Development of a Cost-effective Data Acquisition System using an Open-source Hardware and Matlab/Simulink

Sugkil Seo, Yeong Sang Park and Young Sam Lee
Department of Electrical Engineering, Inha University, Incheon, Republic of Korea

Keywords: USB Communication, Matlab/Simulink, Data Acquisition, Open-source Hardware, S-function.

Abstract: This paper proposes a new cost-effective data acquisition system using open-source hardware and Matlab/Simulink. The proposed data acquisition (DAQ) system has features that it uses the framed data protocol based on hex encoding, it can acquire multiple data which are not of the same type at different sample rates, and the system receives data through USB communication or serial communication. The software of the proposed system consists of the firmware of a microcontroller and user-defined Simulink function block. The firmware of a microcontroller is in the form of a header file, and the data acquisition can be easily achieved by calling a few functions defined in the header file. The developed user-defined Simulink block can get multiple data at different sample rates by configuring the GUI parameters appropriately. For implementation of the system, we use the Arch Max, which is the open-source hardware with an ARM Cortex-M4 core, and also use a user-defined c-code S-function of Matlab/Simulink. For the demonstration of the superiority of the implemented system, we compare the proposed system's performance with that of the data acquisition system provided in Matlab/Simulink Instrument Control Toolbox. Finally we illustrate how presented system can be actually used by applying the proposed system to DC motor control.

1 INTRODUCTION

In the control system, it is very important to acquire data via device for designing system, maintenance and performance verification, etc. The device which acquires the data is called the data acquisition(DAQ) system.

The DAQ device consists of a hardware and a software. The hardware is connected to the target through sensors, at the same time, connected to a PC for the communication. One of the method to communicate with a PC was proposed through the special equipment for the high frequency sampling(Pereira et al., 2008)(Khan et al., 2011). We need the dedicated software for the DAQ device (Mandal et al., 2015). In addition, serial communication was also used for the data transmission. But in this case, the high sampling rate is limited because of the bandwidth of serial communication. So the USB DAQ board was proposed to upgrade the data transmission speed(ABU HASAN, 2012)(Proffitt et al., 2006)(Stanković et al., 2012).

A DAQ software transmits the data from the DAQ device to the PC in form of the protocol. The PC software is designed for the dedicated DAQ device, or use the existing software(Salami et al., 2011)(Lee et al., 2012). One of the existing software, Matlab/Simulink provides Instrument Control Toolbox or Data Acquisition Toolbox which help to communicate with the external device through the TCP/IP or serial communication(MathWorks, 2009)(Storey, 2002).

The monitoring system that uses the function of the DAQ device displays the acquired data through the plotting function of a software in real time. Users can observe the acquired data graphically through the monitoring system. So the monitoring system is helpful to the user, because it gives the insight about the system.

The open-source hardware is that design resources of the product to make the same function and form are freely provided by copyright holder. Even somebody make the copy product for a commercial use, the copyright holder doesn't get the royalty. If people who make the copy product express the copyright, they can freely modify and redistribute the design resources. In this reason, the open-source hardware is already used in commercial purposes, the education and research area(Pearce et al., 2012)(Pearce, 2015)(Claros-Marfil et al., 2016).

Some open-source hardware microcontroller board have built in USB. If the open-source hardware

Seo, S., Park, Y. and Lee, Y.
Development of a Cost-effective Data Acquisition System using an Open-source Hardware and Matlab/Simulink.
DOI: 10.5220/0005978004840491
In *Proceedings of the 13th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2016) - Volume 1*, pages 484-491
ISBN: 978-989-758-198-4

microcontroller board can communicate data with Matlab/Simulink, the cost-effective DAQ system can be developed which uses USB communication. Also, the development processes will be very easy because of open-source hardware's characteristics. If we configure the DAQ system which uses the open-source hardware microcontroller board, then it can acquire the data with a valid sampling rate in the simple theoretical research verification and the engineering education. Aspect of convenience, it has a portability and can be easy installed because of USB's plug and play characteristic. If the firmware of device has a high portability, this DAQ system can be programmed with a controller. So, the user can acquire the state values of the controlled system.

This paper proposes a new cost-effective data acquisition system using open-source hardware and Matlab/Simulink. We propose the algorithm including the protocol, and implement the DAQ device firmware and user-defined Simulink block adapting the proposed algorithm. We compare the proposed system's performance with that of the data acquisition system provided in Matlab/Simulink Instrument Control Toolbox. In addition, we experiment the proposed system by applying it to DC motor control.

