TELKOMNIKA Telecommunication, Computing, Electronics and Control
Vol. 19, No. 3, June 2021, pp. 801~808
ISSN: 1693-6930, accredited First Grade by Kemenristekdikti, Decree No: 21/E/KPT/2018
DOI: 10.12928/TELKOMNIKA.v19i3.18764

Customer application protocol for data transfer between embedded processor and microcontroller systems

Mazin R. Khalil, Laith A. Mohammed, Omar N. Yousif
Technical Engineering College, Northern Technical University Mosul, Iraq

Article Info

Article history:

Received Aug 7, 2020
Revised Nov 5, 2020
Accepted Nov 25, 2020

Keywords:

Customer application protocol
Embedded processor system
Ethernet
Lightweight IP

ABSTRACT

This paper develops a new customer application protocol (CAP) to improve the efficiency of transferring data between embedded processor and microcontroller systems. The established protocol is characterized by its fidelity and simplicity for using a small header to control and monitor the data flow between the two systems. This is achieved by constructing an embedded processor system with an Ethernet intellectual property (IP) core featured by lightweight IP (lwIP) to settle a connection with a microcontroller device. The embedded system is configured on spartan6E FPGAs slice. The system performance is tested by transferring audio samples and displaying them on chipscope media. The performance test of the designed embedded system with the developed customer application protocol showed fast, efficient and high precision data exchange between the processor and microcontroller systems.

Corresponding Author:

Omar N. Yousif
Department of Computer Technology Engineering
Nourthern Technical University
Maj moah, Mosul, Iraq
Email: nabilomar237@gmail.com

1. INTRODUCTION

Now a day, electronic devices that require to communicate with each others are mostly connected to networks with real time system. This is done by using TCP/IP protocol. Lightweight IP (lwIP) is open source TCP/IP networking stack that is used originally and developed by Adam Dankels [1], It seizes small memory size (random access memory (RAM) and read only memory (ROM)) to conform with embedded system prerequisites [1-5]. The lightweight IP (lwIP), available under the Berkeley software distribution (BSD) license, enables the designed processor system to configure and control the layers in TCP/IP protocol.

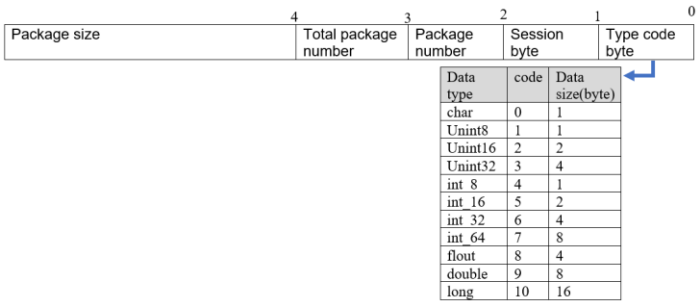
A microcontroller architecture to measure various quantities for distinguishing water quality by a simultaneously distributing data over the internet was developed by [6]. Uploading new files and programs to the embedded web server using file transfer protocol (FTP) was presented by Stipanicev and Jadranka [7]. An embedded direct current (DC) motor position control system with user datagram protocol (UDP) was designed by Ahmed *et al.* [8]. Schema of download operation to the evaluation board using the trivial file transfer protocol (TFTP) server on the advanced reduced instruction set computing (RISC) machine (ARM) based architecture microcontroller LPC2210 was presented by Qiu *et al.*, [9]. Free RTOS operating system based on hypertext transfer protocol (HTTP) was envisaged by Moritz *et al.* [10]. In application programming interface (API) standardized application based on HTTP protocol was developed by Xu [11]. Machine to machine (M2M) communication based on HTTP protocol was discussed by Sharan and Asati [12].

Based on the results obtained from the above mentioned previous works, it is clear that, there is a need to develop a simple and efficient customer application protocol, customer application protocol (CAP) to

facilitate transferring data between a processor and microcontroller system, Arduino mega2560 [13, 14], this target can be performed by constructing a processor system with Ethernet IP core to settle data link with a microcontroller device using embedded design techniques and the developed CAP protocol. The application is developed using C language at the processor side in the software development kit (SDK) platform [2, 15] and using C++ language at the integrated development environment (IDE) at the microcontroller side [16]. The remaining of the paper implies section 2 that presents the CAP, data representation. Section 3 deals with system hardware construction, Result and conclusions are presented in section 4 and section 5 respectively.

