AutoPlug: An Open Experimental Platform for Automotive ECU Testing, Updates and Verification

Rahul Mangharam
Dept. Electrical & System Engineering
University of Pennsylvania
rahulm@seas.upenn.edu

ABSTRACT

In 2010, over 20.3 million vehicles were recalled. Software issues related to automotive controls such as cruise control, anti-lock braking system, traction control and stability control, account for an increasingly large percentage of the overall vehicles recalled. There is a need for new and scalable methods to evaluate automotive controls in a realistic and open setting. We have developed AutoPlug, an automotive Electronic Controller Unit (ECU) test-bed to diagnose, test, update and verify controls software. AutoPlug consists of multiple ECUs interconnected by a CAN bus, a race car driving simulator which behaves as the plant model and a vehicle safety and performance monitor in Matlab. As the ECUs drive the simulated vehicle, the physics-based simulation provides feedback to the controllers in terms of acceleration, yaw, friction and vehicle stability. This closed-loop platform is then used to evaluate multiple vehicle control software modules such as traction, stability and cruise control. With this test-bed we are aim to develop ECU software diagnosis and testing to evaluate the effect on the stability and performance of the vehicle. Code updates can be executed via a smart phone so drivers may remotely "patch" their vehicle. This closed-loop automotive control test-bed allows the automotive research community to explore the capabilities and challenges of safe and secure remote code updates for vehicle recalls management.

Keywords

Real-time control systems, debugging, testing, verification

1. INTRODUCTION

In 2010, safety recalls affected 20.3 million vehicles, according to the National Highway Traffic Safety Administration (NHTSA). All together since 1966, when NHTSA's record of recalls begins, the industry has recalled more than 470 million vehicles. Over the past decade, software issues in the automotive electronic controller units (ECUs) have begun to account for an increasing larger share of the source of recalls. Consider for example, in a 2009 defect notice filed with the NHTSA, Volvo recalled 17,000 vehicles for the following software-related issue: "The engine cooling fan may stop working due to a software programming error in the fan control module." In the short term, the problem could lead to "reduced air conditioning performance." However, if not corrected, it could lead to "loss of cooling system function and engine failure. The driver may not have sufficient time to react to the warning lights or the text message in the instrument panel, increasing the risk of a crash." While such critical recalls require vehicles to be repaired by the dealership, a large proportion of software related recalls may be facilitated by remotely upgrading the software in the vehicle.

There is an urgent need for systematic analysis of software bugs in automotive control systems, a system to *remotely* diagnose the software on a set of ECUs, perform safe and secure remote code updates and finally execute methods for online testing and runtime verification for evidence-based confirmation of the safety and efficacy of the code update. Remote vehicle recalls management will initiate network-wide preventive maintenance, provide an estimate the affected population, deliver a means for continuous on-line performance monitoring and potentially reduce the cost of warranty management significantly.

2. POSITION

Software-related issues with automotive ECUs will continue to be a dominant reason for recalls in the future. The cost of servicing these recalls by manually servicing each vehicle significantly impacts the competitiveness of the automotive manufacturer. Furthermore, currently all vehicles of the given make/year/model are recalled while the issue may only affect a small percentage of the vehicles. There is currently no means to distinguish the affected vehicles from those that continue to be safe and efficient. There is, therefore, an urgent need to explore remote warranty management and remote recalls servicing of software. This can be accomplished by interfacing the vehicle's computer to a smartphone and running remote code diagnostics, code updates/patches, testing and safety certification. While this approach may not be applicable to the most critical issues, it will be able to service non-critical and cabin control issues. The proposed approach involves the following steps (see Fig. 1):

1. Software Bug Detection When the automotive manufacturer is alerted about a potential software issue in a particular, a broadcast with a diagnostic test is sent to the potentially affected vehicle owners. The code is loaded in a

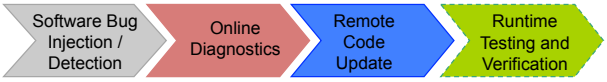

Figure 1: Remote ECU Recalls Management

USCAR Automotive CPS Workshop March 17–18, 2011, Troy, Michigan.

