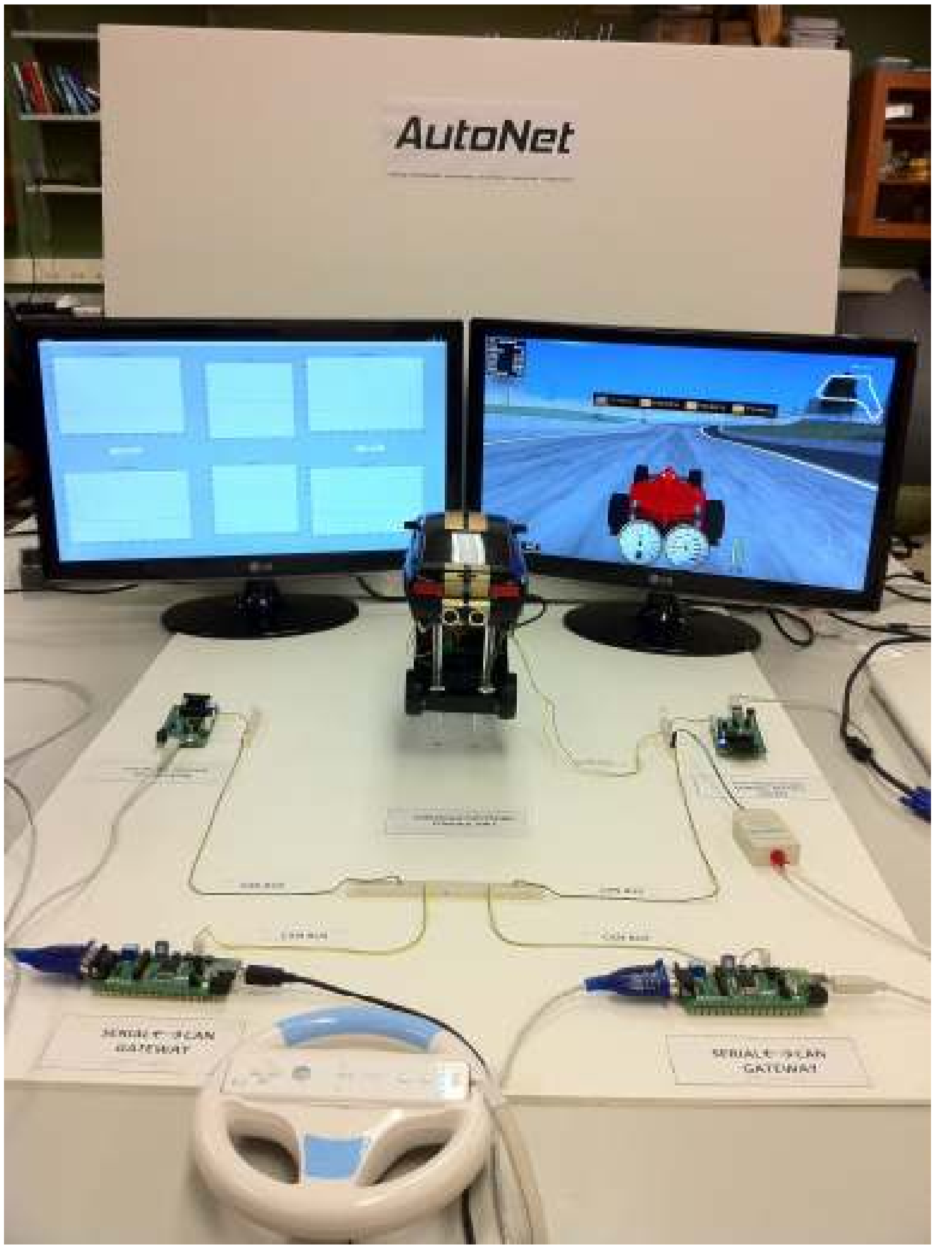
ESE 519 Final Project

AutoNet



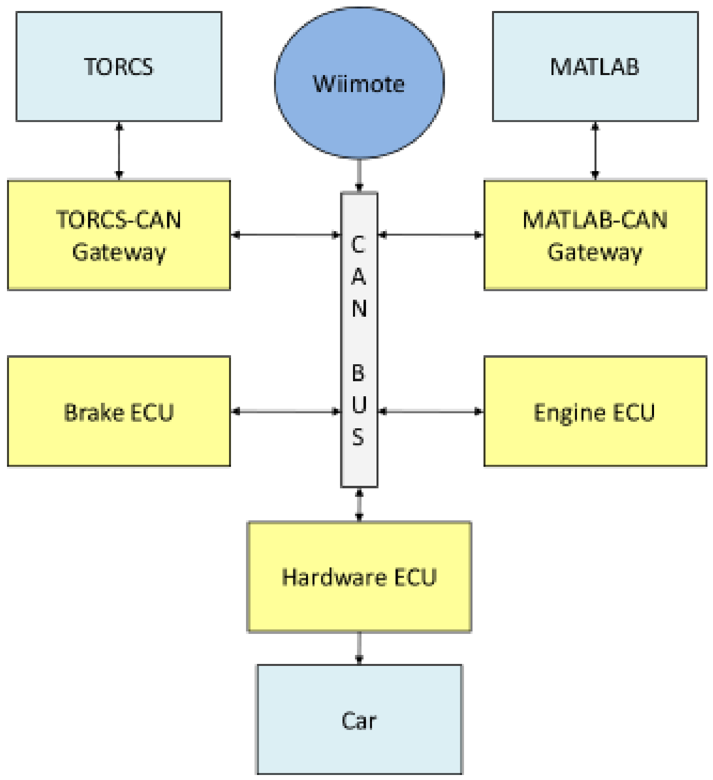
Dheeraj Chandrashekar, Harsh Mehta, Kartik Mohta, Utsav Drolia, Vijetha Kumar

Introduction

Our Project is the first step in creating an automotive ECU test bed, which can be used to develop new software processes that can improve the warranty and recall management. Ultimately, for instance we would like to have the capability to monitor / change / update tasks that are running on an ECU by creating an EVM (embedded virtual machine).

We have created a network of ECUs which can implement control algorithms like ABS & traction control. In place of a real car we have used TORCS (an open source 3D car racing simulator). The inputs are provided through a wiimote steering wheel to make it more interactive. We analyze the relevant data in MATLAB to improve the control parameters.

