MINAR International Journal of Applied Sciences and Technology

Article type : **Research Article**

Date Received : **11/07/2021**
Date Accepted : **08/08/2021**
Date published : **01/09/2021**
: **www.minarjournal.com**
http://dx.doi.org/10.47832/2717-8234.3-3.8

MATLAB BASED HIL FRAMEWORK: A GUIDE TO BUILD A HARDWARE IN THE LOOP DAQ PERIPHERAL

Hassan M. BAYRAM[1] & Bilal A. MUBDIR[2]

Abstract

Testing and validating modern hardware such as some subsystems in modern vehicles is a little challenging especially before assembling them into the final product. To achieve a valid real-time test, the tested hardware or unit must be placed into its real-time environment which is not possible in some cases. Recently, and with the presence of advanced simulation software applications, the hardware environment could be simulated easily to fulfill the real-time test properly. Simulating an environment in one loop with real physical hardware knowing as Hardware-in-the-loop is used nowadays in various development fields, medical, industrial, research, and education. Amongst the aforementioned, HIL is widely used in control systems applications. In the paper, building a framework to enable hardware in the loop (HIL) simulation with the aid of MATLAB/Simulink is discussed. Over serial communication, and inexpensive data acquisition (DAQ) peripheral has been developed using a microcontroller unit. the development of the framework is discussed to be used as a guide for building it by using any microcontroller. The resultant performance appeal to excellent real-time response with quite a small delay of about 70ms in the worst case.

Keywords: DAQ, HIL, Software Applications.

[1] Koya University, Iraq
[2] Sulaimani Polytechnic University, Iraq, bilal.mubdir@spu.edu.iq, https://orcid.org/0000-0001-8096-8027

Introduction

Hardware-in-the-loop (HIL) testing is the technology where real-time testing is applied to a physical signal of a real target device by applying a test scenario with unlimited iterations until reaching the optimum design or the final objective [1].

HIL test has become the dominant test in embedded software in the industrial and automotive fields by connecting the real hardware I/O to the simulation software for its flexibility and cost-effectiveness. Where, many companies now offer HIL commercial platforms, such as NI, Typhoon, dSPICE, and many other companies.

Not limited to industrial or automotive fields, medical devices are now tested in the model-design stage using HIL mythology. Before releasing the final product and due to the critical application of the medical device, many medical manufacturers are going along with testing their embedded software with the products earlier during the design stages effectively to revise the bugs and validate the software and the hardware [2].

In the past few years, HIL becomes essential in developing the Embedded systems, reducing the time consumed in development and even marketing. As the complexity of the products increases, large systems are divided into multiple subsystems, where all are tested simultaneously. HIL appeared to the scene to avoid fully physical tests due to their expensive cost, and fully simulation tests for their inaccurate results.

Referring to figure (1), testing and validating a design during model-based design (MBD) consist of different stages, software, processor, and hardware in the loop until reaching the validation [3]. Furthermore, a detailed guide for MDB can be found in [4].

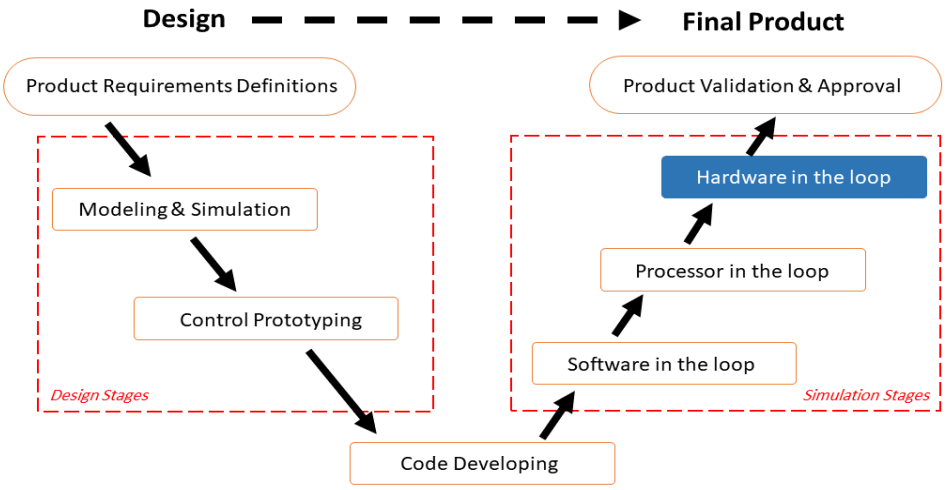

Figure 1: Model-based (MDB) design process

HIL is all about the integration of the real physical world to the software simulation as shown in figure (2) for testing the actual device and validate it [5]. Connecting the real target tested the device and the simulation model together to form the HIL concept finally result in a closed-loop system [6].