2. CUSTOMER APPLICATION PROTOCOL (CAP)

The data is represented in different fashions according to host operation system. It's sent as unsigned integer bytes, and then at receiver side, it is reconstructed to its original form. Customer application protocol (CAP) is used to control the data flow and reconstruction. The CAP protocol, developed here, implies adding a header control to the data sent that consists of data type byte, session byte, package number, total package number and two bytes for package length as shown in Figure 1.



Data type	code	Data size(byte)
char	0	1
Unint8	1	1
Unint16	2	2
Unint32	3	4
int_8	4	1
int_16	5	2
int_32	6	4
int_64	7	8
flout	8	4
double	9	8
long	10	16

Figure 1. Developed customer application header protocol for data transfer between a microcontroller and an embedded processor system

2.1. Data representation

For exchanging data between the two systems, two points need to be considered. The first point is that the microblaze uses big-endian bit-reversed data format [17] while the microcontroller system uses little-endian format to represent the data, also the data transmission in CAP protocol was performed in little-endian mode. Figure 2 shows the big-endian data format representation and Figure 3 shows the location of the data bytes on memory in each mode. CAP protocol takes care of this point by using global variable (extern variable) as a buffer stored in the heap memory. As the heap memory grows [18-20] in a reverse direction to the stack memory growth [21], the problem of different data format is solved. The second point is the data type. CAP protocol header has data type code byte that helps to fetch data from extern buffer as the original data type, both systems have data type code table that can be used as reference. Data type code depends on data type and data size such as int8, and unint16.

2.2. Flow control

The flow control bytes (session byte, package number byte, total package number byte, size package bytes), error message, and acknowledge message enable data flow under monitoring and control of customer application protocol. The session byte prevents interference between new and old data transmitted by the same host. Package number and package length are used in data fragmentation [22-24], the bytes in the original package are numbered from 0 to 65,536. The calculation for the packages offset is determined in (1). The maximum data transfer through CAP protocol is 2^{24}-1 byte because package offset is 24 bits. In case the size of the used package is larger than 16 bits, the scaling factor (S) is added to (2). Consequently, the maximum data transfer depends on scaling factor value. The total package byte is used to determine the end of transmission and free cookies for new transmit.

$$\text{Package offset} = \text{number package} * \text{package size} \quad (1)$$

$$\text{Package offset} = \text{number package} * \text{package size} * S \quad (2)$$

The client has to ensure that every package sent when arrives is error-free, in this case an acknowledgment message as shown in Figure 4 is sent to the clients. The connection will be lost if the client doesn't receive the

acknowledgement message. If any error type of those shown in Table 1 accur an error message as shown in Figure 5 will be sent.

Byte address	n	n+1	n+2	n+3
Byte label	0	1	2	3
Byte significance	MSByte			LSByte
Bit label	0			31
Bit significance	MSBit			LSBit

Figure 2. Big-endian data format

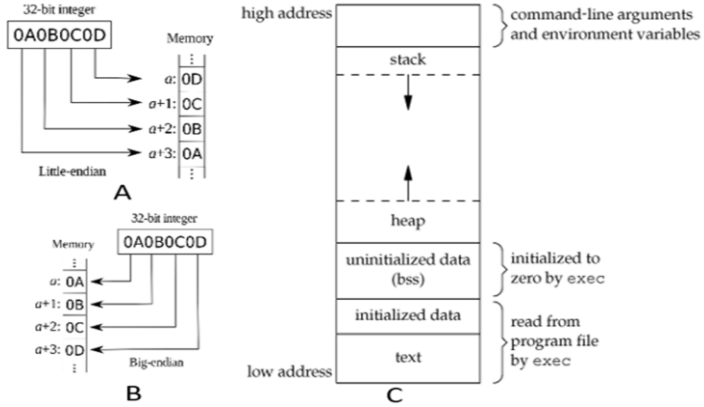


Figure 3. (a) Little-endian data format, (b) Big-endian data format, (c) Memory map for heap and stack memory

Table 1. Error type

NO.	Error type	Code
1	Data type error	1
2	Session	2

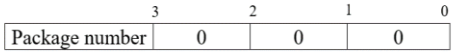


Figure 4. Acknowledge message

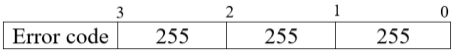


Figure 5. error message

2.3. Cookies

Large amount of data is divided in to more than one package as continuous group packages, especially in API ROW mode, the added cookies are used to hold the information about the last connected host as shown in Figure 6. It contains last IP client, copy of the last package CAP protocol header and numbers of package received. The cookies allow the CAP protocol to combine the transmitted packages data.

