Setting up MATX-based Laboratory Experimental Systems - A Pendulum System with Windows OS -

Masami Iwase * **Shingo Kojima** ** **Teruyoshi Sadahiro** *
Shoshiro Hatakeyama *

* *Department of Robotics and Mechatronics, Tokyo Denki University, Tokyo, Japan, (e-mail: {iwase,sadahiro,sho}@fr.dendai.ac.jp).*
** *Graduate School of Computers and Systems Engineering, Tokyo Denki University, Saitama, Japan, (e-mail: kojima@hatalab.k.dendai.ac.jp)*

Abstract: This paper presents setting up experimental systems based on MATX in laboratories and undergraduate school. This configuration has advantages of introduction cost, maintenance fee, easy reconstruction compared with commercial equipments. The advantages and effectiveness of the proposed experimental system configuration are demonstrated in this paper by using a Furuta pendulum system as an example.

1. INTRODUCTION

Setting up experimental systems in laboratories and undergraduate school with general-purpose devices and free softwares has several advantages from educational viewpoints such as deep observation to controller equipments, and easy reconstruction in accordance with each intended use as well as from cost viewpoints such as introduction cost and maintenance fee, when compared with commercial experimental systems.

In the embedded systems area, as Koopman et al. [1992] says, experts, who are confident in electric devices including I/O devices and micro computers and any necessary softwares concerned with operating such devices, are required to make compact, specialized, low-cost systems with high integration density. There are so many variations of embedded systems generally because they are customized and specialized for each intended use. Sometimes specialized hardwares are developed for a use, and then the softwares associated with the hardwares also have to be developed. Real-time control is essentially required for those embedded systems to operate target plants. Abundant experiences and deep knowledges about system integration are also required. Hence, to built some total experimental system in laboratories and undergraduate school may be good occasion and chance to acquire such experiences.

MATX, a multi-platform free software developed by Koga [1992] and Koga [1999], is a programming language with easy description compatible with C language. It supports numerical analysis algorithms and calculations in science and engineering fields handling integer, real, and complex values, vectors, arrays and matrices, polynomials, rational polynomials, character strings, files, etc. MATX also equips fulfilling functions for control system analysis and design, and provides a real-time control environment for real systems on Windows platform. We focus on the following features of MATX: 1) any I/O devices and hardwares whose device drivers for use on Windows are provided by developers can be used from MATX. 2) the sequence procedure over analysis, design, computer simulations and real experiments can be performed with MATX. 3) MATX has fulfilling functions for control system analysis and design. Those features allow us to make experimental systems based on MATX, and several examples have been reported by Hatakeyama [1999] and Hoshino [1999], for example.

In this paper, we present the built-up process of a Furuta pendulum system in our laboratory. We also describe the configuration of the real-time system with MATX, especially focusing on how the devices with drives for Windows OS use are connected to MATX. The experimental system is built up for swinging up a Furuta pendulum by a state-dependent Riccati equation based control. In the control method, the state-dependent Riccati equation has to be solved at each sampling interval, and it implies that some complicated matrix calculations are necessary to solve the Riccati equation every sampling interval. However, utilizing MATX-based experimental systems, those requirements are easily provided. We intend to demonstrate those advantages of introducing MATX-based experimental systems via the example.

2. EXPERIMENTAL FURUTA PENDULUM SYSTEM

A Furuta pendulum shown in Fig.1 is used as the experimental system. The Furuta pendulum consists of a single pendulum and a rotational motor with an arm. The pendulum is hinged to the arm of the motor via a passive joint. Therefore this system is under-actuated.

The configuration of the total Furuta pendulum system is shown in Fig.2. The motor accepts torque commands as analog voltage signals from the computer, and then a D/A interface is required to plug the motor to the computer. The system has two measurements, the rotational angles of the motor and the pendulum. The rotational angle of the motor is picked by a resolver installed in the motor

Fig. 1. Photo of a single Furuta pendulum. A pendulum is hinged to the arm of the rotating motor. The system has only one actuator and two rotational encoders for angle measurement.