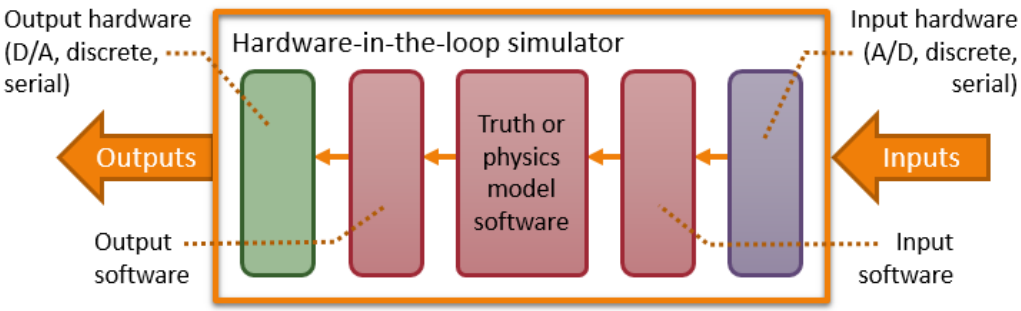

Figure 2: The outline of the Hardware in the loop.

Commercially available HIL platforms in the markets for industrial and even for research and educational purposes are expensive. This paper introduces a full guide for building an inexpensive HIL framework by using any microcontroller with MATLAB/Simulink for providing HIL's functionalities. Any programming language can be used to develop the framework depending on the microcontroller unit (MCU). In this paper, Atmega328P MCU from Atmel [7] has been used which is easy to program using different integrated development environments and very cost-effective.

Related works:

Various definitions for the HIL have been used in various articles, each from its perspective, and the trend of its applications are discussed in the literature. A great review of the available definitions has been reported by N. Brayanov and A. Stoynova in their article [8]. They highlighted the history of HIL in addition to state its requirements and challenges.

Mascio CD and his colleagues have studied the philosophy of the heart to develop mathematical models for the human heart based on HIL. The tested proposed peacemaker with the developed model of the heart where the peacemaker was designed by using Raspberry Pi computer. Their work proves that the peacemaker or any other medical device can be tested and validated without the involvement of real humans. flexible software panel was developed to control and monitor the heart model easily from the computer. The paper results show that the proposed system verify the requirements as defined, where HIL could be a good option for not including humans in real clinical trials [9]

In power system applications, J. Wu et al. have tested modeled active compensator (essential power system component) with relay using real controller hardware. They developed the model of the power active compensator under LabView software from NI and they evaluate it using the HIL method with the external controller. Modeling the compensator from scratch was a challenge for them but with the help of HIL tests, they achieved their final design. Their paper discussed the procedure of the modeling in addition to the test and how the validation was made [10]

In the field of automotive engineering is where the most efforts of HIL occurred to develop advanced Electronic Control Units (ECU) in electrical vehicles, H. Haupt et al. have used the HIL to test the battery management system (BMS) of the electrical vehicle. They utilized a commercial HIL simulator produced by dSPICE to emulate the batter cells by applying a simple batter model for the Lithium-ion type. They receive the voltage of each simulated battery and total current in addition to the temperature of the batteries and outer terminal voltage from the simulator. The parts of the ECU have been bypassed and every sensor and load were modeled to generate a complete simulate model for testing purposes. They claim that this method provides a flexible way to simulate different failure scenarios as is the case in real electronics vehicles [11].

Moreover, in the electric vehicle industry, developing a real-time HIL platform (RT-HIL) for emulate signal generation and measurement based on LabView software from NI is presented by J. Feng et al. in [12]. They used the RT-HIL to control EMC by forming controller area network (CAN) communication at 500kbps speed by using a USB-CAN adapter from a traditional personal computer. Their proposed platform applied to diesel engines and hybrid vehicle controllers supported by personal computers to display and analyze the simulation. They described every implementation detail for developing the HIL board, simulating the signals, and implementation of the analog signal generation and sensor measurement. Their paperwork concludes that their platform is efficient even if it was small scale, but it can test and validate ECUs algorithms.

On the other side, mechatronics is another field in which HIL tests are active and required due to the high cost of the hardware used in robotics systems. Ciprian R. et al. have suggested economic replacement for 2DOF robot by simulating its dynamic model in MATLAB/Simulink and connecting it to real servo motors over target DAQ board called FiO Std board to implement complete HIL simulation to analyze and control the actuation of the robot system. The HIL in their paper is the servo motors, which worked as actuators controlled using pulse width modulation (PWM) signal and they are modified to read their instantaneous position to be feedback to the robot model in MATLAB/Simulink [13].

Proposed HIL Framework

In this paper, Real-time MATLAB/Simulink has been used in the host computer to simulate any mathematical or dynamic model for any system, all output and input signals from/to the real hardware part are interfaced using the MCU as a target DAQ peripheral as shown in Figure (3).

