A Driving Simulator for Teaching Embedded Automotive Control Applications

Paul G. Griffiths
Department of Mechanical Engineering
University of Michigan
Ann Arbor, MI 48109
paulgrif@umich.edu

R. Brent Gillespie
Department of Mechanical Engineering
University of Michigan
Ann Arbor, MI 48109
brentg@umich.edu

***Abstract*— This paper presents the multi-disciplinary nature of embedded control system design and how a course developed at the University of Michigan addresses the broad set of topics needed for embedded control system design to senior engineering undergraduates from a variety of backgrounds. The lab component uses a typical automotive powertrain micro-controller and teaches topics in system dynamics through programming assignments involving dynamical systems simulations. Students interact with and feel their virtual environments though a haptic interface. We present the final project in the course, in which students build a fixed-based driving simulator to test advanced automotive control system designs. This course was developed partially in response to industry demand for students with experience in embedded control. The course has generated remarkable student interest and former students have provided positive feedback about projects and have reported better opportunities in the job market as a result of their experience in this course.**

I. Introduction

Embedded control system design is inherently a multi-disciplinary field, requiring some background in mathematics, computer science (CS), computer engineering (CE), electrical engineering (EE), and mechanical engineering (ME). Effective control education must cover this wide breadth of topics [1]. Motivated by the expressed desire of our colleagues in the automotive industry to hire EE, CS and CE graduates with broad knowledge about micro-controllers, real-time operating systems, sampling, dynamics, simple control algorithms and state-of-art software tools for model-based embedded system development, an advanced undergraduate course on embedded control systems was developed at the University of Michigan. This course has been very successful if measured by our inability to meet the student demand for the course, positive student feedback after completing the course, and anecdotal evidence of improved job prospects of alumni of the course. We present the final project of the class, which ties together topics from control, real-time programming, networking, sensors, and motors. The students create a driving simulator and design a controller for either a steer-by-wire or an adaptive cruise control system that they test on their simulator.

The Embedded Control Laboratory was established in the Electrical Engineering-Computer Science department at the University of Michigan for the purpose of teaching design of embedded systems to senior undergraduate students with backgrounds in EE, CE, CS and ME. It was noted that students in EE receive a good signals and systems background, CE students understand low-level I/O in micro-controllers, and ME students would have a strong background in dynamics. However, all of the topics are necessary for embedded controller design, and as a result, students from different programs have different things to learn, but also are able to teach each other. The Embedded Control Lab complements lectures in an embedded systems class with practical experience programming a micro-controller, interfacing with lab hardware, and networking among multiple processors. Students are taught about low-level programming needed to access a micro-controller's I/O and they learn about types of common sensors and actuators, networking, and some basic concepts in control system design such as sampling, P, PD, and PID control. Toward the end of the course, the students use code generation tools and a real-time operating system to rapid-prototype their software. Using these tools, the students tackle a final project of greater complexity than their regular weekly labs and the project incorporates many of the concepts taught in the course.

One of the devices for the students to control in the Embedded Control Lab is the "Haptic Box", which has a motorized hand wheel and an encoder allowing single-axis virtual environments to be programmed and experienced through the sense of touch, or haptically. Building on the various lab assignments that involve the haptic boxes and that precede the final project, students create virtual mechanical systems, and they learn through physical experience about the dynamical equations for mechanical systems and linear systems [2] [3]. Students program a virtual spring, a virtual spring-damper and a virtual mass-spring-damper, and because the system parameters are set programmatically, the students can adjust the dynamics of the hand wheel while the program is running. Holding onto the hand wheel and feeling the difference between a lightly damped system and an over-damped system is an intuitive mode of learning dynamics. It is also difficult to setup negative physical damping or springs with negative spring-

rates with physical springs and dampers, but the students can feel such systems simply by tweaking the parameters of their program. The experience with virtual mechanical environments provide the students with an intuitive "grip" on the meaning of P or PD control action which they will use in their final project.

