Cost-effective Ethernet Communication for Low Cost Microcontroller Architecture

Original Scientific Paper

Goran Horvat

J. J. Strossmayer University of Osijek,
Faculty of Electrical Engineering
Kneza Trpimira 2b, Osijek, Croatia
goran.horvat@etfos.hr

Damir Šoštarić

J. J. Strossmayer University of Osijek,
Faculty of Electrical Engineering
Kneza Trpimira 2b, Osijek, Croatia
damir.sostaric@etfos.hr

Zoran Balkić

J. J. Strossmayer University of Osijek,
Faculty of Electrical Engineering
Kneza Trpimira 2b, Osijek, Croatia
zoran.balkic@etfos.hr

Abstract – *With the advancement of microelectronic technology and the overall rising trend in the use of low cost microcontrollers the need to share information over the existing infrastructure is more and more emphasized. The problem that persists is how to implement Ethernet communication in low cost microcontrollers while retaining low cost of the device. This paper proposes the use of Microchip's Stand-Alone Ethernet Controller ENC28J60 in order to establish Ethernet communication towards the application located on a Host PC. In order to reduce the induced overhead on the existing microcontroller firmware size, the paper proposes the use of User Datagram Protocol (UDP) alongside with added authentication in the form of Basic Access Authentication using the Base64 algorithm to establish communication. The communication is tested using the Atmel AVR microcontroller architecture (Atmel AVR XMEGA) and the Stand-Alone Ethernet Controller whereas the sent data is displayed on a National Instruments LabVIEW application running on a Host PC. The measurement is carried out by using network protocol analysis and the comparison is made against the existing communication protocol (TFTP). The proposed communication is compared to one of the existing protocols, Trivial File Transfer Protocol (TFTP). The results are visible in a higher data rate and a lower flash size for implementation, representing an advantage over the existing protocols.*

Keywords – *communication, cost-effective, embedded system, Ethernet, LabVIEW , microcontroller, UDP*

1. INTRODUCTION

The rising trend in the production of embedded systems based on microcontroller architecture emphasizes more and more the need to establish effective and low cost communication over the existing infrastructure. According to [1], it is stated that semiconductor and embedded industry is projected to bloom from $3.25 billion in 2005 to $43.7 billion by 2015. With such attractive growth statistics, the field of embedded systems presents an interesting area for research (giving emphasis on communication in embedded systems). Due to the fact that this growth is propelled by the penetration of stand-alone low cost chips such as microprocessors and microcontrollers [1], the question that arises is how to establish an effective and secure way of communication in these low cost configurations.

Embedded systems contain processing cores that are typically either microcontrollers or *Digital Signal Processors* (DSP). In this paper, the term embedded system will focus on microcontroller based low cost embedded systems. These systems are used independently, where CAN, RS-232 and RS-485 are most commonly used communication technologies. Disadvantages of this communication are e.g. a low transmission rate, limited coverage, etc., which cause very difficult performance of flexible remote access and management [2]. However, *Ethernet (IEEE 802.3)* communication presents numerous advantages, such as a high ratio between performance and price, a high data rate, long-distance data transmission [3], thus presenting a viable alternative to the existing communication techniques.

This paper presents the design and implementation of *Ethernet* communication between a National Instruments *LabVIEW* application running on a Host computer and a microcontroller embedded system. The main advantage of the proposed communication is the use of a low cost *Stand-Alone Ethernet Controller* and the existing network infrastructures (*Ethernet* Local Area Network) to relay data between the embedded system and a Host PC, creating a viable and cost-effective alternative to the current microcontroller communication protocols (RS-232 or RS485). Data is transmitted by using UDP, with the added Basic Access Authentication to secure the access to the embedded system.

Ethernet communication is analyzed by using the network protocol regarding a data flow and a data rate. A comparison is made between the existing communication protocols. From the analyzed data rate it is clear that the proposed protocol presents a higher data rate compared to the TFTP protocol.

The following section describes the advantages of using *Ethernet*-based communication in the microcontroller embedded system. Also, the advanced implementation of the proposed system is shown by using the *Atmel's XMEGA* microcontroller and the SPI *Ethernet* module, and the communication protocol is proposed. Section 3 gives the results of *Wireshark* analysis with an emphasis on the data rate and the data flow, while Section 4 gives the conclusion.

