J. Mechatron. Electr. Power Veh. Technol. 06 (2015) 75-82

Journal of Mechatronics, Electrical Power, and Vehicular Technology

e-ISSN: 2088-6985
p-ISSN: 2087-3379

www.mevjournal.com

ALGORITHM OF 32-BIT DATA TRANSMISSION AMONG MICROCONTROLLERS THROUGH AN 8-BIT PORT

Midriem Mirdanies *, Hendri Maja Saputra, Estiko Rijanto
Research Centre for Electrical Power and Mechatronics, Indonesian Institute of Sciences (LIPI)
Komp LIPI Bandung, Jl. Sangkuriang, Gd. 20. Lt. 2, Bandung 40135, Indonesia

Received 24 July 2015; received in revised form 13 October 2015; accepted 21 October 2015
Published online 30 December 2015

Abstract

This paper proposes an algorithm for 32-bit data transmission among microcontrollers through one 8-bit port. This method was motivated by a need to overcome limitations of microcontroller I/O as well as to fulfill the requirement of data transmission which is more than 10 bits. In this paper, the use of an 8-bit port has been optimized for 32-bit data transmission using unsigned long integer, long integer, and float types. Thirty-two bit data is extracted into binary number, then sent through a series of 8-bit ports by transmitter microcontroller. At receiver microcontroller, the binary data received through 8-bit port is reconverted into 32 bits with the same data type. The algorithm has been implemented and tested using C language in ATMega32A microcontroller. Experiments have been done using two microcontrollers as well as four microcontrollers in the parallel, tree, and series connections. Based on the experiments, it is known that the data transmitted can be accurately received without data loss. Maximum transmission times among two microcontrollers for unsigned long integer, long integer, and float are 630 µs, 1,880 µs, and 7,830 µs, respectively. Maximum transmission times using four microcontrollers in parallel connection are the same as those using two microcontrollers, while in series connection are 1,930 µs for unsigned long integer, 5,640 µs for long integer, and 23,540 µs for float. The maximum transmission times of tree connection is close to those of the parallel connection. These results prove that the algorithm works well.

Keywords: transmission algorithm; 32-bit data; data transmission; 8-bit port; microcontroller; C language.

I. INTRODUCTION

In a complex system, the use of multiple microcontrollers is typically required to handle each sub section. Therefore, a communication among microcontrollers is needed [1, 2]. The number of the microcontrollers that can be connected and the number of data that can be sent are very determines in the communication media selection. The communications media on the microcontroller is limited and data size that can be sent is usually 8-10 bits.

Research that applied communication between microcontrollers via UART using Zigbee wireless have been carried out by Reddy [3] and Thakur [4]. A similar thing has been done by Saputra [1] using a YS-C20K type of wireless module. In Leeman research [5], communication between microcontrollers via UART was performed using RS-232 cable, while Solanke [6] using a wireless RF at frequency 433.92 MHz. Data communication via UART / YS-C20K will be troublesome if the communication is done among many microcontrollers simultaneously. Research that applied communication via the CAN bus have been done by Prickett [7] and Kutlu [8], however, communication using this medium requires an additional interface. Moreover, it is difficult to be implemented on an 8-bit microcontroller with the minimum system which does not provide an embedded CAN controller i.e. ATmega8/ ATmega8535.

Communication among the microcontrollers also can be made using an 8-bit port. Research that applied communication among the microcontrollers through an 8 port have been done by Saputra [1] and Mirdanies [2], however the number of data are limited to 8 bits only (0-255 decimal). Communication among the microcontrollers through multiple 8-bit ports can also be done by adding port expander [9], but this

* Corresponding Author.Tel: +62-22-2503055
E-mail: midriem.mirdanies@lipi.go.id

doi: 10.14203/j.mev.2015.v6.75-82

method is not optimal due to the addition of several devices and coding is not practical.

This paper proposes a new algorithm for 32-bit data transmission among microcontrollers through one port which is consisted of 8 bits. The data types used are long integer, unsigned long integer, or float types [10]. This method is used to overcome the limitations of the microcontroller I/O for connectivity among microcontrollers without the use of additional interfaces, and requirement of data transmission more than 10 bits. Communication can be done among many microcontrollers simultaneously using parallel, tree, or series connection. Experiments have been done using four ATMega32A microcontroller boards [11].