Architecture



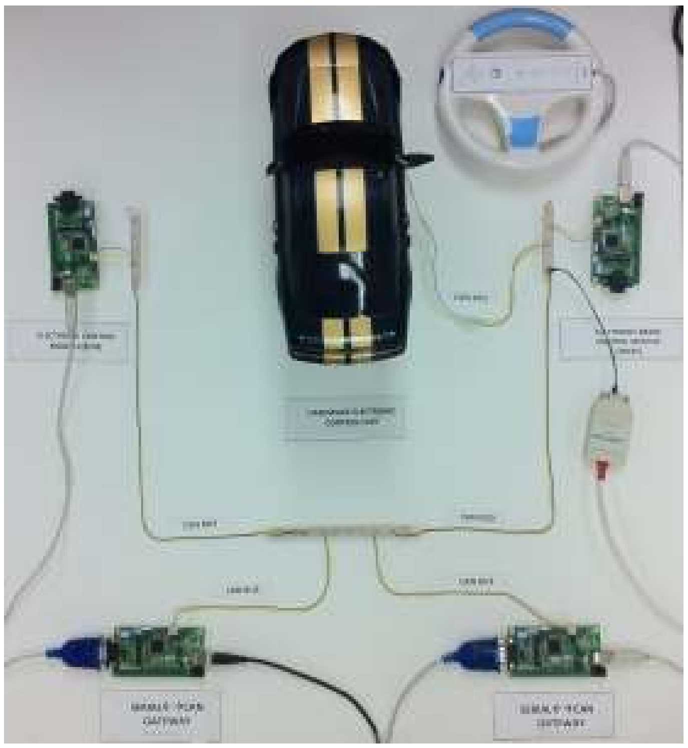


Fig 1. ECU network architecture and the actual setup

We have 5 ECUs (HCS12) connected through a CAN bus (500 Kbps). Their functions are listed below:

1. Engine control Module: Throttle Control, Auto Transmission & 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.

TORCS

TORCS (**The Open Racing Car Simulator**) is an open source 3D car racing simulator. The simulation features a simple damage model, collisions, tire and wheel properties (springs, dampers, stiffness,), aerodynamics (ground effect, spoilers,) and much more.

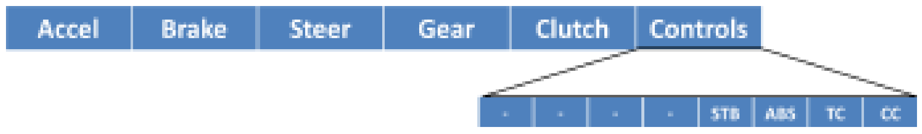
We used TORCS to get the required data like individual wheel speeds, car speed, engine rpm, current gear, yaw rate.

