ACTUATORS AS MECHATRONICS OBJECTS FOR MOTION SOFTWARE CONTROL

Aurel FRATU
Transilvania University of Brasov, Romania

Abstract. This paper presents the integration between the components hardware and the driven functions software, resulting in integrated systems called intelligent mechatronic systems. Their development involves finding an optimal balance between the basic mechanical structure, sensor and actuator implementation, automatics information processing and control. A major importance is the simultaneous design of mechanics and electronics, hardware and software and embedded control functions resulting in an integrated system.

Keywords: actuators, mechatronic objects, components integration, software control

1. Introduction

Mechatronics is the synergistic integration of physical systems – sensors, actuators, electronics and control systems, computers systems, from beginning to end of the design process; thus enabling complex decision making. Basis of this integration process the design problems have been transferred from the mechanical domain to the electro and computer software domains.

Mechatronic system design deals with the integrated design of a physical system, including sensors, actuators, and embedded digital control system. The integration is applied at both hardware components and information, both on-line and off-line.

2. Actuators for motion control systems

Actuators are most often found in motion control systems. In these systems, the ultimate objective is to drive the mechatronic system along some reference trajectory. The role of the actuator in such system is to establish the flow of power by means of some control actions, in response to process models or sensory data, so that the desired trajectory is effectively accomplished.

One feasible way of changing the mechanical state of a mechatronic system is through an effective exchange of energy with its surroundings. The exchange of energy can be accomplished by active interaction with other systems. An actuator is a device that modifies the mechanical state of a mechatronic system with which it is coupled.

An actuator can be seen as a system that establishes a flow of energy between an electrical port and a mechanical port [1]. The actuator converts some sort of input power into mechanical power. The power exchanged between the input ports and output ports will be completely defined by two conjugate variables: an effort (force, torque, voltage) and a flow (velocity, angular rate, current).

The use of electrical energy at the input port of actuators has a lot of advantages:

1. *Compatible energy domains*. Most motion control systems incorporate actuators which are controlled electronically. In consequence, the output energy nature of the control part is the same with the energy nature existing at the input actuator port.
2. *Fast operation of electric devices*. Electronic and electric devices are characterized by fast processes. Electrical processes are much faster than the mechanical processes. This characteristic improves the controllability of actuators.
3. *Availability of components*. The electronic components used in the control systems are well-known and easy obtainable.

In view of the above considerations, the actuator concept that we will use throughout this paper comprises both the electrical sub-system and the mechanical sub-system.

3. The role of the actuator in an intelligent control system

Motion control systems can be regarded as the paradigm in the application of actuators [2]. The main objective of a motion control system is to drive the mechatronic system into work space.

Motion control systems drive the mechatronic system according to the reference trajectory by means of a combination of functions: sensing,

processing and actuation. Some of these functions may not be present in a particular motion control system. The role of the different devices, in an intelligent control system, is discussed in the following paragraphs [3].

3.1. Sensing function

Sensors are devices that monitor the status of a parameter of the mechatronic system. In a general motion control scheme, the reference trajectory must be compared to the actual one. Reactive measures to counteract deviations can then be implemented on the basis of this comparison.

The use of sensors in feedback control motion systems provide means for improving the robustness of the whole process. The sensors enable the implementation of disturbance rejection strategies. Feed-forward control schemes (sensor-less) are susceptible of being affected both by imprecision and by external and internal disturbances. On the contrary, feedback schemes are much more robust against external and internal disturbances.

3.2. Processing function

The processing function in a motion control system is done by the controller. The controller provides the equivalent to the intelligence in a control system. It usually receives the reference trajectory as an input and computes the required action to drive the mechatronic system according to the reference trajectory.

The controller in feedback schemes obtains information on the status of the mechatronic system through sensors. On the basis of this information, the deviation from the reference trajectory is calculated and corrective actions implemented.

3.3. Actuation function

The actuator is the only obligatory component of a motion control system. The actuator establishes a flow of energy between the electrical and the mechanical domains.

The function of the actuator is to impose a state on the mechatronic system, ideally without being affected by the load. The mechatronic system is driven according to the reference trajectory, by either increasing or decreasing her energy level.

4. Universal mechatronics objects

Mainly, one address mechatronic system as a whole composition of subsystems which correspond to the technical disciplines involved: mechanical, electrical, control and software engineering. Frequently, to design a mechatronic system is used a team, where each member of the team is skilled in a relevant discipline.

Given a functional specification for the system, and an understanding of the issues involved, to compose the system from subsystems can be reasonably achieved by the appropriate expert. The synergy between subsystems of the system is built into the manner in which the system is composed from individual subsystems.

