



DOI: 10.22144/ctu.jen.2016.041

A FRAMEWORK FOR TRANSFERRING ALGORITHMS DESIGNED ON MATLAB/SIMULINK TO ARM MICROCONTROLLER EMBEDDED SYSTEMS

Nguyen Van Khanh

College of Engineering Technology, Can Tho University

Article info.

Received date: 12/11/2015
Accepted date: 30/11/2016

Keywords

College of Engineering Technology, Can Tho University

ABSTRACT

Recently, demand of implementation stand-alone embedded systems has increased sharply. This article represents a method for building a framework that can implement control algorithms designed on Matlab/Simulink to embedded systems. An inverted pendulum stabilizing controller, running on a STMicroelectronics low-cost STM32F4 Discovery development kit, is used as a demonstration. The control algorithm, after designed and simulated on Matlab/Simulink, is configured to generate the corresponding embedded C code by using Matlab Real-time Embedded Coder. This code is combined with microcontroller peripheral libraries to make a complete Keil C project. In this project, the C language main function is generated automatically by using TLC – Target Language Compiler; however, users must write additional code to complete the whole coding. Experimental results of the demonstrative algorithm show that by using the proposed framework the control algorithm only has 0.001 ms of sample time error. Response quality of the inverted pendulum is as good as simulation results in term of fast response, low overshoot and steady error.

Cited as: Khanh, N.V., 2016. A framework for transferring algorithms designed on matlab/simulink to arm microcontroller embedded systems. Can Tho University Journal of Science. Vol 4: 36-45.

1 INTRODUCTION

The microcontroller STM32F407VG has been used in the STM32F4 Discovery kit, announced in 2011 by STMicroelectronics. It is widely used by students and researchers in universities. This is a low-cost, powerful microcontroller integrated an ARM Cortex – M4 core with speed up to 168 MHz and a hardware floating – point unit (FPU) which is especially for DSP (STMicroelectronics, 2011). The official development kit STM32F4 Discovery is designed using this microcontroller for low cost and suitable for fresh of studies (STMicroelectronics, 2015).

Recently, microcontrollers have been used in many studies. For example, Geekiyanage and Jayarathne published a robot platform which was used for

smart applications in the 7th International Conference on Sensor Technology in 2013 (Geekiyanage *et al.*, 2013). This robot used STM32F4 Discovery as a central processing unit to collect and analyze data from built-in sensors (ex. Encoder, SRF04, MPU6050, HMC5883L). Based on analysis results microcontroller calculates the trajectory and controls the robot. All robot data are also sent back to personal computer via serial communication.

Also in 2013, swam robots were published which can move together to detect fire (Chattunyakit *et al.*, 2013). The main processor of these robots is also STM32F407VG. These robots can detect fire by processing the images took from a built-in CMOS camera combine with a thermal couple. The experimental results show that the robot can move and detect fire well.

Besides that, the use of microcontrollers to implement and real-time execution algorithms also attract researchers. Typically, in the 7th Vietnam Conference on Mechatronics (VCM), Dr. Nguyen Chi Ngon has published a study about using MSP430 microcontroller family to implement a discrete PID which was utilized by a C programming language (Ngon and Truong, 2014). The experimental results of this study show that the discrete PID controller has good responses.

Although a lot of studies used microcontrollers to implement real-time algorithms or discrete controllers have been published, but most of them mainly based on C language. This method takes a lot of time for writing and debugging codes. Especially, when studies need to use others algorithm, the code must be rewritten from beginning. To overcome these disadvantages, this article represents the structure of a framework which enables to implement the algorithm designed and simulated by Matlab/Simulink on embedded systems. This framework will help researchers to take advantages of design toolboxes and built-in functions of Matlab/Simulink for their studies. So, it reduces the coding time for implementing embedded algorithms.

2 DEMONSTRATED ALGORITHM

The inverted Pendulum model in Process Control labs (Department of Automation Technology, Can Tho University) is nominated to evaluate the proposed framework. The inverted Pendulum, an example of under-actuated mechanical system, is a very popular experimental model in education and research. The system by its nature is nonlinear, so it is used to illustrate a lot of control problems in nonlinear control field (Krstic *et al.*, 1995).

There are many control algorithms which run on the inverted pendulum system. In this article, backstepping controller (Khalil, 1996; Tsai and Lin, 2003) is employed to control this model. Backstepping is also a powerful and flexible method to design nonlinear controllers (Nghia, 2007).

Let consider the inverted pendulum as shown in Figure 1.