2. ETHERNET COMMUNICATION IN LOW COST MICROCONTROLLERS

Ethernet (IEEE 802.3) is the most mature and widely used LAN technology. It provides the ability to connect the embedded device to the network device such as hub and switch, thus realizing flexible real time control and monitoring [4]. The main problem with using *Ethernet* communication in embedded systems is the lack of *Ethernet* support, i.e. NIC in low cost embedded systems [2] and [3]. *Network Interface Controller* is the core that implements the *Ethernet* protocol, and most of the low cost embedded systems do not provide necessary interfaces to communicate with *Ethernet*-based networks.

On the other hand, a small number of microcontrollers incorporate the necessary hardware and software requirements to enable *Ethernet* communication; such as the Microchip PIC18F97J60 family [5]. However, the presented problem is based on implementing *Ethernet* communication in the existing systems that do not incorporate the desired Ethernet functionality. The effective way to resolve this problem can be seen in the use of a *Stand-Alone Ethernet Controller*.

Furthermore, related work on this subject in the literature is versatile and covers a wide area. First of all, the emphasis on the embedded system's limited resources is discussed in [2], where the communication protocol section is not discussed. On the other hand, the authors in [3] implement a complete TCP communication in a high performance embedded system that supports a preinstalled operating system (i.e. *uClinux*). This presents a good method for data exchange and a cost-effective solution; however, it presents a problem for implementation in limited resources embedded systems. Similar work is also presented in [11] and [16]. Furthermore, the authors in [16] propose the use of ENC28J60 presenting a very cost-effective solution in combination with TCP communication. As stated before, due to limited resources in embedded systems, the implementation of the TCP stack induces higher additional overhead, so a logical choice for the communication protocol was the UDP protocol [4].

2.1. COST-EFFECTIVE ETHERNET COMMUNICATION

An important aspect in establishing Ethernet communication in low cost embedded systems satisfies the requirements for cost-effectiveness of the implementation. When designing an embedded system planned to be mass produced as a low cost system, all additional costs added to the overall price of the systems must be mitigated as much as possible. To give a comparison, an existing technology used for establishing Ethernet communication in embedded systems has a pricing range of $99 and more, whereas more cost-effective solutions do exist in other forms [6].

When calculating the overall cost of an embedded system, the important fact is how much the implemented communication will raise the overall price of the system, due to the multiplier of production quantity combined with the higher percentage of the total system cost it represents [7]. In general, if the embedded systems are in the price range of up to $300 USD, then implementing the communication proposed by [6] will raise the expenses by at least 33%. Due to a large increase in expenses of using standard equipment, this paper proposes the use of cost-effective Ethernet controllers, creating an increase in expenses of only 2% (according to [8], the retail module price ranges between $6.00 and $24.00 USD). The choice presents a cost-effective and easy to implement solution for existing embedded systems that do not incorporate the integrated Ethernet interface.

The solution presented here is the use of Microchip's ENC28J60 Stand-Alone Ethernet Controller, which uses the SPI interface to relay data to and from the microcontroller embedded system. ENC28J60 incorporates integrated MAC and PHY layers with 10BASE-T support and the ability to support UDP. The proposed solution is the presented method of communication in this paper along with the Atmel's XMEGA microcontroller as an embedded device. Hardware platform which enables Ethernet communication is shown in Figure 1.