By default, TORCS did not allow us to control individual wheel brake pressures so we hacked into the source code to enable it.

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. The message hierarchy and the packet details are specified below

A. Driver input (CAN ID – 0x60): Acceleration, brake, steering angle, gear, clutch and controls on/off as bitmasks.



B. Brake Message (CAN ID – 0x61): Computed individual brake pressures.

Brake FL	Brake FR	Brake RL	Brake RR

C. Acceleration Message (CAN ID – 0x62): Computed acceleration, Auto transmission gear and clutch.

Accel	Gear	Clutch

D. TORCS parameters (CAN ID – 0x70): Car Speed, engine RPM, individual wheel speeds and yaw rate.

Car Speed	Engine RPM		Wheel Speed				Yaw rate
	High	Low	Front Left	Front Right	Rear Left	Rear Right	

E. Acceleration Correction message (CAN ID- 0x80): Correction to the computed acceleration by the Traction control module. (Same format as Acceleration message).

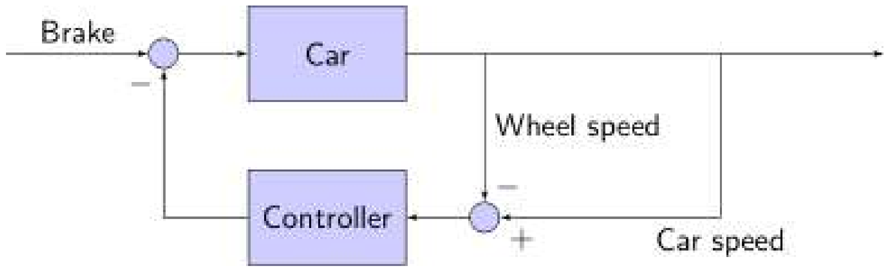
Data Plotting

The Telemetry module captures CAN messages over the bus and assembles an 18 byte packet which is sent over serial to a PC running MATLAB. The packet contains all the required data which allows analyzing and fine tune the control parameters.

CAN interrupts were used so that we do not miss any packets over the bus and a serial connection of 19200 Kbps was used with the PC.

Controls

ABS



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 below, of the wheel slip during braking show the effect of ABS on/off.