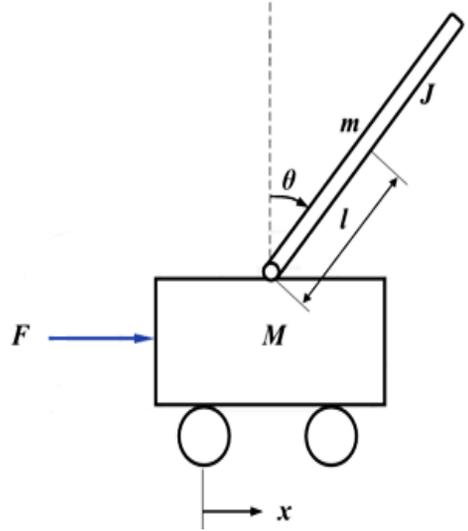


Fig. 1: Inverted pendulum system

Applying Lagrange's equation, the differential equation of the inverted pendulum is derived as following equations.

$$\begin{cases} (M + m)\ddot{x} - ml\dot{\theta}^2 \sin\theta + ml\ddot{\theta} \cos\theta = F - b\dot{x} \\ ml\ddot{x} \cos\theta + (J + ml^2)\ddot{\theta} - mgl \sin\theta = 0 \end{cases} \quad (1)$$

where M, m, l, b, J, x and θ are respectively cart mass, pendulum mass, pendulum length, friction coefficient, moment of inertia, cart displacement and pendulum angle.

Linearizing (1) around $\theta = 0$ point leads to:

$$\begin{cases} (M + m)\ddot{x} + ml\ddot{\theta} = F - b\dot{x} \\ ml\ddot{x} + (J + ml^2)\ddot{\theta} - mgl\theta = 0 \end{cases} \quad (2)$$

Equations in (2) are rewritten as the following state equation:

$$\dot{z} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-(J+ml^2)b}{J(M+m)+Mml^2} & \frac{-m^2gl^2}{J(M+m)+Mml^2} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{mlb}{J(M+m)+Mml^2} & \frac{mgl(M+m)}{J(M+m)+Mml^2} & 0 \end{bmatrix} z + \begin{bmatrix} 0 \\ \frac{J+ml^2}{J(M+m)+Mml^2} \\ 0 \\ -ml \\ \frac{J+ml^2}{J(M+m)+Mml^2} \end{bmatrix} u, \quad (3)$$

where $\dot{z} = [\dot{x}_1 \ \dot{x}_2 \ \dot{x}_3 \ \dot{x}_4]^T$, $z = [x_1 \ x_2 \ x_3 \ x_4]^T$, $x_1 = x, x_2 = \dot{x}, x_3 = \theta, x_4 = \dot{\theta}, u = F$.

The objective of the back-stepping controller is to stabilize the inverted pendulum at up-right equilibrium ($\theta = 0$) and tracking the cart displacement reference signal. By using the design method is shown in (Khanh, 2014), the equation of control law is obtained:

