More Robust Firmware Development of Mechatronic System at Early Stage using Virtual Environment

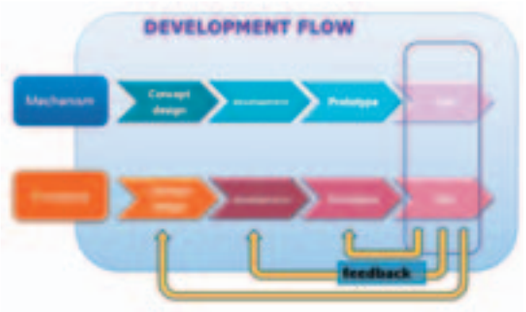
J. H. Choi

Professor, Mechanical Engineering, Department
Kyunghee University (Korea)
E-Mail: jhchoi@khu.ac.kr

1995 Ph.D. at Mechanical Engineering, University of Illinois, Chicago, IL, USA
1997 Senior Researcher, 1st R&D Division, Agency for Defense Development, Korea
1999 (~Present) Professor, Department of Mechanical Engineering, KyungHee University, Korea
2000 (~Present) Chairman of FunctionBay Inc., Korea

■ABSTRACT

In firmware development of mechatronic systems, the computer simulation and virtual environment have become important to reduce the development cost and the lead time. Since most mechanical systems are managed by microprocessors and they are required to do more and more complicated tasks, the virtual prototype using computer simulation techniques has already been replacing the real prototype rapidly. However, in practical development for robust firmware, in practice, there are still difficulties in the development of robust firmware using virtual prototypes after the initial structure of the software development because the mechatronic systems are sophisticated inter-disciplinary systems which include several different fields of physics, such as mechanical, electrical, and electronic.
In this investigation, a more efficient collaboration concept of the simulation environment for stable firmware development at an early stage is proposed, and it is shown using several examples to be possible. Thus, through these examples, the needs of the virtual environment for the robust firmware development of mechatronic system and its advantages are introduced.



[Figure 1]
contemporary Development Flow

1. Introduction

Recently, robust firmware is getting more and more important. Most mechanical products include microprocessor and have the firmware to operate the system. These products are frequently called ‘Mechatronic products’ . Products such as automobiles, robots, printers, cameras, even washing machines can now be called mechatronics products which are controlled by firmware, because all of them are controlled by microprocessors and firmware rather than just by mechanical control systems.
For mechatronic products, usually the quality of the firmware is a key element because control is often deeply interrelated with the quality. Furthermore, even a small bug in the firmware can cause a severe accident.

Furthermore, nowadays mechatronic products are getting more and more complex. So firmware is also getting more and more complex and larger in scale. On the other hand, it is usually desired that the development costs and the development lead time are reduced. [6], [8]

In mechatronic products, the firmware is usually used to control the mechanical system of the product. But firmware development is a different discipline from the development of mechanical systems. So it is very difficult to perform integrated testing both of them together during the early stage of the development. In order to test and validate the firmware, the mechanical system should be ready in advance and the firmware must be compiled and downloaded to the mechanical system. This means that the firmware must be able to be compiled and executed to control the mechanical system to test if it can control the system as intended at an early stage of the development.

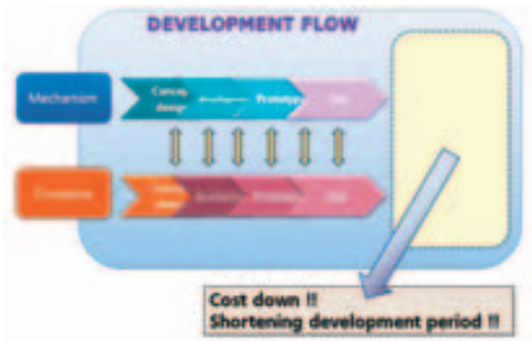
On the other hand, from the viewpoint of a mechanical system engineer, the mechanical system which works as the firmware controls must be developed before the integration test. Figure 1 shows the contemporary development process in mechatronic industries. Eventually, the integration test between the firmware and the mechanical

system usually is done at the late stage of the development. Since the integration test cannot be started until a late stage of the development, even if any bug or any conflict of the specification or interface is discovered, it is not easy to change the specification or fix the bug.