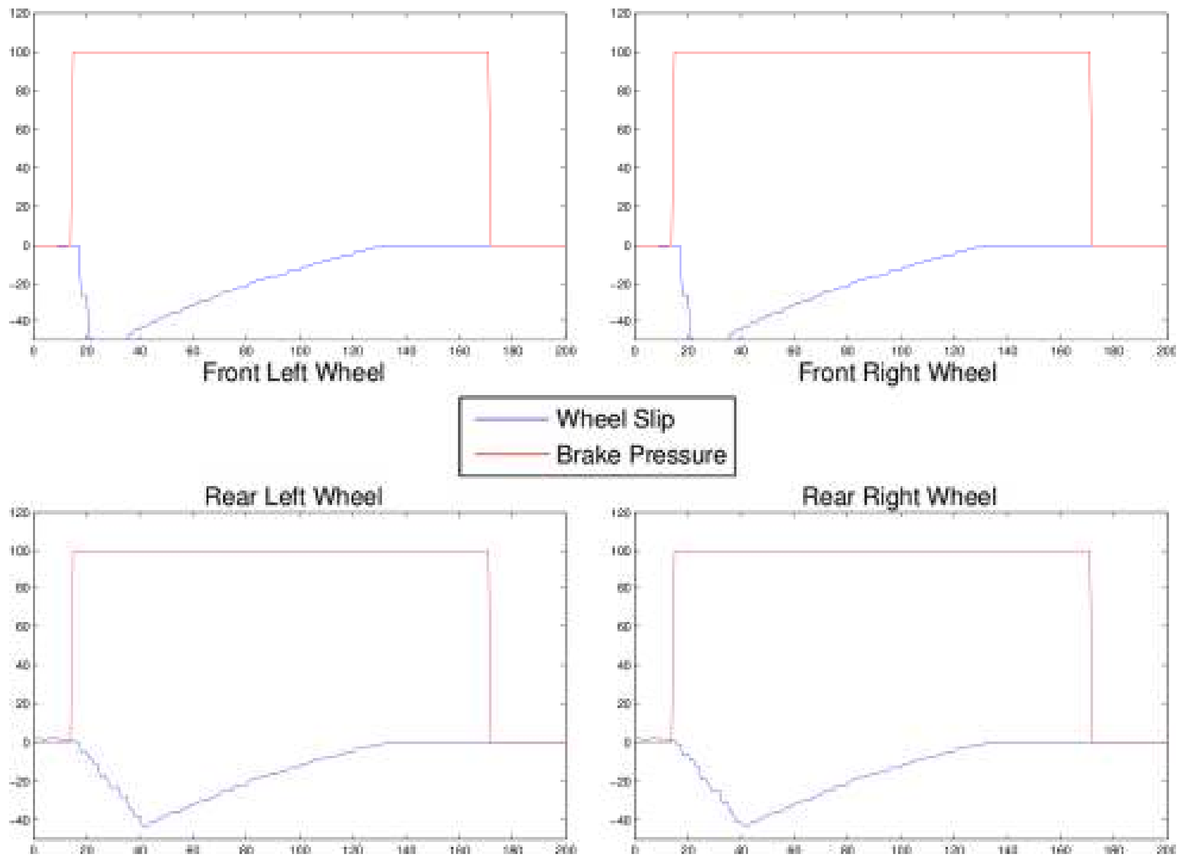


Fig 2. Braking with ABS OFF

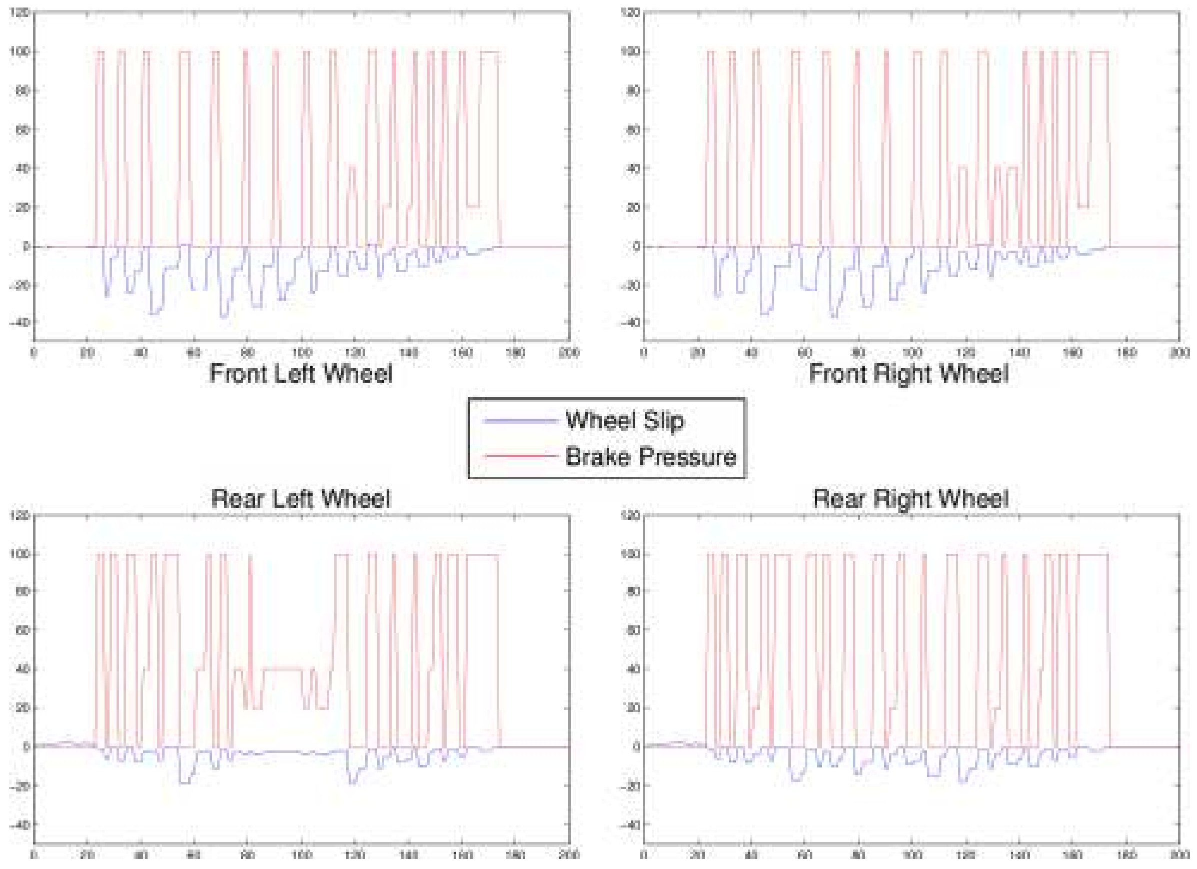
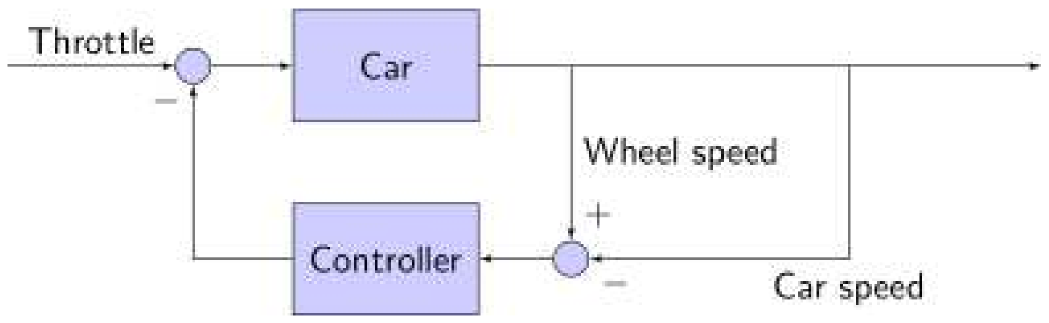