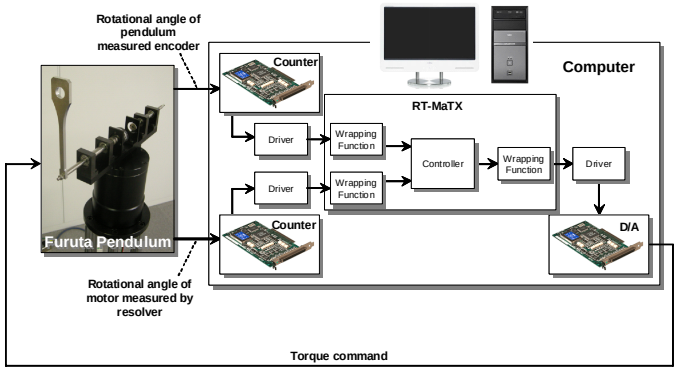

Fig. 2. A block diagram shows the experimental system configuration.

Table 1. List of devices used in the experimental system.

Direct drive motor	DR5B-030G (YOKOGAWA)	max torque 30 [Nm], max speed 5.0 [rps]
Resolver (motor)	-	557056 [p/rev]
Rotary encoder (pendulum)	MES-30-4500PST4 (MTL)	4500 [p/rev]
I/O board	PCI-360116 (Interface)	D/A (16bit), Counter (1ch, 24bit)
Counter board	PCI-6201 (Interface)	Counter (4ch, 24bit)
Computer	- (Hand-made)	OS: Windows XP SP3, CPU: Core2 Duo 2.33GHz, Memory: 2.0GB RAM

inside. The angle of the pendulum is measured by a rotary encoder. To capture those angles, a counter board is required. In our system, the devices and equipments used in the system are general-purpose products which are easy to be get, and can be replaced by similar others. The equipments and devices with specifications are listed in Table 1.

A mathematical model of the Furuta pendulum is necessary to analyze and design its control system. The coordinate system, parameters and variables in this paper are taken as shown in Fig.3. The notations and quantities of parameters are also listed in Table 2. As shown in Åström [2006], the equations of motion of the Furuta pendulum system is given by

$$\begin{aligned}
&\left(J + m_p\ell^2\right)\ddot{\theta} - m_p r\ell\cos\theta\ddot{\varphi} \\
&\quad = (J + m_p\ell^2)\dot{\varphi}^2\sin\theta\cos\theta + m_p\ell g\sin\theta - c_p\dot{\theta} \\
&\left((J_a + m_p r^2) + \left(J + m_p\ell^2\right)\sin^2\theta\right)\ddot{\varphi} - m_p r\ell\cos\theta\ddot{\theta} \\
&\quad = \tau - c_a\dot{\varphi} - 2\left(J + m_p\ell^2\right)\dot{\varphi}\dot{\theta}\sin\theta\cos\theta - m_p r\ell\dot{\theta}^2\sin\theta
\end{aligned} \tag{1}$$

where φ, θ and τ are the motor rotational angle, the pendulum tilting angle and the motor torque, respectively.

We often meet situations in which a current system doesn't match to solve a problem at hand. In that case, we need modify the system desirably, and want to modify it as soon as possible, if feasible, with available something accessed easily. We think integration also plays a prominent role in problem resolution, although development of high performance devices and equipments for the special use are important. Hence we much pay attention to integration with MATX in the following sections.

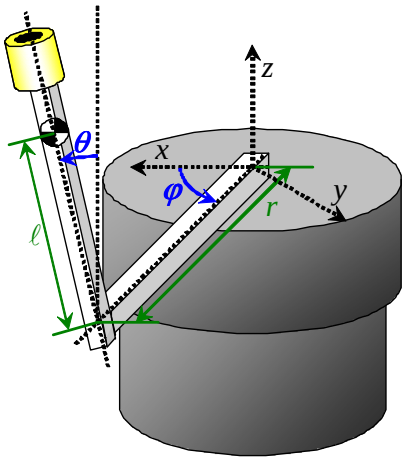

Fig. 3. Schematic figure and a coordinate system of a single Furuta pendulum.

Table 2. Parameters in the Furuta pendulum.