Once a set of subsystems has been configured, the fundamental problem remains of the specifying the interfaces between subsystems. In research and development projects, these integration and interfacing issues become more prevalent and new tools and methodologies will be needed to deal with them.

In this paragraph, we present Universal Mechatronic Objects (UMOs) tool for interfacing of the subsystems [4]. These UMOs can be applied to any mechatronic system to form the interface between the electrical and the computer control subsystems. By this means an effective software control of the mechatronics systems can be built.

4.1. The electrical subsystem/control subsystem interface

They are many researches focused upon the interface between the electrical subsystem and the control subsystem. The figure 1 illustrates the various hardware and software layers involved. The first software layer above the hardware consists of device drivers which interact directly with the computing hardware.

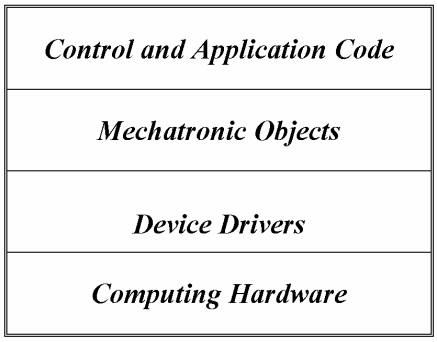


Figure 1. Layers of software for a mechatronic system

The device drivers are considered part of the electrical subsystem because their development requires intimate knowledge of how the actuators and sensors are interfaced to the computing hardware. The control developer will need to read the sensors information and write the command to the actuators.

As can be seen in figure 1, mechatronics objects are shown to be the interface between the device drivers (i.e., the electrical subsystem) and the control software (i.e., the control subsystem). Given this arrangement, we analyse how we design the mechatronics objects to encapsulate the access functions for actuators and sensors so that the device drivers are hidden from view of the control developer.

Further, the application-specific details of the control algorithms and application software are hidden from view of the electrical developer.

The assembly of the mechatronics objects can be accomplished by the team member responsible for system integration, before either the electrical or control developers begin their work. We propose that by defining the boundary between the electrical and control subsystems by the using of a standard set of UMOs, the entire project will progress more quickly, efficiently and reliably toward the desired functional system realization.

4.2. Object oriented programming

An object oriented paradigm is appropriate for the definition of the electrical/control subsystem boundary. Object oriented programming, in contrast with the traditional structured programming methodologies is modular in a way which allows the assembly of all variables and functions relating to a concept. For example, an actuator (sensor) has associated with it certain variables and functions which all must be present for the programmer to use it effectively, but which are not necessary if the actuator (sensor) is not used.

The required variables, called member variables, might include the actuator's name, the last value commanded to the actuator, and a variable whose value indicates whether the actuator is synchronous or asynchronous.

The required functions, called member functions, might include an initialize function that sets-up the actuator for use, a write function, and a finalize function that is called when the program is about to exit. In such a case, it is useful to encapsulate these variables and functions into a single module, called an object. An object that is derived from a particular class is called an instance of the class.

Object Oriented Programming (OOP) is a topic of much programming books and progress in the last years. The advantages of OOP, over more traditional structured programming methods are by now well documented.

Object oriented design can be implemented using several different programming languages. Borland DELPHI is the language of choice for most of the literature that we have reviewed, because of its many built-in features which facilitate OOP [5].

By application of the object oriented paradigm the significant characteristics of actuators and sensors can be mapped directly to member variables and member functions of object classes in OOP.

In this paper, DELPHI language has been designed to facilitate OOP with the built-in type class. One advantage of the use of DELPHI and OOP is the notion of inheritance. We can define a class which encapsulates all of the member variables and functions required for a wide spectrum of actuators. The class is entirely independent of any particular actuator. Then we can derive an instance of the class for each particular actuator in an intelligent system. Each of these actuator objects will inherit the member variables and functions of the parent class with no additional coding.

The modularity enforced through the application of OOP allows software to be constructed so that improvements can be implemented through local modifications only. Thus for example, if changes are made to the device driver code, no changes will be necessary in the control code and vice versa.

Moreover, because the programming objects relate directly with the actual system components (i.e., actuators and sensors), no new abstract concepts need to be mastered. Further benefits will add if UMOs are applied consistently across multiple projects incorporating a wide variety of hardware, because control software from one project can be voluntarily reused on another project.

For the realization of industrial control systems, one can promote the application of the object-oriented paradigm to all phases of control system development, including functional specification, design and implementation. One can elucidate the application of object oriented formalism in translating verbal system specifications into object definitions at a high level of abstraction. This approach addresses universal actuators and sensors.