II. METHOD/MATERIAL

An example connection between two microcontroller can be seen in Figure 1. Several pins used on the port partialy functioned as identifier while others as data. Pins configuration for unsigned long integer, long integer, and float types can be seen in Figure 2, Figure 3, and Figure 4. Pins used as data in both unsigned long integer and long integer data types are pin 0-4, while in float data type are pin 0-3. In the second block of data transmission, pin 0 in long integer and float serves as an identifier that the data sent is positive (0) or negative (1). Pin 4 on float serves as the identifier that the data sent is an integer (0) or fractions (1). Pin 5 in any data type serves as the identifier from the transmitter indicating that the data is ready to be read by the receiver. Pin 6 as an identifier from the transmitter indicating that the data sent is the last data, and pin 7 serves as the identifier from the receiver which indicates that the data has been read/received.

Connection among microcontrollers described in this paper is not limited for two microcontroller, but can be used for communication with many microcontrollers at the same time in parallel, tree, or series connections. An example of each connection type used can be seen in Figures 5, 6, and 7.

Specifically in tree connection, an additional confirmation step is required for sending data to a specific microcontroller. This step to ensure that the data transmitted to the microcontroller target is correct. Therefore, the number of microcontrollers that can be installed is limited to $2^6 = 64$ units. Pin configuration used in this step can be seen in Figure 8. Pins 0-5 are used to store data, while the pin 6 is the identifier from the transmitter that the data is ready to be read and pin 7 is the identifier from the receiver that the data has been read.

A. Algorithm of 32-bit Data Transmission

Data transmission flowchart among microcontrollers for unsigned long int, long int, and float types can be seen in Figure 9, Figure 10, and Figure 11.

