Experiences of a CPS Course on Embedded Control

Årzén, Karl-Erik

Published in:
[Host publication title missing]

2013

Experiences of a CPS course on Embedded Control

Karl-Erik Årzén
Department of Automatic Control
Lund University
Lund, Sweden
Email: karlerik@control.lth.se

***Abstract*: The paper presents a CPS-oriented course that integrates control theory and embedded computing. The course combines concurrent real-time programming and analysis with discrete-time sampled control theory and real-time networking. The contents of course, including the course projects are described in the paper as well as the experiences with the course.**

Keywords: cyber-physical systems, education, embedded systems, real-time systems, control

I. INTRODUCTION

Cyber-Physical Systems (CPS) are systems consisting of cyber components and physical components that are tightly integrated and networked at all scales and levels. The physical and cyber components interact through sensors and actuators allowing the cyber components to monitor and control the behavior of the physical components. The networking is a consequence of geographical and/or logical distribution and may be realized as physical networks, communication networks, and/or social networks. Hence, the combination of computing, control, and networking is essential for CPS.

Control and embedded systems are both disciplines in which tight interaction with the physical environment is necessary. In traditional control the computing system implementing the controller is viewed as a machine that is able to realize the abstraction of a discrete-time difference equation in an ideal way. The fact that computations take time, which in many cases is not constant, and the fact that the amount of computations that can be performed in parallel is limited by the amount of processors available is often disregarded. A similar situation holds for networked control where control loops are closed over a communication network which is often idealized as a constant delay, although for most network protocols this delay is far from being constant, and, in particular for wireless networks, can even be infinite due to lost packets. In a CPS-based approach to control the temporal non-determinism caused by the implementation platform and its effect on the control performance should be taken into account in the analysis and design.

The Department of Automatic Control has pioneered the teaching of real-time systems and control at Lund University, starting already in the mid-1970s. The current course – Real-Time Systems – has been given in its present form since 2007. The course contains many of the aspects that are currently the focus in CPS curriculum development, e.g., modeling of physical systems, control analysis and design, real-time networks, simulation, and system implementation. However, since the course was developed before the CPS boom reached Europe, the term CPS is not used explicitly in the course name.

The paper is organized as follows. In Section II the overall situation at Lund University is presented including a description of the background knowledge that the students have when they come to the course. The course contents are described in Section III. The course contains a project. Examples of recent projects are given in Section IV. Finally, in Section V, the experiences of the course are discussed.

II. STUDENT BACKGROUND

Lund University offers a five year integrated engineering education. However, during the last two years the students follow a specialization which is very similar to a Master's programme. The Real-Time Systems is an elective course that may be taken in year four or five. It is followed by students from several education programmes, although the majority comes from Computer Engineering, Electrical Engineering, Engineering Physics, and Mechanical Engineering. The course is part of several specializations including Signals, Systems and Control; Control and Automation, Embedded Systems, Software Systems, and Mechatronics. Since the course is elective the number of

students varies from year to year, with an average of around 80. Around 50% of these are Computer Engineering students. The course runs over a full semester and corresponds to 10 ECRTS credits (the normal course load during a semester is 30 credits).

At Lund University all the students that may sign up for this course have taken the same mandatory basic course in control. This course is based on continuous-time and cover both state-space and input-output (transfer function) modeling formalisms. However, the mathematics background is not the same among the students. The engineering physics students, who also have the best grades from high school, takes more and more advanced mathematics courses.

The fact that the computer engineering students have taken the same control course as the other students is very important. In addition to this the computer engineering students also have a mandatory course on concurrent real-time programming. This has, however, proved to be of less important with respect to how well these students perform in this course.

III. Course Contents

Real-time systems is an area that is of vital importance to all control engineers. All control systems are real-time systems. It is therefore essential for control engineers to have a thorough understanding of computers and real-time systems. It is also important for computer engineers to understand the control theory in order to implement `controllable' systems. Concepts that are important are, e.g., feedback, stability, delays and dynamics.

The aim of the course in Real-Time Systems is to study methods for design and implementation of computer control systems with the focus on embedded systems. The course consists of two main parts: Real-Time Programming and Computer-Controlled Systems. After the course the students should have sufficient knowledge to implement small embedded control systems on their own, and also have a thorough understanding of the system aspects of large control systems.

The course runs over one semester (14 weeks). The first half of the course consists of 17 90-minute lectures, 5 computer-based exercises, 6 problem-solving exercises and three 4-hour laboratory exercises. Seven of the lectures are focused on real-time programming and embedded systems covering topics such as concurrent programming, real-time kernels, inter-process communication mechanisms, priority inversion and inheritance, interrupts, time handling, fixed-point arithmetic, real-time scheduling theory, and real-time networks.

