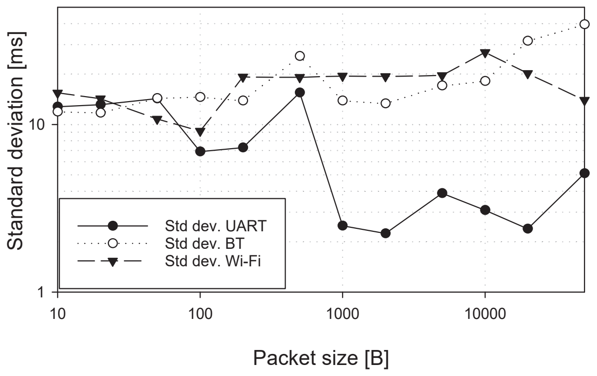

Fig. 8. Standard deviation for packets of different lengths (first measurement system).

The standard deviation values for wireless transmission remain at the level of approx. 9 ms to 40 ms. In wired transmission, this value is significantly improved (from 2 ms to 5 ms) in the case of transmission of packets with a size of 1 kB of data or larger. A momentary increase of the standard deviation was also observed for two types of transmission (wired and Bluetooth) when the packet size was 500 B. This test was performed in a situation where the function implementing the command-response transmission was continuously checking whether the full frame was retrieved. Therefore, in this situation, the work of Matlab software focused entirely on data transmission

Test no. 2 helped to determine the effective working time of Matlab which is required to handle data transmission, but at the same time allowing for data processing. The test method is shown in Fig. 9.

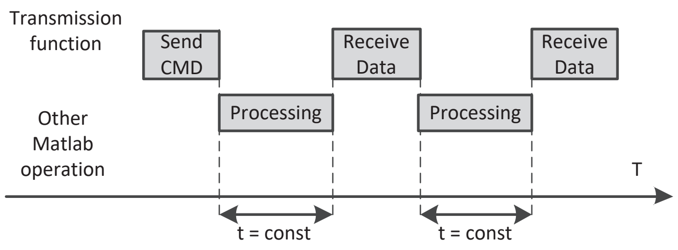

Fig. 9. Test of Matlab usage required for handling data transmission.

Calling the function handling the transmission was implemented with a fixed time interval (t = const). Values of the measured system usage times and the achieved data transfer throughput of the UART interface are shown in Fig. 10.

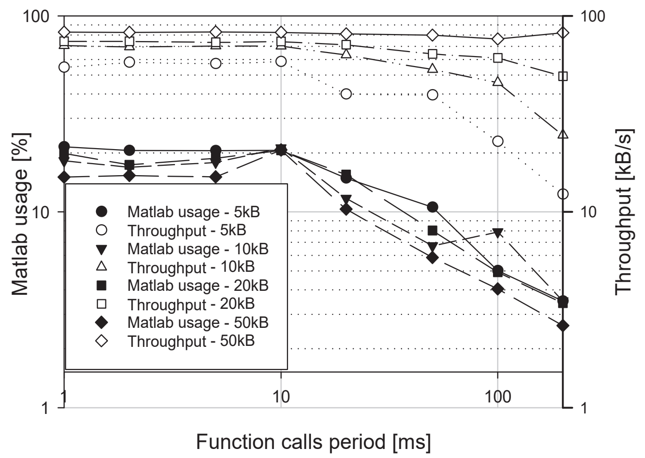

Fig. 10. Usage of the Matlab system and throughput for different function call times.

For each period of calling the function which handles transmission, the throughput obtained increases with the size of the packet. If the function call interval is 10 ms, the largest increase in Matlab usage of approx. 20 % was observed for all transmissions. From this value, system usage decreases to a level of a few percent. In the case of other interfaces, a similar trend was obtained in regard to the system usage. A summary of results for packets with a size of 5 kB is shown in Table I. The smallest system usage was obtained when devices communicated via Bluetooth. In the case of the WiFi interface, a slight decrease in usage of the system was observed compared to the wired interface.

TABLE I.
USAGE OF THE MATLAB SYSTEM FOR THREE TESTED INTERFACES (FOR PACKET SIZE 5 KB)

Call period [ms]	*UART*	*BT*	*WiFi*
0	99.68	99.63	99.58
1	21.48	16.49	20.27
2	20.61	16.40	19.07
5	20.56	16.37	20.55
10	20.63	16.82	20.46
20	14.87	9.76	12.24
50	10.57	6.49	8.08
100	5.01	4.24	5.69
200	3.51	3.21	3.66

In the last test, the stability of a counter in Matlab, used as an object triggering periodic data transmission, was determined. Fig. 11 shows the standard deviation of the jitter values occurring during data transmission. The results are similar for each interface tested. It should be noted that for short periods between sending commands (e.g. 20 ms), the deviation is approx. 15 ms. Therefore, it is comparable with the value of the intervals between transmission calls. As presented earlier, the fourth study determine the transmission time of one data packet and the effective throughput when packets of various sizes are transmitted.