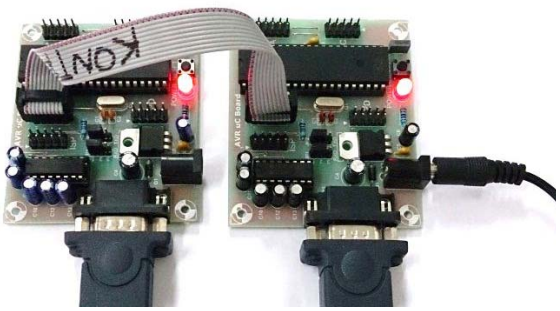

Figure 1. Connection between two microcontrollers

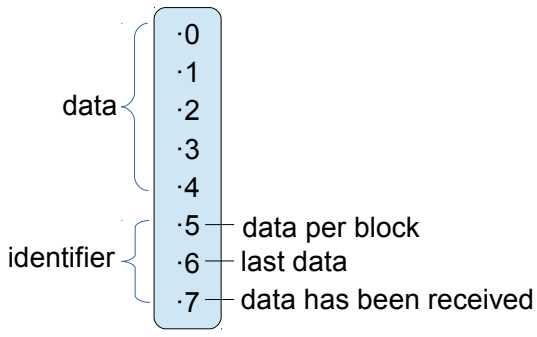

Figure 2. Pins asignment for unsigned long integer type

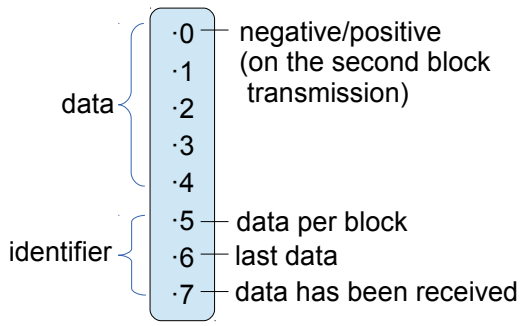

Figure 3. Pins assignment for long integer type

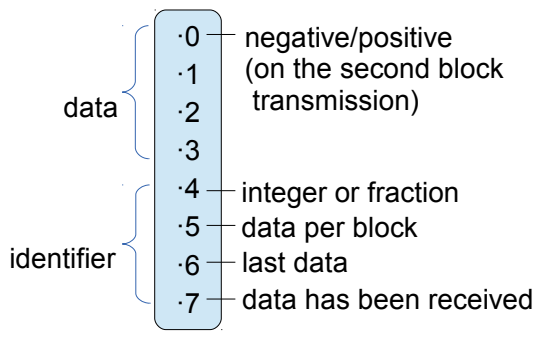

Figure 4. Pins assignment for float type

Figure 5. Series connection

Figure 6. Parallel connection

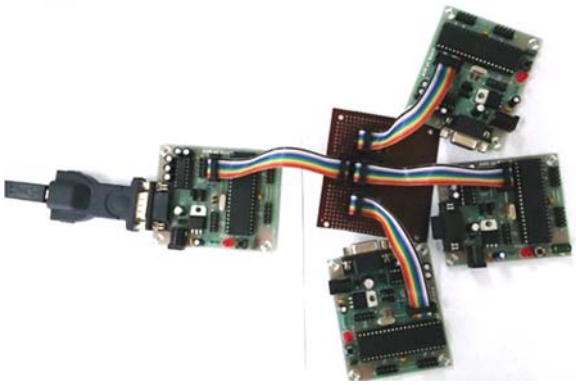

Figure 7. Tree connection

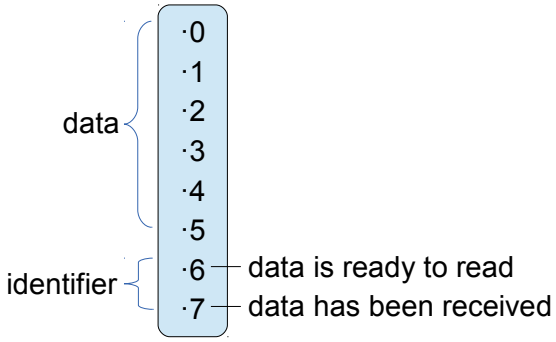

Figure 8. Confirmation step for sending data to a specific microcontroller in tree connection type

At the transmitter microcontroller, the 32-bit data is extracted into binary number using equation 1 and equation 2, then it is sent through a series of 8 bits.

$$binCounter = input\_value \,\%\, 2 \quad (1)$$

$$input\_value = input\_value/2 \quad (2)$$

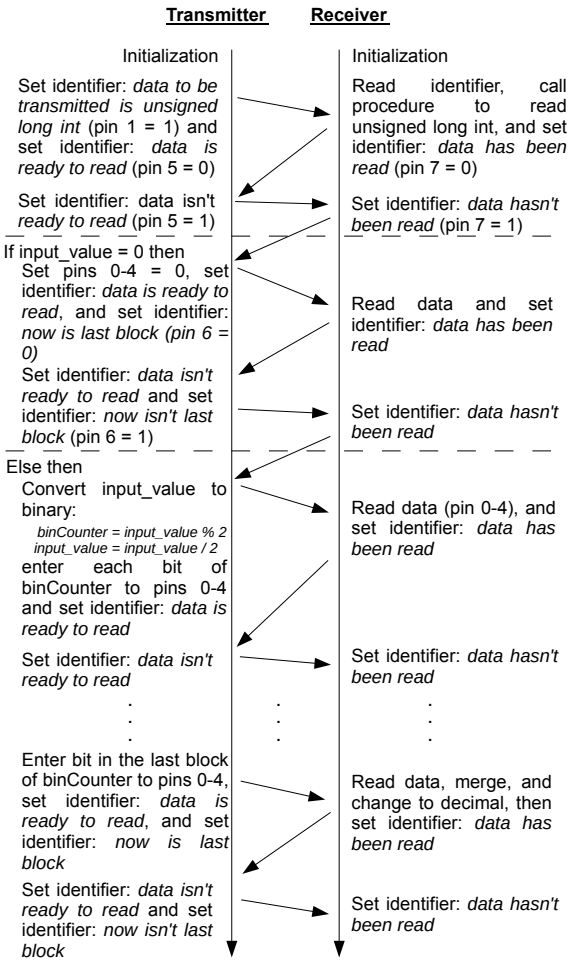

Figure 9. Flowchart of unsigned long int data transmission

where $binCounter$ is a binary value that will be stored on data pins. The equations is repeated until $input\_value = 0$.

At the receiver microcontroller, the binary data received through 8-bit port is reconverted into 32 bits with the same data type. This algorithm has an acknowledgment or an identifier that ensures the data has been received so errors can be avoided. The waiting process for this acknowledgement is 100 ms, if no acknowledgement arrived, then process will be stop and system will return to 0 value (that means an error/mistake occured).

There are differences in the phases of data transfer in each type as shown in Figure 9, Figure 10 and Figure 11. Unsigned long int consists of two phases: transmission of the identifier of data type and data/value. Long int consists of three phases: transmission of the identifier of data type, identifier of positive or negative sign, and data/value. Whereas float consists of five phases: transmission of the identifier of data type, identifier of positive or negative sign, decimal value, number of zero value behind the comma, and fractional value.