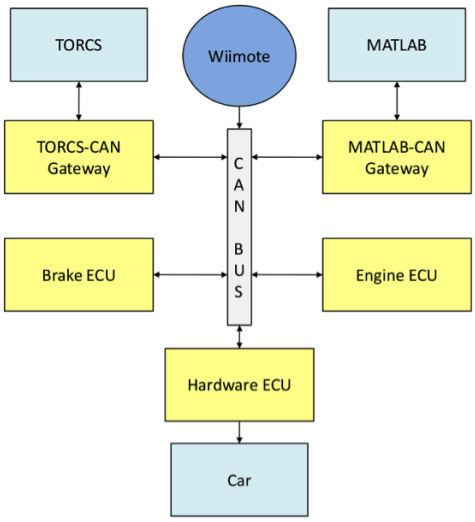

Figure 2: AutoPlug system architecture

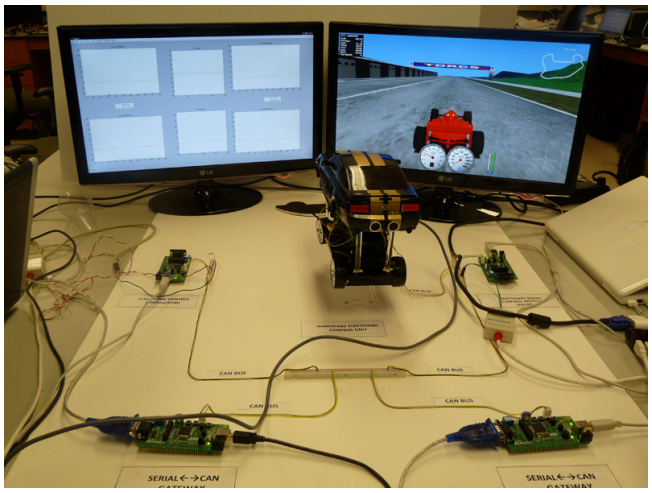

Figure 3: AutoPlug ECU Test-bed Platform

secure manner from the owner's smartphone into the vehicle via a WiFi gateway interfacing the vehicle's computer.

2. Online Diagnostics The diagnostic code runs a set of invasive and non-invasive tests which determine if the vehicle is to be recalled or not.

3. Remote Code Updates If a recall is necessary, the owner has the option to conduct the recall remotely by downloading a patch for the ECU software. The patch is applied automatically to the set of target ECUs.

4. Runtime testing and verification provide evidence based proof that the applied patch is safe and does not violate the properties of the affected ECUs.

3. RESEARCH CHALLENGES

With the above procedure, there are significant challenges with ensuring that the communication and write actions to the vehicle are safe and secure. As the onus of maintaining safe software in the vehicle is on the manufacturer, there must be both functional and formal approaches to guarantee the safety of the system before, during and after the software patch. We currently assume the patch will be applied only when the vehicle is parked and the engine is off.

4. INNOVATIONS AND ABSTRACTIONS

We propose the early steps in creating an open-source automotive ECU test-bed to be used for the development of new software processes to improve the warranty and recall management of vehicles. This platform will help investigate new automotive architectures for remote vehicle software monitoring, testing and updates for more efficient vehicle recalls management. We have created a network of ECUs which implement control algorithms for anti-lock braking system (ABS), traction control, stability control and cruise control. In place of a real vehicle we have used The Open-source Race Car Simulator (TORCS) which is a 3D driving driving simulator that provides physics-based feedback to the ECU network. A Nintendo Wii mote is used as a steering wheel to make it more interactive. Matlab is used to set parameters and analyze the output.

5. SYSTEM ARCHITECTURE

As shown in Fig. 2, we have five Freescale HCS12 microcontroller based ECUs connected through a 500Kbps CAN bus. Their functions are listed below:

1. Engine control Module: Throttle Control, Auto Transmission and Cruise Control.
2. Brake control Module: ABS, Traction Control, Stability Control.
3. TORCS Module: Serial conversion and filtering of CAN messages required for TORCS and driver input.
4. Telemetry Module: Serial conversion and filtering of CAN messages required for graphing in MATLAB.
5. Hardware control Module: Drives a model car to replicate the steering input and rear wheel speeds.

5.1 Race Car Plant Model

TORCS simulation features a simple damage model, collisions, tire and wheel properties (springs, dampers, stiffness,), aerodynamics (ground effect, spoilers,) and much more. We used TORCS to get the required data such as individual wheel speeds, car speed, engine RPM, current gear, yaw rate. By default, TORCS did not allow us to control individual wheel brake pressures so we modified the source code to enable it.

5.2 User Input

A Wii mote with a steering wheel was used to provide user input over the CAN bus (acceleration, brake, steering angle, gear, clutch and controls). The Wii mote interfaces with the PC using Bluetooth. We used an open source library (wiiuse) for Linux which captured the button press and motion events on Wii mote.

5.3 CAN messaging

Since we have distributed controls we need to ensure that the messages have a priority based hierarchy. We took advantage of the arbitration mechanism of the CAN protocol to ensure that the identifier fields of the higher priority messages will dominate the lower priority messages. More details on the message formats are on the http://autoplug.org.