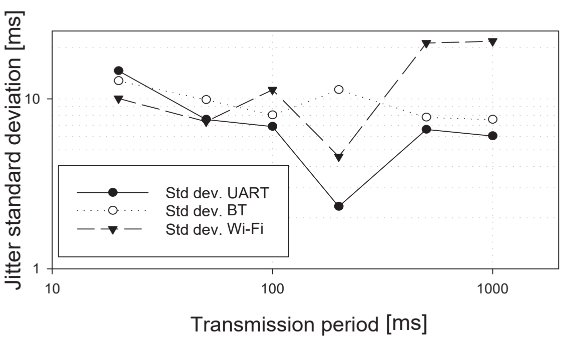

Fig. 11. Standard deviation for the transmission of data with different periods of repetition.

The Fig. 12 shows the resulting of transmission time and effective throughput for one packet when the Matlab software, receives only data. Software at the master-side does not perform any data processing.

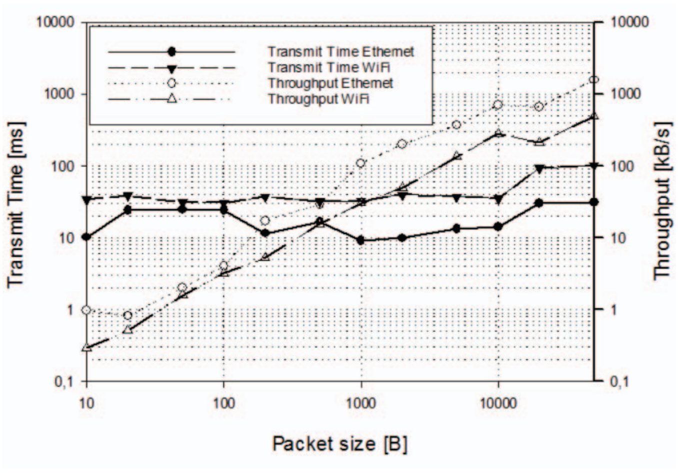

Fig. 12. Transmission time and effective throughput for one packet (Matlab only received the data).

With the increase of the size of the transmitted packet, increasing the effective data throughput. This increase is observed until the packet size is about 20 kB of data. Both types of transmission where the packet size is about 20 kB, there was a slight decrease of effective throughput. For each transmitted packet size, wired connection (Ethernet) offered a higher effective throughput compared with wireless connection (WiFi). The highest throughput achieved for the wired network and was about 1570 kB/s and in the case of a wireless connection was approximately 480 kB/s. Regardless of the type of network connection, the maximum throughput achieved for the largest size of the test packet. In the case of WiFi, almost the entire range of the transmitted packet sizes, reached similar transmission times. In most cases (packets with sizes ranging from 10 B to 10 kB), the time was about 33 to 35 ms. Only a packet size greater than 10 kB have been transmitted with an average time of more than 90ms. For a wired connection, the transmission times of a single package contained a much greater extent compared to a WiFi connection. The smallest transmission time (less than 20 ms) of single packet, for a wired connection obtained when the packet size was 10 B, and size is in the range of from 200 B to 10 kB. The largest transmission time of one packet has been achieved when the transmitted packets have a size greater than 10 kB. For a wired transmission, time of 31 ms was obtained, but for the wireless transmission, the obtained time was about 100 ms.

In the fourth study also performed a second test to determine the transmission time of one data packet and the effective throughput when packets of various sizes are transmitted but now additionally Master-side software (Matlab) process data. Each received amount of data was 100 times processed using the FFT and IFFT algorithms. The Fig. 13 shows the obtained result.

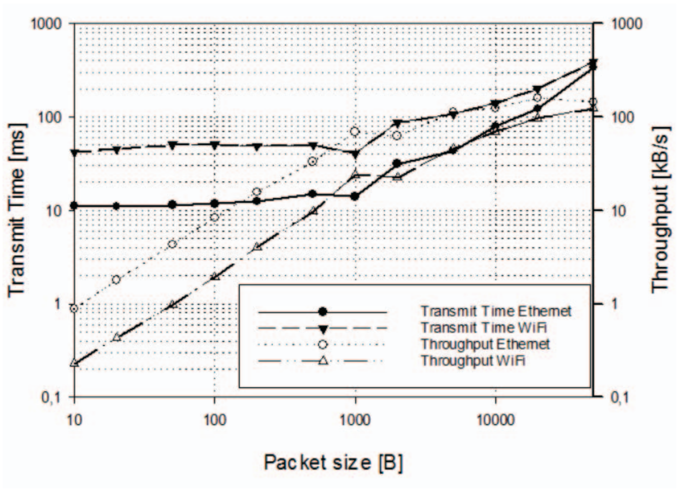

Fig. 13. Transmission time and effective throughput for one packet (Matlab received and processed the data).