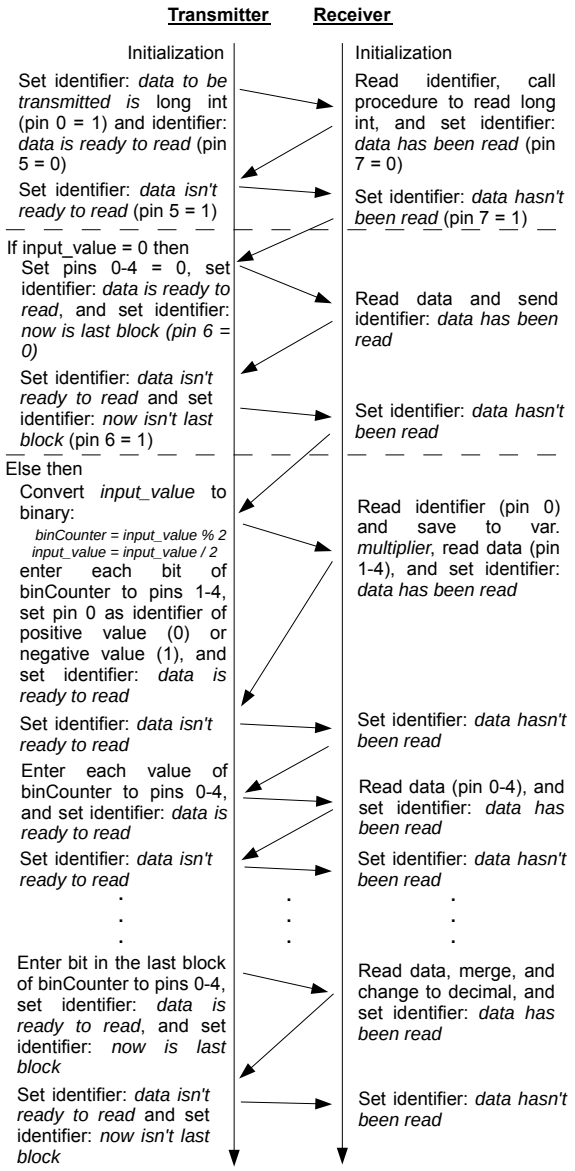

Figure 10. Flowchart of long int data transmission

There are an additional steps in tree connection before sending data, i.e. unsigned long integer, long integer, or float types, that can be seen in Figure 12. *Input_value* mentioned in Figure 12 is a number of specific microcontroller (0-63). Afterwards, the next process is the same as in Figure 9, Figure 10, or Figure 11.

B. Data Transmitter/Receiver Procedures

In order to implement the algorithm, a program has been created using C language with CodeVision Advanced AVR v3.10 IDE. Four functions have been created for data transmission, those are

```
int dataSend_unsigned_long_int(unsigned
    long int input_value)
int dataSend_long_int(long int
    input_value)
int dataSend_float(float input_value)
int dataSend_MikroKe(int micro)
```