Mass of pendulum	m_p	0.065 [kg]
Distance from the pivot to CG of pendulum	ℓ	0.123 [m]
Length of arm	r	0.215 [m]
Moment of Inertia of pendulum	J	0.0001554 [kgm^2]
Moment of Inertia of arm	J_a	0.0669 [kgm^2]
Gravity acceleration	g	9.81 [m/s^2]
Viscus friction coefficient of the pivot	c_p	2.102×10^{-4}[Nms/rad]
Viscus friction coefficient of the motor	c_a	0.0926779 [Nms/rad]

3. CONFIGURATION OF REAL-TIME SYSTEM WITH RT-MATX

3.1 Connectivity

The proposed configuration of real-time experimental systems with MATX is illustrated in Fig.4. Any hardwares

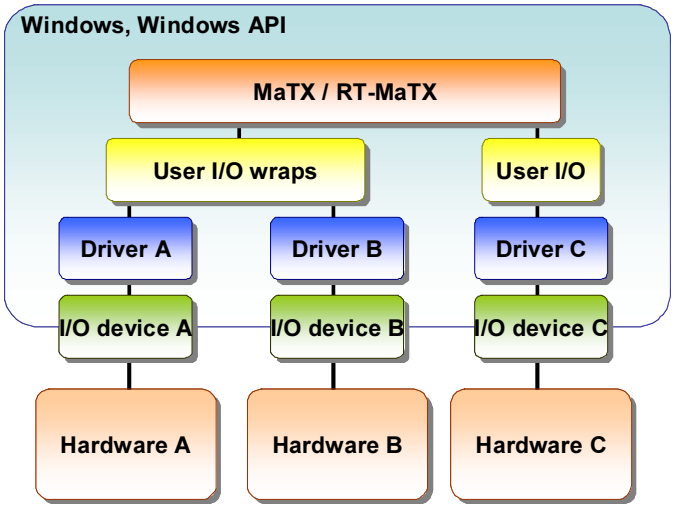

Fig. 4. The schematic figure shows the configuration of real-time systems with RT-MATX.

```
// Head files
#include "Gpcda.h"    // for I/O common
#include "Fbida.h"    // for D/A
#include "FbiEnc.h"   // for counter

// Declare functions called from MaTX
extern "C" int __cdecl DA_Open(void);
extern "C" int __cdecl DA_Close(void);
extern "C" int __cdecl DA_Out(WORD);
extern "C" int __cdecl Counter_Open(void);
extern "C" int __cdecl Counter_Close(void);
extern "C" int __cdecl Counter_Get(int);

// Libraries to be linked
#pragma comment(lib, "FbiDaDC.lib")
#pragma comment(lib, "FbiDa.lib")
#pragma comment(lib, "fbienc.lib")
```

Fig. 5. The header part of a sample code for user defined wrapping functions.

are available for the configurations as long as the corresponding device drivers are provided with libraries which can be used from C and C++. This feature is very useful to introduce new devices and equipments, and has some advantage because commercial products such as MATLAB require special drivers and sometimes need to wait for providing those special drivers. In the configuration, we just prepare a small user defined wrapping functions in C or C++ languages, and it is always very easy. In Fig.4, pairs of hardwares, I/O devices and drivers should have connectivity. The connectivity between MATX and drivers should be ensured by the user defined wrapping functions. Therefore what we need prepare from software aspect is only two programs: user programs for MATXand user defined wrapping functions.

In this configuration, some user defined functions are required to bridge a gap between MATXand device drivers provided like Windows API. The default calling convention in Windows API is `__stdcall`. On the other hand the default calling convention for C/C++ programs is `__cdecl`. In almost case, device drivers for windows are written in the `__stdcall` convention, however, executables in MATX are written in the `__cdecl`. Hence we need the user defined functions playing wrapping functions. What we need remind in such programs is only to declare external functions with `__cdecl`. An example code such is given in Fig.5.

3.2 RTMATX

MATX provides a real-time environment which is called RT-MATX. Codes for RT-MATX are completely compatible with ones for MATX. The real-time implementation with RT-MATX consists of mainly three functions: the main function, `main()`, the function executed in real-time, `on_task()`, and the function executed in non real-time, `off_task()`. Additionally a function called when a user wants to break the execution is `break_task()`. A skeleton for the real-time implementation is given in Fig.6. `on_task` may be called each sampling interval which is set by `rtSetClock()`. The function name, `on_task`, should be registered by `rtSetTask()`. The similar procedure for `break_task()` is required.

`on_task()` may contain some function calls for getting information from sensors and for putting input signal to actuators. In `on_task()`, users can also write any codes for MATX including function calls, matrix calculations etc. as long as those codes can be executed within the set sampling interval. This fact permits us to much easily implement sophisticated controllers or complicated controllers which sometimes require messy calculation, large memory, complicated if-then rules and so on. In the next section, a State-Dependent Riccati equation control, which require to solve a matrix Riccati equation depending a nonlinear state-space representation every sampling interval, shows up. However, we can write only a few codes to realize this thanks to (RT)MATX. `off_task()` plays a user interface function for showing and changing parameters, and control of the mode.