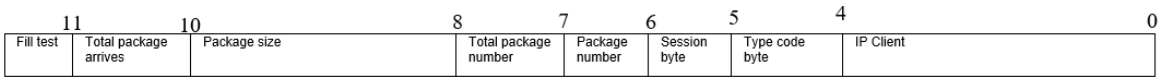


Figure 6. Cookies structure

2.4. Software application development

The flow chart of the customer application protocol developed in server software development kit (SDK) environment is depicted in Figure 7. The flowchart contains the codes in c language with note for each step. It explains implementation of CAP protocol in server side. Figure 8 presents the flow chart of the customer application protocol developed in client integrated development environment (IDE) environment. The flowchart contains the codes in C++ language (Arduino) with note for each step. It explains implementation of CAP protocol in client side.

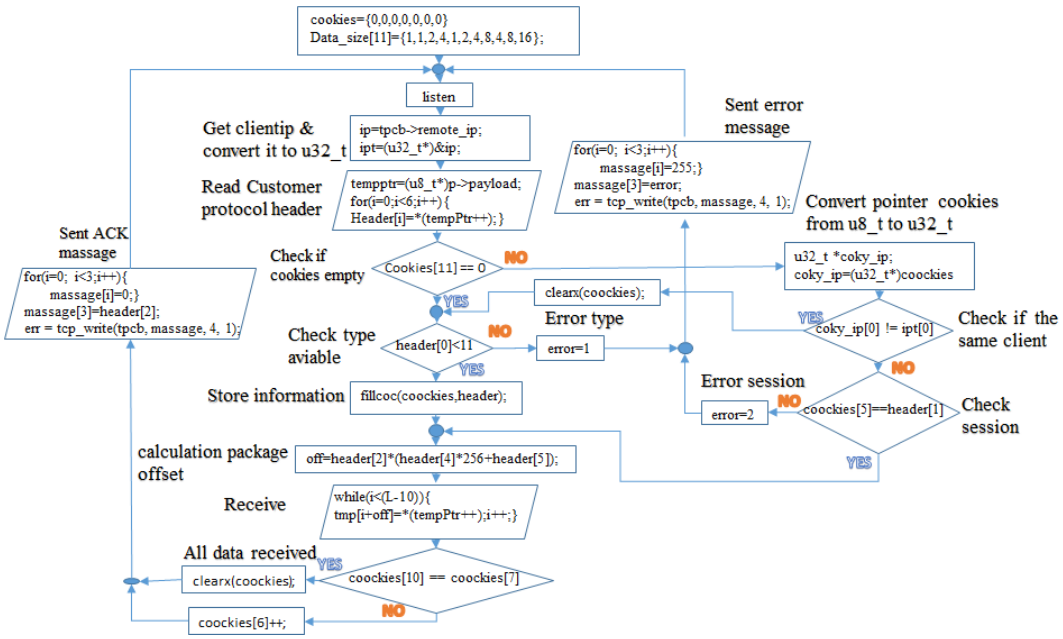


Figure 7. The flowchart for the customer application protocol of the server