6. AUTOMOTIVE CONTROLS

The test-bed is currently able to operate the vehicle with a choice of stability control, traction control, anti-lock braking

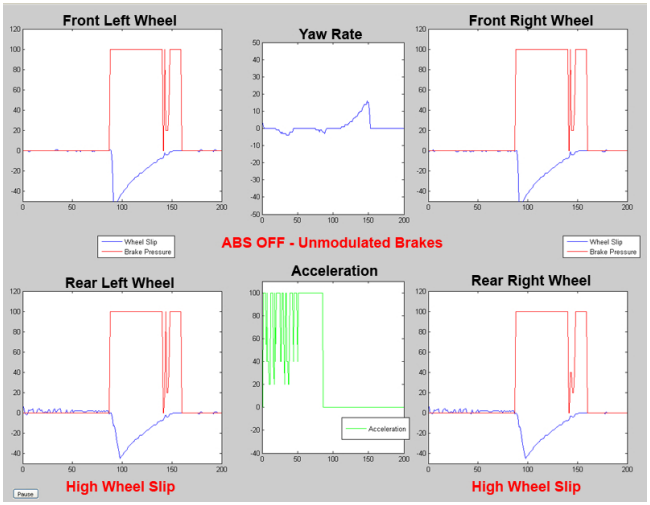

Figure 4: Anti-lock Braking System OFF: Unmodulated brakes result in high wheel slip

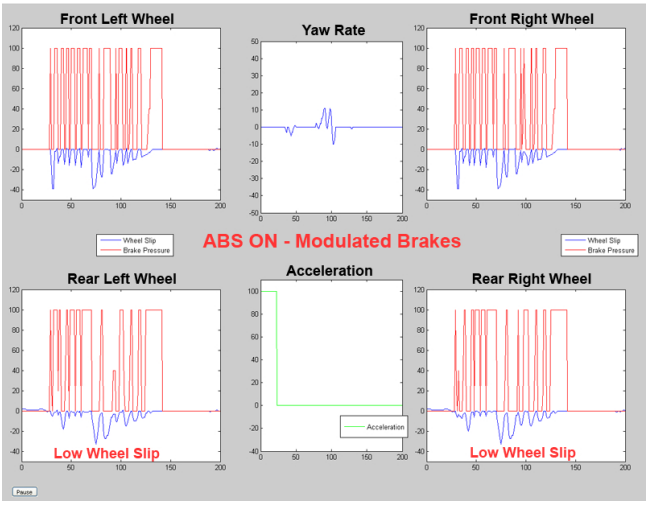

Figure 5: Anti-lock Braking System ON: Brake modulation reduces the wheel slip

system and cruise control. We describe two of the above features below.

6.1 Anti-lock Braking System

ABS tries to ensure that during braking the wheels of the car do not lock (i.e. slip with respect to the ground) so we try to match the wheel speeds with the linear speed of the car during braking by modulating the brake pressure on the wheels. ABS is implemented in the Brake Control Module. A simple proportional control is implemented with a gain of 20 and threshold of 3 m/s. The plots in Fig. 4 of the wheel slip during braking show the effect of ABS off. With the ABS switched on, the vehicle's brakes are modulated to minimize loss of traction due to wheel slip (see Fig. 5).

6.2 Traction Control

Traction control helps to maintain wheel traction with the ground during acceleration and cornering. Again as in ABS we try to match the wheel speeds with the linear car speed, but here we modulate the throttle instead of the brake pressure. We use proportional control with a gain of 20 and threshold of 3 m/s. The traction control happens in the Brake module ECU which sends a correction to the computed acceleration in the Engine control ECU.

Other controls such as stability control and cruise control were implemented

7. SAFE SOFTWARE UPDATE PROCEDURE

The test-bed is currently being used to evaluate software testing and verification procedures for safe patching of controls software on the ECUs. Each ECU runs the nanoRK (http://nanork.org) real-time operating system (RTOS) and is capable of executing tasks with a fixed period and worst case execution time using the rate monotonic scheduling (RMS) scheme or the earliest deadline first scheduling (EDF) scheme. Tasks may be activated, terminated, suspended at runtime by remote messaging over the CAN bus. Furthermore, additions to the RTOS allow new tasks to be issued over the CAN bus to monitor/replace/assist existing tasks. This provides the basic mechanisms to apply code updates.

We are able to introduce bugs in the controls software, such as, time delays, sensor noise and parametric changes to the proportional-integrative-derivative (PID) controller algorithms on the ECUs. This results in unstable or less responsive systems. We are then able to issue software tests tailored to monitor the ECU at runtime. This provides feedback as to whether a software patch is necessary or not. The patch is applied by migrating a task over the CAN bus to the appropriate set of ECUs. We are currently able to execute runtime verification procedures to demonstrate that the updated software maintains the safety and efficacy properties as intended. The platform will be made available to the research community and the goal is for it to serve as an experimental test-bench for future remote vehicle software processes.

8. CONCLUSION

Automotive recalls due to software-related issues are on the rise and will continue to affect a very large number of vehicles. An open automotive ECU architecture is proposed for use by the community at large. This will facilitate new and efficient methods to detect, diagnose, update, test and verify automotive ECU software.

9. BIOGRAPHY

Rahul Mangharam is the Stephen J Angello Chair and Assistant Professor in the Dept. of Electrical and Systems Engineering and Dept. of Computer and Information Science at the University of Pennsylvania. He directs the Real-Time and Embedded Systems Lab at Penn. His interests are in real-time scheduling algorithms for networked embedded systems with applications in automotive systems, medical devices and industrial control networks.