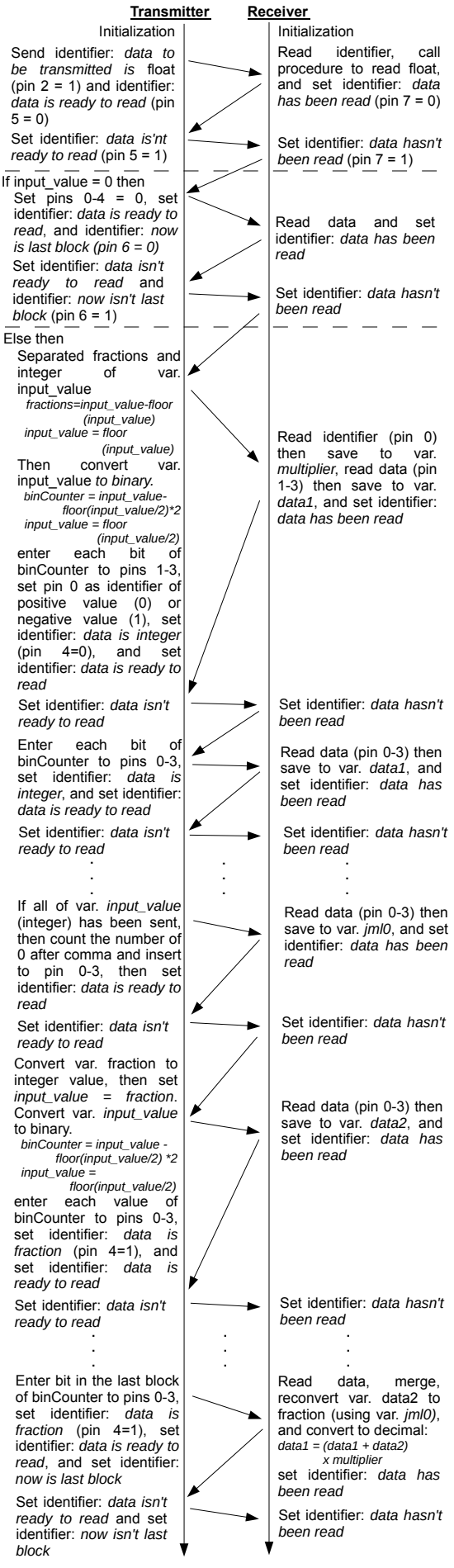

Figure 11. Flowchart of float type data transmission

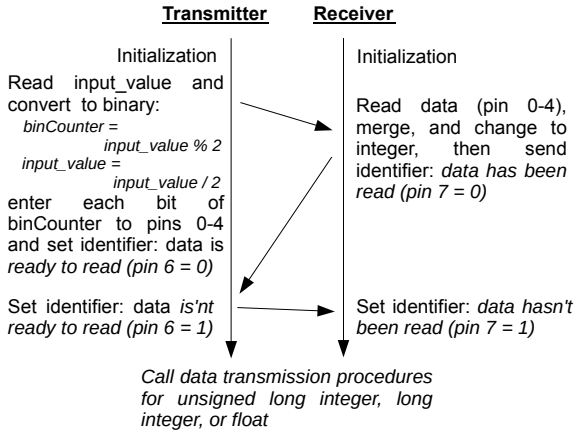

Figure 12. Flowchart of confirmation step for sending data to a specific microcontroller in tree connection

where `input_value` filled with values that will be sent. `micro` filled with numbers of specific microcontroller that will receive the data. All these functions will return to 1 if it is successful or otherwise will return to 0.

The receiver microcontroller simply call one of the following two procedures.

```
int dataReceive()
int specificDataReceive (int micro)
```

If the connection type used is tree type, then call the procedure `int specificDataReceive (int micro)` on the receiver microcontroller. Parameter `micro` is filled with the microcontroller number (0-63). If the connection used is not tree type, then call the procedure `int dataReceive ()` on the receiver microcontroller. This function will automatically call other functions according to the data type that the transmitter used, and return 1 if successful and otherwise is 0. Functions that are called are as follows.

```
long int dataReceive_long_int()
unsigned long int dataReceive_unsigned_
                  long_int()
float dataReceive_float()
```

The function returns a value that is received whose type is unsigned long int, long int, or float.

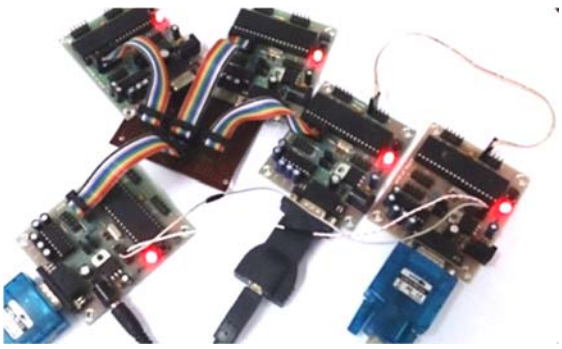

Figure 13. Connection for calculation of transmission time

III. Result and Discussion

Experiments have been conducted to test the accuracy and speed of data transfer between two microcontrollers (Figure 1) as well as using four microcontrollers in series (Figure 5), parallel (Figure 6), and tree (Figure 7). The data transmitted and received is displayed on a PC terminal via com 3 and com 5 to view the results.

Port B is used for experiments between two microcontrollers. For experiments using 4 microcontrollers ports are assigned as follows. In series connection, the transmitter uses port A while the receiver uses port B. In parallel connection, the transmitter uses ports A, B, and C, while the receiver uses port B. In tree connection, both the transmitter and receiver use port B.

Measurement of data transmission time is performed using the timer microcontroller [11] as shown in Figure 13. The time calculation algorithm can be seen in Figure 14.

At the start of the data transmission process, the transmitter set port D.6 = 0, then timer will start calculate using 16-bit timer, after the data transmission process is completed then set the port D.7 = 0, finally data transmission time can be calculated from the time interval.

A. Data Accuracy Experiments

Several data from minimum to maximum for each data type are used in experiments. Figure 15 and Figure 16 are described example data which sent from the first microcontroller and data received by the second microcontroller.

It can be seen that the data transmitted from the first microcontroller can be received without any damage or data loss by the second microcontroller. The range of values that can be used for each data type are listed in Table 1.