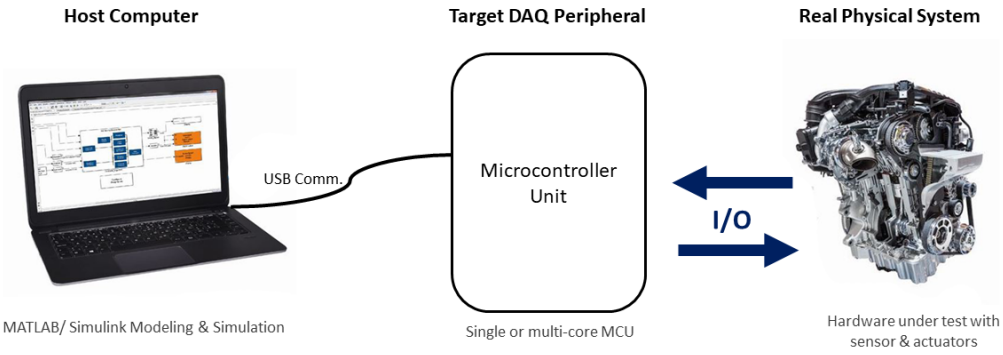

Figure (3): Typical real time Hardware in the loop testing system

In the experimental work of our paper, Atmega328P MCU has been chosen to be used as DAQ peripheral for its inexpensive cost, availability in the markets, and adequate I/O resources compared to its price. The Atmage328P has 14 digital I/O ports [7], where it is possible to connect to 14 digital actuators or sensors. Also, it has 6 analog input channels for analog sensors and 6 possible output analog channels for analog actuators. Reading sensors or controlling actuators is all about programming the MCU itself, for that reason we assumed that this part must be made by the developer according to the application considered in the HIL.

Simulink HIL Configuration

The main objective in this section is to prepare the simulated model in MATLAB/Simulink to be ready to communicate with the target DAQ peripheral and so the real hardware in real-time. An official support package developed by MathWorks itself for accessing the resources of Arduino board (which Atmega328P MCU-based development board) is available for free and easy to use [14]. Nevertheless, the package is very limited and provides access only for the ports without any ability to send/receive variables data. To overcome the limitation issues, this guide provides a raw method for accessing every port and variable in the MCU is presented.

Considering sending single “float” data from Simulink to the DAQ unit or vice versa over USB cable, the first stage is to convert the USB protocol to the default Serial protocol (UART). This could be done using a UART circuit, for instance; FT232RL or any other equivalent IC as shown in figure (4).

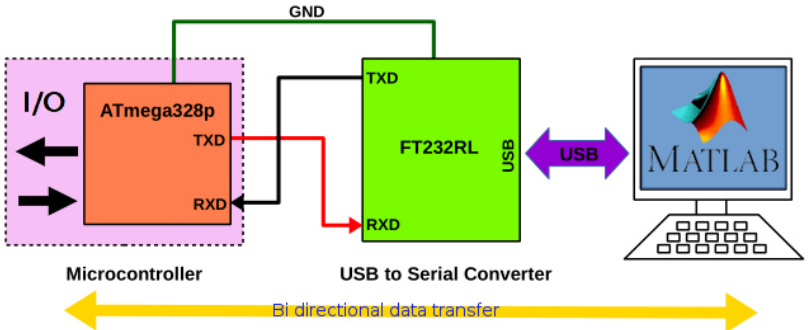

Figure (4): the communication between MCU and the host computer over USB

The host computer needs to be configured according to the MCU serial communication settings. To do so, the “Serial Configuration” block is used in Simulink to specify the baud rate of the serial

data communication, number of bits, synchronization bits, and other serial communication settings shown in figure (5).

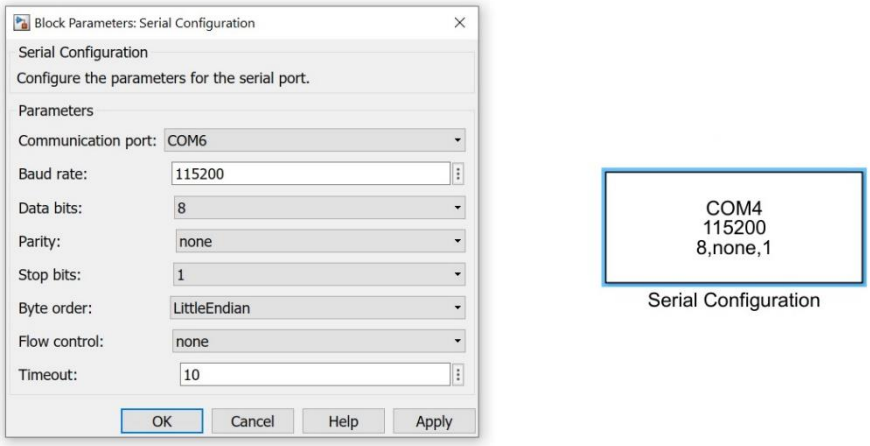

Figure (5): Serial configuration block and its settings

Now, to send data from the DAQ unit to the Simulink another block called “Serial Receive” is used. This block must be configured carefully with the DAQ unit to let the data deliver properly to the DAQ. Referring to figure (6), the Communication port must be selected according to the chosen option in the Serial Configuration block.

