Model-Based Real-Time Embedded Control Software for Automotive Torque Management

Carlos Zavala, Pannag Sanketi, Daniel Lamberson, Anouck R. Girard and J. Karl Hedrick

Abstract— **The development and implementation of model-driven architectures for integrated real-time, embedded, hybrid control software is described using an automotive system as the application instance. Specifically, engine torque management, in which electronic throttle control and air/fuel ratio control are integrated to form a unifying control strategy, is discussed. The associated models are presented and simulated and the outcome is compared with the experimental results obtained using a suitable testbed. We emphasize the integration of software along the development process and point out issues found during the implementation on a particular hardware platform.**

***Index Terms* — Automotive Control, Embedded Software, Hybrid Systems, Control Architecture, Model-Driven Software, Platform based design, Torque management**

I. Introduction

Simply stated, an embedded controller is a special-purpose controller that is embedded in a greater system [1]. Embedded controllers are often found in appliances (microwave oven, refrigerators, television and VCRs, stereos), computer equipment (laser printers, modems), robotics, automobiles (engine control, diagnostics), environmental control (greenhouse, factory, home), instrumentation, aerospace, etc.

There are ever increasing demands being placed on the functionality, complexity and reliability of embedded systems. Modern-day automobiles now contain controllers for applications such as torque control, vehicle stability and traction control, electronic control of windows and driver-seat settings.

The current practice of design and implementation of embedded systems is to have the control design process and the implementation happen separately as shown in figure 1a. The implementation on the target processor is most likely to be done by someone other than the control designer. Changes made to the control code during the process of converting the model to implementation code and those that occur during the final testing phase are not fed back to the simulation model.

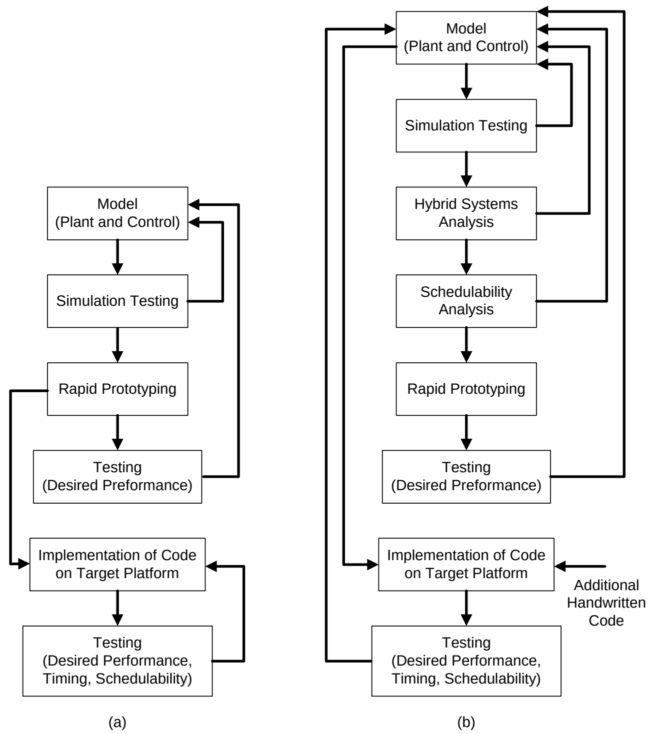

Figure 1. Process for design and implementation of embedded software (a) current process (b) model based approach [2]

This process leaves a disconnect between the simulation model and the implementation code and many of the problems associated with the target platform are not found until testing begins, leading to various subsequent problems. Thus, there is a need for an approach that would lead to reduction in the amount of hand written code, reduction of redundant work in moving from application to application and decreased testing time. It should comprise improved modeling of the physical constraints of the control hardware.

The model-based process we present in this paper, shown in figure 1b, places emphasis on integrating various phases of the design process, hence leading to faster production of quality

C. Zavala and P. Sanketi are graduate students with the vehicle dynamics laboratory at the University of California at Berkeley (email: czavala@vehicle.me.berkeley.edu, pannag@vehicle.me.berkeley.edu)

D.Lamberson is an alumnus of the vehicle dynamics laboratory at the University of California at Berkeley (e-mail: dalmberson@vehicle.me.berkeley.edu).