Fig.7 shows relationship between files and MATX. MATX consists of four systems, `matx`, `matc`, `rtmatx` and `rtmatc`. `matx` and `rtmatx` are interpreters which can accept command lines interactively or run any scripts and source codes written in `*.mm` format. `matc` and `rtmatc` are compilers to make an executable from source codes in `*.mm` or `*.c` formats.

At analysis, synthesis and simulation stages, so much and frequent feedback is required to fix, modify and improve programs. Users can do this without any compiling process, and also check and verify execution results and status easily from command lines because `matx` and `rtmatx` are interpreters. This fact can be regards as an advantage of MATX.

Once the program is fixed, an corresponding executable can be obtained by compiling the source code with `matc` or `rtmatc`. The executable can run faster than the interpreter `matx`, so is essentially necessary for real-time operating. `matc` and `rtmatc` can also compile any source codes written in common C/C++ languages, and can link adequate libraries to make the executable. Note that some linker options are sometimes required when compile source codes by `rtmatc`. A sample is given as follows.

```
// Declare external functions
Integer DA_Open();
Integer DA_Close();
Integer DA_Out();
Integer Counter_Open();
Integer Counter_Close();
Integer Counter_Get();

// Break task
Func void break_task()
{
  rtStop();          // Stop the online task
  DA_Out(0);         // Clear DA output
  DA_Close();        // Close DA I/O
  Counter_Close();   // Close Counter I/O
}

// Online task
Func void on_task()
{
  // write online task here
}

// Online task
Func void off_task()
{
  // write offline task here
}

// Main
Func void main()
{
  Real dt;
  void on_task(), off_task(), break_task();

  dt = 0.01;
  dt = rtSetClock(dt); // Set sampling time
  rtSetTask(on_task);    // Set online task
  rtSetBreak(break_task); // Set break task

  DA_Open();        // Open D/A I/O
  Counter_Open();   // Open counter I/O

  rtStart();        // Start real-time task

  off_task();       // Call offline task

  rtStop();         // Stop real-time task

  DA_Out(0);        // Clear D/A output
  DA_Close();       // close D/A I/O
  Counter_Close(); // close Counter I/O
}
```

Fig. 6. A skeleton of RT-MATX program.

```
rtmatc -v *.mm *.lib -link -NODEFAULTLIB:libcd
 -NODEFAULTLIB:libcmt
```

In usual case, drivers for I/O devices are provided as a library file, `*.lib`. User defined wrapping functions are prepared in `*.c` or `*.cpp` forms. Finally an executable is

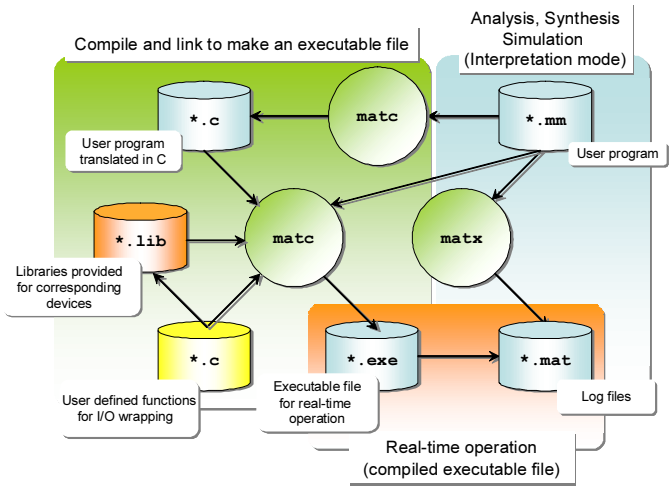

Fig. 7. The diagram shows the relationship among files concerned and the use of `matx` and `matc`.