Six of the lectures are focused on sampled control theory including zero-order-hold sampling, aliasing, sampling period selection, the z-operator, pulse transfer functions, PID, state feedback, observer design, discretization of continuous-time controllers, controller implementation techniques, and discrete event-based control using Statecharts and Grafcet.

The remaining lectures focus on the interaction between control and scheduling. Here the Jitterbug toolbox that allows analysis of how delays caused by, e.g., network communication and task preemptions, effect the control performance, is used [1]. Also the TrueTime simulation toolbox, [1] is used to study how task scheduling effects control performance. TrueTime makes it possible to simulate multi-tasking real-time kernels and real-time networks embedded as S-functions within Simulink. Tasks, representing, e.g., controllers, are implemented as Matlab m-files or C-code, communicating with the continuous-time Simulink blocks modeling the physical system under control using simulated AD and DA converters.

During the computer exercises, the students learn concurrent and real-time programming using Java on a standard desktop Linux PC as the programming environment. The students implement small multi-threaded Java-based control systems including graphical user interfaces. The developed controllers are used to control virtual processes. The virtual processes are real-time simulation models that include an animated user interface. One example of a virtual process is a virtual ball and beam process that is controlled by cascade-connected PID controllers, see Fig. 1.

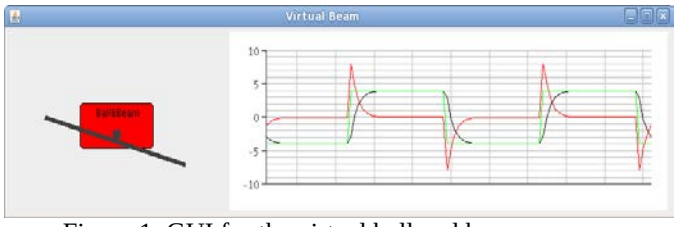

Figure 1: GUI for the virtual ball and beam process.

In the exercises the students can see how context switches and priority settings effect the timing of the controller task, and how that in turn effect control performance. The problem-solving exercises are more

traditional in nature, in the sense that the students solve control analysis and design problems and scheduling problems using pen and paper, possibly supported by Matlab.

During the first laboratory exercise the task is to take the previously developed Java-based control system for the virtual ball and beam process, and instead apply that to a real ball and beam process, see Fig. 2.

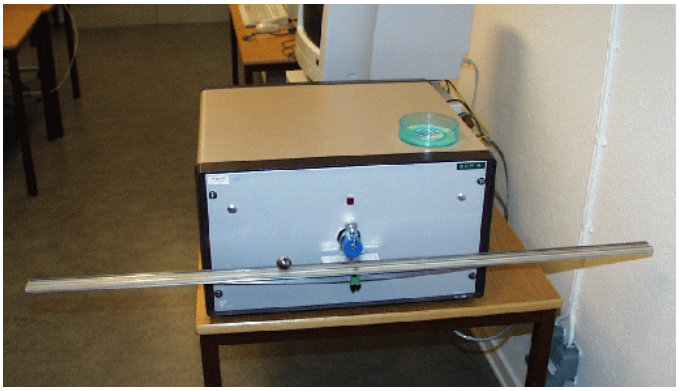

Figure 2. Ball and Bean Process

Depending on the background of the students they either continue to use standard Java or they use LJRT, a Java to C compilation framework [2]. This laboratory normally goes quite smoothly. The reason for the Java-focus is mainly that Java is the programming language that all the students coming to the course know beforehand. Using Java of course does not give any hard real-time guarantees. However, modern hardware is so fast that Java works very well in practice for all our laboratory processes. In the case hard real-time guarantees are important; the Java-to-C approach can provide this.

The topic of the second laboratory is sequential discrete-event control. JGrafchart, an in-house developed programming environment for graphical programming and execution of IEC 61131-3 Sequential Function Charts (Grafcet) is used. The task is to implement a sequential control program for sorting and sequencing differently colored beads in a physical bead-sorter process. In the third laboratory the topic is fixed-point implementation of a state-feedback controller for a mechanical servo process in C. The platform used is an ATMEL AVR Mega16 microcontroller with PWM-based signal output.