A. Girard is a visiting post-doctoral researcher and lecturer at the University of California at Berkeley (e-mail: anouck@eecs.berkeley.edu)

J. K. Hedrick is a Professor of Mechanical Engineering at the University of California at Berkeley (e-mail: khedrick@me.berkeley.edu).

code and minimization of the cost associated with fixing the problems. We start with hybrid system models of the plant and controller, which allows verification, simulation and automatic code generation. The approach is applied to automotive control, more specifically to torque management. The engine torque management strategy presented here involves a clever combination of throttle position control and air to fuel ratio control.

In this paper, the automotive and controller models are presented, implementation of the controllers on automotive engines is discussed in terms of experimental platform capabilities and software implementation; and experimental results are shown.

II. Model and Model-Driven Controller Architecture and Design

The engine model used for control development includes the dynamics of the throttle, air-fuel ratio and the vehicle dynamics. This is shown in figure 2. The engine torque management strategy involves a high level control which outputs a desired throttle position and a desired air to fuel ratio. The strategy combines these two low level controllers, which are discussed further. The torque management strategy takes advantage of the drive-by-wire system by interpreting the pedal position as torque demand. The throttle and fuel controllers can take these set points as inputs. The details of the model are available in [3].

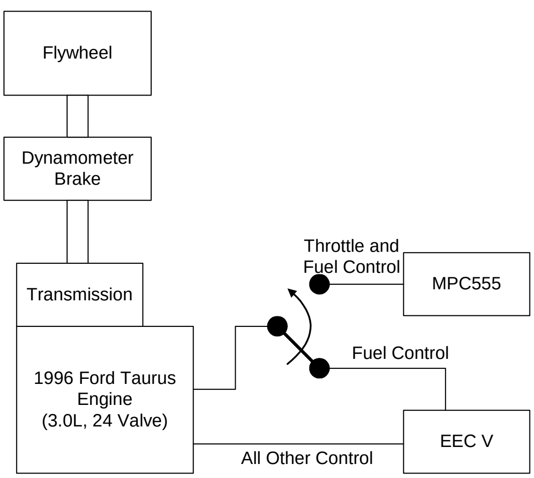

Figure 2. Schematic on engine model and controller

For implementation of the strategy, the pedal position to desired engine torque map is set to be identical to the engine response with the mechanical throttle.

The electronic throttle control (ETC) system is a drive-by-wire system in which the direct linkages between the accelerator and the throttle or the steering wheel and the steering gear are replaced with pairs of sensors and actuators. The top most level of the ETC system model contains the driver, actuator plant, sensor and controller models and specifies the interconnection of these models. This level of the model contains a mixture of continuous and discrete time signals. The signals available to the sensors are strictly those that can be measured with the hardware and the controller receives these signals with noise. The electronic throttle plant is modeled as a second-order, nonlinear system. The model takes into account coulomb friction and saturation due to physical constraints on the throttle plate. The top-most level of the controller model is designed using sliding mode control technique. It deals with the scheduling of the controller tasks and the input and output that the controller has with the environment. A sub-model within the controller model specifies the triggering of the three control tasks for the electronic throttle control. The feedback around the actuator and plant model reveals the coupling between the electric dynamics of the motor and the mechanical dynamics of the throttle plate.

The air to fuel ratio control requires the control action to be triggered asynchronously. A toothed wheel is used to tell the controller what the current engine position is. A subset of the powertrain model is used for the fuel controller design. Standard nonlinear control design techniques are used to design a sliding mode air-fuel ratio controller. The transmission is modeled for fourth gear only; so all simulations are performed for throttle angles resulting in vehicle speeds above 30 mph. The controller is modeled in the discrete time domain.

III. Software Development

The essential architecture of embedded systems consists of programmable cores, I/O subsystem and memories. A family of architectures that allows substantial re-use of software is what we call a hardware platform. To be able to achieve the level of application software re-use, the hardware platform has to be abstracted at a level where the application software sees a high-level interface to the hardware that we call Application Program Interface or API. A software layer is used to perform this abstraction. This layer covers multiple parts of the hardware platform: the programmable cores and the memory subsystem via a Real-Time Operating System (RTOS), the I/O subsystem via the Device Drivers, and the network connection via the network communication subsystem. This layer is called the software platform. The system platform is formed by the integration of the hardware and the software platforms [4].