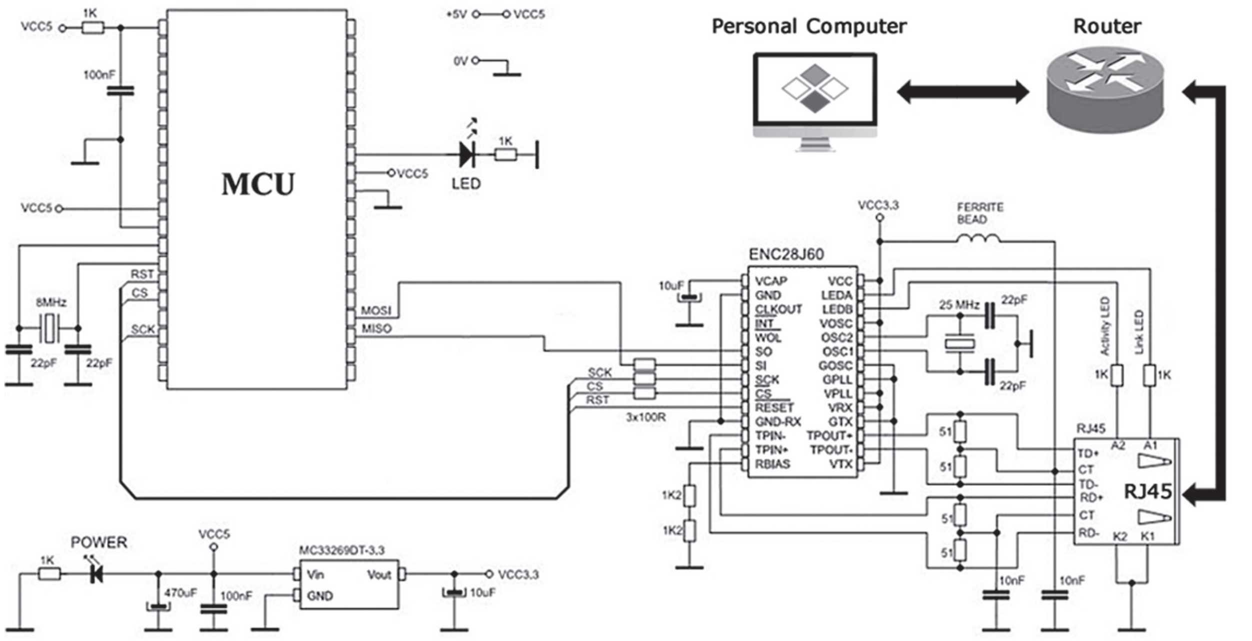


Fig. 1. Hardware platform for Ethernet communication

The realization of the proposed communication is based on establishing communication between the embedded system and the *Stand-Alone Ethernet Controller.* Due to the fact that the *Stand-Alone Ethernet Controller* uses a Serial Peripheral Interface to exchange the data, the implementation is further simplified, as the SPI is a widely used standard in embedded systems. The implementation of the communication protocol was carried out in the embedded system by using a MikroC compiler and the communication was established towards a *LabVIEW* application running on a Host PC.

Furthermore, the communication protocol used in data exchange must be specified. The *Stand-Alone Ethernet Controller* incorporates PHY and MAC layers; however, it does not incorporate any higher layer functions. These layers include the Internet and the Transport layer, and their implementation must be included in firmware development. Taken into consideration that simplicity of this design is the key feature, the most easily implemented and effective protocol for data exchange is the *User Datagram Protocol* (UDP). UDP is a connectionless protocol that can be used in embedded systems to replace existing standards such as RS-232 or RS-485. The main reason behind this claim is the simplicity of sending and receiving UDP packets. This process encompasses encapsulating data in UDP, IP and Ethernet headers, as well as parsing of the received packets. Due to the simplicity of the UDP implementation, the processor load and time delay is minimized.

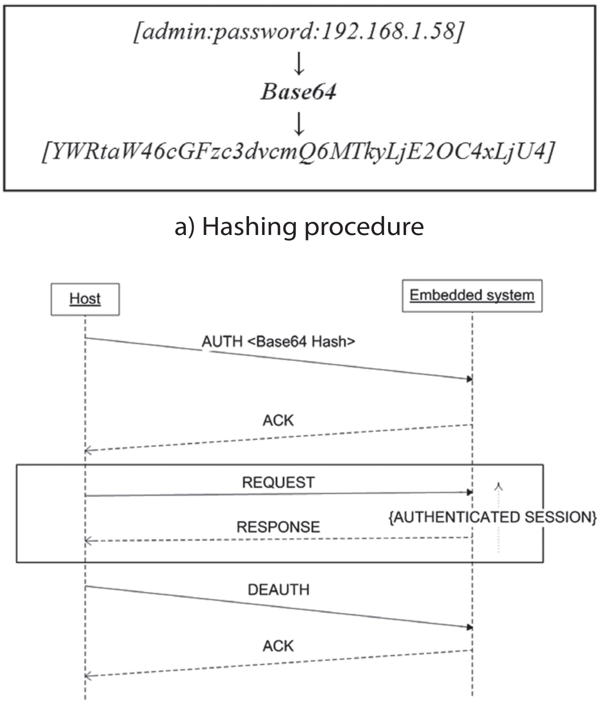
The overall simplicity of the proposed communication method is seen from the application point of view. From here, encapsulation and data transmission is performed by the lower layers (included in precompiled *Ethernet* libraries), so the process of establishing communication is further simplified. As stated, the communication is enabled by using precompiled libraries designed to support the Microchip's *Stand-Alone Ethernet Controller ENC28J60* in a microcontroller architecture which supports the UDP communication [9]. The practical application is demonstrated by using a microcontroller embedded system based on the Atmel's AVR XMEGA and *MikroC* (ANSI C) compiler (proprietary to electronic equipment manufacturer *MikroElektronika* [8]).

An additional advantage of the proposed method is the ability to implement the proposed Ethernet communication to various microcontrollers through the use of precompiled libraries (e.g. MikroElektronika's compilers offer support for PIC, dsPIC/PIC24, PIC32, AVR, C51 and ARM *Ethernet* support).