$$u = \frac{1}{ml+(J+ml^2)k_1}(h_1x_1 + h_2x_2 + h_3x_3 + h_4x_4) \quad (4)$$

where

$$h_1 = -k_1d_2(J(M + m) + Mml^2),$$

$$h_2 = mlb + (J + ml^2)bk_1 - k_1d_1(J(M + m) + Mml^2),$$

$$h_3 = mgl(M + m) + m^2gl^2k_1 +$$

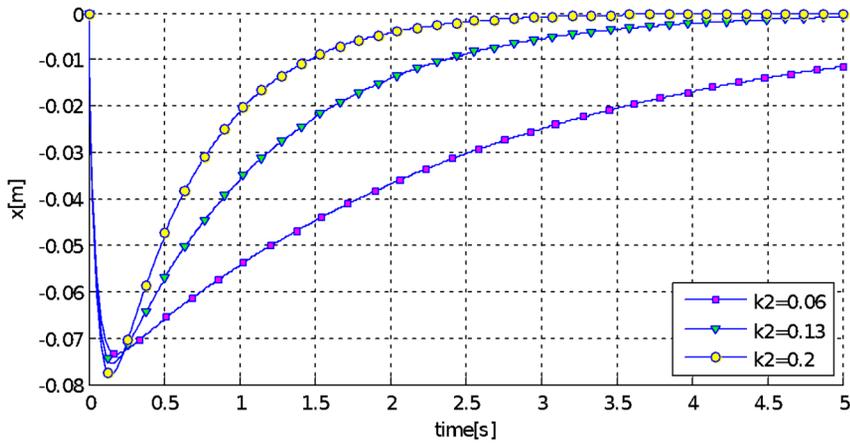
$$d_2(J(M + m) + Mml^2),$$

$$h_4 = d_1(J(M + m) + Mml^2),$$

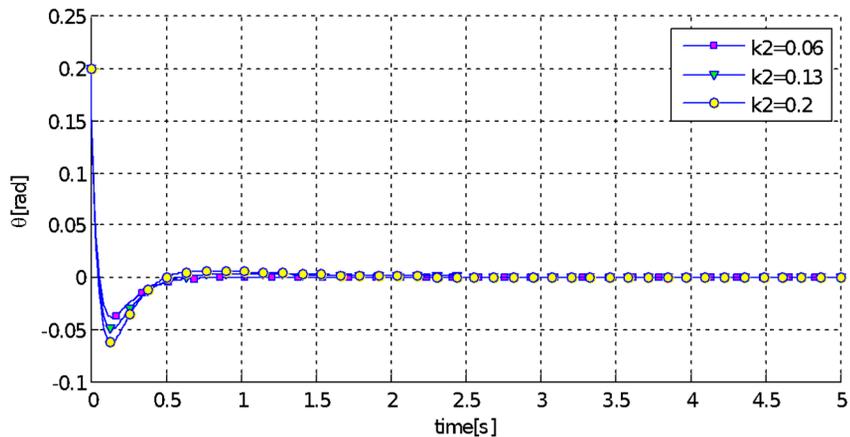
k_1, k_2, c_1, c_2 : positive design constants,

$$d_1 = c_1 + c_2, d_2 = c_1c_2 + 1.$$

The simulation results of this control law on Matlab/Simulink are shown in Figure 2. There are four parameters of this controller, which can adjust the response of the inverted pendulum. In this study, the try-and-error method is used to figure out these values. In fact, when $d_1=50, d_2=300, k_1=3.0$ and k_1 equals 0.06, 0.13, 0.2 consecutively, the inverted pendulum will get the desired responses. As can be seen in subfigure 2a that the displacement of the inverted pendulum stabilizes quickly when k_2 changes from 0.06 to 0.2. When $k_2=0.02$, the inverted pendulum only needs about 2.5 s to reach 0 m again. However, subfigure 2b shows that k_2 value does not affect to the pendulum angle response, but it slightly increases the overshoot of this response.



a)



b)

Fig. 2: The simulation response of backstepping controller. a) Displacement response, b) Tilt angle response

3 IMPLEMENTATION

3.1 Hardware description

Figure 3 shows the prototype of the inverted pendulum. The designed model has an iron chassis, a motor for moving the position of the cart, two high resolution incremental encoders for measuring cart

displacement and pendulum angle. Figure 4 illustrates that this model is controlled by STM32F4 Discovery development kit designed based on STM32F407VG microcontroller. Two encoders are read directly by two built-in decoder modules. The DC motor is controlled by PWM module via the power circuit using LMD18200.



Fig. 3: The prototype of an inverted pendulum model

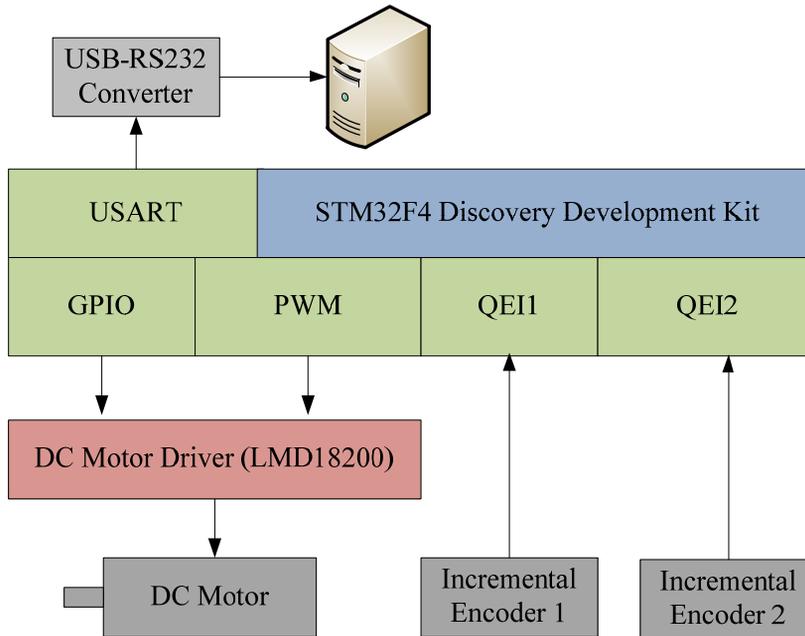


Fig. 4: Hardware description

Parameters of the inverted pendulum are given in Table 1.

Table 1: The inverted pendulum parameters

Parameters	Value	Units
M	1.2	kg
m	0.1	kg
l	0.26	m
b	0.08	-
J	0.00386	kgm ²

3.2 The proposed framework to embed systems

Simulink is a graphical programming environment for modeling, simulating and analyzing for multi-

domain dynamic systems (MathWorks, 2016a). It is integrated many useful toolboxes of various fields, especially in automation control. Simulink deploys Model-Based Design (MathWorks, 2016b) to design complex control systems. Using Simulink, it is so easy to model, analyze and synthesize the plant's controller, integrate all phases of deploying controller. Especially, Simulink can form embedded C code from Simulink program to help integrate them to embedded systems.