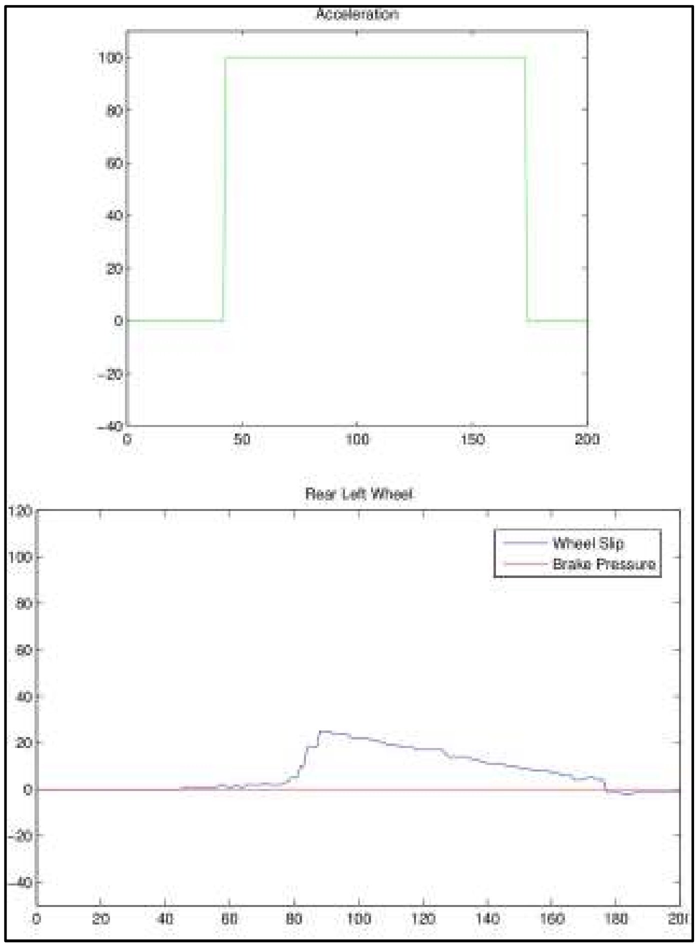
Fig 3. Braking with ABS ON

The brake pressure is clearly modulated with the ABS on, minimizing wheel slip, which allows more control during braking.

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. The plots below show the effect of Traction on and off (brake is zero and full acceleration is applied).