2.2. AUTHENTICATION AND COMMUNICATION PROTOCOL

To establish communication throughout an *Ethernet* environment an effective communication protocol for data exchange must be implemented. As stated before, data encapsulation is performed by the lower layers and precompiled libraries, so the microcontroller firmware is presented with direct data sent from the Host application. This communication is easily represented as a buffered RS-232 communication and in this form it can effectively replace existing microcontroller communication protocols.

Due to the fact that the *Ethernet* and LAN is a versatile environment coexisting with various systems, protocols and a large number of users, an effective way of securing the connection is required. By definition, UDP is a simple but unsecured communication protocol, so an additional security mechanism must be implemented to prevent unauthorized access. The simplest yet effective way of authorizing users is the use of the *Basic Access Authentication* algorithm together with the Base64 algorithm [10]. The Base64 function takes as an argument the username, password and IP address separated by semicolon (e.g. admin:password:192.168.1.58). The improvement from the original implementation of Basic Access Authentication is the IP address added to the hash string. The included IP address verifies the Host, thus preventing replay attack from different IP addresses. The authentication process is shown in Figure 2.



a) Hashing procedure

b) Authentication process

Fig. 2. Base64 hashing and Authentication flow

The Host sends an authorization request in the form AUTH <*hash*> (*AUTH YWRtaW46cGFzc3dvcmQ6…*). The embedded device responds with ACK if the authentication is successful or with ERROR if the process is unsuccessful. If the authentication was successful, the Host IP address is authorized and all data traffic from and to the designated IP address is allowed. If a user attempts to send a request to the Host from an unauthorized IP address, the Host will respond with the Host unauthorized response and will not permit any form of communication. The authenticated is able to send requests to the embedded device and receive data. Upon session completion, the Host sends a DEAUTH command closing authentication session of the Host's IP address. The microcontroller responds with ACK and the session is closed.

In the time frame of the authenticated session, the data exchange is enabled. The implemented protocol for data exchange is based on accessing data stored on a *microSD* memory card. Due to the fact that the accessed data are formatted in the FAT16 file system, the protocol is adapted to support the used file system, based on the existing protocol (Trivial File Transfer Protocol). The communication protocol supports the following: reading file content, writing to a file (Append mode and Rewrite mode) and deletion of a file. The stated support is realized by using UDP requests: RRQ (Read ReQuest) and WRQ (Write ReQuest). Also, the checksum was added in the EOF packet in the form of a total byte count during the transfer (e.g. EOF 000002E5). Due to the unreliable nature of the UDP protocol, the checksum could be designed so that it could detect received content out of order (e.g. a CRC type of checksum). In doing so, the order of messages could be preserved by means of shuffling packets and recalculating checksum on the Host side. On the microcontroller side, this would not be plausible, due to the limited resources of a microcontroller. The protocol communication flow is shown in Figure 3.

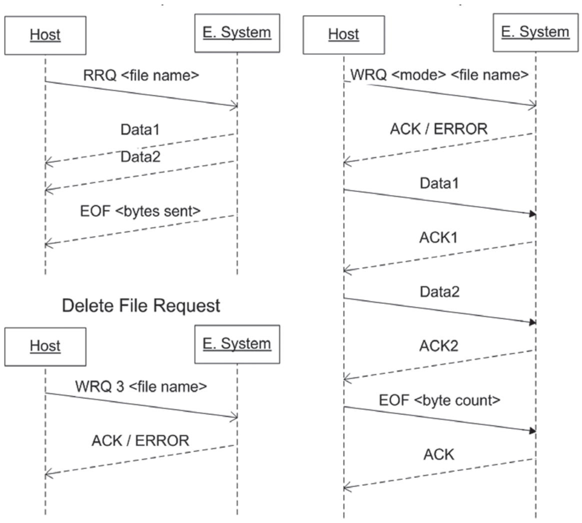


Fig. 3. Proposed protocol requests

In the proposed communication protocol, communication is always initialized from the Host. When a *Stand-Alone Ethernet Controller* receives a request in a UDP packet, the processing in performed inside an interrupt routine and the received request is processed. If the IP address is not authorized, the communication will not be possible. When the authorization process is completed, communication is enabled.

In order to establish a two-way communication, the Host sending the requests has to support the proposed protocol. The Host PC implements the proposed communication protocol using the National Instruments *LabVIEW* platform [12] and data exchange is enabled [13]. The *LabVIEW* application has the ability to access files related to the Access control system (authorization logs and users database) and to display or export data. Also, the application exhibits the functionality of manipulating user's database by using implemented write requests. The graphical "G" code is shown in Figure 4.