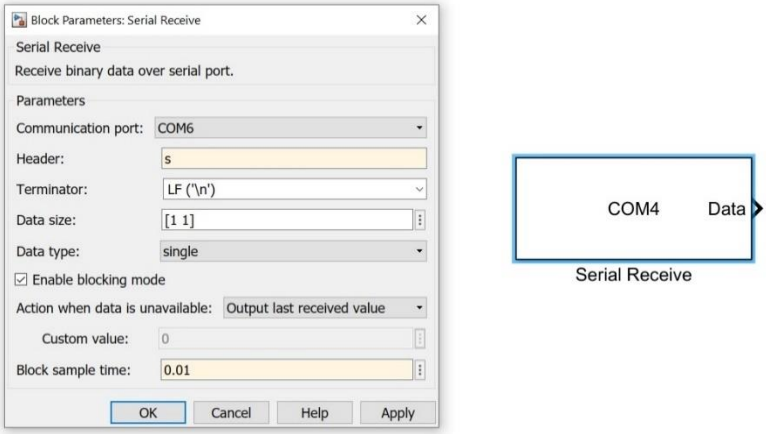

Figure (6): Serial Receive block and its settings

Other settings are illustrated in Table (1), and they are very important in setup the DAQ unit’s framework. Every single setting can make difference in data communication speed, so the simulation speed. Misconfigure the receive block or the DAQ unit could lead to wrong synchronization and fault in simulation.

Table (1): Setting parameters for the “Serial Receive” block in Simulink

Setting Parameter	Description
Header	This data (Character/String) could be appended to the payload package as a header. It used as an additional layer of synchronization with the target DAQ.
Terminator	This data also attached to the end of the payload message.
Data Size	The number of data package, if single data to be received it must be [1 1], if multiple packages of data are to receive then it could be [1 N], where N is the number of single data type to be received from DAQ.
Data Type	Here is the place where the data type must be defined, “float” data in MATLAB for example defined as “single”.
Enable blocking	This option blocks the simulation until receiving the full package of data

mode	
Block sample time	The duration on which the DAQ is sending data to Simulink

The “Serial Receive” block should be followed by “Data Type Conversion” block to convert the received data into “Double” type as shown in figure (7), to process the receive data and make it compatible with the data flow in Simulink for other simulation purposes, like applying mathematical operations, or display the received data on the scope.

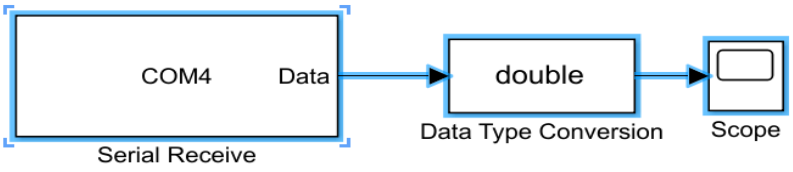

Figure (7): Final block diagram for receiving data from target DAQ unit.

On the other hand, to send data from the Simulink environment to the target DAQ unit, “Zero-Order Hold” block has been used as Analog to Digital converter (ADC) and sample the data at a specific rate that represents the duration for snap the sample of the data, which defined the rate of sending data. Moreover, the “Data Type Conversion” block is used after the “Zero-Order Hold” to convert the sampled data to be sent into “Single” type data which is equivalent to “float” type than before sending that data the data bytes must be packed into 4 bytes payload and that has been done by using “Byte Packing” block then “Serial Send” block has been used to send the data to the DAQ unit. The configuration for each block is revealed in figure (8) along with the final block diagram, where the important settings parameters are previously described in Table (1).

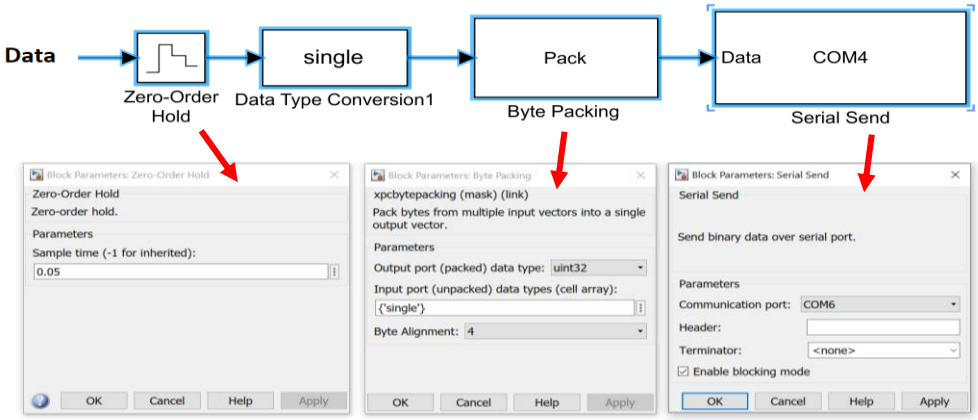

Figure (8): Final block diagram for sending data to target DAQ unit.