2 STRUCTURE OF SYSTEM

The proposed DAQ system uses the open-source hardware microcontroller board that can use serial communication or USB communication, and communicates the acquired data with user-defined Simulink block. The concept of system is Figure 1. In the DAQ system, the microcontroller board sends the acquired data to PC through unidirectional communication. An ideal DAQ system should have no data missing, can acquire multiple data which are not of the same type at different sample rates. The proposed system combines the acquired data and the next data with the information of the data type and the channel number by using separator, and sends it to PC. Through this process, the distinction of the series data become clear, access time of the DAQ device to a PC decreases, and the DAQ system can transmit the acquired data without missing. Therefore, the DAQ system should satisfy the following conditions.

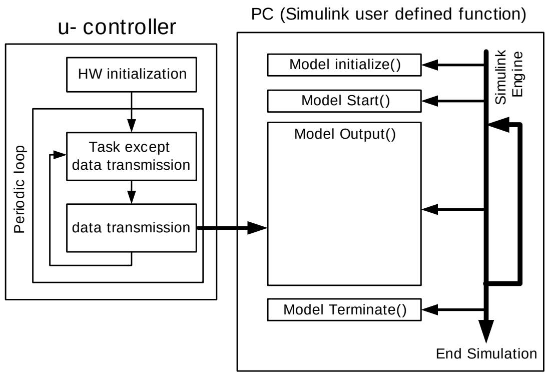


Figure 1: The conceptual diagram of the proposed DAQ system.

1. The DAQ system can acquire multiple data at different sample rates.
2. All the channels should be able to have another type of data.
3. There is no need to match the sampling time and the data transmission interval.
4. The block returns the data corresponding to sampling each a period of sampling.
5. Receiving and converting data is executed at least in Simulink.

For this conditions, we propose following protocol and the method that generates a data packet and the algorithm for receiving data, and implement the DAQ block using user-defined Simulink block.

2.1 Protocol

In protocol configuration, the protocol including encoding changes the raw data and lengthens the total length of data, and increases the data transmission load. But, we can use some values that are not appeared after encoding as a control command, and the control commands make clear the data transmission. The proposed protocol is based on hex encoding, and is represented in Figure 2(a). The sensed data are changed through hex encoding and reconstructed by adding the information of the data type and the channel number. In hex encoding, the encoded data are represented only as a hexadecimal number, other numbers can be used to command. We use the 'T', 'Q' as the control command by inserting the 'T', 'Q in the front and rear of the encoded data. The 'T','Q' which are used as a separator separate the data that are sampled at different time like Figure 2(b). We add a 'Z' at the end of the combined the data as a terminator, and use it as a data packet. If the data packet is transmitted, as shown in Figure 3, each data can be distinguished by the acquired sampling time using the 'T' and 'Q'. The data between 'T' and 'Q' have the information about type and channel, even some channel data are not sampled at specific sampling time, the receiving part know which data are transmitted and are not transmitted. If these functions can be implemented as user-defined Simulink block, system doesn't have problem from different sampling time.

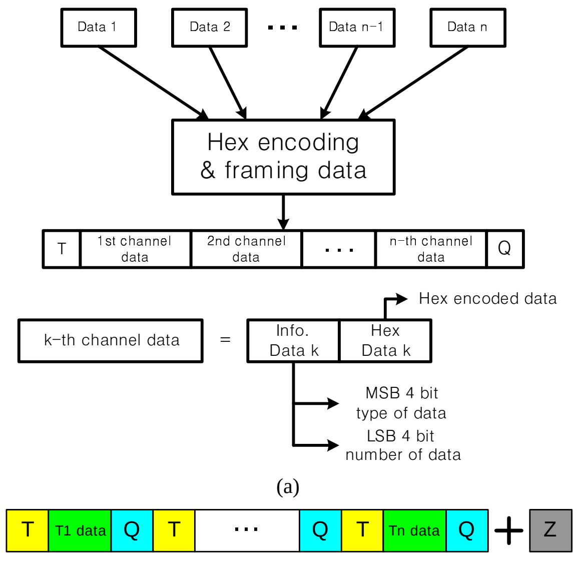


Figure 2: (a) : Hex encoding and data framing, (b) : Data packet.