It is important to note that the Host device could be a similar microcontroller embedded system, in which case data transfer is established between two embedded systems and it presents an internal communication network functioning without the need for a server.

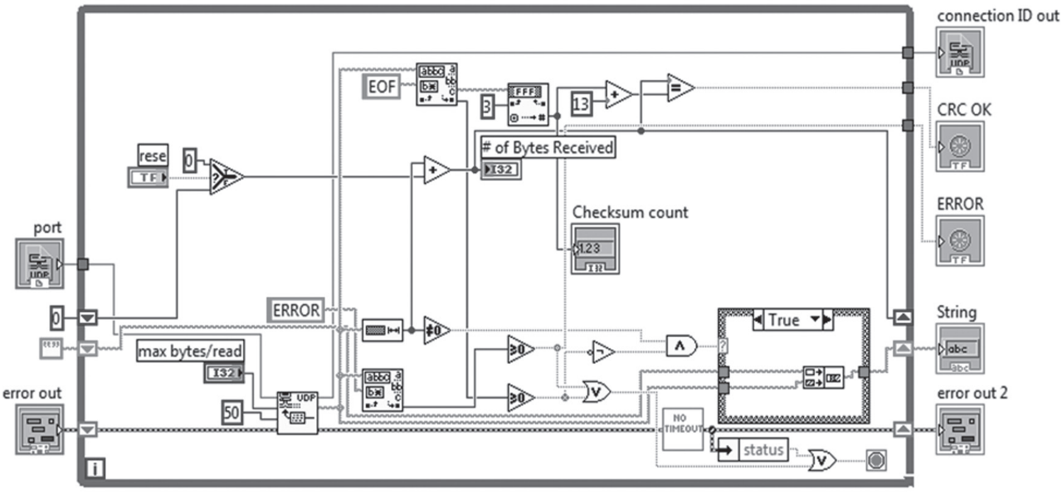


Fig. 4. "G" code of the LabVIEW application

It is important to note that the Host device could be a similar microcontroller embedded system, in which case data transfer is established between two embedded systems and it presents an internal communication network functioning without the need for a server.

2.3. TESTED MICROCONTROLLER EMBEDDED SYSTEM

The tested embedded system is based on microcontroller architecture and the *Stand-Alone Ethernet Controller ENC28J60*. The microcontroller embedded system used in this paper is an Access control system designed to acquire data from multiple RFID readers connected in a wireless *ZigBee* network [13]. In the original form, the access control system is realized using a BigAVR2 laboratory development system based on an ATMega128 microcontroller clocked at 8MHz alongside with the MMC Data card module, the Real Time Clock module, the *ZigBee* RF module and the ETH28J60 *Stand-Alone Ethernet Controller* [11], [13], [14] and [16].

The system used is realized using the ATXMega128A1 microcontroller and *MikroMedia for the XMEGA* development board, courtesy of equipment manufacturer *MikroElektronika (Figure 5)* [8]. *MikroMedia for XMEGA* is a compact high-quality multimedia development platform for XMEGA devices. It has numerous on-board modules that allow users to write multimedia applications and can be used both for development and as a final product [8]. The board contains an *Atmel XMega* microcontroller, *microSD* memory card periphery, a TFT LCD touch screen and several other modules. The use of a new microprocessor opposed to the original version demonstrates flexibility and diversity of the implemented communication protocol.

The microcontroller embedded system (Fig. 5) represents an Access control system whose primary purpose is to authenticate and authorize users by using RFID technology [13].

The user database is stored on external memory as well as the authorization logs due to the fact that the microcontroller is generally low on memory capacity to store the entire data. The form of the external memory is the micro Secure Digital (*microSD*) memory card (various capacities supported) formatted in the FAT16 format. Also, the electronic equipment manufacturer *MikroElektronika* proposes the use of a *MikroC* compiler that includes precompiled libraries which support a *microSD* data card and FAT16 file system manipulation. After implementing the proposed protocol in the microcontroller embedded system, testing of the protocol is enabled.