Similarly to the first part of this study, there was an increase effective throughput with increasing size of the package. Also it observed a moment where there is a lack of continuity growth of effective throughput. This moment is when packet size is about 2 kB. There was also a greater difference in throughput between a wired and wireless, for the corresponding size of transmitted packets. In the range of 10 B to 1 kB was observed linear transmission times of the data packet. For a wire transmission it was about 12 ms, and for wireless transmission it was about 48 ms. Over 1 kB packet size, transmission time of one packet grew up to more than 337 ms for Ethernet, and 393 ms for WiFi connection.

The above times were averaged, but the standard deviation of the interfaces used in a second type of measurement system is shown in Fig. 14. Depending on the type of connection and load of Master system, there was a different standard deviations. The largest standard deviation was observed in the case of network connection type WiFi and the Master software handles the received measurement data. In this case, the standard deviation over a wide area of the chart was about 28 to 37 ms. The largest spread of the standard deviation has been observed in the case of network connection type WiFi and the Master software does not process the data. In this case, the standard deviation was in the range of from about 14 to more than 30 ms. The lowest value of the standard deviation was observed when the network connection was Ethernet and Master software also handles the received data. In almost the entire range characteristic of the standard deviation does not exceed 10 ms. In the case of network connection type Ethernet, and the software does not process the data, we observed the lowest standard deviation of less than 5 ms (for packet sizes: 2 kB, 5 kB, 20 kB and 50 kB).

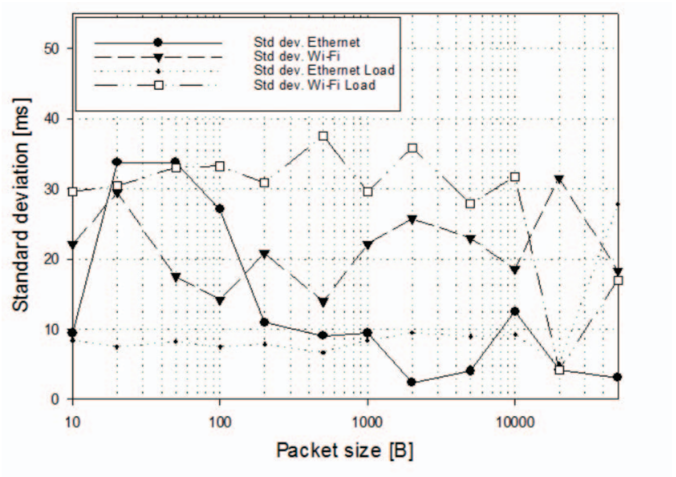

Fig. 14. Standard deviation for packets of different lengths (second measurement system).

After the completion of all four tests, no errors during data transmission were observed.

VI. Summary

The proposed method of communication makes it possible to achieve an error-free and fast data transmission. Also, the process of communication between two devices is monitored on an ongoing basis. As a result, this method can be used in data acquisition systems with transmission in real time. Data transfer size may change dynamically depending on the settings or load on the system that carries out some other tasks, such as data processing. The communication method presented in the article can be implemented with different serial interfaces and various computer software. The article presents test results for the UART/USB, Ethernet, Bluetooth and WiFi interfaces and for software written in Matlab. The tests allowed to assess the suitability of using selected communication interfaces for two type embedded system. The first system (1) was based on full an autonomous microcontroller (STM32 with ARM Cortex-MF core). The second one (2) was based on embedded system with high performance microprocessor (Raspberry Pi 3 System-on-Chip BCM2837 with ARM Cortex-A53 core) controlled by Linux operating system. Also, the experimental tests allowed the usage (measured as the duration of handling communication procedures) of Matlab when handling this method of communication was determined. For first (1) tested system, during the transfer of small data packets (less than 1 kB), the average transmission time in the command-response cycle remains at a constant level for each of the interfaces and is the shortest for UART (approx. 40 ms) and the longest for WiFi (approx. 100 ms). When sending larger packets (greater than 1 kB), a larger role is played by the data transfer rate of the selected interface. Tests of Matlab usage when handling the proposed method of communication in the worst case amounts to approx. 20 % for UART. However, in the case of calling a function that handles communication every 100 ms and the transmission of larger packets (50 kB), the usage is only 4 %. For second embedded system (2), the highest throughput achieved for the wired (Ethernet) network and was about 1570 kB/s and in the case of a wireless (WiFi) connection was approximately 480 kB/s. The largest transmission time has been achieved when the transmitted one packet have a size greater than 10 kB, for Ethernet interfaces it is about 31 ms and for WiFi interfaces it is about 100 ms. When Matlab also performs data processing (each received amount of data was 100 times processed using the FFT and IFFT algorithms), the transmission time is increased to 337 ms for Ethernet, and 393 ms for WiFi. It should be noted, that in this case, was still retained real-time requirements.