When combining both receiving and sending blocks together, the external real-time hardware could be connected in one loop with the Simulink model through the DAQ unit to form a complete HIL, figure (9) reveals the full diagram of the real-time HIL configuration for sending/receiving single float variable.

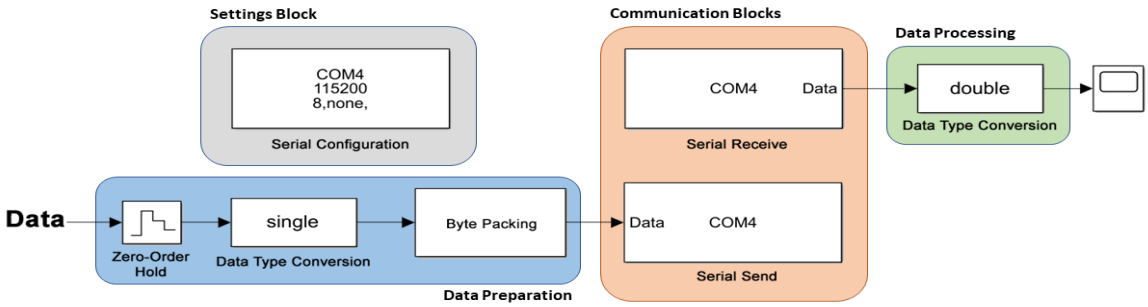

Figure (9): Complete Simulink real time HIL model for sending/receiving single float variable.

When the HIL system has multiple numbers of variables to be communicated between the real hardware and the simulation model, the model of the figure (9) is modified to accept multiple data variables to be sent or receive. For the receiving side, to receive multiple variables (N), the only setting parameter to be changed is the “Data Size” on the “Serial Receive” block and it became [1 N], where (N) is the total number of variables of float type. For example, to receive six float variables from the DAQ unit as shown in figure (10), the data size set would be [1 6] followed by a “Demux” block to decode the received variables into six separated variables.
However, the “Serial Send” block doesn’t need any change to send multiple variables. The change is on the way to preparing the data before the sending process. The data must be gathered in one vector using “Vector Concatenate” and the output feed to the “Zero-Order Hold” block, then data converted to “Single” type using “Data Type Conversion” block and finally before the “Serial Send”, the data are passed through “Byte Packing” block, all with the same previous settings. The final diagram of the real-time HIL configuration for sending/receiving multiple float variables is shown in figure (10).

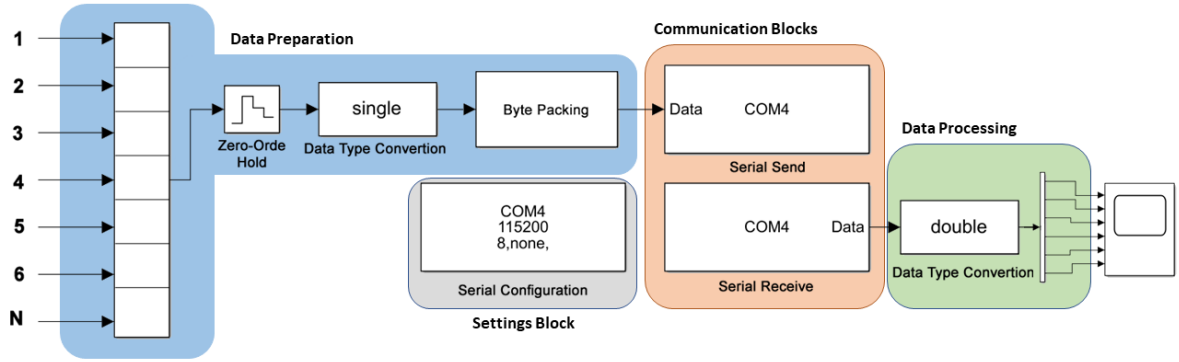

Figure (10): Complete Simulink real time HIL model for sending/receiving multiple float variables.