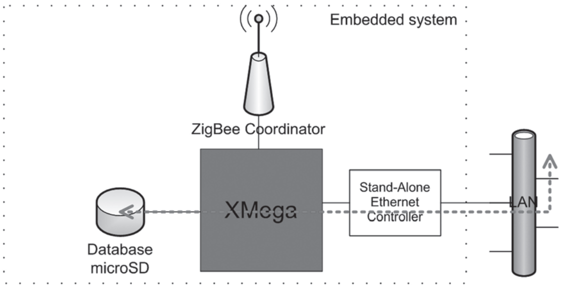


Fig. 5. Tested laboratory device block diagram

3. NETWORK PROTOCOL ANALYSIS

In order to analyze the performance of the proposed *Ethernet* communication, network protocol analysis was performed [15]. As stated in Section 2.3, the tested architecture is composed of an Access control system connected to the *Ethernet* LAN. In order to test the basic communication the network setup consisted of a network switch located in a real life LAN network. The Access control system was connected to the switch alongside with the PC used to request data and log data transmission. The proposed setup was established as a basic network setup in order to perform basic network testing. Extended testing with network condensation analysis was not conducted.

Network protocol analysis was performed in the following manner. First, the data flow was analyzed as well as the flow of Basic Access Authentication. As stated in the previous section, the authentication process must be performed before any data transmission occurs. In the beginning of the authentication process, a Host computer sends an authorization request in the form specified in the previous section. If the authorization process is successful, the embedded system responds with ACK and further communication is enabled. Upon completion of data transfer, the Host computer closes the session by sending a DEAUTH message to the embedded system. The embedded system ACK response concludes the authenticated session.

Furthermore, the data transmission rate is analyzed by using a different file size in both Read and Write mode (RRQ, WRQ). The analyzed parameters include Data rate, Transfer time and Average delay of the transmission. The first analysis depicts the UDP packet transfer through time. As stated, the protocol used for data exchange is UDP based and due to the nature of transmission all the data must be fragmented into packets. Figure 6 shows packet transfer in time, where the Read and Write requests are processed.

Figure 6 clearly depicts the communication protocol described in Section 2.1 (Fig. 3): A Host requests data from an embedded device (time 0.000s) and the embedded device responds with a series of packages seen in time frame 0.020s – 0.250s. These packages contain data fragmented to the maximum defined size of 1200 bytes. The rest of the data is sent in the remaining packet. This packet is seen as a penultimate packet with less size than previous packages. The last packet in line is the EOF packet that defines the end of the file and the data transfer as well. As Figure 6 shows, packet sizes seem to be relatively the same due to the use of a logarithmic scale.

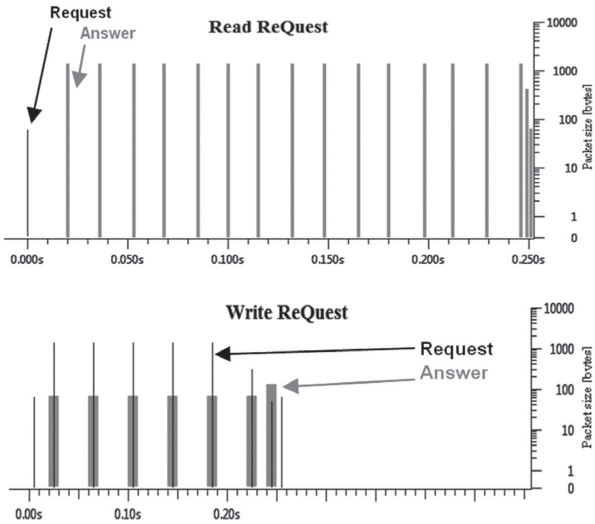
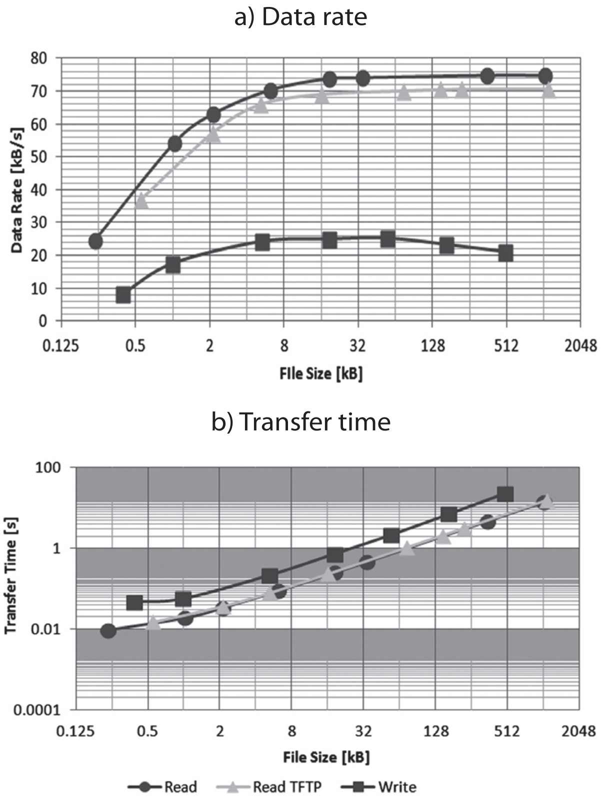