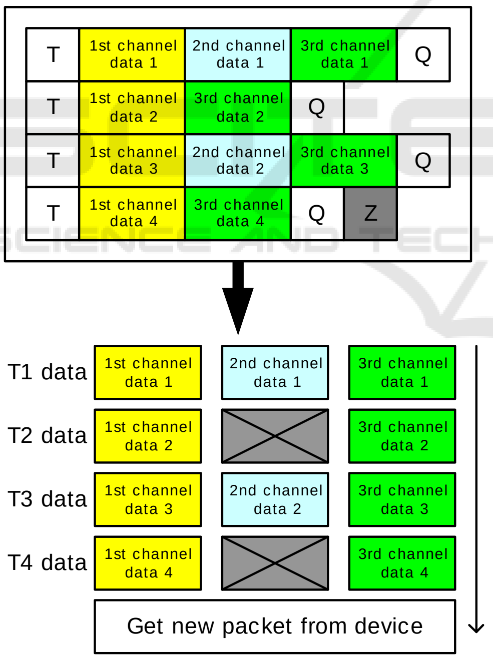
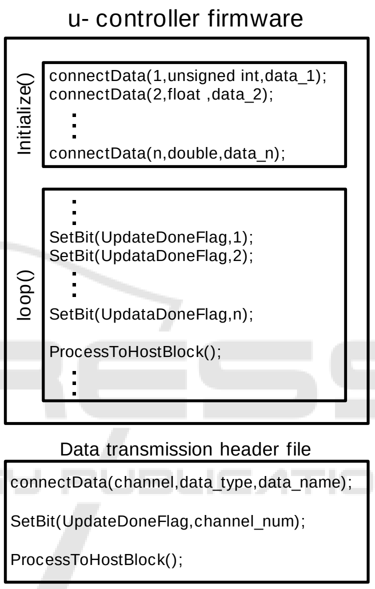


Figure 3: Decoding of a data packet.

2.2 DAQ Device Firmware

Generally, a DAQ system can be used through the simple GUI format. However, the proposed system dose not target a specific microcontroller, it is hard to develop the system that only uses GUI format. So, for simplifying implementation of the system, we configure the DAQ device firmware as a header file format. Figure 4 represents overview of the firmware. At start, the DAQ device connects acquired data variables and the information of the channel number and the data type using the defined function in header file. After the loop beginning, the DAQ device calls the function for updating flags that represent new data are sensed. When every data are updated, the device calls another function for making the data packet and sending packet to PC. As with the previous process, if firmware is configured in the header file format for the function declaration and the data transmission, we can implement the DAQ system without the consideration of hardware and protocol in the stage of use. The user can use the system by programming like Figure 5.



```
connectData(1,unsigned int,data_1);
connectData(2,float ,data_2);
   ⋮
connectData(n,double,data_n);
```

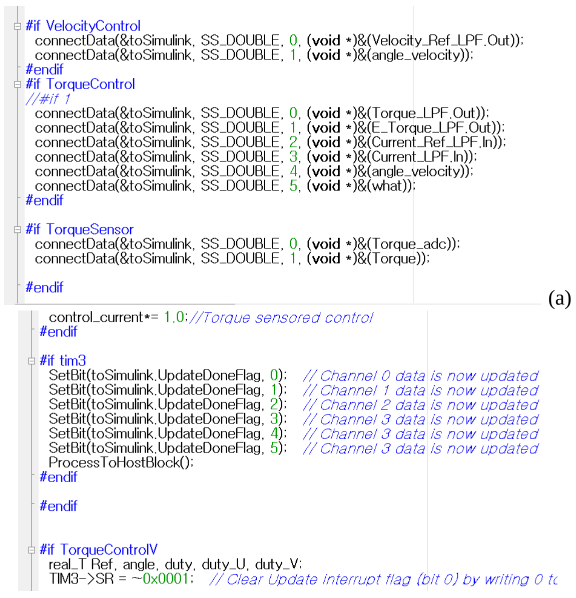
```
   ⋮
SetBit(UpdateDoneFlag,1);
SetBit(UpdataDoneFlag,2);
   ⋮
SetBit(UpdataDoneFlag,n);

ProcessToHostBlock();
   ⋮
```

```
connectData(channel,data_type,data_name);

SetBit(UpdateDoneFlag,channel_num);

ProcessToHostBlock();
```

Figure 4: The overview of firmware.