Target DAQ Programming

As mentioned before, the MCU is acting as a DAQ to enable communication between the host computer and the real hardware. The Atmega328 used in this research has been programmed using C language in different cases, sending single-variable data, receiving single-variable data, sending multiple variables data, and receiving multiple variables data.
When sending a single variable to the host computer, the data package must be led by header if it is considered in Simulink and followed by terminator if considered too. When a single float variable is intended to be sent, the variable bytes must be sent one by one in four parts then followed by terminator if considered. Noting that every sending procedure must be done at a rate equal to the sampling rate of the simulation, which can be achieved by delaying the process of sending data for the desired sampling time TSampling as shown in the figure (11).
On the flip side, when receiving the data from the host computer the same process will be followed in a reverse direction with some extra operations. In the reading receiving data process, the delay for sampling time TSampling is not essential, whenever the data arrived, it could be processed. When a single float variable is intended to be received, every four bytes received must be combined to get the full variable. Again, if header or terminator are considered, they must be excluded, and conditions used to extract the variable data as shown in figure (12).

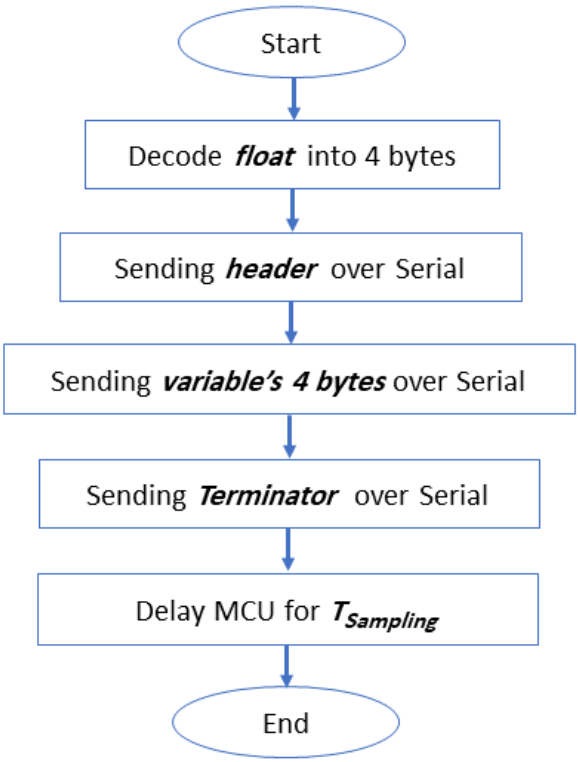

Figure (11): Flow chart of sending data process from DAQ

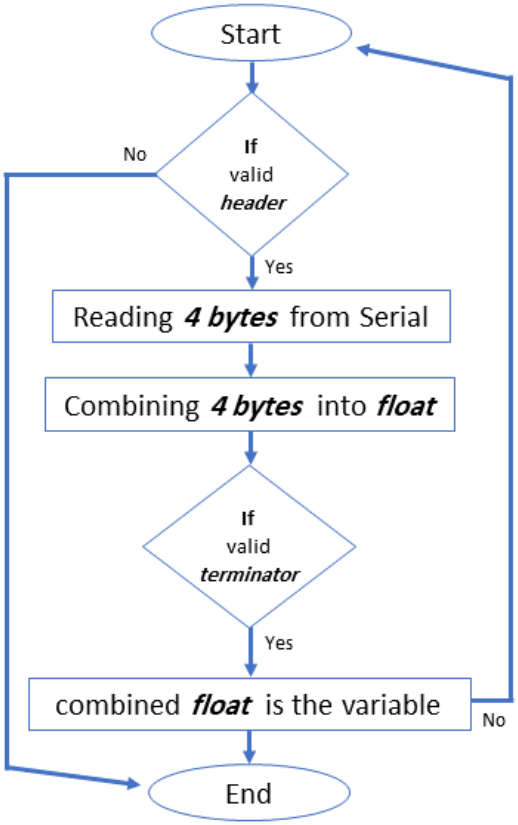

Figure (12): Flow chart of receding data process by DAQ

In the case of multiple variables, the process explained previously in figures (11) and (12) are repeated for N times equal to the number of variables but the repeated process for all variables must be performed within the one segment of sampling time TSampling. However, there is no need to repeat appending or checking header and terminator repeatedly, only one time with the whole package of the sent or received variables. Both single and multiple variables codes for the Atmega328P are available online [15].

Finally, the date to be sent to the host computer could be any data. Digital or analog ports that are connected to sensors or could be read and send periodically to the host commuter or according to specific criteria or variables resultant from processing algorithms could be sent also. The same case for the received data, it could be actuators signals or processing parameters all is dependent on the purpose of the application and the design of the developer.

Results & Discussion

Evaluating the operation of the proposed framework was done by using the single variable model in figure (f9) and the multiple variables model in figure (f10). The experiments were performed to send a sawtooth signal to the target DAQ for the single variable case and sending three different variables to the target DAQ unit (sinewave, sawtooth, and squire wave) for the multiple variables case. In both cases, the test was carried out to receive the echo of original signals from the target DAQ. The purpose of this procedure is to check the real-time response of the framework for a given sampling rate and baud rate. The sample time selected for the experiments was 50ms

(20Hz), and the baud rate for serial communication was 115200 bps. The resultant response of the echo signals for both cases is shown in figures (13) and (14).