Fig. 6. Read Request and Write Request packages through time

In a lower graph in Fig. 6, the same protocol is shown in Write mode. The protocol flow shown in Fig. 4 is presented in time through packet transfer and the communication is vividly represented. One indistinct fact is the overlapping of the Sent data (thin black packet) and received ACK (thick red packet). The reason for this overlapping is the following: When a Host sends a Write Request (WRQ) to the embedded device, the device has to process the request and inform the Host if the file is found and if data transfer is ready. When the device responds with ACK, the Host immediately sends the data, thus creating a visible overlap.


a) Data rate

b) Transfer time

Fig. 7. Data Rate and Transfer Time versus File Size for TFTP and the proposed protocol

The time between packet transmissions and the embedded system depends on the system's processing speed. In this case, the processing time is approximately 50ms. From this timing and the timing of the received packets, it is possible to graph the data rate versus the number of packets sent, i.e. the size of a file. It is possible to calculate this relation due to several measurements of traffic on various different file sizes. Fig. 7 shows the given relation of transfer time and data rate.

From Fig. 7 it is clearly seen that the data rate improves by increasing the size of a file. The reason for this is the need to fragment data into packets and the added headers to the transmission. After a certain file size the data rate tends to saturate to approximately 75kB/s for the Read request and 25kB/s for Write request, as seen in Fig. 7. For smaller files the transfer speed tends to drop but the transfer time saturates around 10ms for Read request and 50ms for Write request.

A comparison of the proposed method and one of the existing file transfer protocols (e.g. TFTP [17], [18], [19]) is seen in Fig. 7. The proposed protocol exhibits a higher data rate versus the implemented *Trivial File Transfer Protocol* in read requests. The main reason for this is the fact that the TFTP uses sequential acknowledgements after each transmitted data block, creating an additional processing delay. This affects the overall data rate and reduces the data rate by 5%. By leaving out the sequential acknowledgement, the data rate is improved and the load on the embedded system is reduced. An important fact to note the bottleneck in this communication is the *microSD* memory card (Reading and Writing), which in turn diminishes the overall performance of this system. Also, the data rate of the Write request is additionally reduced by using sequential acknowledgement messages and parsing of the incoming data, on the side of the embedded device.

Regarding *Ethernet* communication in general, the module supports 10 Mbps data rates, resulting in the maximum possible data rate of 1.25 MB/s. Due to the fact that communication depends on the Round Trip Time parameter (RTT is 1.3ms), the theoretical maximum data rate that can be obtained is 774 packets per second resulting in the data rate of 900kB/s (7Mbps). This approaches the theoretical limit of the *Ethernet* module (10Mbps) and the speed of the SPI interface (8MHz) enabling a designer to implement high speed data transfer towards the *Ethernet* network.

4. CONCLUSION

This paper presents a theoretical approach and a practical implementation of *Ethernet*-based communication in low cost microcontroller based embedded systems. The proposed method utilizes the *Ethernet* network to transfer data between the Host computer and the embedded system. The proposed communication is based on UDP data transfer and it utilizes a *Stand-Alone Ethernet Controller ENC28J60* to relay data from and to the *Ethernet* network. The ease of implementation is seen through the use of standard SPI communication between the *Stand-Alone Ethernet Controller* and the embedded device, simplifying the implementation in existing embedded systems.

Another advantage is the added authentication that similar communication protocols do not incorporate (e.g. TFTP). The added feature of Basic Access Authentication additionally secures communication and enables data transfer through a versatile *Ethernet* environment. One disadvantage of using Basic Access Authentication is a possibility of the Man-in-the-Middle attack. If an attacker acquired the Base64 hash, the username and password could be easily extracted from the hash. This could be avoided by using more sophisticated authentication methods.

Data transfer for the proposed protocol was proven to be fairly large related to the processing power of the embedded system. A maximum throughput of 600kbps (75kB/s) for Reading and 200kbps (25kB/s) for Writing approaches the limit for broadband communication (>1Mbps), providing fast data transfer. According to the *Round Trip Time* (RTT) measurement, the maximal theoretical throughput is 7Mbps, approaching the theoretical limit of the *Stand-Alone Ethernet Controller* (10Mbps). Compared to the existing TFTP protocol, the proposed protocol improves the data rate by 5% by eliminating the sequential acknowledgements. It is important to note that the data rate is reduced by the *microSD* data card, which represents a bottleneck in this communication. Accordingly, the overall throughput of *Ethernet* communication could be improved significantly. Compared to the data rate of RS-232 (250kbps) and RS-485 (up to 10Mbps for 12m and 100k for 1200m), *Ethernet* communication is well over the RS232 protocol and within the specified range of RS485 [23].