The final project assigned to students in past semesters has been an advanced automotive control application: either steer-by-wire or adaptive cruise control, which the students tested on a driving simulator. The driving simulator consisted simply of a motorized wheel with an angular encoder to serve as a steering wheel, a Motorola MPC555 micro-controller to run a real-time simulation of a vehicle and the steering controller, and a graphics program running on a PC to provide the driver's view of the road. The simulator made use of existing equipment in the Embedded Control Lab. For the steer-by-wire project, the communication between the steering wheel controller and steering system controller, which included a simulation of the vehicle's wheels and road forces, had to take place over a Controller Area Network (CAN) bus. For the adaptive cruise control project, instead of using radar to detect other vehicles, students virtually sensed their classmates' vehicles via CAN messages broadcasted across a CAN bus. The students responded enthusiastically to these projects and apparently enjoyed them in part because they were "real-world" but also because they could experience their designs both visually and haptically through the graphics and the motorized wheel.

II. Driving Simulator Hardware

The driving simulator runs on a Motorola MPC555-based development board for simulation of vehicle dynamics, a PC for graphical display and a "Haptic Box" that serves as a force-reflecting steering wheel. This setup uses only pre-existing hardware in the Embedded Control Laboratory and does not require additional hardware.

The Haptic Box is a general purpose, one degree of freedom, impedance-based, haptic interface that uses a DC motor to generate torque on a hand wheel and has an angular encoder on the motor's shaft to provide position feedback. One of the lab's Haptic Boxes is shown in Fig 1. It was designed and built by the Haptix Laboratory at U of M to serve as a simple low-cost haptic device ($600) to teach students about systems dynamics [2]. It has proven to be remarkably general purpose; several human factors experiments have been run on the device, it has taught students about instabilities arising from sampling and encoder resolution, demonstrated dynamics of coupled mass spring damper systems, served as the master and slave in bilateral teleoperation systems, and for the driving simulator it has served as a motorized steering wheel.

The setup for one lab station is shown in Fig. 2. The host PC is used primarily to load programs onto the MPC555 micro-controller and run a debugger for the MPC555 through a parallel port connection. A proprietary piece of

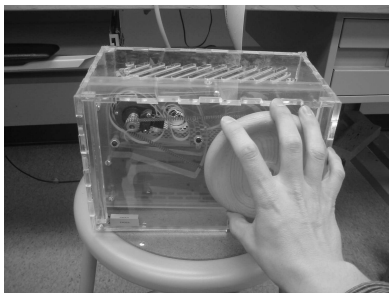

Fig. 1. One of the force-reflecting Haptic Boxes designed at the University of Michigan for teaching and research purposes. Students experience system dynamics by manipulating a virtual environment that they program.

hardware–of which there are several available–connects the parallel port to the micro-controller; we use a Macraigor Wiggler. A serial port connection enables programs to log data on the PC or to provide a simple text-based interface. The MPC555 micro-controller is the target processor that controls the Haptic Box. The particular MPC555-based development board used is the Axiom CME555, which can be purchased from Axiom Manufacturing for $599. See http://www.axman.com. An additional break-out board designed at U of M provides access to the pins of the MPC555 through buffer chips that prevent students from accidentally short-circuiting or otherwise destroying the more expensive MPC555 chip. Specialized connectors have been built into the break-out boards that combine a PWM output and the inputs for quadrature decoding into one ribbon cable. This ribbon cable plugs into the Haptic Box and connects the PWM signal to a power amplifier for the DC motor and connects the angular encoder's output to the MPC555's Time Processing Unit (TPU) module, which performs (fast) quadrature decoding (FQD).

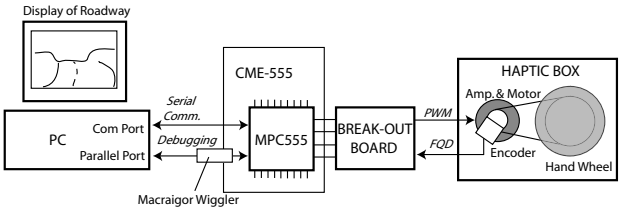

Fig. 2. Hardware setup in the Embedded Control System Laboratory is used to create a driving simulator, where the Haptic Box is a force-reflecting steering wheel.

Communication with the MPC555 is accomplished either through a serial connection, a debugging session, or communication across a CAN bus. Programs are loaded onto the processor from the PC using SingleStep, a debugger GUI for the MPC555 provided by Windriver. Once a program is running, the serial and the CAN communication modules can be initialized and begin communicating. A simple text-based program interface can be developed over the serial connection and accessed on the PC using a

terminal program such as HyperTerminal. For the driving simulator, the serial communication connects the vehicle simulation on the MPC555 with the graphics on the PC. At a data rate of 115k bits per second, the link is more than sufficient for providing updates to the vehicle's location and orientation, and additional data can be logged to a file for subsequent inspection. The CAN bus allows communication among processors in the lab enabling the implementation of distributed control systems.