In order to take the advantage of design system toolboxes of Simulink and reduce design time and cost of embedded controllers, this paper proposes a

general framework which is used to design a real-time controller by combining Simulink toolboxes and a platform-specific microcontroller. Figure 5 shows a block diagram of a closed loop control system. Assume that it is desired to design a real-time controller for a plant using the diagram in Figure 5. The proposed framework is a complete embedded C project comprised three parts of code. The first part uses peripheral libraries to read feedback signals from the plant. The feedback data are deployed to calculate the error e signal by subtracting from reference signal r . The second part is the controller code which is the most difficult part if it

is written manually. So, it is suggested that the controller will design and simulate on Simulink until having a good response. Such, final controller is customized to generate embedded C code by using Real-time Embedded Coder. The final part is also used peripheral libraries to convert output u to analog signal controlled the plant. All three parts of this code will be integrated into the C code project to compile and run on embedded systems. Based on the response of real-time controller, if it does not meet the designed objective, go back to modify its parameters on Simulink to improve the response.

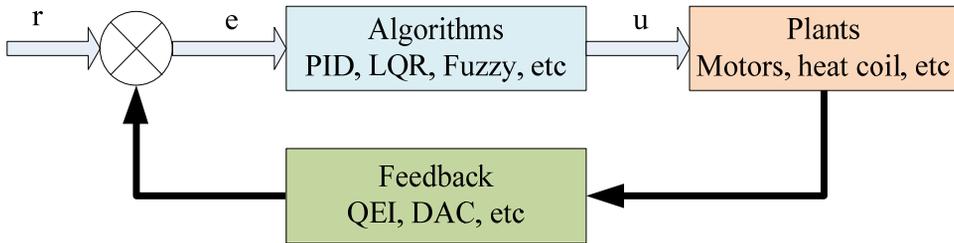


Fig. 5: Close loop control diagram

To help the proposed framework to deploy easily, all of the libraries are previously developed and integrated into a C project for STM32F407VG microcontroller. However, if users want to use other microcontroller platform, they can write their own libraries. So, used the proposed framework user only designs and simulates controller on Simulink, then integrates the generated C codes to project, writes some codes to link controller with the project. Figure 6 shows the steps of designing embedded controller using the proposed framework.

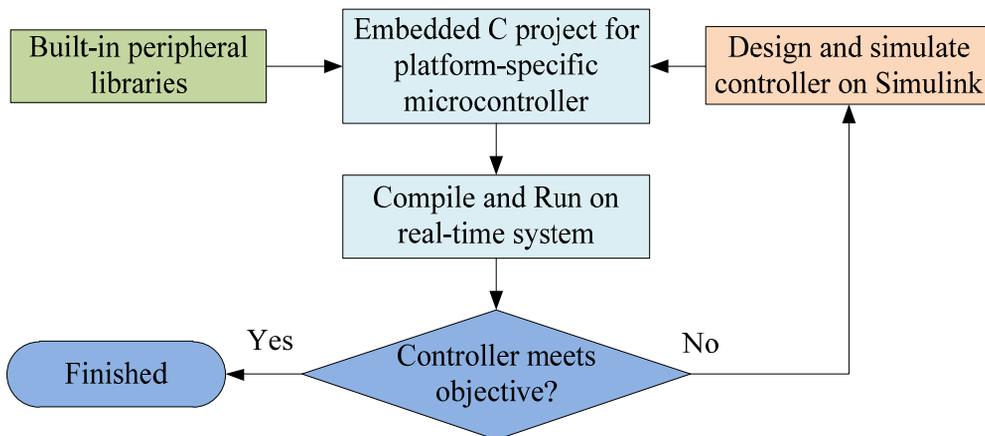


Fig. 6: Design flowchart

To access the inputs/outputs of the controller designed in Simulink, signals must be defined on every input/output and configured to generate into global variables. This helps access signals in C code easily. This paper proposes two Simulink

structures of controller based on how the controller is generated. First, if all of the controller will be generated, input and output signals are defined as shows in Figure 7.

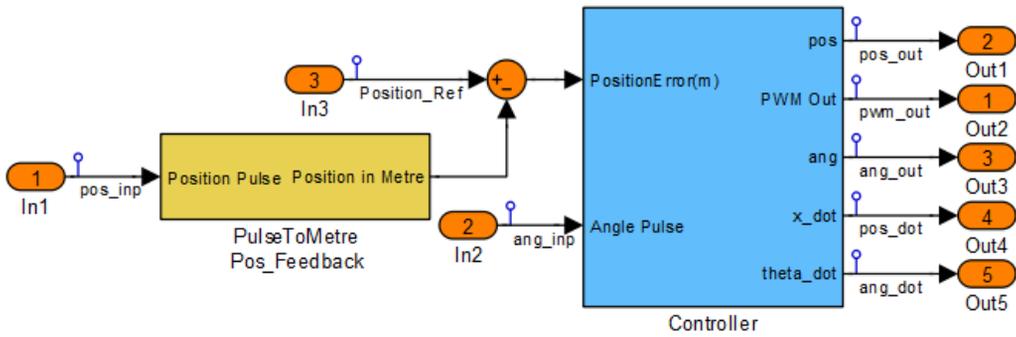


Fig. 7: Simulink structure for all generation

Second, if controller is splitted into some blocks which are generated to C code independently, the controller is redesigned into some subsystem as subfigure 8a. In this case, each subsystems must be defined their own input/output signals as shows in subfigure 8b detailed designed block of Enable_PWM subsystem. This style is very helpful

when the controller is complex and needs to design in group. In this case, the Gain block must be added to avoid compiling errors. It is noted that subsystem function codes must be called in right order to reconstruct algorithm correctly on embedded systems.