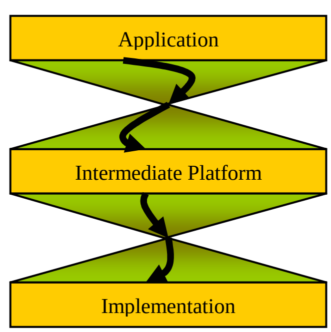

Figure 3. Hardware platform based design

Hardware platform-based design looks at ways to face application specifications and architecture definition in a non-traditional (bottom-up or top-down) methodology that gives flexibility to the search of optimal design parameters.

Figure 3 shows the hardware based design approach, the bottom of the figure shows the set of architectures that could implement that application. The arrow shows the design process, which minimizes a cost function (including production cost, design cost, etc) while looking for reasonable solutions.

Platforms eliminate large loop iterations for affordable design. They shape design space with new forms of regularity and structure that enable design potential for lower cost and first pass success. The number and location of intermediate platforms is the essence of platform- based design

We used a model-based approach founded on the basis of a hardware platform design throughout the control and software development. The model based approach underlines the use of analysis tools in the simulation environment and uses the model as the source for implementation. In this approach, the model is updated at each step in the analysis and implementation process. The simulation environment counts with schedulability analysis tools and timing analysis tools to test various aspects of the control. Thus, the control designer can do more testing in the relatively cheap simulation environment. Rapid prototyping tools and automatic code generators also give more certainty by reducing the possibility of manual mistakes. This requires modeling constructs as defined in [5]. If changes are required during the testing phase, the model is updated and the code is generated again. In this way, the model contains a representation of the code running on the processor. As components are added to the system, the original model will be altered. However, since the model fully represents the code running on the processor, the redundant work is minimized.

Different tools account for timing (Giotto), schedulability (Airies), and, if the models are phrased as hybrid systems, hybrid system analysis and verification (Checkmate).

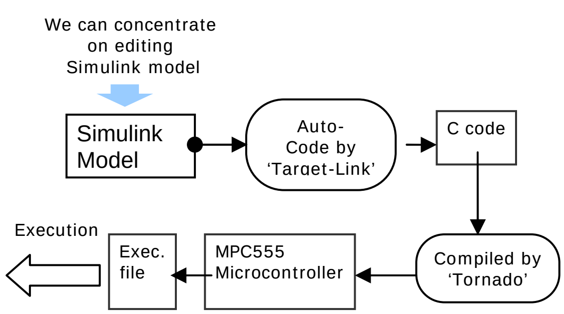

Figure 4. Software development process

The process design diagram is shown in figure 4. In this specific configuration, the source is a simulink model, from which code is developed in two lower levels: C language code and assembly language code. In this process, changes can be made relatively cheaply in the simulation environment before the testing phases begin. Thus, increased analysis time required in the model based approach represent more advantages in terms of time required for implementation.

The following results present the use of this type of model based approach for the development of a control system for electronic throttle control and air to fuel ratio control.

IV. Implementation and Experimental Results

A. Experimental Platform

The powertrain test facility consists of a 1996 Ford Taurus engine (3.0L, 24 Valve, DOHC) and 4-speed automatic transmission. The engine has been outfitted with a BMW electronic throttle (part # 13.54-1 439 224). A 149.2 kW (200 hP) dynamometer is used to control the output of the transmission. Also, to better represent the inertial load of a vehicle, a 13.5 kg-m^2 (317 ft-lb^2) flywheel was added downstream of the dynamometer. Implementation of the controls is done using a MPC555 processor. The MPC555 is viewed as the next standard processor to be used in the automotive industry.

B. Experimental Results

The experimental results of the electronic throttle control are shown below. Figure 5 shows the throttle response to a sinusoidal input of the pedal position. The engine speed is also shown for reference.