2.3 Algorithm for Block Implementation

Figure 6 represents the flowchart of each user-defined Simulink block. The algorithm of the block is satisfying previous conditions. First branch uses the StartFlag that is used for checking the first loop of system as a condition of branching, this case is executed only once for the data acquiring process. If the StartFlag is true, communication initializing function is run for receiving the data from the device and the StartFlag becomes false. Second branch is used to check block's process operating number and the remaining data to decoding through the OrderIndex and the NoDataFlag. The OrderIndex is execution number of block in Simulink, and The NoDataFlag is true when non-decoded data packet is not exist. When the NoDataFlag is true and OrderIndex is zero, second



(a)

```
#if VelocityControl
  connectData(&toSimulink, SS_DOUBLE, 0, (void *)&(Velocity_Ref_LPF.Out));
  connectData(&toSimulink, SS_DOUBLE, 1, (void *)&(angle_velocity));
#endif
#if TorqueControl
//#if 1
  connectData(&toSimulink, SS_DOUBLE, 0, (void *)&(Torque_LPF.Out));
  connectData(&toSimulink, SS_DOUBLE, 1, (void *)&(E_Torque_LPF.Out));
  connectData(&toSimulink, SS_DOUBLE, 2, (void *)&(Current_Ref_LPF.In));
  connectData(&toSimulink, SS_DOUBLE, 3, (void *)&(Current_LPF.In));
  connectData(&toSimulink, SS_DOUBLE, 4, (void *)&(angle_velocity));
  connectData(&toSimulink, SS_DOUBLE, 5, (void *)&(what));
#endif

#if TorqueSensor
  connectData(&toSimulink, SS_DOUBLE, 0, (void *)&(Torque_adc));
  connectData(&toSimulink, SS_DOUBLE, 1, (void *)&(Torque));

#endif
```

(b)

```
  control_current*= 1.0;//Torque sensored control
#endif

#if tim3
  SetBit(toSimulink.UpdateDoneFlag, 0);   // Channel 0 data is now updated
  SetBit(toSimulink.UpdateDoneFlag, 1);   // Channel 1 data is now updated
  SetBit(toSimulink.UpdateDoneFlag, 2);   // Channel 2 data is now updated
  SetBit(toSimulink.UpdateDoneFlag, 3);   // Channel 3 data is now updated
  SetBit(toSimulink.UpdateDoneFlag, 4);   // Channel 3 data is now updated
  SetBit(toSimulink.UpdateDoneFlag, 5);   // Channel 3 data is now updated
  ProcessToHostBlock();
#endif

#endif


#if TorqueControlV
  real_T Ref, angle, duty, duty_U, duty_V;
  TIM3->SR = ~0x0001;   // Clear Update interrupt flag (bit 0) by writing 0 tc
```

Figure 5: Examples of actual use((a) : initialize, (b) : loop).

branch goes to yes, and block receive the packet like Figure 2(b) from buffer. Before the received packet are entirely decoded, the NoDataFlag is false every loop, so second branch goes to no. Third branch goes to yes when the block's OrderIndex is zero. If third branch goes to yes, the part of the data packet is decoded. At the end of the flowchart, the decoded data are came out from the block refer to the channel number step by step. By the previous process, the system can minimize the number of data receiving and decoding. Because the data packet can be separated using the 'T' and 'Q' like Figure 3, the algorithm can be implement corresponding to the previous conditions. Generating and transmitting data packet make delay between the target system and the acquired data, but reduce the load from device access.

2.4 Implementation of Communication Block

We configure the communication block using the proposed algorithm, communication protocol and Matlab/Simulink c-code S-function. Simulink determines priority number of every block that consist of model according to defined rule, and the individual block's initialization and output calculation is executed in accordance with the priority. In consideration of the characteristics of Simulink, the method considering the priority should be included. For making user-defined Simulink block, we use provided the format about input-output size, sampling time etc, and implement block including the method that can receive channel number and sampling time. We configure the block that can be connected with the USB device driver through the function that OS provides. Also, like Figure 7, we make the function parameter that can be easily changed using GUI.