Figur

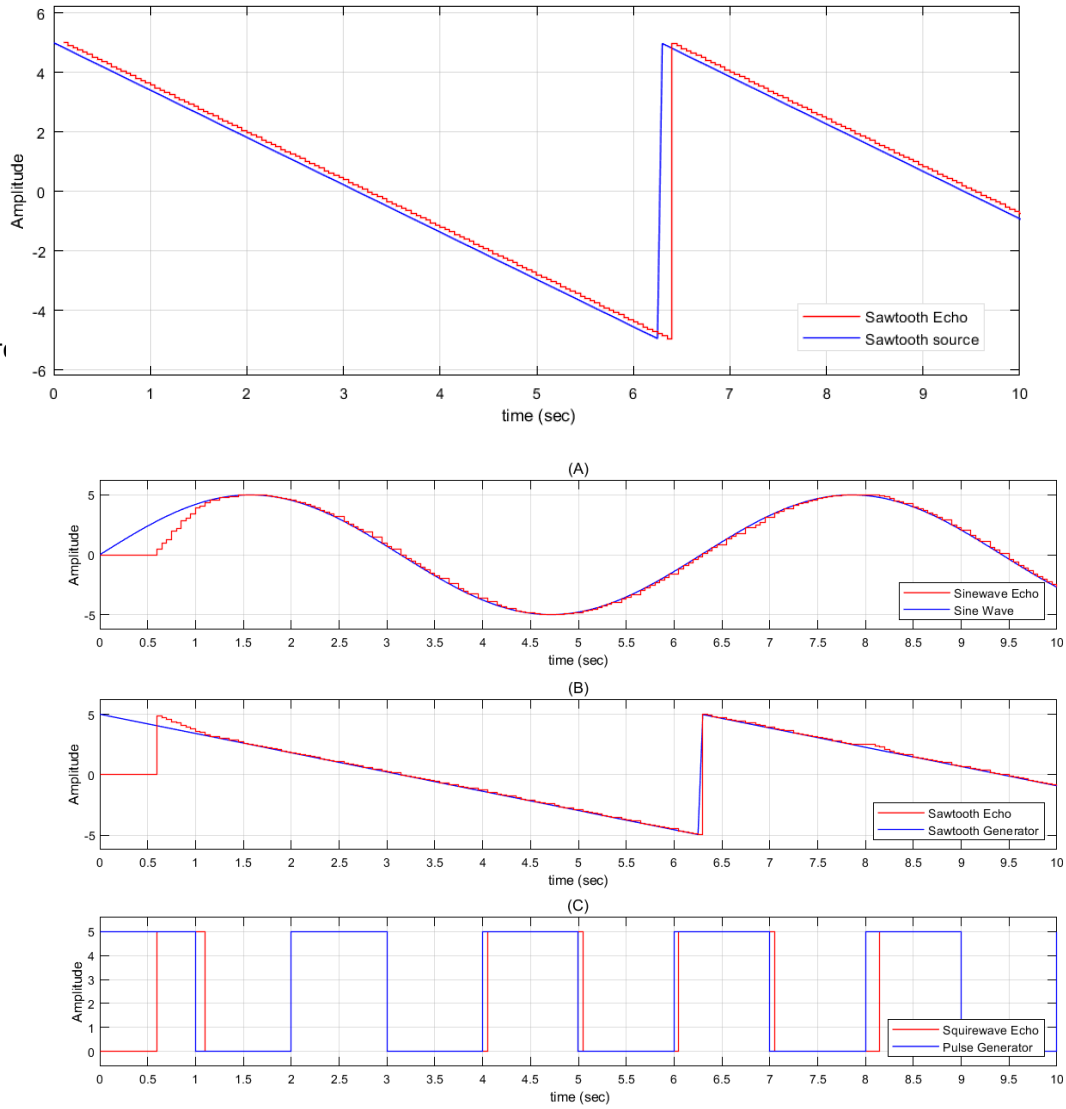

Figure (14): The echo responses of original multi signals from target DAQ

With a close look at the results of the figure (15), by making a precise comparison between source signals from Simulink with the echo returned signal from the target DAQ, it is seen that the delay is about 50ms between the two signals. Where the echo signal is lagging by 50ms, and this is the selected sampling time.