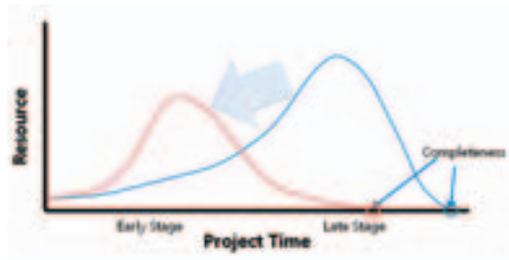
Bugs in the firmware can cause lots of troubles, and wastes time and money. So it is essential to find and fix the bugs at as early a stage of development as possible by thorough validation and debugging. In addition, under the current process frequently used by most companies, conflicts in the specifications are often not found until the late stage, and when this happens, it is very difficult to fix the problem. Either a quick but inadequate firmware solution must be made or a very expensive re-design must be made.

Therefore, an innovative environment in which to perform the integration test from the early stage is needed. The virtual environment using the virtual mechanical systems which do not require the use of real hardware can be an efficient solution. [1], [2]

There are many advantages to using a virtual mechanical system. Firstly, the virtual mechanical system can be built earlier than the real hardware and costs cheaper than the real hardware to develop. And it is much easier to change the model. Secondly, the virtual mechanical system is very useful for reproducing problems under the same circumstance and it is possible to generate very dangerous situations or situations which are very difficult to cause or replicate with the real hardware. Thirdly, with real hardware, it is difficult to see the behavior of parts inside the machine during testing. On the other hand, it is very easy to investigate the behavior of these internal parts in the virtual system. The proposed firmware development flow is shown in Figure 2. Using this proposed development flow, the firmware and the mechanical system can be tested together even in very early stages of the development. There can be continual communication and interaction throughout the development process.



[Figure 2]
Development Flow using Virtual Environment



[Figure 3]
Improvement of the resource distribution

Figure 3 shows the required resource diagram or completeness of software during project time. Usually software engineers are extremely busy at the end of development which can be a cause for the introduction of bugs very easily. Moreover it is very difficult to modify the product at the end of development since it is not easy to figure out the main problem and just asking other team to fix it

Using the proposed development flow in Figure 2, the resource usage curve can be shifted to the left and overall resource usage will decrease. And it can be much comfortable and efficient to obtain robust software of the system.

In addition, the environments where the product will be used can be very diverse. Lots of the uncertainties such as the temperature, wear or tear of the machine, or foreign substances can change the operational condition of the product. In order to develop a robust product, lots of tests are required to make the system operate optimally under these uncertainties. But it can be almost impossible to do with real hardware at an early development stage. This can be achieved by using the virtual environment and it can save time and cost significantly.

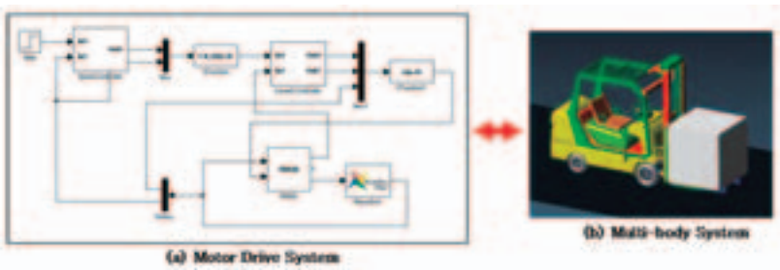
2. Integrated Virtual Environment For Mechatronic System

The virtual environment for the mechatronics area has been developing for long time. Several years ago, similar methods which used co-simulation interface techniques between RecurDyn™ (Plant), Simulink® (Controller), and Simplorer® (Circuit) were introduced. With this interface, it was possible to test and validate the control algorithm developed by Control & Circuit simulation programs with the virtual plant modeled in a multibody dynamic tool. These techniques were also used to develop the control algorithm for a tank [1] and a LIM (Linear induction Motor) drive system for a rail vehicle [2].