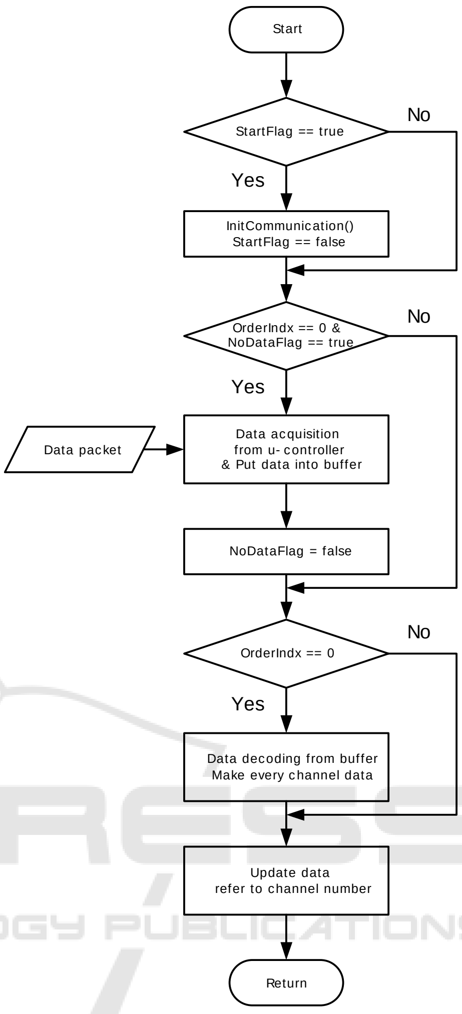
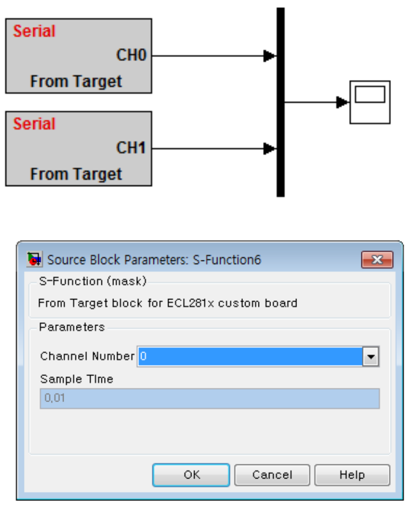


Figure 6: The flowchart of the user defined block.

3 PERFORMANCE TEST

We compare the performance of proposed system with that of DAQ system provided in Matlab/Simulink Instrument Control Toolbox. We used an open-source hardware Arch Max as a DAQ unit for the test. The Arch Max has the features that it is supported by mbed project based on ARM and open-source project, the user of the Arch Max can use free development tool via the internet, and it support drag and drop programming. The Arch Max use the mi-



Source Block Parameters: S-Function6

S-Function (mask)

From Target block for ECL281x custom board

Parameters

Channel Number: 0

Sample Time: 0.01

Figure 7: The implemented block and block's GUI.

crocontroller STM32F407 that is ARM Cortex-M4, 32bit,and 168MHz maximum clock speed and support USB full speed and USB High speed. In this test, we set clock speed 168MHz and use USB full speed using CDC class.

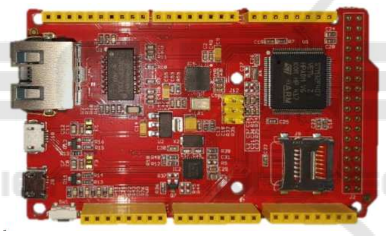


Figure 8: The open-source hardware Arch Max(ARM STM32F407).

3.1 Compare Data Packet Generation

As shown in Figure 9, the DAQ system consists of serial communication block in the Matlab/Simulink Instrument Control Toolbox. This system uses the binary data transmission protocol adding terminator, and some settings can be changed via the GUI. And

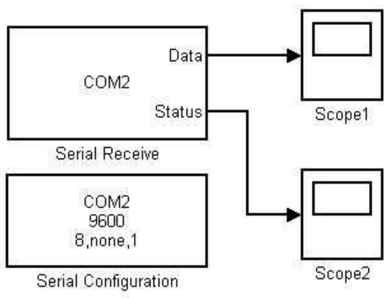


Figure 9: The DAQ system using the Matlab/Simulink Instrument Control Toolbox.

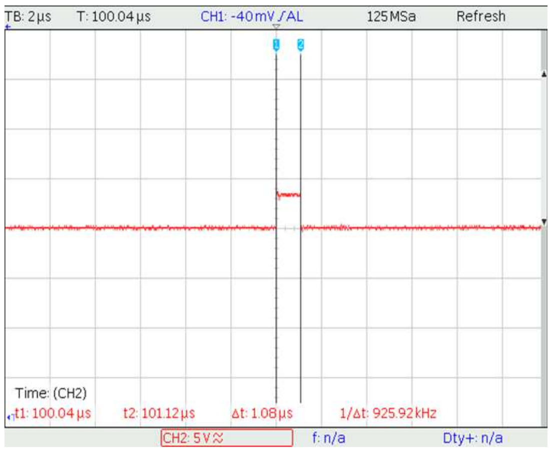
the other system that we proposed uses the protocol as shown in Figure 3.

We check the load of the packet generation and transmission through measuring the time taken. If the time taken of the packet generation gets longer, it will have implications on the performance of the DAQ system. Figure 10 shows the time taken that changes from 5 floating point data to the data packet and transmits. If we ignore the GPIO toggling time, we can measure the time taken using the GPIO toggle back and forth of the packet generation and transmission. In the binary data transmission case, it takes about 1 micro-second. In the presented protocol case, periodically, the time taken appears longer than others because the system collects several sampled data and periodically changes from the acquired data to the packet. So, we measure the longest time and that is about 8 micro-second. If we use a 1kHz sampling rate, the time taken of both cases are not over 1% of the available time. So the packet generation problems are hardly happen.