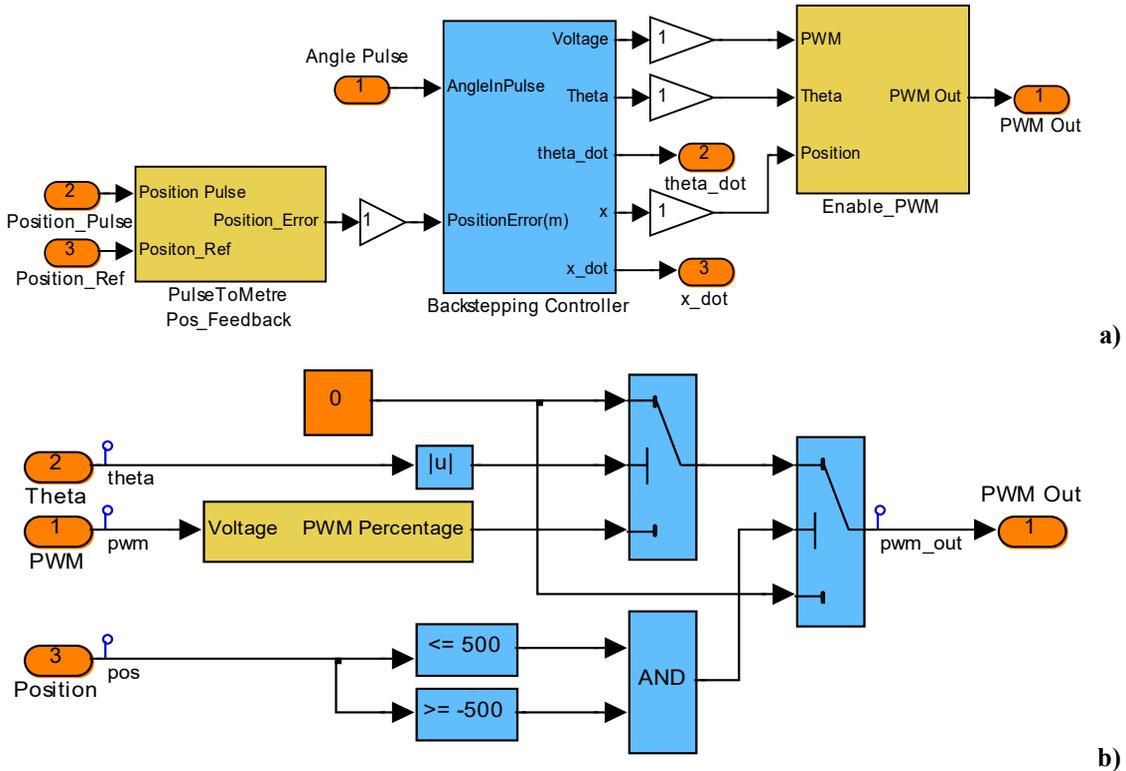


Fig. 8: Simulink structure for block generation; a) Simulink program of algorithm; b) Signal definition of Enable_PWM subsystem

After designing and defining signals, Simulink parameters must be configured to generate controller to embedded C language. There are two groups of parameters must be configured by user as shown in Table 2. Based on Simulink design, as mentioned above, there are two generating ways. First, use command Tools\Real-Time Workshop\Build model if whole controller is generated. Second,

click right-mouse on subsystems of controller and choose Real-Time Workshop\Build Subsystem if controller is designed into some subsystems. Generated codes are combined with built-in libraries in a C project to compile and program to embedded systems as shown in Figure 9. If controller does not meet objectives, return to modify its parameters and evaluate result again.