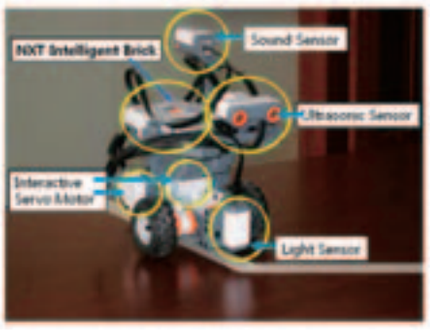
In 2008, Ch™ [3], the script engine which completely implements the C language was adopted as the script engine of CoLink (CoLink is the control modeling tool developed by FunctionBay Inc.,), and the virtual environment using RecurDyn™, CoLink™, and Ch™ script engine was proposed [4]. In this paper, a realistic nonlinear dynamic model with an electrical motor and an electronic control system was successfully simulated using this integrated virtual environment. The electric forklift vehicle model was modeled in RecurDyn™, a PMSM (Permanent Magnet Synchronous Motor) was modeled in CoLink, and the controller was modeled with C code embedded in a Ch block in CoLink. Figure 4 shows the schematic diagram for electric forklift integrated system simulation.

The integrated system simulation methods and the virtual environment were validated in this study. In this study, all of the elements, including block modeling, script modeling, and CAD modeling, were all performed in one integrated virtual environment. Furthermore, the firmware was almost completely developed using only this virtual development environment.

Since 2009, a new co-simulation interface between ZIPC™ and RecurDyn™ has been developed. The purpose of this interface is to enable testing and validation from an early stage using the STM (state transition matrix) model provided by ZIPC™ with a RecurDyn™ model. In particular, since ZIPC™ can simulate the STM model even before C source code is developed, this new environment has the advantage that it is possible to perform the integration test between the firmware and the mechanical system from the early stage even without the source code. Figure 5 shows the concept of the hybrid car model using the virtual environment. In this model, the vehicle consists of rigid bodies, flexible bodies, and various kinematic constraints. Also, motors and circuits are modeled in software such as CoLink, Simulink®, and Simplorer®. The control firmware developed by ZIPC™ can be included in this environment as well. The



[Figure 4]
Integrated model of Forklift system



[Figure 6]
Real Mindstorms model

[Figure 5]
Hybrid car model using the virtual environment

simulation results show significant advantage using such a virtual environment to develop more complete firmware at the early development stage.

3. Validation by Lego MindStorms® examples

Mindstorms® is the programmable robotics kit consisting of Lego blocks, motors, sensors, and other parts as well. Mindstorms® was used to validate the usefulness of the proposed virtual environment using ZIPC™ and RecurDyn™.
Figure 6 shows the Mindstorms® robot which can follow a line and the firmware which processes data from the sensor and controls the motors [5]. This robot was modeled in RecurDyn™, and the firmware to process the sensor events was developed using ZIPC™.

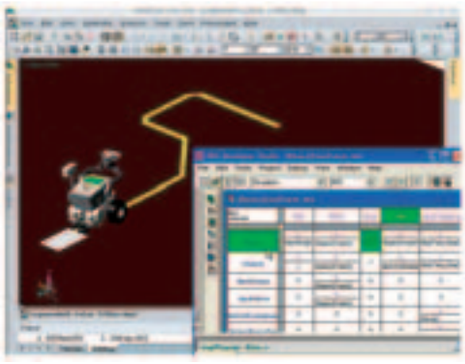
For the sensor modeling described in Table 1, the light sensor was modeled in the C language mathematically and embedded in CoLink through the Ch™ script engine. The sound sensor and the ultrasonic sensor events were not modeled in the virtual plant (RecurDyn™). For these 2 sensors, the sensor events were input into the ZIPC™ interface during the simulation manually and the action commands were sent to RecurDyn™ [6].