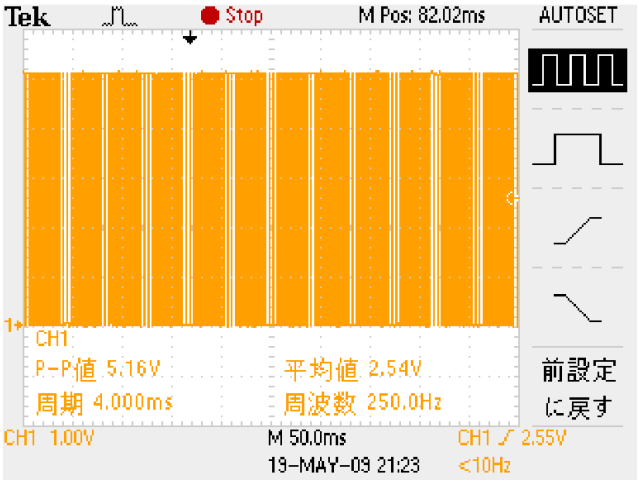

Fig. 8. A captured screen of an oscilloscope measuring a pulse pattern generated by RTMATX with 4.0[msec] sampling interval.

obtained via `matc`. The executable may be used in almost all real-time systems.

3.3 Jitter variation test

Basically, RTMATX realizes the real-time environment on the Windows operating system. Accurate timers and well-conceived interrupt structure are necessary for realizing good real-time performance. Windows has several timers, and RTMATX seems to use a multimedia timer. The time resolution of the multimedia timer is 1[msec]. However, its accuracy isn't well known. Therefore we tried to check the accuracy of the real-time performance. We generated a on-off pulse pattern with 4.0 [msec] time cycle and 50% on-off duration ratio, and output the pattern as voltage signal from D/A board. The signal was measured by an oscilloscope as shown in Fig.8. The jitter variation of the measured signal was analyzed. Fig.9 is the histogram of the jitter variation in this case. From the figure, the real-time performance of RTMATX is almost accurate because 95% situations were performed with about 4.0[msec] sampling interval. The remaining 5% situations are distributed around 5.0[msec]. As the result, even though the real-time

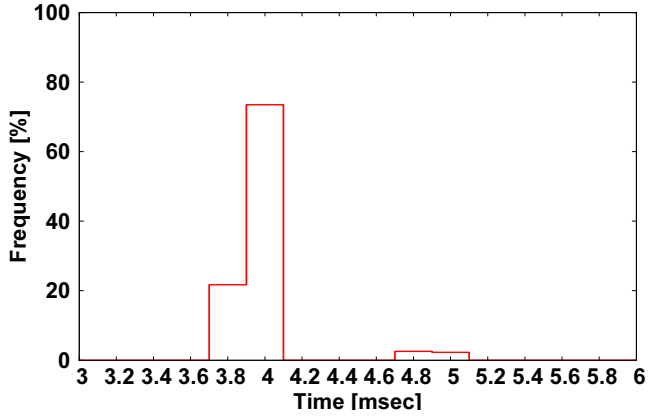

Fig. 9. The histogram of jitter variation when a RTMATX program was running with 4.0[msec] sampling interval. The vertical quantities shows each frequency with percentage. The frequency around 4.0[msec] is about 95%. Only 5% situations with longer sampling interval appeared.

performance of RTMATX is almost reliable, sometimes longer sampling period cases may appear. In those longer sampling cases, the jitter variation seems to be concentrated around 1.0[msec] that is equal to the time resolution of the multimedia timer. Therefore RTMATX can realize the real-time environment in which the minimum time resolution is 1.0 [msec] and the accuracy is also 1.0 [msec].

4. EXAMPLE: SWING-UP OF FURUTA PENDULUM BY SDRE CONTROL

Swinging up a single Furuta pendulum by the State-Dependent Riccati Equation (SDRE) approach on the real-time environment based on RT-MATX is presented as an example. A single Furuta pendulum system has been shown in Fig.3. The SDRE approach is named after a state-dependent Riccati equation which has to be solved every sampling interval to get a stabilizing control law. Basically, a state-dependent coefficient (SDC) form in the state-space representation is required in the SDRE approach. Note that the parameterization of SDC is not unique and the optimality under the SDRE approach depends on the choice of the parameterization. Therefore a control law obtained by the SDRE approach is regarded as a quasi-optimal control. We can refer to Çimen [2008] for the details about the SDRE control. Here we chosen a SDC parameterization as follows:

$$
\begin{aligned}
M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q)q &= \tau \\
q &= [\varphi\ \theta]^T \\
M(q) &= \begin{bmatrix} J_a + m_p r^2 + (J + m_p\ell^2)\sin^2\theta, & -m_p r\ell\cos\theta \\ -m_p r\ell\cos\theta, & J + m_p\ell^2 \end{bmatrix} \\
C(q,\dot{q}) &= \begin{bmatrix} (J + m_p\ell^2)\dot{\theta}\sin(2\theta) + c_a, & m_p r\ell\dot{\theta}\sin\theta \\ -(J + m_p\ell^2)\dot{\varphi}\sin\theta\cos\theta, & c_p \end{bmatrix} \\
G(q) &= \begin{bmatrix} 0, & 0 \\ 0, & -m_p\ell g\sin\theta/\theta \end{bmatrix}.
\end{aligned}
\tag{2}
$$