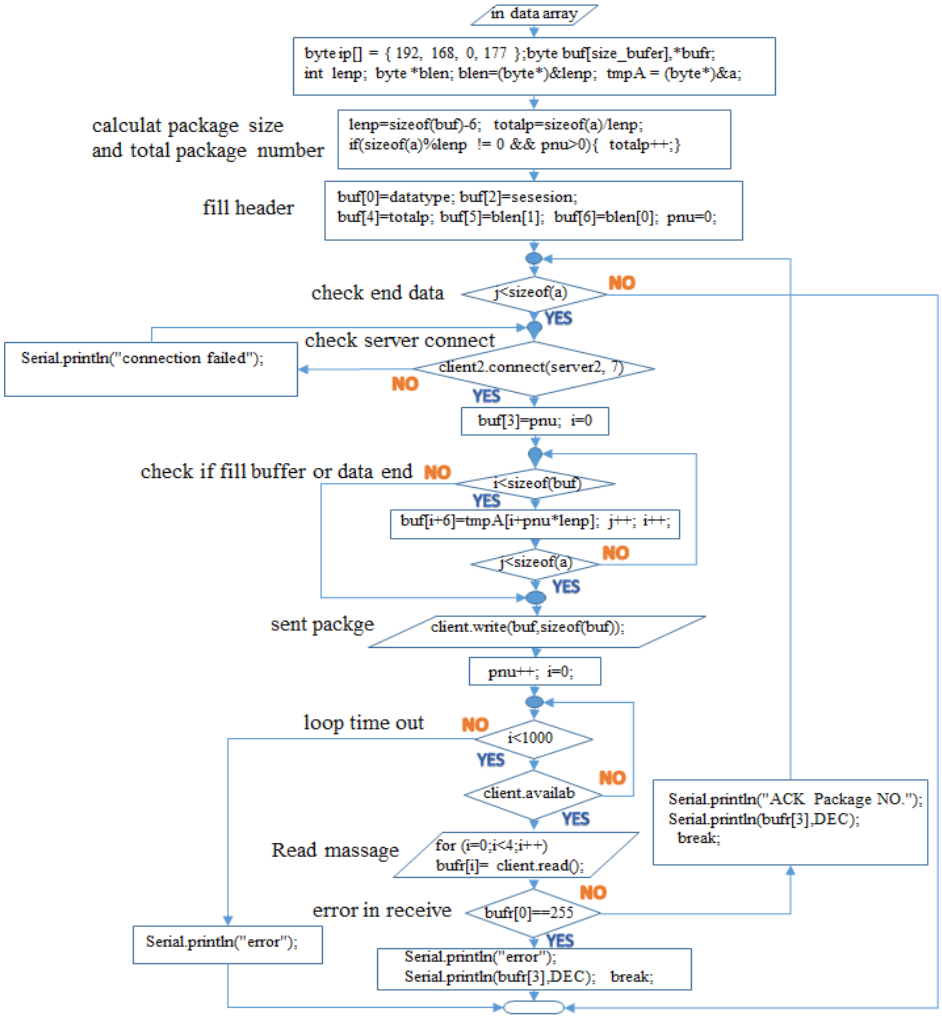


Figure 8. The flowchart of the customer application protocol of the client

3. SYSTEM HARDWARE DEVELOPMENT

The designed embedded processor system is shown in Figure 9. The block diagram of the designed embedded processor system with Ethernetlite IP core that act as a media access controller (MAC) operating with interrupt active mode [25, 26]. The designed system uses AXI4 interconnect module that act as system bus [27], memory controller MCB-DDR2 [28] to control a 128 M byte dual data rate memory (DDR-SDRAM) READ/WRITE operations, an interrupt controller (INTC) [29] to deal with the Ethernetlite IP core and timer interrupt signals, AXI4 timer [30] and other necessary peripherals.