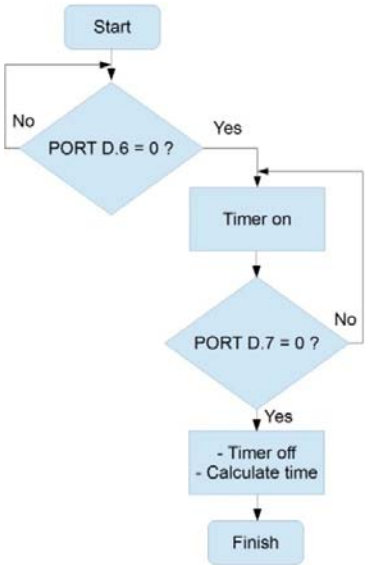

Figure 14. Algorithm of data transmission time calculation

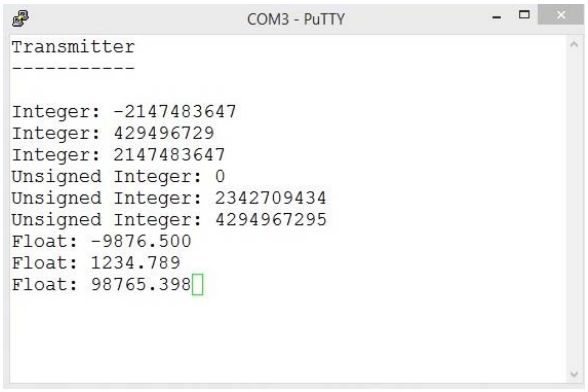

Figure 15. Data transmitted from the first microcontroller

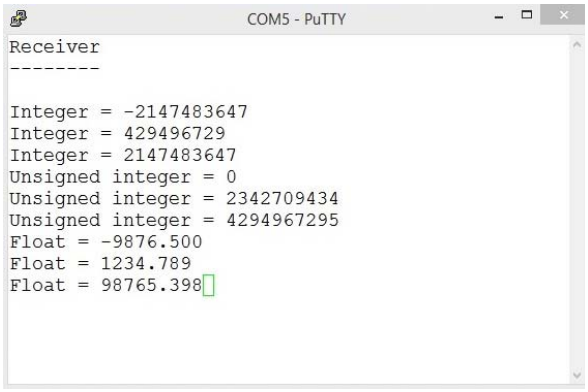

Figure 16. Data received by the second microcontroller

B. Data Transmission Time Experiments

Experiment of transmission times is done by sending sampling value from minimum to maximum. The number of data used in experiment is 11 for each type. In the float, the maximum number of digit used is seven digit [12, 13].

The transmission time between two microcontrollers for unsigned long int, long int, and float can be seen in Tables 2, 3, and 4. Based on Table 2, Table 3, and Table 4 it can be seen that maximum data transmission times for 32 bits between two microcontrollers for unsigned long integer, long integer, and float are 630 µs, 1,880 µs, 7,830 µs. Unsigned long integer has the fastest time of data transmission than long integer and float. This is related to the number of processes/steps used in float is more than long integer, and number of processes used in long integer is more than unsigned long integer.

The experiments results of data transmission time using four microcontrollers in series, parallel, and tree connections using unsigned long int, long int, and float can be seen in Figures 17, 18, and 19.

X-axis in Figure 19 is float value which is represented as data sequence as can be seen in Table 4. Based on Figures 17, 18, and 19, it can be seen that data transmission time of unsigned long integer, long integer, and float are almost

Table 1.
Data types with the range of values

Data types	Value
Unsigned Long Int	0 to 4,294,967,295
Long Int	-2,147,483,647to2,147,483,647
Float	±1.175e-38 to ±3.402e38 (seven digit precision) [12,13]

Table 2.
Data transmission time of unsigned long int

No	Unsigned Long Int	Time (µs)
1	0	50
2	390,451,572	550
3	780,903,145	570
4	1,171,354,717	620
5	1,561,806,289	620
6	1,952,257,861	610
7	2,342,709,434	620
8	2,733,161,006	630
9	3,123,612,578	630
10	3,514,064,150	630
11	4,294,967,295	630

Table 3.
Data transmission time of long int

No	Long Int	Time (µs)
1	-2,147,483,647	1820
2	-1,717,986,919	1880
3	-1,288,490,189	1880
4	-858,993,460	1820
5	-429,496,730	1740
6	0	50
7	429,496,729	1730
8	858,993,459	1820
9	1,288,490,188	1870
10	1,717,986,918	1860
11	2,147,483,647	1810

Table 4.
Data transmission time of float

No	Float	Time (µs)
1	-999,999.9	7290
2	-12,345.67	7830
3	-123.456	6220
4	-12.3	2920
5	-0.1	1750
6	0	460
7	0.1	1720
8	12.3	2890
9	123.456	6180
10	12,345.67	7810
11	999,999.9	7260