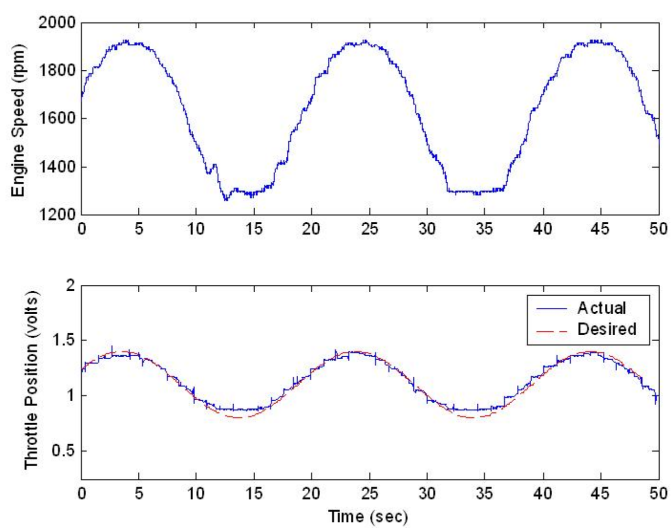

Figure 5. Throttle position control and engine speed

Figure 6 shows the results of the factory AFR controller (part of the original electronic control unit which is displayed here for reference). As shown, the fuel commanded increases with throttle angle and air flow. This test consists also of a sinusoidal input to the pedal position. The regions where the lambda sensor response looks denser or averages lower voltage, correspond to low air/fuel ratio and more gasoline being injected in the engine.

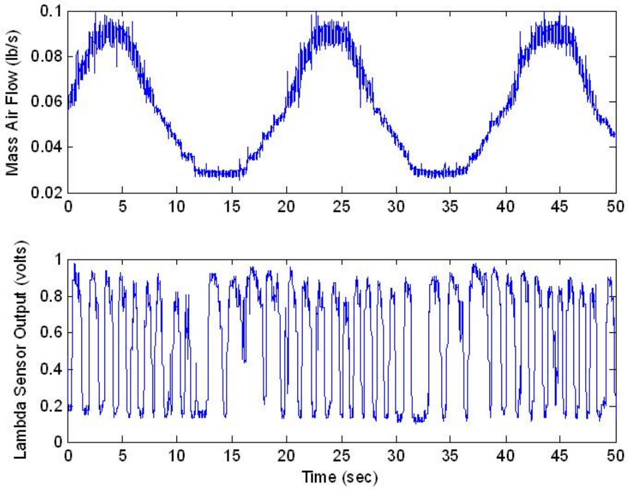

Figure 6. Production AFR control

As shown, the output of the oxygen sensor changes from low to high throughout the test. This indicates that the air to fuel ratio is cycling from lean to rich as desired.

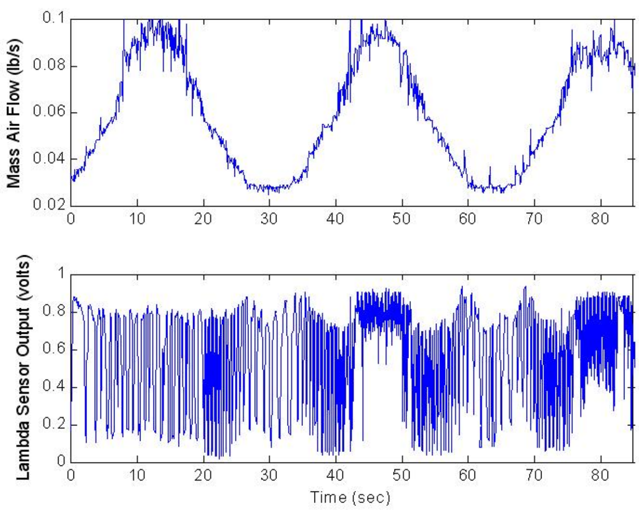

Figure 7. Berkeley AFR control

Figure 7 shows the results of this same test using the controller presented in this paper. In this test, the fuel control is switched between the production control to the Berkeley control at 20 seconds. As shown, the Berkeley fuel controller forces the air to fuel ratio (the lambda sensor output) to cycle between lean and rich operation much more quickly that the production control. While this is the proper function of the fuel control, it is noted that the output of the lambda sensor is shifted higher to some extent. This indicates that, although the air to fuel ratio is cycling between lean and rich operation, it is lean for the majority of the test. Since engine torque is related to air to fuel ratio, the engine torque is also affected.