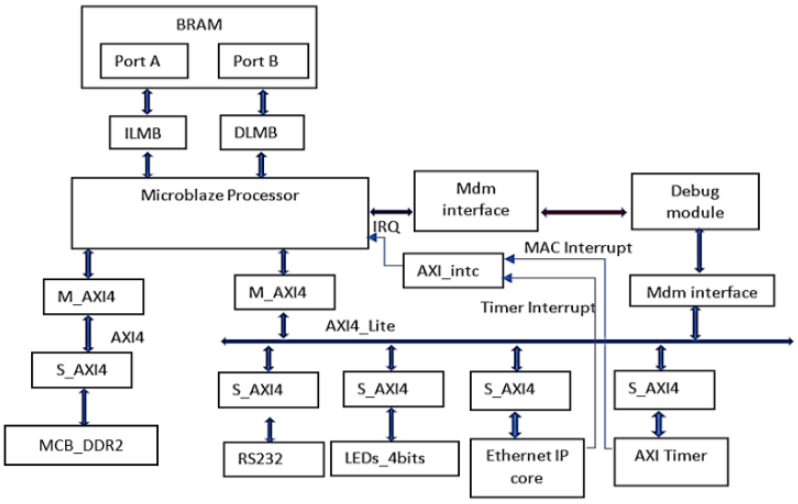


Figure 9. Block diagram of the hardware part of the designed processor system

4. RESULTS AND ANALYSIS

The Data Exchange Link is tested by sending an audio record (4 kByte size) from Matlab (PC) to microcontroller with 5 packages, and is then transmitted from microcontroller to microprocessor designed system with 16 packages. All transmissions are done under customer protocol. Figure 10 shows the audio signal created in Matlab media. Figure 11 shows a samples of audio signal created in Matlab, samples of the audio