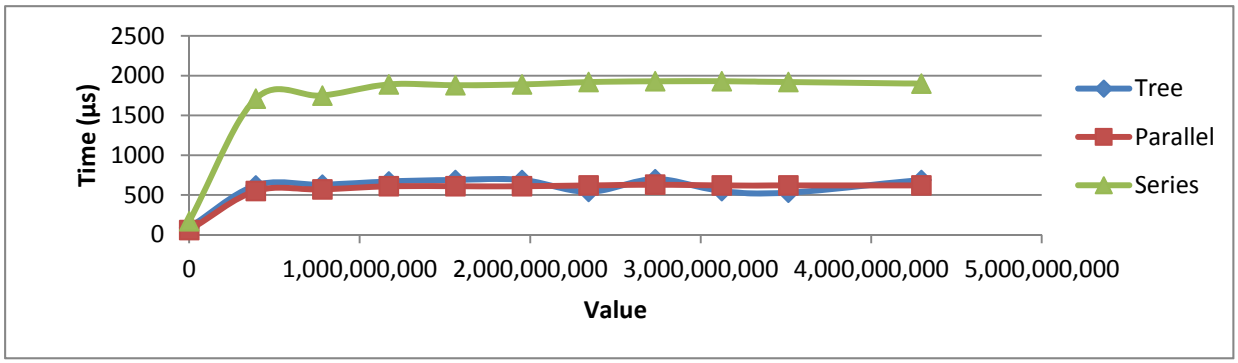

Figure 17. Unsigned long int type data transmission time

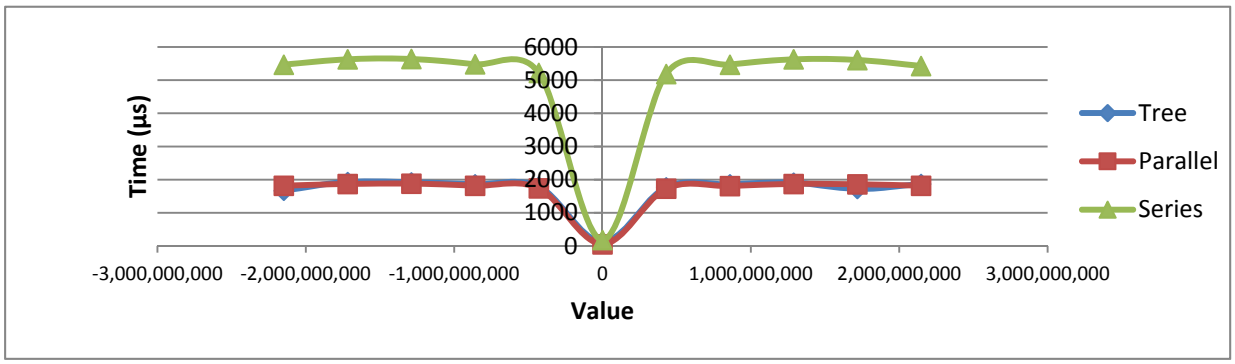

Figure 18. Long int type data transmission time

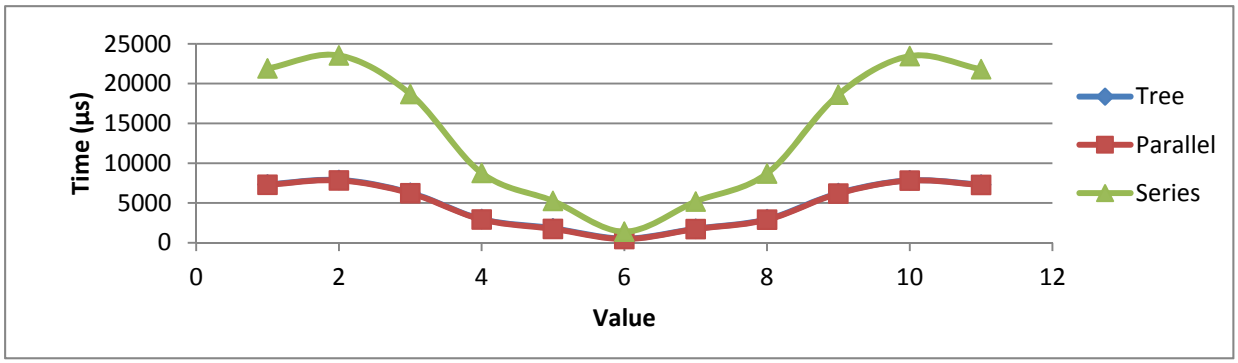

Figure 19. Float type data transmission time

equal between tree and parallel connections with the average absolute difference on unsigned long integer is 69 µs, long integer is 66 µs, and float is 54 µs, while series connection is slower with an average difference on unsigned long integer is 1,160 µs to parallel, long integer is 3,334 µs to parallel, and float is 9,543 µs to parallel. It is due to the data transmission performed gradually over three microcontrollers. The maximum transmission time on (a) unsigned long integer using parallel connection is 630 µs, tree is 700 µs, and series is 1,930 µs, (b) long integer using parallel connection is 1,880 µs, tree is 1,930 µs, and series is 5,640 µs, (c) float using parallel connection is 7,830 µs, tree is 7,890 µs, and series is 23,540 µs.