III. Driving Simulator Software

The software for the driving simulator can be divided into target and host software that communicate together through a serial connection. By "target" and "host" we mean the MPC555 and the PC, respectively. The software on the target consists of a vehicle simulation and a controller. Models of these two components are combined into one Simulink model and code is generated using Mathwork's Realtime Workshop code generation technology. A Simulink blockset provided by New Eagle Software called RapidHawk (now Motohawk) supported generation of code for OSEKWorks, a real-time operating system provided by Windriver. A Simulink blockset written by one of the authors added support for some hardware features of the MPC555. The generated code is compiled using the Diab compiler, linked with OSEKWorks, and downloaded to the target. Although the students use this suite of tools to create their vehicle simulation and controller, all that is required is a compiler and debugger for the power-pc architecture, which are freely available as part of the GNU C compiler, gcc. See http://www.macraigor.com/gnu_faq.htm. However, the Rapidhawk blockset and the OSEKWorks operating system were generously donated by New Eagle Software and Windriver, respectively, and thus, our students benefited from the opportunity to develop embedded control systems with state-of-the-art software.

The background of the students is mainly computer and electrical engineering so we did not expect them to derive the kinematics and dynamics of a car. Instead we provided them with the nonlinear differential equations describing the kinematics of the bicycle vehicle model without tire-slip, which they in turn implemented and tested in Simulink. A side-lesson for the students in this process was how the use of a simulation tool like Simulink can significantly reduce the development time of a control system as they can design and test without ever running code on the target. This lesson was particularly well appreciated as the students had to code relatively complicated programs such as real-time simulations of the dynamics of mass-spring-damper systems, and they discovered how difficult it is to diagnose a problem when the program is running inside a micro-controller with no monitor, keyboard or mouse. Once they had tested their vehicle model and controller through simulation, they built the software and loaded it onto the target.

The graphics software was written by one of the authors and it makes use of the OpenGL library native in Windows 2000 and XP. See the course website http://www.eecs.umich.edu/courses/eecs461. The graphics program shows a view looking over a car hood down a concrete roadway with a single yellow centerline, grassy embankments on the right and left, and optional orange cylinders that can serve as obstacles. A sample screen-shot is shown in Fig. 3. Roadway geometry is read in from a file, and any smooth path can be generated as the graphics program simply extrudes the road and embankment texture between cross-sections. Each cross-section has three spatial coordinates and three coordinates for orientation. Although the path shape is quite general, the software does not support intersections. A smooth animation is accomplished by double-buffering and generating the graphics in one thread of execution and handling communication with the target in another thread. The communication thread reads a formatted message from the target through the serial port. We provided students with a Simulink block that could properly format a vector of input signals and send them to the graphics program through the serial link. For the students projects, we limited the road geometry to two-dimensions and so the graphics software required regular updates of the vehicle position in only two-dimensions along with a single vehicle heading angle to determine the direction of the view. Besides providing a display of the roadway, the host software also logged the serial communication to a file for later examination. Remaining bandwidth in the serial link was used to log additional variables. Students could read the logged data into Matlab and graph vehicle coordinates, error signals, control values or other data that might help them diagnose problems.

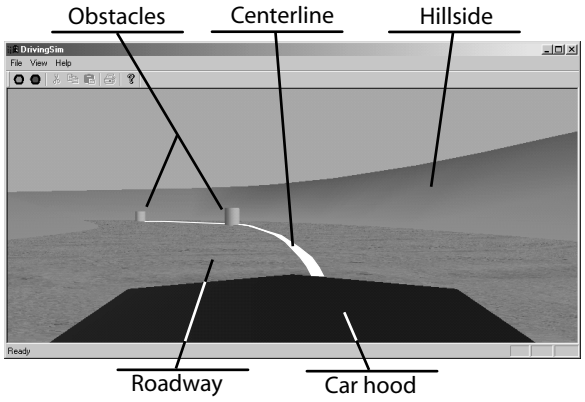

Fig. 3. An OpenGL animation of the roadway visible over the hood of the simulated vehicle (labels added).