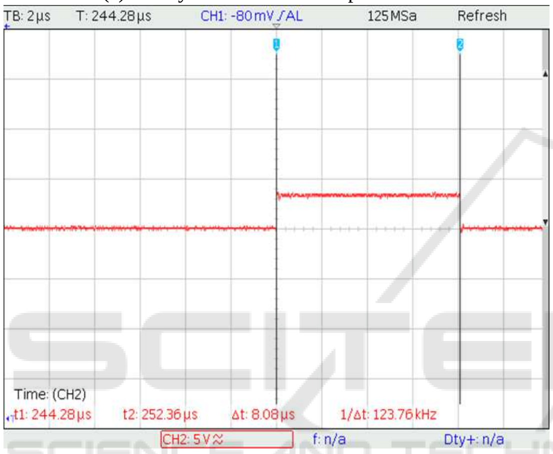
3.2 DAQ Performance Test

Figure 11 represents the result of receiving 5 floating point data about 100Hz sampling and 1000Hz sampling through Matlab/Simulink Instrument Control Toolbox. The results have data missing, and result of 100Hz sampling frequency shows indentation because of low sampling frequency. The result of 1000Hz sampling frequency has better smoothness than previous case, but it's data missing is longer than the previous. Figure 12 is the results of receiving 5 floating point data using proposed system about 1000Hz sampling and 5000Hz sampling. The experiment results doesn't have the data missing both 1000Hz sampling and 5000Hz sampling. Depending on the experiment results, the proposed system has more better data acquisition ability than that of Matlab/Simulink Instrument Control Toolbox.

Figure 13 is the visualized data drop. The method which measures the data drop is sending the integer data that is increased 1 per sampling with other data and checks the difference of adjacent two numbers. If the difference is n, then $n-1$ sampled data are dropped. In Figure 13, the case that uses Matlab/Simulink library shows 242 data drop when block receive data 1kHz. This means 242 data drop per 1.242 second, so the data drop rate is about 19.5%. In contrast, the result of the proposed system shows no data drop even sampling frequency is 10 times faster. In repeated experiments, about 20% data drop happen when Matlab/Simulink library is used, and data drop doesn't happen when the presented system is used.



(a) Binary data transmission packet case.



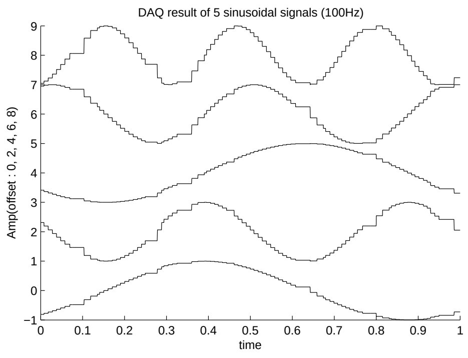
(b) User-defined data transmission packet case.

Figure 10: Time taken to generate a packet of 5 floating point data.

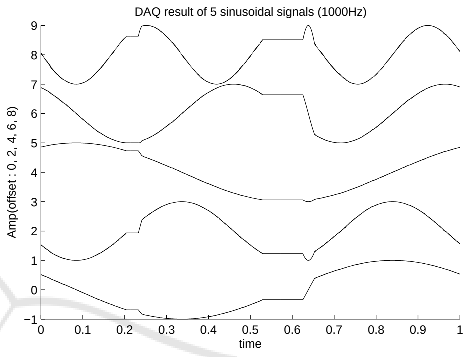
3.3 Actual DAQ System Test with DC Motor Velocity Control

Figure 14 represents the configuration of DC motor experimental system for the practical application of the proposed system. The experimental system consists of the DC motor, the encoder board, the motor driver board and the Arch Max with the Arch Max adaptation board.

The experiment is to control the angular velocity of DC motor in no-load condition. We configure the PI controller. In the experiment, we give the sinusoidal reference to system, sample angular velocity 1000Hz sampling frequency and configure the feedback loop. At the same time, we configure the monitoring system using Matlab/Simulink scope block. With reference to the plot in Figure 15, the user can easily design the PI controller by changing the coefficients.