Based on the above experiments results, in order to maximize the range of available data and to minimize the transmission time, the following data type selection is recommended: In the case of transferred data is positive integers, then use unsigned long integer type. If transfered data is negative and positive integers, long integer type is preferable to be used. Finally, the float type should be used when the transferred data has fractions.

IV. Conclusion

A 32-bit data transmission among microcontrollers using an 8-bit port can be realized by using the algorithm described in this paper. The data types used are long integer,

unsigned long integer, or float types. The algorithm has been successfully implemented using the C language with CodeVision AVR v3.10 Advanced IDE. It has been successfully tested for the communication among ATMega32A microcontrollers. Based on the experiment results, it is known that the data transmitted using 32-bit long integer, unsigned long integer, or float can be accurately received without errors or data loss. Maximum transmission times between two microcontrollers for unsigned long integer, long integer, and float are 630 µs, 1,880 µs, and 7,830 µs. Unsigned long integer has the fastest time of data transmission than long integer and float. Maximum transmission times using four microcontrollers in parallel connection for unsigned long integer is 630 µs, long integer is 1,880 µs, and float is 7,830 µs. Maximum transmission times in tree connection for unsigned long integer is 700 µs, long integer is 1,930 µs, and float is 7,890 µs. Maximum transmission times in series connection for unsigned long integer is 1,930 µs, long integer is 5,640 µs, and float is 23,540 µs.

ACKNOWLEDGEMENT

Authors would like to thank to Rifa Rahmayanti and the Research Centre for Electrical Power and Mechatronics - Indonesian Institute of Sciences (LIPI) that has supported this research and all those who have helped conducting this research.

REFERENCES

[1] R. P. Saputra *et al*., "DC brushless motor control design and preliminary testing for independent 4-wheel drive REV-11 robotic platform," *Journal of Mechatronics, Electrical Power, and Vehicular Technology*, vol. 2, no. 2, Dec 2011, pp. 85-94.

[2] M. Mirdanies and R. P. Saputra, "Control system of solar tracking mechanism using combination of astronomy algorithm and light sensor," in *Seminar Nasional Rekayasa Energi, Mekatronik, dan Teknologi Kendaraan (RIMTEK 2013)*, Bandung, 2013, pp. 213-222.

[3] M. R. Reddy *et al*., "Touch screen and Zigbee based wireless communication assistant," *International Journal of Combined Research & Development (IJCRD)*, vol. 1, no. 4, Aug 2013, pp. 6-10.

[4] D. S. Thakur and A. Sharma, "Voice recognition wireless home automation system based on Zigbee," *IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)*, vol. 6, no. 1, June 2013, pp. 65-75.

[5] M. Leeman *et al.*, "Bridging the educational gap in embedded systems curricula: Developing an e-commerce audio streaming system," in *Ninth Annual IEEE International Conference and Workshop on the Engineering of Computer-Based Systems*, Lund, 2002, pp. 211-220.

[6] M. Solanke *et al*., "Automatic override of speed and brake control and ABS system," *International Journal of Thesis Projects and Dissertations (IJTPD)*, vol. 2, no. 2, June 2014, pp. 04-08.

[7] P. W. Prickett *et al.*, "A microcontroller-based end milling cutter monitoring and management system," *The International Journal of Advanced Manufacturing Technology*, vol. 55, no. 9, Aug, 2011, pp. 855–867.

[8] A. Kutlu, "MicroLab: a web-based multi-user remote microcontroller laboratory for engineering education*," *International Journal of Engineering Education*, 2004, pp. 879-885.

[9] Mikro Elektronika, "PORT Expander Manual," Zemun, Manual Book 2014.

[10] C. Pozrikidis, *Introduction to C++ programming and graphics*, 1st ed. University of California, San Diego: Springer Science+Business Media, LLC, 2007.

[11] Atmel. (2015, Oct.) ATmega32A 8-bit AVR microcontroller Datasheet Complete. [Online]. http://www.atmel.com/images/atmel-8155-8-bit-microcontroller-avr-atmega32a_datasheet.pdf

[12] IEEE, "IEEE standard Floating-Point Arithmetic," *IEEE Std 754-2008*, Aug 2008, pp. 1-58.

[13] Microsoft. (2014) Floating point types. [Online]. https://msdn.microsoft.com/en-us/library/aa691146%28v=vs.71%29.aspx