Another approach applies the object oriented paradigm to the development phases of a real-time industrial automation application. His assignment of physical equipment to software objects encompasses the lowest levels including individual actuators and sensors [6].

4.3. Case study - a robotic arm driving an object in a virtual environment

In a motion control system, the actuation function is accomplished by the actuators. As usual applications for these mechatronics systems are give in figure 2, as example a gripper system and a robot joint, which incorporate both: electrical and mechanical elements.

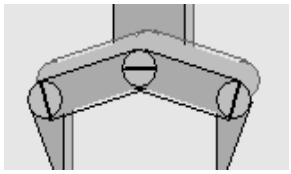
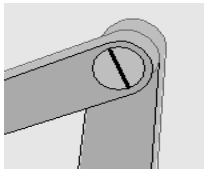

a)

b)

Figure 2. Actuators system for a gripper system (a) and a robot joint (b)

A software control system, with actuators as mechatronics objects, can be illustrated in a robotic application. In this application a robotic arm, equipped with actuators, drives a virtual object.

The robotic arm, presented in figure 3, is equipped with the owned sensors system.

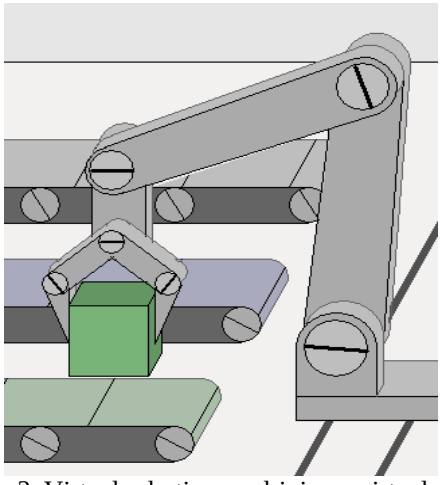


Figure 3. Virtual robotic arm driving a virtual object

The robot's actuators can impose an effort (force) on the manipulated object. However, the velocity of the robotic arm will be determined by the inertial characteristics of the mass. Here, the mass is acting as admittance; it receives an effort and determines the velocity. According to the principle of causality, the action of the motion control system on the robotic arm, viewed as a whole, must accept a flow (velocity) and imposes an effort (force or torque) [7].

5. Conclusions

Actuators are memory-mapped inside the computer and thus accessed by reading and / or writing to memory locations. For this description, there is a great arrangement of commonality at this level of abstraction.

The OOP serve as a common basis for the controller design. The OOP based on UMOs concept has to be taken into account within the synthesis task of the motion control systems.

In most of the control applications only some state variables are measurable and seldom are some signals corrupted by noise. In these situations the actuators have a limited accuracy, and some parameters are only known inaccurately or are even varying slowly. For this reason the OOP serve as informatics tool to adjustment the mechatronics systems accuracy.

References

1. Zupan, M., Ashby, M., Fleck, N. (2002) *Actuator classification and selection - the development of a database*. Journal of Advanced Engineering Materials, Vol. 4, No. 12, 2002 (July 2007), p. 933-940, ISSN 0309-7420
2. Fujita, K., Akagi, S. (1995) *A Framework for Component Layout and Geometry Design of Mechanical Systems: Configuration Network and its Viewing Control*. Proceedings of the Design Engineering Technical Conferences, p. 515-522, ISBN 9780791849040, 1995, ASME Publishing House, Boston, USA
3. Rzevski, G. (2003) *On Conceptual Design of Intelligent Mechatronic Systems*. Journal of Mechatronics, Vol. 13, No. 10 (December 2003), p. 1029-1044, ISSN 0957-4158
4. Patrick, F., Muir, M., Horner, J. (1998) *Mechatronic objects for real-time control software development.* Proceedings of the 1998 SPIE International Symposium on Intelligent Systems and Advanced Manufacturing: Mechatronics Conference, Editor, p. 251-265, ISBN 9780819429797, November 1998, Boston, USA
5. Auslander, D.M., Ridgely, J.R., Ringgenberg, J.D. (2002) *Control Software for Mechanical Systems: Object-Oriented Design in a Real-Time World*. Prentice Hall Publishing House, ISBN-13 978-0-13-786302-0, Boston, USA
6. Fratu, A., Fratu, M. (2008) *Programarea vizuală în mediul Delphi cu aplicații în robotică* (*Visual programming in Delphi environment with applications in robotics*). Transilvania University Publishing House, ISBN 978-973-598-315- 4, Brașov, Romania
7. Stacey, M.K. et al.: *Intelligent Support for Conceptual Design: A Flow Modelling Approach.* Proceedings of International Conference on Engineering Design – ICED11, p. 261-266, 1997, Tampere, Finland