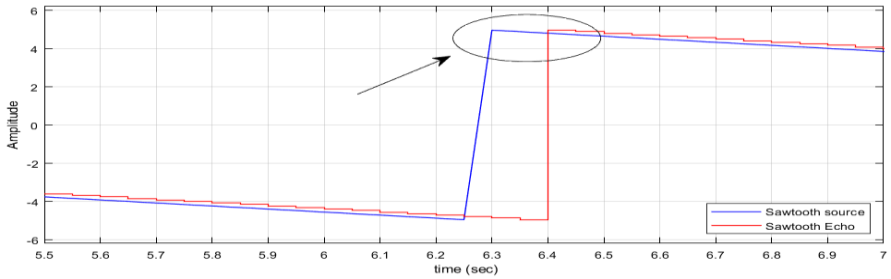

Figure (15): Close look at the echo response of original sawtooth signal from target DAQ

Although the 50ms appeared between the original and echoed signal, the real delay is half, 25ms. This is because the appeared delay on the graphs is for the two-way communication, so the actual response time is 25ms. If any data is to be sent or receive from one side to the other, it takes the half value of the sampling time.

Conclusion

HIL technique provides a rapid method for testing and verification of control systems in various application fields. The proposed framework provides an inexpensive and easy-to-build method for including the real hardware into MATLAB/Simulink for real-time simulation. The proposal consists of a microcontroller unit operated as a DAQ unit, interfaced with the real hardware, and communicated with a host computer that has MATLAB/Simulink model configured to exchange data. The number of channels or variables that could be exchanged between the real hardware and the MATLAB/Simulink model is flexible, can be reduced or increased according to the requirements very easily. The practical results have revealed a superior performance with a fast response of 25ms when sampling rate 20Hz.

Acknowledgment

All experimental tests and the hardware implementation were carried out at the Training and Development department, UrukTech Electronics Company, Iraq.

References

National Instruments Corp., “What is hardware-in-the-loop?” Www.ni.com, 28-Jun-2017. [Online]. Available: https://www.ni.com/en-lb/innovations/white-papers/17/what-is-hardware-in-the-loop-.html. [Accessed: 06-Aug-2021].

The Mathworks, Inc. (R2021a). “Basics of Hardware-In-The-Loop simulation - MATLAB & Simulink,” Mathworks.com. [Online]. Available:https://www.mathworks.com/help/physmod/simscape/ug/what-is-hardware-in-the-loop-simulation.html. [Accessed: 06-Aug-2021].

T. VanGilder, “Why HIL is becoming critical for medical device testing,” Genuen.com. [Online]. Available: http://www.genuen.com/blog/why-hil-is-becoming-critical-for-medical-device-testing. [Accessed: 07-Aug-2021].

The MathWorks, Inc., “Model-Based Design for Embedded Control Systems”, White paper, 2020, available: https://www.mathworks.com/campaigns/offers/model-based-design-embedded-control-systems.html

Jason Benfer, “What is Hardware-in-the-Loop (HIL) Testing?”, White paper, Genuen Company, 2021. Available: http://www.genuen.com/blog/what-is-hardware-in-the-loop-hil-testing

Peter Waeltermann, “Hardware-in-the-Loop: The Technology for Testing Electronic Controls in Vehicle Engineering”, dSPACE Inc., April 11, 2016.

Atmel Corporation, “8-bit AVR Microntroller with 4/8/16/32K Bytes In-System Programmable Flash,” ATmega 328P datasheet, 2019.

Nikolay Brayanov, and Anna Stoynova, "Review of hardware-in-the-loop - a hundred years progress in the pseudo-real testing," Journal Electrotechnica& Electronica, vol. 54, No. 3-4, pp. 70-84, Bulgaria, 2019.

C. D. Mascio and G. Gruosso, “Hardware in the loop implementation of the oscillator-based heart model: A framework for testing medical devices,” Electronics (Basel), vol. 9, no. 4, p. 571, 2020.

J. Wu, Y. Cheng, A. K. Srivastava, N. N. Schulz and H. L. Ginn, “Hardware in the Loop Test for Power System Modeling and Simulation,” 2006 IEEE PES Power Systems Conference and Exposition, 2006, pp. 1892-1897, doi:10.1109/PSCE.2006.296201.

Hagen Haupt, Markus Plöger, JörgBracker,” Hardware-in-the-Loop Test of Battery Management Systems”,IFAC Proceedings Volumes,Volume 46, Issue 21,2013,Pages 658-664,ISSN 1474-6670, ISBN 9783902823489.

Jing Feng et al., "Principles and application of the real-time hardware-in-the-loop simulation platform based on multi-thread and CAN," 2008 IEEE International Symposium on Industrial Electronics, 2008, pp. 2225-2230, doi: 10.1109/ISIE.2008.4677041.

Rad, C., Maties, V., Hancu, O., Lapusan, C., 2012. Hardware-In-The-Loop (HIL) Simulation Used for Testing Actuation System of a 2-DOF Parallel Robot. AMM 162, 334–343.

The MathWorks Inc., “Simulink Support Package for ArduinoHardware”, Reference Documentation, Version 21.1.0 (R2021a), March 2021.
Mubdir, Bilal (2021) HIL-Atmega328 (Version 1.0) [Source Code]. Available on: https://github.com/bilalmubdir/MATLAB-Based-HIL/releases/tag/1.0