IV. Example Automotive Control Applications

Given the proximity of the University of Michigan, Ann Arbor to major automotive manufacturers and tier-one suppliers, there is significant student interest in automotive engineering and partnering between industry and the

research work at the university. Working on automotive applications while in school makes students more marketable, and experience with embedded systems is currently in demand. The driving simulator seemed like a great way to mix embedded systems design and control design in a visual and haptic experience. Students had the opportunity to tackle the design of an automotive control system, then prototype it and actually feel and see its behavior with the simulator.

Networking is one of the topics taught in the class and we wanted the final projects to include a networking aspect. The MPC555 has two CAN modules on-board that allow the micro-processor to be attached to a CAN bus, a common communication bus in American cars. The first class to try the simulator was given the task of designing a steer-by-wire controller, where the wheel and steering actuator dynamics were simulated on one processor along with a controller to receive commands and reflect forces to another controller running on another processor. The two parts of the controller communicated to each other via the CAN bus. The second project was the design of an adaptive cruise control system. In this project, each group had simulated its vehicle and controller on one processor and virtual sensing of other vehicles was accomplished by each processor broadcasting its vehicle's location on the CAN bus in a fashion similar to the way radar systems broadcast target information.

For both projects, the students were presented with the bicycle model for representing the simplified kinematics of a four-wheeled car. This model assumes that there is just one front wheel and one rear wheel like a bicycle as shown in figure 4. We allowed the students to ignore tire slip – although the more ambitious students tried deriving the equations of motion using tire-slip and some even included tire-patch saturation. The bicycle model has three degrees of freedom: the position (x,y) of the vehicle center in the X-Y plane and the heading ψ. If the front wheel speed is u, and the steering angle is δ_f, then the kinematics are given by:

$$\dot{x} = \frac{-ud}{L} \sin\delta_f \sin\psi + u\cos\delta_f \cos\psi \quad (1)$$

$$\dot{y} = \frac{ud}{L} \sin\delta_f \cos\psi + u\cos\delta_f \sin\psi \quad (2)$$

$$\dot{\psi} = \frac{u}{L} \sin\delta_f \quad (3)$$

A. Steer-by-wire over a CAN Network

For the steer-by-wire project, students had a distributed control problem with a communication link between two controllers via a CAN bus. One controller had the responsibility of actuating the front (steerable) wheels of the simulated vehicle. The wheels were modelled as a rotational inertia driven by a torque. A self-aligning torque acted on the steering system and this torque had to be reflected back onto the steering wheel by communication across the network. A PD controller was suggested to the students, where the set-point for the front wheel is the steering wheel angle and the applied steering force is reflected back to the steering wheel.

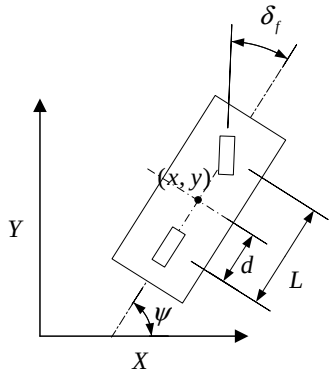

Fig. 4. Bicycle vehicle model

A number of complicating factors are embedded within this problem which the students discover by themselves when realize the control system. For instance, the students have to choose sampling rates for both controllers and consider the network bandwidth that they will then require. All the students share the bus which has a total bandwidth of 500 kbps. A sample rate too slow leads to instability given the round-trip delay between the two controllers. However, the CAN protocol allows nodes, in effect, to yell over-top of other nodes by transmitting messages with higher IDs, so a controller transmitting at a very high frequency can monopolize the bus. The students learn first-hand that the networking protocol can affect a control system's performance when another student's controller floods the CAN bus, starving their controllers of communication.

One logistical difficulty in the steer-by-wire project is the need for two processors per student group. A feature of the MPC555's TouCAN module (the CAN controller) is the ability to receive its own transmissions. This allowed students to test their distributed controllers without really distributing them across two processors, although combining both onto one processor reduced the maximum frequency at which they could run their controllers.

B. Adaptive Cruise Control

The adaptive cruise control project turned the lab into a more collaborative–if sometimes hostile–experience. All the simulated vehicle controllers had to interact with each other through the CAN network to share one virtual roadway. From a functional perspective, an ACC system has a typical cruise control that maintains the vehicle speed, but it also detects traffic ahead and maintains some appropriate spacing if needed. For this project to work, each simulated vehicle had to broadcast its position and velocity information onto

the CAN bus for the rest of the controllers to hear. Then, if the ACC system was enabled, either a speed control or a distance controller is activated depending on some appropriate switching logic.