signal transferred to microcontroller media and samples of the audio signal transferred to microprocessor media. In comparison with previous works, the CAP is considered flexible and powerful for data transfer. It is also accurate and characterised by its low complexity with lower package control number. Figure 12 displays the data received by the microprocessor system on chipscope windows. Each window display one K byte of data.

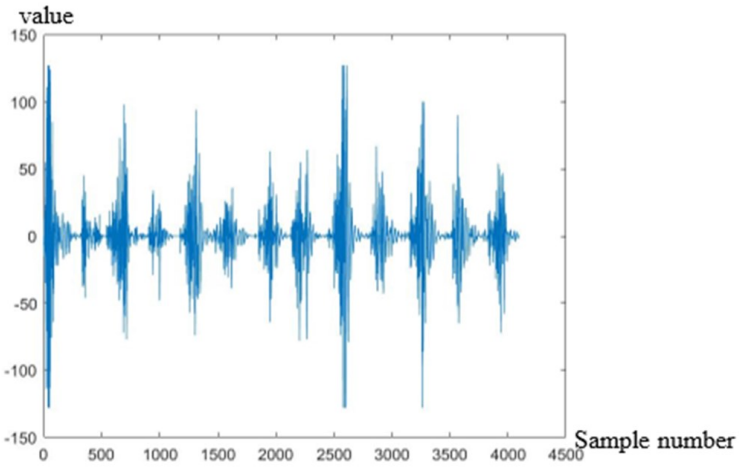
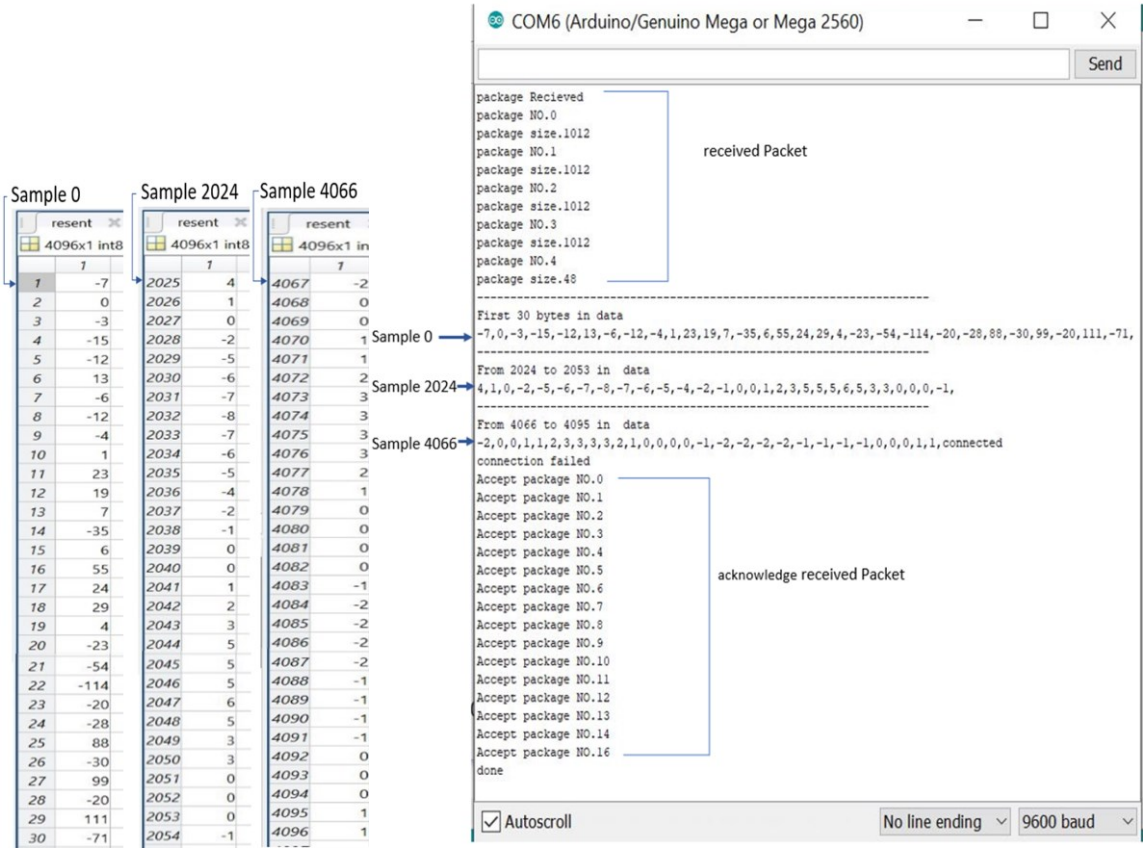
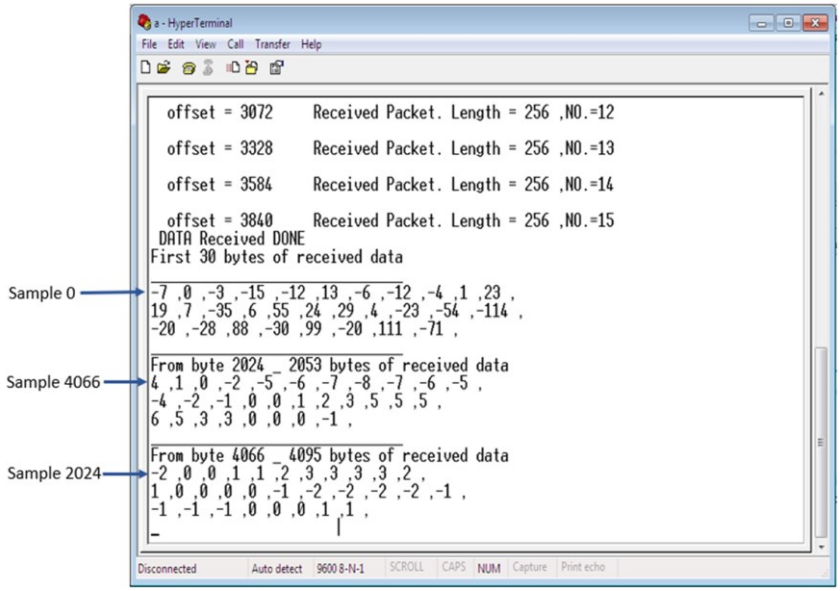