With (2), we have a SDC state-space representation:

```
// Get information from sensors
phi = -Counter_Get(3)/90000.0*(2.0*PI);
th  = -Counter_Get(1)/18000.0*(2.0*PI);
dphi = (phi - pphi)/dt;
dth  = (th - pth)/dt;
pphi = phi;
pth  = th;
xh   = [phi, th, dphi, dth]';
// SDC representation
M = [[m1*r1^2 + J1 + m2*l1^2
      + (m2*r2^2 + J2)*sin(th)^2,
      -m2*l1*r2*cos(th) ]
     [-m2*l1*r2*cos(th), m2*r2^2 + J2]];
H = [[(m2*r2^2 + J2)*sin(2*th)*dth + c1,
       m2*l1*r2*sin(th)*dth]
     [-(m2*r2^2 + J2)*sin(th)*cos(th)*dphi,
       c2                    ]];
G = [[0,                0 ]
     [0, -m2*g*r2*sin/th ]];
A = [[Z(2,2), I(2,2)]
     [-M~*G,  -M~*H ]];
B = [[Z(2,1)][M~*[[1][0]]]];

// Optimal control
Q = diag(10.0, 5.0, 10000.0, 100.0);
R = [1.0];
{Ad,Bd}=c2d(A,B,dt);       // discretization
{Fd,Pd} = dlqr(Ad,Bd,Q,R);// feedback gain
u = [-Fd*xh];              // input
// Output input signal
if(u(1)/k>=6.0){
  V1 = 6.0;
} else if(u(1)/k<=-6.0){
  V1 = -6.0;
} else{
  V1 = u(1)/k;
}
DA_Out(Integer( V1*3276.8+32768 ));
```

Fig. 10. A part of on-line task codes for swinging-up a Furuta pendulum by the SDRE control.

$$
\begin{aligned}
\dot{x} &= A(x)x + B(x)u \\
x &= \begin{bmatrix} q^T & \dot{q}^T \end{bmatrix}^T \\
A(x) &= \begin{bmatrix} 0, & I \\ -M(q)^{-1}G(q), & -M(q)^{-1}C(q,\dot{q}) \end{bmatrix} \\
B(x) &= \begin{bmatrix} 0 \\ M(q)^{-1}\begin{bmatrix} 1 \\ 0 \end{bmatrix} \end{bmatrix}.
\end{aligned}
\tag{3}
$$

A SDRE controller can be obtained by discretization of (2) and updating the feedback gain every sampling period. At k-th sampling period, we freeze the state x as x_k. In that moment, the SDC representation is given by $\dot{x} = A(x_k)x + B(x_k)u$. Suppose the input takes a constant value, u_k, during a sampling interval. We approximately discretize the frozen SDC like $x_{k+1} = \Phi(x_k)x_k + \Gamma(x_k)u_k$. Finally the SDRE controller can be obtained by $u_k = -F(x_k)x_k$ solving the corresponding matrix discrete-time Riccati equation with weight state-dependent matrices $Q(x_k) \geq 0$ and $R(x_k) > 0$. This sequential procedure can be described

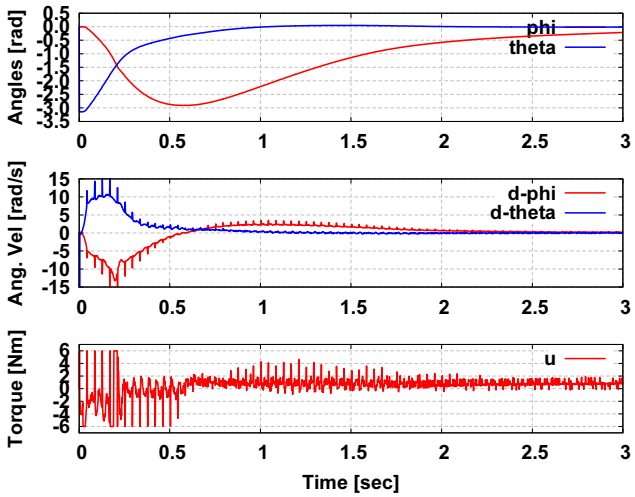

Fig. 11. Experimental results of swinging up a Furuta pendulum with the SDRE control.

in the on-line task of RTMATX, `on_task()` as shown in Fig.10. We think it has been described simply, easily and straightforwardly. The experimental results are shown in Fig.11. The figures illustrates the SDRE control with RTMATX works very well.