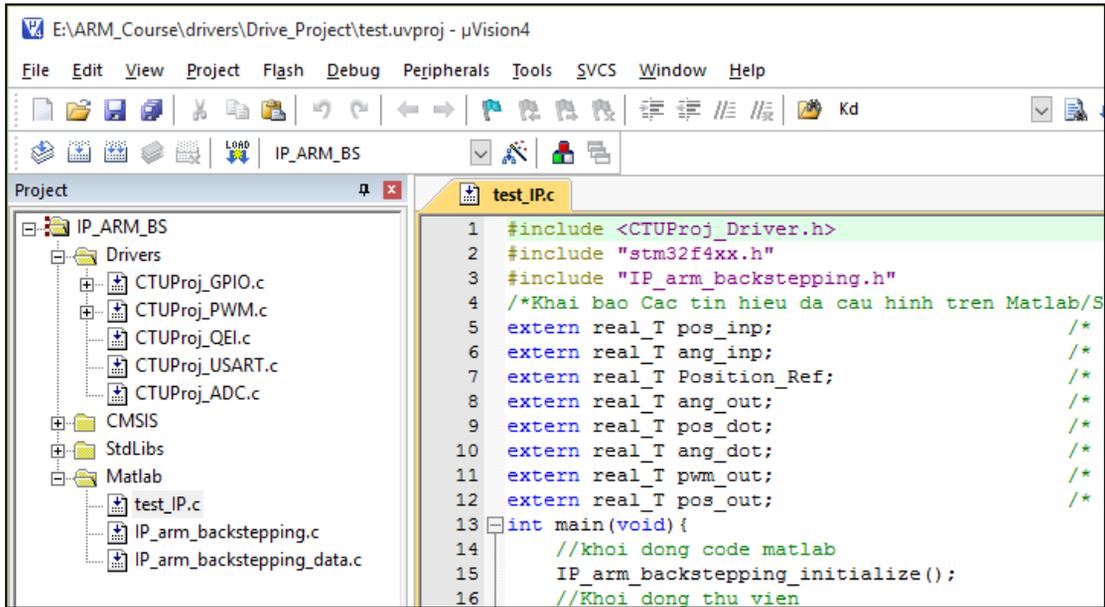


Fig. 9: Final Keil C project of the proposed framework

Table 2: Simulink parameters

Group	Parameter	Value
Solver	Type	Fixed-step
	Solver	discrete
	Fixed-step size	0.01
Real-Time Workshop	System target file	ert.tlc
	Generate code only	checked

4 EXPERIMENTAL RESULTS

In this section, some experimental results of the inverted pendulum controller are conducted to sup-

port the proposed framework. Figure 3 shows the inverted pendulum system used in the experiment. The model parameters are given in Table 1. The controller parameters are slightly changed in k_1 and k_2 , and their values equal 0.29 and 0.06 respectively. There are three main experiments. The first experiment is to measure the sample time of an embedded system to make sure that it is produced exactly on the target system. A 32-bit timer module is used to measure this time and show in Figure 10. The graph shows that the sample time is generated exactly has a very small error, 0.01 ms.

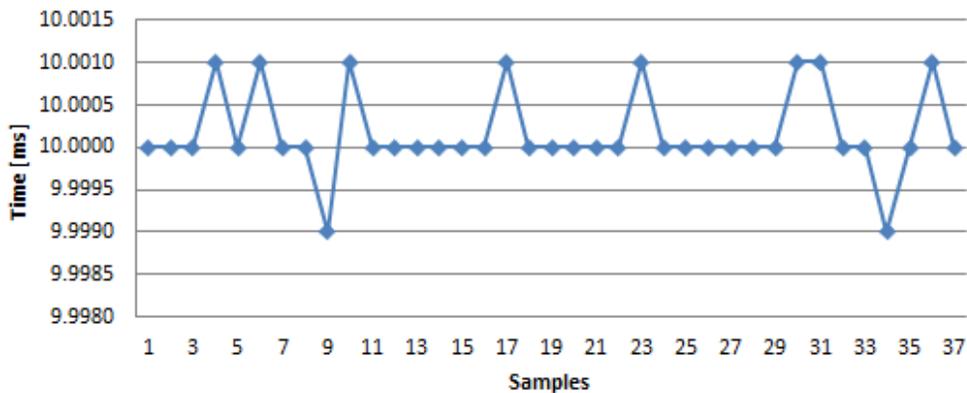


Fig. 10: Sampling time result

The second experiment is to measure the step response of the realtime controller. The initial value of pendulum angle and cart position are 0.2 rad and 0 m respectively. After cart is stabilized, a small force is also impacted to pendulum in order test the reaction of controller to external noises. The result shows that the controller stabilized pendulum at

up-right and moved the cart back to zero after less than 2 s and as shown in Figure 11. When there is an external force which impacts to the pendulum, controller moved cart toward quickly to avoid the pendulum drop down and moved back to zero again. Table 3 shows the comparison between simulation and experimental results. The simulation

response in the case $d_1=50$, $d_2=300$, $k_f=3.0$ and $k_l = 0.2$ is used to make this comparison. The table shows that the controller has the same qualities in both cases except in overshoot specification. Based on both simulation and experimental results, it is shown that cart is also controlled back to 0 m (i.e

this is cart displacement input reference of controller). In practical, however, cart only needs 1.5 s to reach back 0 m, while it needs 3 s in simulation. The steady error of cart displacement is zero in simulation, but it is 0.053 m in experiment