Figure 10. Original audio record data



(a) (b)



(c)

Figure 11. (a) Samples of audio data in Matlab, (b) Samples of audio data in microcontroller, (c) Samples of audio data in microprocessor design system

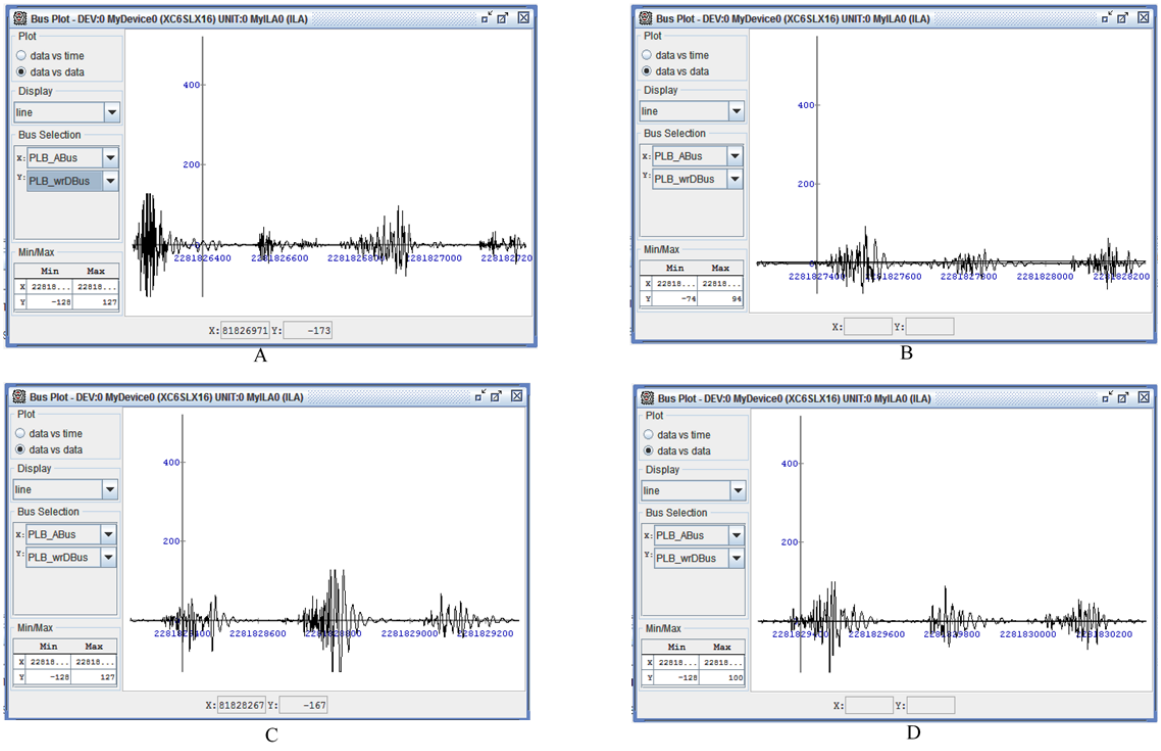


Figure 12. (a) 1ST KB of received data, (b) 2nd KB of received data, (c) 3rd KB of received data, (d) 4th KB of received data

5. CONCLUSIONS

A sophisticated data transfer system was designed to exchange data between microcontroller and embedded microprocessor systems using ethernet IP core and a developed customer application protocol. The proposed system is characterized by following: lwIP facilitates applying a TCP/IP protocol for the designed system. The developed customer application protocol offer a proficient control on transfer system. The data transfered in customer application protocol was maximized to 16 MB with ability to be expanded. The system is with high precision data transfer such that percentage error in the received data is (0.01-0.2%).

REFERENCES

[1] A. Dunkels, "Design and Implementation of the lwIP TCP/IP Stack," *Swedish Institute of Computer Science,* pp. 40-46, 2001.
[2] J. Xu, "The research and implementation of embedded TCP/IP protocol stack," in *Proceedings of the 2012 International Conference on Computer Application and System Modeling,Atlantis Press*, 2012, pp. 0584-0587.
[3] Q. Hui and Q. Li, "Implementation of LwIP TCP/IP protocol stack based on S1C33E07," in *Software Engineering and Knowledge Engineering: Theory and Practice, Springer, Berlin, Heidelberg,* pp. 635-642, 2012.
[4] J. Moon and M. Yoon, "an Implementation of a Configurable Serial-To-Ethernet Converter Using Lwip," *Science International (Lahore),* vol. 29, pp. 103-107, 2017.
[5] S.Pasca *et al.*, "Architectural challenges and solutions for collocated LWIP-A network layer perspective," in *2017 Twenty-third National Conference on Communications (NCC), IEEE,* 2017, pp. 1-6.
[6] O. Postolache *et al.*, "An Internet and Microcontroller-Based Remote Operation Multi-Sensor System for Water Quality Monitoring," *Proc. IEEE Sensors*, vol. 1, 2002, pp. 1532-1536.
[7] D. Stipanicev and M. Jadranka, "Networked embedded greenhouse monitoring and control," in *Proceedings of 2003 IEEE Conference on Control Applications, 2003. CCA 2003, IEEE,* 2003, pp. 1350-1355.
[8] I. Ahmed *et al.*, "Internet-based remote control using a microcontroller and an embedded ethernet," *Proceedings of the 2004 American Control Conference,* vol. 2, 2004, pp. 1329-1334.
[9] S. Qiu *et al.*, "Building TFTP server on embedded system," in *2008 4th International Conference on Wireless Communications, Networking and Mobile Computing, IEEE,* 2008, pp. 1-4.
[10] G. Moritz *et al.*, "Web services on deeply embedded devices with real-time processing," in *2008 IEEE International Conference on Emerging Technologies and Factory Automation, IEEE,* 2008, pp. 432-435.
[11] Q. Xu, "A Design and implement of IAP based on HTTP," in *2011 International Conference on Computer Science and Service System (CSSS), IEEE,* 2011, pp. 1918-1922.