(a)

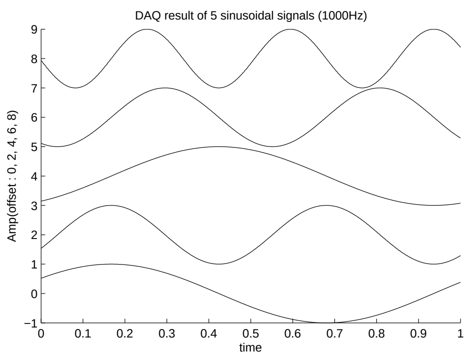


(b)

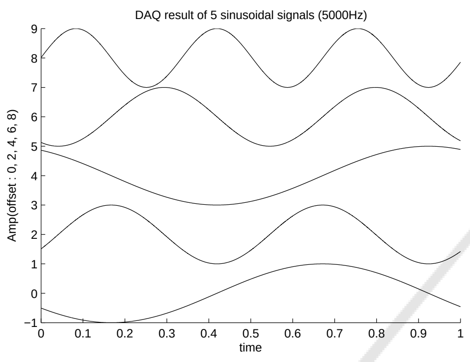
Figure 11: The result of the Simulink library block((a) : 100Hz sampling, (b) : 1000Hz sampling).

4 CONCLUSIONS

In this paper, we proposed a new cost-effective data acquisition system using open-source hardware and Matlab/Simulink. The proposed DAQ system has the following features : it uses an open-source microcontroller board that supports USB communication or serial communication, it is implemented for use with Matlab/Simulink so that one can utilize all the convenient functions provided by Matlab/Simulink. To implement the proposed system, we used an open-source hardware Arch Max as a DAQ unit, developed the device firmware and a DAQ block using user-defined c-code S-function. The firmware is in the form of a header file, which includes several handy functions. One can program the data acquisition by calling the provided functions in a well defined sequence. One can configure the developed Simulink block so that he can obtain multiple data at different rates by changing GUI parameters appropriately. Through the experiment, we showed that the proposed DAQ system has better performance than the system provided by Matlab/Simulink. In addition to the acquisition of sensor data, the proposed DAQ system



(a)



(b)

Figure 12: The result of the presented user defined function block((a) : 1kHz sampling, (b) : 5kHz sampling).

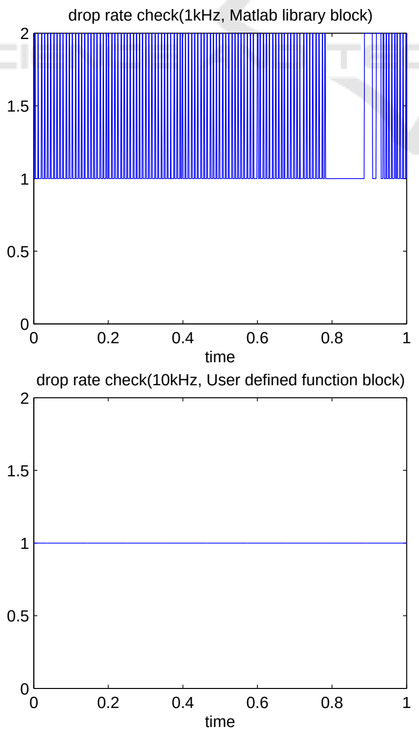


Figure 13: Visualized data drop.

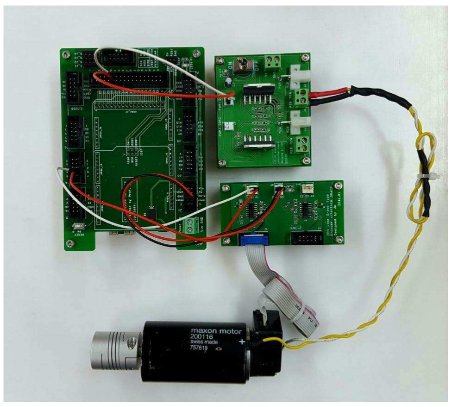


Figure 14: The experiment setup of the DAQ system for DC motor control.