The VIP function of ZIPC™ was used for the co-simulation between RecurDyn™ and ZIPC™. And the firmware downloaded to the robot was generated using the ZIPC™ generator from the ZIPC™ model co-simulated with the RecurDyn™ model as shown in Figure 7. Figure 8 shows the Simulation using an STM in ZIPC™

In this model, the line tracer algorithm embedded in this robot was intentionally written to contain a bug. The bug produced caused erratic motion in the robot. Remarkably similar behavior with the real robot caused by the intentional bug was captured in the virtual model as well. From this, it can be inferred that this virtual environment is useful for finding bugs in firmware code at the early development stage without the use of real hardware.

Degrees of the Freedom	10
# of bodies	6 (rigid)
# of joints	4 (revolute)
# of sensors	3 (sound, light, ultrasonic)
# of actuators	3 (servo motors)

[Table 1]
model specification of line tracer robot



[Figure 7]
Co-simulation of ZIPC™ and RecurDyn™

[Figure 8]
Simulation using STM in ZIPC™

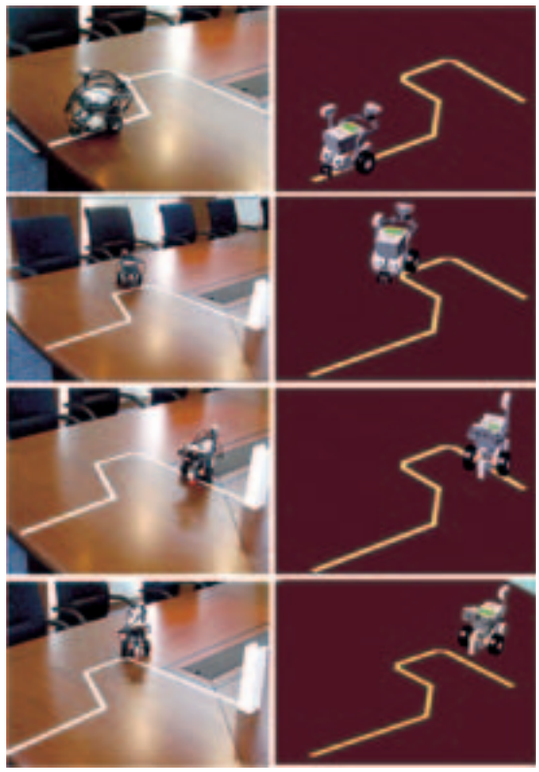
	Firmware	Plant
Real System	C source code	Real Mindstorms®
Virtual System	ZIPC™ Model	RecurDyn™ Model

[Table 2]
comparison between Real system and Virtual system

The figure 9 shows a comparison between the test results of the real robot and the simulation results of the virtual model.

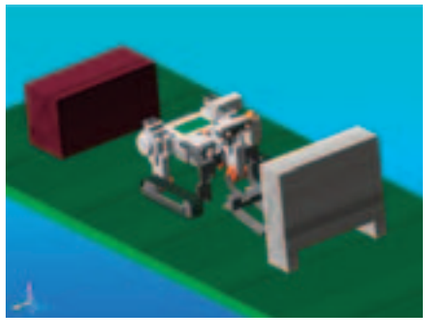
Another Mindstorms® example is a simple forklift model that has tracks that have a huge number of degrees of freedom. This vehicle is a very complicated nonlinear system [7]. The specifications of the system is illustrated in Table 3. Even with this kind of complicated virtual mechanical model, the co-simulation results showed the extremely similar behavior with the real model. Figures 10 and 11 show the virtual and real motion of the system.

This example has many more degrees of freedom than the first example because this model has lots of track bodies. For the sound sensor and the touch sensor, the triggering events were modeled with a signal chart. And the light sensor and the ultrasonic sensor were modeled with function expressions to calculate the distance between the position of the robot and the obstacle (box). Like the real robot model, these distances were sent to ZIPC™ to be processed.