[12] R. Sharan and D. Asati, "Development of Embedded Web Server Configured on FPGA Using Soft-core Processor and Web Client on PC," *International Journal of Engineering and Advanced Technology (IJEAT)*, vol. 1, pp. 295-298, 2012.
[13] M.Kusriyanto and D. Bambang, "Smart home using local area network (LAN) based arduino mega 2560," in *2016 2nd International Conference on Wireless and Telematics (ICWT), IEEE*, 2016, pp. 127-131.
[14] Z. Raiyan *et al.*, "Design of an arduino based voice-controlled automated wheelchair," in *2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC), IEEE,* 2017, pp. 267-270.
[15] S. Zoican and M. Vochin, "LwIP stack protocol for embedded sensors network," in *2012 9th International Conference on Communications, COMM 2012-Conference Proceedings,IEEE,* 2012, pp. 221-224.
[16] A. Macker *et al.*, "ARDUINO Based LPG Gas Monitoring Automatic Cylinder Booking with Alert System," in *2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI), IEEE*, 2018, pp. 1209-1212.
[17] L. Moss, *et al.*, "Automation of communication refinement and hardware synthesis within a system-level design methodology," in *2008 The 19th IEEE/IFIP International Symposium on Rapid System Prototyping, IEEE*, 2008, pp. 75-81.
[18] M. Mihara *et al.*, "Negative heap pump for low voltage operation flash memory," in *1996 Symposium on VLSI Circuits. Digest of Technical Papers, IEEE,* 1996, pp. 76-77.
[19] G. Duck and R. H. Yap, "Heap bounds protection with low fat pointers," in *Proc. CC 2016 25th Int. Conf. Compil. Constr.,CC2016*, 2016, pp. 132-142.
[20] W. Li *et al.*, "A wear leveling aware memory allocator for both stack and heap management in pcm-based main memory systems," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), IEEE*, 2019, pp. 228-233.
[21] Y. Afek *et al.*, "The velox transactional memory stack," *IEEE micro*, vol. 30, pp. 76-87, 2010.
[22] J. Yoon *et al.*, "Data fragmentation scheme in IEEE 802.15. 4 wireless sensor networks," in *2007 IEEE 65th Vehicular Technology Conference-VTC2007-Spring, IEEE,* 2007, pp. 26-30.
[23] Q. Li, Qing *et al.*, "Process and data fragmentation-oriented enterprise network integration with collaboration modelling and collaboration agents," *Enterprise Information Systems*,vol. 9, pp. 468-498, 2015.
[24] M. Atmani *et al.*, "Modelling and analysis data fragmentation in IEEE 802.15. 4 slotted CSMA/CA protocol without ACK mode," *International Journal of Critical Computer-Based Systems* ,vol. 7, pp. 4-21,2017.
[25] L. Min *et al.*, "Realization of led urban lighting network based on powerlink industrial ethernet," in *2014 7th International Conference on Intelligent Computation Technology and Automation, IEEE,* 2014, pp. 445-448.
[26] T. Corrêa *et al.*, "Hardware/Software Implementation Factors Influencing Ethernet Latency," in *2018 IEEE 16th International Conference on Industrial Informatics (INDIN), IEEE,* 2018, pp. 323-328.
[27] R. Bhaktavatchalu *et al.*, "Design of AXI bus interface modules on FPGA," in *2016 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT), IEEE*, 2016, pp. 141-146.
[28] E. Ragab *et al.*, "DDR2 Memory Controller for Multi-core Systems with AMBA AXI Interface," in *2018 30th International Conference on Microelectronics (ICM), IEEE,* 2018, pp. 224-227.
[29] K. Naga and K. Tarangini. "Design and development of AXI based multi channel interrupt controller2," *International Journal of Managment, IT and Engineering* vol 3, pp. 108-125, 2013.
[30] B. Kanigoro *et al.*, "Overview of Custom Microcontroller using Xilinx Zynq XC7Z020 FPGA," *TELKOMNIKA Telecommunication Computing Electronics and Control,* vol. 13, pp. 364-372, 2015.