The proposed algorithm and dedicated communication protocol were originally designed to communicate with a single device. (e.g. PC and external microprocessor system). In the case of multiple data streams where each of them has a different communication interface (BT, Wi-Fi, UART), it is possible to create several dedicated functions that implement the proposed algorithm. All transmitting functions must be called in Matlab sequentially. At a given time, a single transmission is supported.

If multiple external devices use the same communication interface e.g. Bluetooth, then the distinction between multiple data streams must be made using the appropriate command (Command field) and the higher layer protocol. In this case, all devices within the communication interface cannot transmit data simultaneously. Communication with devices is also sequential.

It is also possible to transmit multiple streams from one device, where a dedicated command and a higher layer protocol must be used.

Experimental tests carried out have shown that a suitable compromise is possible between the usage of Matlab while maintaining a suitable throughput. The communication protocol presented and the function for handling transmission can be used wherever there is a need for an efficient communication of a system based on Matlab with, for example, distributed measurement sensors.

Acknowledgment

This work has been supported by the Military University of Technology, Warsaw, Poland, as a part of the project PBS661.

References

[1] MathWorks, MATLAB® Product Family, www.mathworks.com, 2016
[2] A. Panda, H. Wong, V. Kapila, S. H. Lee, “Matlab Data Acquisition and Control Toolbox for Basic Stamp Microcontrollers”, 2006 45th IEEE Conference on Decision and Control, 2006, pp. 3918-3925.
[3] M.N. Elya, M. Loqman, M. Aqilah, S. Murniati, “Development of simple setup for model identification using Matlab Data Acquisition”, 2013 IEEE International Conference on Control System, Computing and Engineering (ICCSCE), 2013, pp. 52-57.

[4] N. Belgacem, S. Assous, F. Bereksi-Reguig, "Bluetooth portable device and Matlab-based GUI for ECG signal acquisition and analisys", 2011 7th International Workshop on Systems, Signal Processing and their Applications (WOSSPA), 2011, pp. 87-90.

[5] A. Z. Alkar, M. A. Karaca, „An Internet-Based Interactive Embedded Data-Acquisition System for Real-Time Applications", IEEE Trans. on Instrum. and Measurement, 2009, Vol. 58, Issue 3, pp. 522-529.

[6] P. Cao, K. Song, J. Yang, K. Zhang, "A real-time data transmission method based on Linux for physical experimental readout systems", 2012 18th IEEE-NPSS Real Time Conference (RT), 2012, pp. 1-5.

[7] J. Zhang, J. Wu, Z. Han, L. Liu, K. Tian, J. Dong, "Low Power, Accurate Time Synchronization MAC Protocol for Real-Time Wireless Data Acquisition", IEEE Transactions on Nuclear Science, 2013, Vol. 60, Issue 5, pp. 3683-3688.

[8] K. Różanowski, M. Sawicki, T. Sondej, "Wireless Measurement Modules for Multichannel Drivers Monitoring System", Przegląd Elektrotechniczny, R. 89, No. 12/2013, pp. 138-141.

[9] ST Microelectronics, STM32 http://www.st.com/web/en/catalog/mmc/FM141/SC1169/SS1577/LN11, 2016

[10] FTDI, FT232RL Data Sheet, http://www.ftdichip.com, 2016

[11] Microchip, RN41 Data Sheet, http://ww1.microchip.com/downloads/en/DeviceDoc/rn-41-ds-v3.42r.pdf, 2016

[12] Zentri, ADS-MWx06-WiConnect-106R, http://ack.me, 2016

[13] https://www.raspberrypi.org, 2017

Krzysztof Sieczkowski was born 20th July 1990 in Łowicz. He received B.Sc. and M.Sc. degrees in electronics and telecommunications from Military University of Technology, Poland, in 2014 and 2015, respectively. He is currently working toward the PhD degree in electronic engineering. His main scientific interests are measurement and analysis of biometric signals, especially detection strokes in blood pressure and energy-efficient microprocessor systems.

Tadeusz Sondej received the M.Sc. degree in electronic engineering and the Ph. D. degree in applied sciences from the Military University of Technology (MUT), Warsaw, Poland, in 1997 and 2003 respectively. Since 1998, he has been with the MUT, where he has been working on design and programming of embedded systems. His current interests are in the field of design, optimization and programming of System-on-Chip based digital systems, especially for biomedical applications.