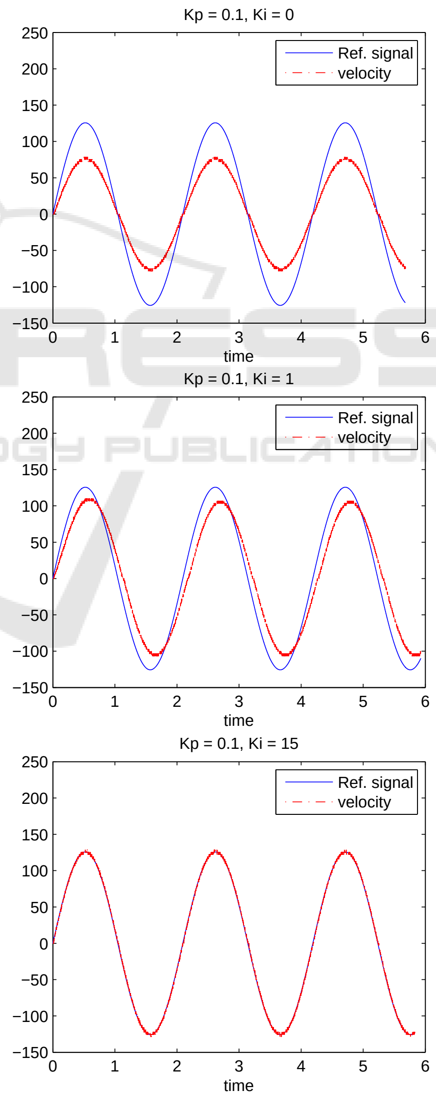


Figure 15: Experiment results of DC motor PI velocity control(Top : Kp = 0.1 , Ki = 0 Middle : Kp = 0.1, Ki = 1 Bottom : Kp = 0.1, Ki = 15).

can be used to monitor all the variables in the control algorithm. This allows the user to have better insight in designing control system by providing plentiful information. This aspect was clearly illustrated by the controller design example. Since the proposed system is based on open-source hardware, it is more cost-effective than existing DAQ systems using PCIe or expensive DAQ boards. The proposed system can be successfully used for teaching purpose in undergraduate or graduate courses.

ACKNOWLEDGEMENTS

This research was supported by the MSIP(Ministry of Science, ICT and Future Planning), Korea, under the ITRC(Information Technology Research Center) support program (IITP-2016-H8601-16-1003) supervised by the IITP(Institute for Information & communications Technology Promotion)

REFERENCES

ABU HASAN, R. (2012). *Development of USB Biosignal DAQ System with Matlab Interface*. PhD thesis, Universiti Teknologi Malaysia.

Claros-Marfil, L. J., Padial, J. F., and Lauret, B. (2016). A new and inexpensive open source data acquisition and controller for solar research: Application to a waterflow glazing. *Renewable Energy*, 92:450–461.

Khan, F. A., Hafeez, Z., Mirza, A., and Ain, Q.-u. (2011). Design of fpga based daq card using pci express protocol. In *Multitopic Conference (INMIC), 2011 IEEE 14th International*, pages 211–216. IEEE.

Lee, Y.-S., Yang, J.-H., Kim, S.-Y., Kim, W.-S., and Kwon, O.-K. (2012). Development of a rapid control prototyping system based on matlab and usb daq boards. *Journal of Institute of Control, Robotics and Systems*, 18(10):912–920.

Mandal, S., Sau, S., Chakrabarti, A., Saini, J., Pal, S. K., and Chattopadhyay, S. (2015). Fpga based novel high speed daq system design with error correction. In *VLSI (ISVLSI), 2015 IEEE Computer Society Annual Symposium on*, pages 80–85. IEEE.

MathWorks, I. (2009). Matlab & simulink instrument control toolbox. *Neural Network Toolbox. The MathWorks Inc*.

Pearce, J. M. (2015). Commentary: Open-source hardware for research and education.

Pearce, J. M. et al. (2012). Building research equipment with free, open-source hardware. *Science*, 337(6100):1303–1304.

Pereira, R., Sousa, J., Fernandes, A., Patrício, F., Carvalho, B., Neto, A., Varandas, C., Gorini, G., Tardocchi, M., Gin, D., et al. (2008). Atca data acquisition system for gamma-ray spectrometry. *Fusion engineering and design*, 83(2):341–345.

Proffitt, J., Hammond, W., Majewski, S., Popov, V., Raylman, R., and Weisenberger, A. G. (2006). Implementation of a high-rate usb data acquisition system for pet and spect imaging. In *Nuclear Science Symposium Conference Record, 2006. IEEE*, volume 5, pages 3063–3067. IEEE.

Salami, M.-J. E., Tijani, I., and Jibia, A. U. (2011). Development of real-time software interface for multicomponent transient signal analysis using labview and matlab. In *Mechatronics (ICOM), 2011 4th International Conference On*, pages 1–5. IEEE.

Stanković, M., Manojlović, S., and Jovanović, Z. (2012). Acquisition system for analysis and design of electrical servo system based on usb daq card dt9812. *FACTA UNIVERSITATIS, Series: Automatic Control and Robotics*, 11:69–79.

Storey, B. D. (2002). Using the matlab data acquisition toolbox. *URL: http://faculty.olin.edu/bstorey/Notes/Card. pdf (accessed January 2008)*.