Our future work involves securing additionally authentication and data stream by applying encryption algorithms and a stronger authentication method. Also, the Man-in-the-Middle attack could be avoided by using *nonce* hashing and more secure *hash* functions (e.g. SHA-1).

ACKNOWLEDGEMENTS

This work was sponsored by private funding together with funds provided by electronic equipment manufacturer *MikroElektronika* (Belgrade, Serbia) in form of equipment donation including laboratory based development systems, compiler licenses and various modules used, that made this work possible.

5. REFERENCES

[1] J. Parab, S.A. Shinde, V.G Shelake, R.K. Kamat, G.M. Naik, "Practical Aspects of Embedded System Design using Microcontrollers", Springer 2008, XXII, p. 150.

[2] Z. Chuan-Sheng, F. Chong, "Implementation of a General Ethernet Interface for Embedded System", Proc. of HIS 2009 Ninth International Conference on Hybrid Intelligent Systems, Vol. 3, Shenyang, LiaoNing, China, August 2009, pp. 58-62.

[3] C. Wu, B. Wang, "Design and Implementation of Embedded Ethernet Accessing in Data Collection and Processing System Based on ADSP-BF533", Proc. of International Conference on Embedded Software and Systems Symposia (ICESS2008), Chengdu, Sichuan, China, July 2008.

[4] J. Sommer, S. Gunreben, F. Feller, M. Kohn, A. Mifdaoui, D. Sass, J. Scharf, "Ethernet – A Survey on its Fields of Application," Communications Surveys & Tutorials, IEEE, Vol.12, No.2, Second Quarter 2010, pp. 263-284.

[5] Electronic equipment manufacturer Microchip. Online available: http://www.microchip.com (December 2011).

[6] NetBurner, Networking in 1 day. Online available: http://www.netburner.com/ (January 2012).

[7] P.J. Koopman, "Embedded System Design Issues", Proc. of International Conference on Computer Design (ICCD 96), Austin, Texas, USA, 1996.

[8] Mikroelektronika, Serbia, Electronic equipment manufacturer. Online available: http://www.mikroe.com (November 2011).

[9] D. Šoštarić, D. Vinko, D. Žagar, "JavaScript Virtual Web Page for Wireless Sensor Node under AVR Microcontroller Architecture", Proc. of WMNC 2010 Wireless and Mobile Networking Conference, Budapest, Hungary, October 2010, pp. 1-5.

[10] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication", RFC Editor, 1999.

[11] W. Mei, Z. Ruimei, H. Huiting, "Design and Realization of Embedded Ethernet Communication System Based on ARM", Proc. of ISISE 2009 Second International Symposium on Information Science and Engineering, Shanghai, China, December 2009, pp. 287-289.

[12] National Instruments LabVIEW Software Platform. Online available: www.ni.com (December 2011).

[13] G. Horvat, D. Šoštarić, D. Žagar, "User authorization system using ZigBee WSN and AVR architecture", Proc. of 19th Telecommunications Forum TELFOR 2011, Belgrade, Serbia, November 2011.

[14] Atmel AVR XMEGA ATXMega128A1 Datasheet, www.atmel.com, November 2011.

[15] Wireshark: the world's foremost network protocol analyzer, Online available: http://www.wireshark.org (December 2011).

[16] B. Tan, B. Yuan, B. Zhu, "The Design of Ethernet Controller Interface Circuit Based on ENC28J60", Proc. of Second International Symposium on Networking and Network Security, Jinggangshan, China, 2-4, April 2010, pp. 035-038.

[17] K. Sollins, "The TFTP Protocol (Revision 2)", RFC 1350, July 1992.

[18] G. Malkin, A. Harkin, "TFTP Option Extension", RFC 2347, May 1998.

[19] G. Malkin, A. Harkin, "TFTP Blocksize Option", RFC 2348, May 1998.

[20] P. Kocher, R. Lee, G. McGraw, A. Raghunathan, S. Ravi, "Security as a new dimension in embedded system design", Proc. of 41st Design Automation Conference, 7-11, July 2004, pp. 753-760.

[21] Summary of RS-232, RS-422, and RS-485 Interface Standards, Toshiba America Information Systems, Security Products Group, Application Note, August 2002.