[Figure 9]
Real Mindstorms vs. Virtual Mindstorms

[Figure 10]
Real Mindstorms model



[Figure 11]
RecurDyn™ model of the lift robot

Degrees of the Freedom	541
# of bodies	96 (rigid, track)
# of joints	7 (revolute)
# of sensors	4 (sound, light, ultrasonic, touch)
# of actuators	3 (servo motors)

[Table 3]
model specification of the lift robot

It is obvious even at the earliest stages of development of the forklift robot that its fork will have to rise, hold its position, and then drop in order for the robot to lift, transport, and put down a box. It is possible to represent this motion of the fork with a very simple linear model, as can be seen by the dotted line in Figure 12. This model does not contain any uncertainties, such as the structural flexibility of the robot, which can cause many dynamics effects such as vibration of the lifting fork. Designing firmware based on such simplified models is inherently error prone because these uncertainties can cause the firmware's algorithm to be unable to control the real device. Through a complete model which includes such properties as structural flexibility in a virtual model, many of these uncertainties can be captured even at an early stage of firmware development. The solid line (vibrating line) in Figure 12 shows

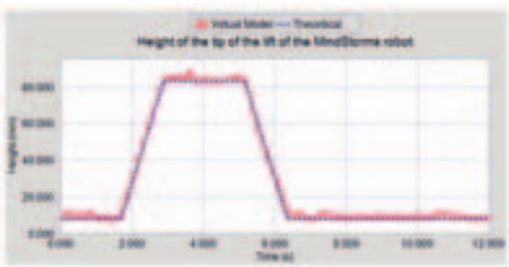
this kind of effect – the vertical vibration of the tip of the lifting fork of the virtual model over time.
Uncertainties like this are always inherent in real mechanisms, and they can play a very important part in the development of the device. Instead of using the simplified model, if a virtual model of the device is used in the early development, the initial development of the firmware can already incorporate many of these uncertainties.

4. Summary and Future Development

The virtual environment has advantages for use in firmware development. With the virtual environment, it is possible to test and validate the firmware from the early stage of the development. And it can save time and cost dramatically.

To use the virtual environment, it is required to use several kinds of the software. But it is true that it is not easy to skillfully as integrate several software packages simultaneously. So the next task is to develop the easy-to-use environment for both the firmware designer and the mechanical engineer. Because not all users are familiar with both mechanical engineering and firmware design, the user interface must be very easy to understand and intuitive for each of these users [8].

In addition, there are formulation and software design topics in the development of the virtual development environment that require further investigation. For example, the firmware operates at a time scale of microseconds to milliseconds, but in general mechanical system simulation software operates at a time scale of milliseconds to seconds. So, it is important to find a way how to synchronize these different time scales.



[Figure 12]
Height of the tip of the lift

REFERENCES

[1] Kunsoo HUH, Jinhwan CHOI and Honghee YOO, 2003, “Development of a Multi-body Dynamics Simulation Tool for Tracked Vehicles” (Part II, Application to Track Tension Controller Design), JSME vol. 46, no2, pp. 550-556

[2] Jaehym Lee, Hyungsoo Mok, Changwan Kim, 2008, “A Modeling of Linear Induction Motor Based Railway Propulsion System for α-Simulation of Electric-Mechanical System” , KSAE 2008, pp. 598

[3] http://www.softintegration.com/

[4] D.J. Yun, Hyungsoo Mok, K. H. Cho, and J. H. Choi, 2008, “Dynamic simulations for Electric Forklift System driven by PMSM drive Using RecurDyn CoLink” , 4th Asian Conference on Multibody Dynamics 2008, pp. 7-11.

[5] Taero Cha, S. T. Kim, D. J. Yun, 2010, “Virtual Firmware development system for Lego® Mindstorms® vehicle using RecurDyn® & ZIPC® at early design stage” , 2010 KSAE proceeding, KSAE10-B0172

[6] IEEE/MESA Conference “Firmware Development of Mechatronic System in Virtual Environment using Ch” 2010 Quintao China.

[7] S. T. Kim, D. J. Yun, K. H. Cho, J. H. Choi, 2010, “The inter-disciplinary simulation environment including the firmware and the mechanical system” , ACMD2010

[8] OTIS Central R&D Center, Technical meeting & Discussion “Real Digital Prototyping for elevator system” 2010