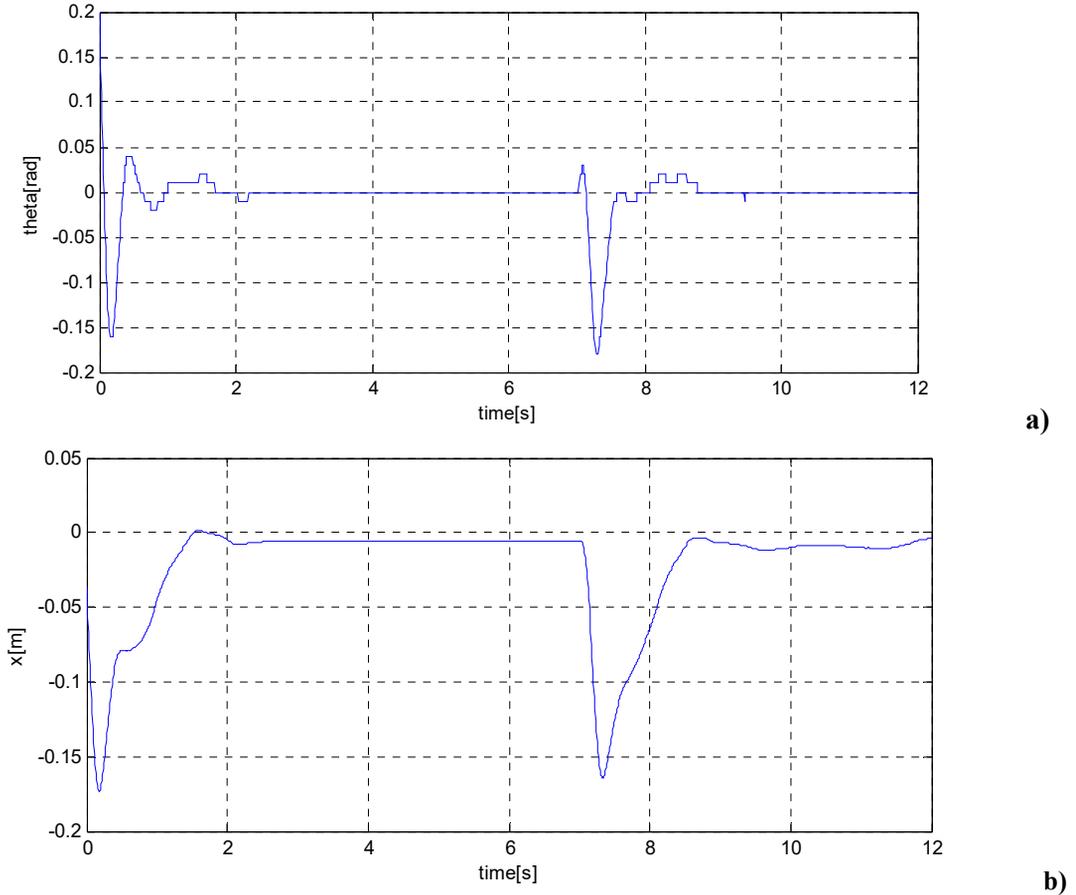


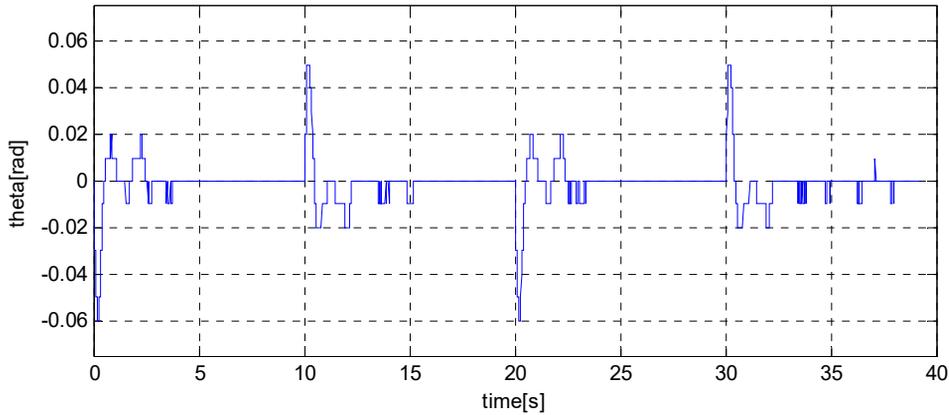
Fig. 11: Step response of the inverted pendulum; a) pendulum angle, b) cart displacement