IV. Course Projects

During the second half of the course the main activity is a project. The projects are performed in teams of 2-4 students. Often the students in a group come from different education programmes, but it is not required. The size of the projects corresponds to approximately 2.5 full weeks of work. Normally the students have 30-40 different projects to choose among. In most projects the task is to implement a particular controller for one of the laboratory processes available, e.g., a Furuta-type inverted pendulum, a linear inverted pendulum, a helicopter dynamics process, a quadcopter, different versions of ball and beam processes, flexible servos, tank processes etc. The type of controller is tailored to which other control courses the students have taken or currently are taking in parallel with this course. Some examples are PID control, state feedback control, adaptive control, and model-predictive control (MPC). Most of the projects also involve some type of modeling and identification. The implementation platform is either Java on a standard desktop PC, C on a microcontroller such as ATMEL AVR or Raspberry Pie; or Lego Mindstorm NXT programmed either in Java or in a subset of C. In the Lego case the actual design and construction of the process to be controlled is also a part of the project. In many cases these projects are focused on various types of mini-Segways, e.g. the double-pendulum Segway shown in Fig. 3. In case the students choose ATMEL AVR as the implementation platform the controller is implemented using fixed point arithmetic and the built-in support for timers, counters, and interrupts. Serial communication is used between the AVR and a PC, where the GUI is implemented. In many cases the projects involve networked control loops, closed over Bluetooth, Wifi, or 4G/LTE. Sometimes the controller is implemented in a smartphone, utilizing the screen for the GUI and the build-in IMU sensors to implement gesture-based operator interaction.

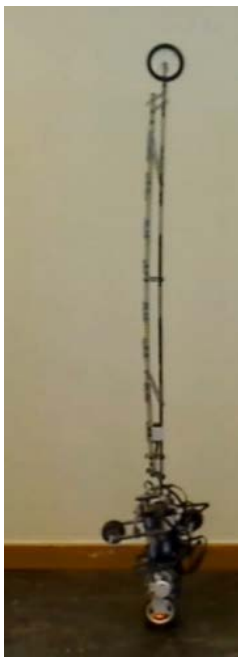

Figure 3: Double-pendulum mini-Segway

In some cases also pure real-time programming projects are allowed, not including any control part. Some examples of projects of this type that have been performed are implementation of a kernel supporting EDF (Earliest Deadline First) scheduling on an AVR

Mega8 microcontroller, and implementation of the priority ceiling protocol.

The course is examined through a 5 hour open-book exam. The exam is a mixture of control-oriented problems, real-time oriented problems, and problems that combine real-time and control issues. The course is based on two text books: an in-house text on Real-Time Computing and a slightly expanded educational version of [3].

V. EXPERIENCES

The course is quite popular among the students, to a large extent due to the fact that the course includes a practical project where the students are allowed to design and implement a controller for a real physical plant. The course fulfills most of the characteristics to be part of a CPS curriculum. For example, it combines control theory with embedded computing and real-time networks. It also looks upon the interference between task scheduling and control performance and gives the students experience of modeling physical systems.

The experience from the course shows that it is possible to teach multi-thread programming using only a few lectures, providing that these are combined with hands-on exercises and laboratories. It also shows that it is possible to include relatively advanced level discrete-time control topics, in a course for students specializing in embedded systems or software systems. Here, however, we have a large advantage at Lund University by the fact that the CS students and the EE students all have taken the same mandatory continuous-time basic control course.

One would believe that the computer engineering students should perform best in the course since they already have taken a mandatory course in real-time programming when they begin. This is, however, not the case. The better grades of the Engineering Physics students more than well compensate for this. Also the Mechanical Engineering students perform as well as, or sometimes better, than the Computer Engineering students. However, at the same time the top students in the course are often Computer Engineering students, so the variation is large.

Although the course works well in its current shape there are certain parts that one would like to include. One example is to consider multi-core platforms in the concurrent programming and scheduling lectures. Currently the course mostly covers the uniprocessor case. Another issue that would be natural to have is a modeling module based on Modelica [4], in particular since our department has a substantial Modelica activity at the research level. A possibility in this case would be to also include automatic controller code synthesis.

More information about the course can be found at the home-page

http://www.control.lth.se/Education/EngineeringProgram/FRTN01.html

[1] Anton Cervin, Dan Henriksson, Bo Lincoln, Johan Eker, Karl-Erik Årzén: "How Does Control Timing Affect Performance? Analysis and Simulation of Timing Using Jitterbug and TrueTime." IEEE Control Systems Magazine, 23:3, pp. 16–30, June 2003.

[2] Anders Nilsson: “Tailoring native compilation of Java for real-time systems”, Doctoral Dissertation, Department of Computer Science, Lund Univiversity, 2006

[3] Wittenmark, B., Åström, K. J., Årzén K. E. (2002), *Computer Control: An Overview*, IFAC Professional Briefs, freely available from http://www.ifac-control.org/

[3] Mattsson, S.E., Elmqvist, H., Otter, M., Physical system modeling with Modelica. Control Engineering Practice, vol. 6, pp. 501-510, 1998