5. SOME EDUCATIONAL ASPECTS

The proposed configuration requires some knowledge about lower layers such as I/O structures of devices, and communication with device drives. The knowledge is important to make embedded systems consisting of hardwares plugged to microcomputers. Exclusive commercial real-time environments tend to throw those factors into a black box, but we think it is not good for education. From that aspect, the proposed system may give users much good chance to learn those structures and communications.

The configuration based on MATX has also several educational advantages. The first one is that the configuration provides a seamless environment for analysis, synthesis, simulation through real-time implementation. The second point is that MATX is compatible with C-langrage, and then enhances users to easily extend functions of MATX. User defined wrapping functions for connection of device drivers for Windows OS to MATX is one of those extentions. Hence we can think this configuration is well suited for application. Actually, in our laboratory, the configuration has been applied to a pendulum on cart system, a multiple pendulum system, a linear motor system, a heating system, a fuel-cell system, a segway-type robot, a snake-like robot, a SCARA-type manipulator, a unicycle simulator, etc. These facts illustrated that the configuration is worthwhile to construct it for educational aspect, and supports to set up any experimental systems for laboratories and undergraduate schools by students themselves.

6. CONCLUSION

This paper described the configuration of experimental systems based on MATX and demonstrated its effectiveness and advantages. MATX provides the real-time environment called RTMATX. In the configuration based on RTMATX, any devices can be utilized as long as the corresponding device drivers for Windows OS use are provided. However, user defined wrapping functions are required to plug MATX to the device drivers. We noticed the several notes when the user defined wrapping functions were developed.

The real-time performance with RTMATX was investigated from the jitter variation point of view. As the result, we found that the time resolution was 1.0 [msec] and the accuracy was also 1.0 [msec]. The frequency of abnormal sampling period cases seemed only up to 5.0%. Of course if any heavy load programs were running, different results might show up.

Swinging-up a Furuta pendulum by a SDRE control was demonstrated as a sample case of the proposed environment. In that example, the matrix SDRE should be solved every sampling period, however, this was realized easily thanks to the good description of MATX. The experimental results illustrated the effectiveness of the proposed environment.

ACKNOWLEDGEMENTS

The authors would like to express our thanks to Prof. Masanobu Koga, who developed and is maintaining MATX, for his great contribution to CAD for control system designs and real-time implementation environment with MATX.

REFERENCES

P. Koopman et al. Undergraduate embedded system education at Carnegie Mellon. *ACM Tras. on Embedded Computing System (TECS)*, Vol. 4, Issue 3, pp. 500–528, 2005.

M. koga and K. Furuta. A high-performance programming language (interpreter and complier) for science and engineering computations. *Proc. IEEE Int. Symp. on Comp.-Aided Contr. Sys. Design*, Napa, pp. 15-22, 1992.

M. Koga. MATX/RtMATX: A Freeware for Integrated CACSD. *Proc. of CACSD'99*, Kohala Coast-Island, Hawai'i, U.S.A., pp. 451–456, 1999.

T. Hoshino and K. Furuta. Modeling and Simulation of Mechanical Systems – Combination of a Symbolic Computation Tool and MaTX –. *Proc. of CACSD'99*, Kohala Coast-Island, Hawai'i, U.S.A., pp. 462–467, 1999.

S. Hayakeyama and Y. Pan. MATX Aided Control Education. *Proc. of CACSD'99*, Kohala Coast-Island, Hawai'i, U.S.A., pp.480–485, 1999

K. J. Åström, M. Iwase, K. Furuta and J. Åkesson. Safe Manual Control of Pendulums - A Human Adaptive Mechatronics Perspective -. *International Journal of Assistive Robotics and Mechatronics*, Vol. 7, No. 1, pp. 3–11, 2006.

T. Çimen. State-Dependent Riccati Equation (SDRE) Control: A Survey. *Proc. of the 17th IFAC world congress*, Seoul, Korea, July, pp. 3761–3775, 2008.