Table 3: The control quality of angle

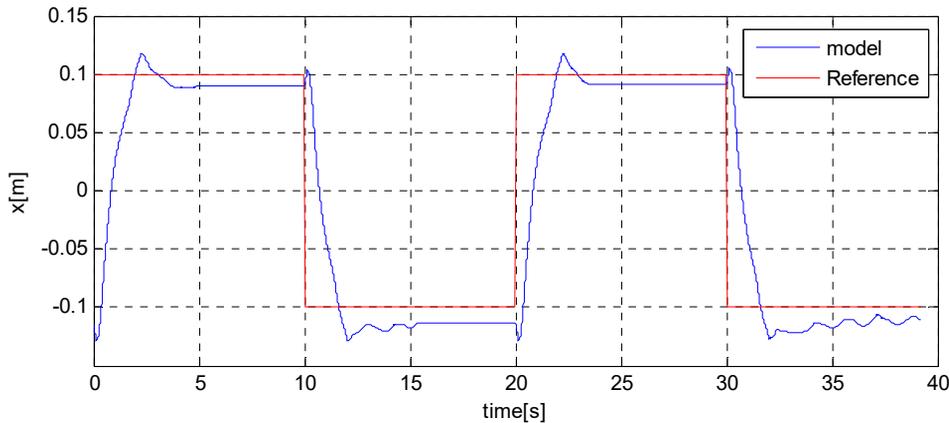
Standard	Simulation	Experiment
Rising time	0.041 s	0.057 s
Steady time	0.39 s	1.58 s
Steady error	0	0
Overshoot	26 %	75%

Last experiment is to track the cart follow a pulse which changes state every 10 s. The result is given

in Figure 12. The figure shows that the controller can track the cart to a pulse reference input while try balancing the pendulum at the up-right equilibrium. The maximum steady error of cart displacement is 0.01 m (Fig. 12a). Pendulum angle is fluctuated as displacement reference input changes, but it is quickly stable and has no steady error (Fig. 12b).



a)



b)

Fig. 12: Pulse tracking response of the inverted pendulum; a) pendulum angle, b) cart displacement

In conclusion, the real-time controller designed by the proposed framework operated well on practical model. In fact, the sample time is formed exactly on embedded systems. This is an important factor to evaluate the controller because it influences the calculation of the controller. Experimental results are also proved that the nominal controller operated well not only in simulation, but also in practical with good response.

5 CONCLUSIONS

The proposed framework for embedded systems is presented in this work. This framework facilitates researchers to employ algorithms designed in Simulink, a powerful designed tool, for embedded systems by integrating C code was formed from them to an embedded C project. The experimental results show that the nominal controller which is designed successful by using the proposed framework operated well on the inverted pendulum system. The pendulum is balanced at the up-right equilibrium with no steady error. The cart can track input reference signals and operate in a limited range. The maximal steady error of card displacement is 0.01

m. This proposed framework provides a good solution for embedded systems and a suitable tool for rest of embedded designers.

ACKNOWLEDGEMENTS

First of all I am thankful to Can Tho University for financial and logistical support and for providing necessary guidance concerning for my project whose code is T2015-17. I am also grateful to the Department of Automation Technology for the provision of expertise, and technical support in the implementation.

REFERENCES

- STMicroelectronics, 2011. Reference manual STM32F405xx, STM32F407xx, STM32F415xx and STM32F417xx advanced ARM-based 32-bit MCUs. 1315 pages.
- STMicroelectronics, 2015. STM32F4DISCOVERY, assessed on 20 October 2015. Available from <http://www.st.com/web/catalog/tools/FM116/SC959/SS1532/PF252419>
- Geekiyana, P., Jayarathne, H.T., Jayasinghe, L.A.D.I.T., Amarasinghe, Y.W.R., 2013. Development of Upgradable Mobile Platform for Smart Ap-

- plications. The Seventh International Conference on Sensing Technology, pp.841-847
- Chattunyakit, S., Kondo, T., Nilkhamhang, I., 2013. Development of Robotic Platform for Swarm Robots in Fire Detection Application. The Kasetsart Journal. 47: 967-976
- Ngon, N., Truong, N.M., 2014. Applying the MCU-MSP430 to develop digital PID controller kits. The 7th Vietnam Conference on Mechatronics. 7: 96-100 (in Vietnamese)
- Krstic, M., Kanellakopoulos, I., Kokotovic, P.V., 1995. Nonlinear and Adaptive Control Design. Wiley-Interscience Publication. New York. 576 pages
- Khalil, H.K., 1996. Nonlinear Systems. Prentice Hall. New Jersey. 750 pages
- Tsai, F.K., Lin, J.S., 2003. Nonlinear Control Design of 360-Degree Inverted Pendulum Systems. The Fourth International Conference on Control and Automation. 634-638
- Nghia, D.H., 2007. Multivariable control systems. Vietnam National University. Ho Chi Minh City. 68 pages (in Vietnamese)
- Khanh, N.V., Hao, N.V., Phong, N.N., 2014. Stabilization control an inverted pendulum using backstepping controller. Can Tho University Journal of Science. 31: 18-25 (in Vietnamese)
- MathWorks, 2016. Simulink, assessed on 19 January 2016. Available from http://www.mathworks.com/products/simulink/?s_tid=hp_fp_sl
- MathWorks, 2016. Model-Based Design, assessed on 19 January 2016. Available from <http://www.mathworks.com/solutions/model-based-design/>