C. *Expected Torque Management Results*

The following plots show the results of the torque management strategy when air to fuel ratio is adjusted for short duration torque transients. In figure 8, the engine response is shown. The throttle flow and manifold pressure vary to account for a change in the accessory torque. However, the transients in the engine speed and turbine torque are eliminated. This is due to the fact that the air to fuel ratio is being used to control torque during these transients. The AFR control is actuating to cancel oscillations in both engine speed and turbine torque. The change in air to fuel ratio is shown in figure 9.

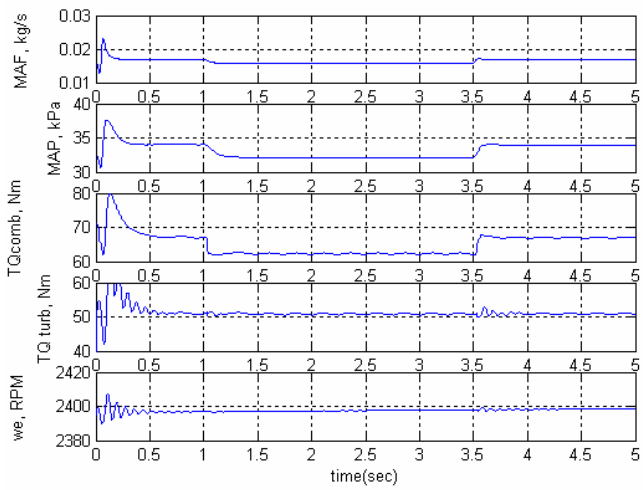

Figure 8. Torque management results

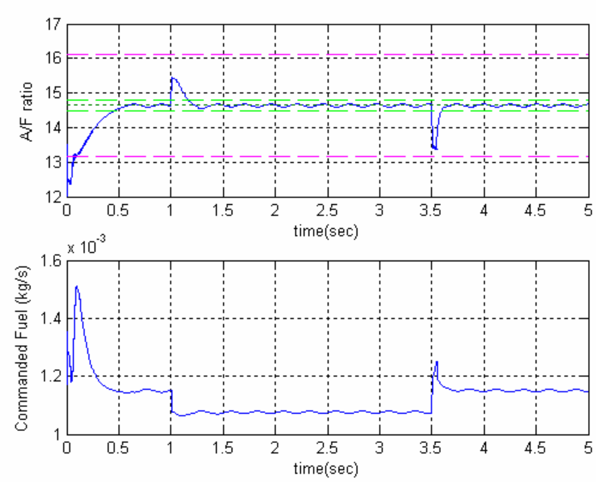

Figure 9. AFR regulation in torque management

V. CONCLUSIONS

The model-based approach to the development of real-time, embedded, hybrid controller design was presented in this paper. The analysis was formulated under the hardware platform design model that maps the functionality of a control algorithm into a specific hardware architecture. The advantages of model based design are illustrated during the design of throttle and

AFR controllers. The torque management strategy was also introduced as a way of integrating throttle and AFR control for the improvement of the torque profile. Torque management expected results are presented together with experimental results for throttle and AFR control.

REFERENCES

[1] http://www.wikipedia.org/wiki/Embedded_system, from Wikipedia, the free Encyclopedia

[2] D. Lamberson, N.P. Teske, J.K. Hedrick, Intelligent Transportation Systems, 2003. Proceedings. 2003 IEEE , Volume:1, 12-15Oct.2003 Pages:655 - 659

[3] Girard, A.R., Spry, S.C., Kretz, P.R. Dickey, S.R., Empey, D.M., Misener, J.A., Variaya, P.P. and Hedrick, J.K., "Vehicle-to-Vehicle Open Experimental Platform Reference Manual", Report to DARPA on the MoBIES Project, available in its latest version for the web at: http://robotics.eecs.berkeley.edu/~anouck/v2v_report.pdf

[4] A. Sangiovanni-Vincentelli, "Defining Platform-based Design", EEDesign, February 2002.

[5] Paul Griffiths. Embedded Software Control Design for and Electronic Throttle Body. Masters Thesis. University of California, Berkeley, 2002.