Basic PID controllers were supplied to the students for the speed control and distance control and their task, arguably tougher than design of a PID controller, was to select an appropriate switching logic. For many students, this was their first experience with hybrid systems, and we let them discover on their own the problems of chatter in the switching and instability due to switching.

The collaborative nature of the project was interesting because students were subject to the problems in other groups' simulations. Cars would sometimes randomly appear and demonstrate to everyone that some students did not have the right kinematics or dynamics for their car. This led to a lot of excitement, laughing and some hair-pulling as students tried to understand the strange behavior of their vehicle. An additional processor on the network allowed the lab instructors to run a CAN monitor, written by one of the authors (see the course website given), to identify groups running amok on the bus. This sort of bus monitoring tool is a most in the lab setting to settle debates between students about who is transmitting what.

V. Evaluation

For many students, this course is their first experience in the field of control systems. We think that the a hands-on haptic and visual experience is a particularly intuitive way to connect abstract concepts like the roots of a system's characteristic equation with physical behavior like damped oscillations or unstable behavior. The students built on their experience with virtual environments like springs and spring-damper systems to understand the use of P and PD controllers to achieve a control objective. Feeling the control action of a P or PD control through the Haptic Box provided another means for teaching the concept in an intuitive manner. Many students also had the opportunity to feel negative spring rates and negative damping coefficients when they accidently omitted a negative sign!

The breadth of the course prohibited including much discussion of control theory. Instead, some basic ideas such as P and PD control were presented by talking about the mechanical analogs and programming virtual springs and virtual spring-dampers. We appealed to students' intuition to get them started in the right direction for the final project. Beyond the basic controller, we asked the students to develop some additional control feature for their final project. This is where the students really stretched to out-shine each other. As examples, some groups designed steering controllers to drive their car down the center of the road. There were also lane departure controllers that tried to keep the car off the shoulder. Some groups used orange barrels in the road to test obstacle avoidance controllers; some groups tried to combine the idea of haptic collision warning (shaking the steering wheel to alert the driver) with active obstacle avoidance. In tackling these tough problems, there were many opportunities to foster students' nascent interest in control by showing them different control design techniques. We showed students how to tune PID controllers, how to understand open-loop Bode plots and how to use Matlab to design a state-feedback control law.

The driving simulator combined both a visual and haptic way of experiencing control design. For a variety of reasons, we cannot let students try their hand at designing and testing steer-by-wire or adaptive cruise control systems on real vehicles. But in a graphical and haptic simulation, they were able to attack some relatively sophisticated control problems, test their software on an appropriate microprocessor for the application, and experience their design in a manner more satisfying and fun than only examining plots of signals from simulation. The real-world control problem and the multi-modal way of experiencing their control system provides the students with a more engaging and valuable learning experience [4].

From our perspective, this course is successfully providing students with valuable background in embedded control systems. Certainly there is an overwhelming demand for the class, with a typical wait-list as long as the enrollment limit of 48 students per semester. After the course, students have responded particularly enthusiastically about the final project and say that they are listing the experience in this course on their resume to improve their prospects in the job market. One student reported back to us that her experience with the MPC555, rapid-prototyping, and the automotive application of the final project was what secured her a job. Besides making the students more marketable, we believe the course is providing unique exposure to the wide set of topics necessary for designing embedded control systems that is called for by industry and leading figures in the field of automatic control [1] [4].

References

[1] P. Antsaklis, T. Basar, R. DeCarlo, N. H. McClamroch, M. Spong, and S. Yurkovich. Report on the NSF/CSS workshop on new directions in control engineering education. *IEEE Control Systems Magazine*, 19(5):53–58, October 1999.

[2] R. B. Gillespie, M. B. Hoffman, and J. Freudenberg. Haptic interface for hands-on instruction in system dynamics and embedded control. In *Proceedings of the 11th Sympsium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pages 410–415. IEEE Computer Society, March 2003.

[3] C. Richard, A. M. Okamura, and M. R. Cutkosky. Getting a feel for dynamics: Using haptic interface kits for teaching dynamics and controls. In *Proceedings of ASME DSC Division*, pages 153–7. ASME, November 1997.

[4] D. S. Bernstein. Enhancing undergraduate control education. *IEEE Control Systems Magazine*, 19(5):40–43, October 1999.