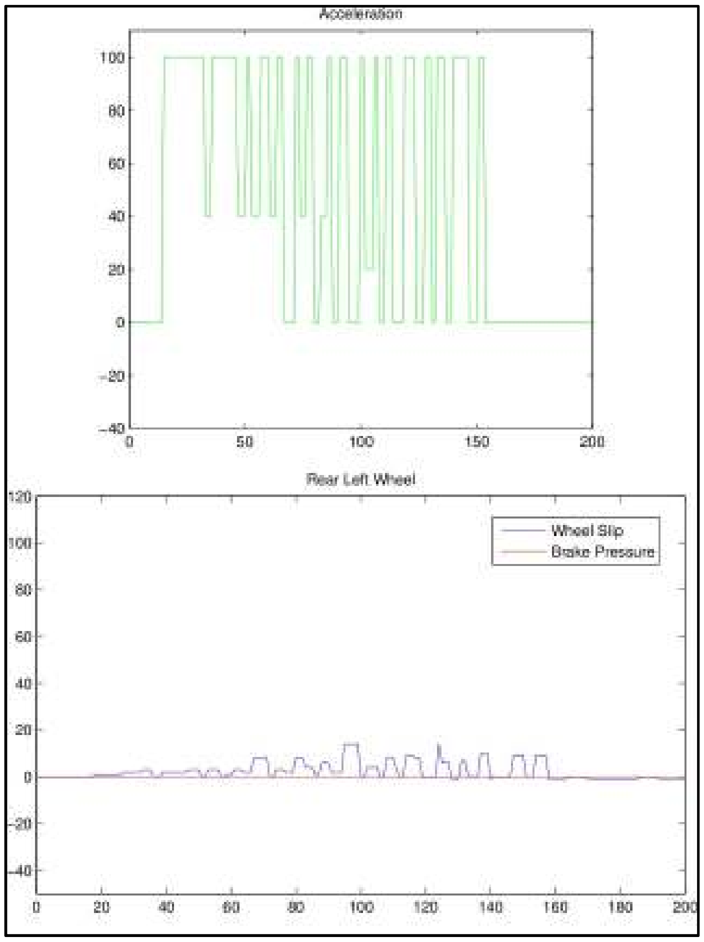
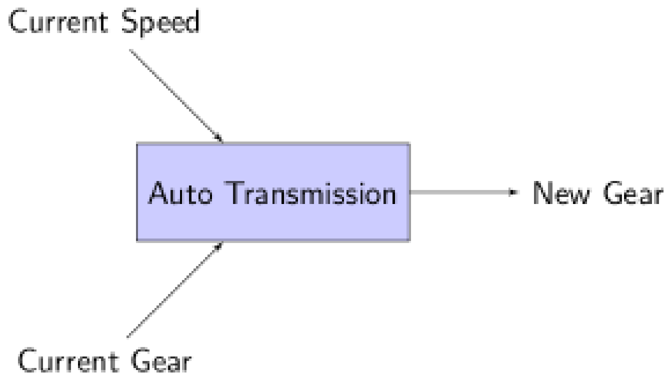
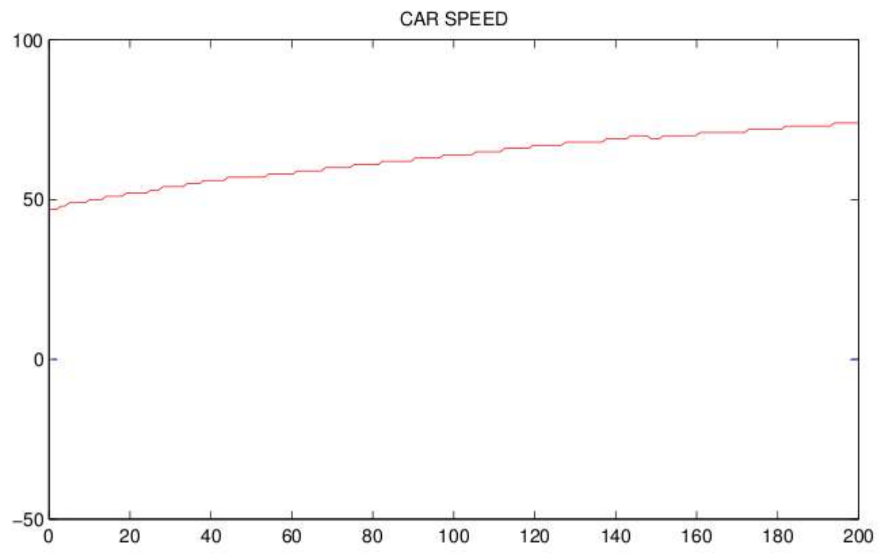


Fig 4. Traction control OFF (left) and Traction control ON (right)

Automatic transmission



We have defined a band for the engine rpm at each gear, so when the rpm exceeds the higher limit of the band we shift up and vice versa.



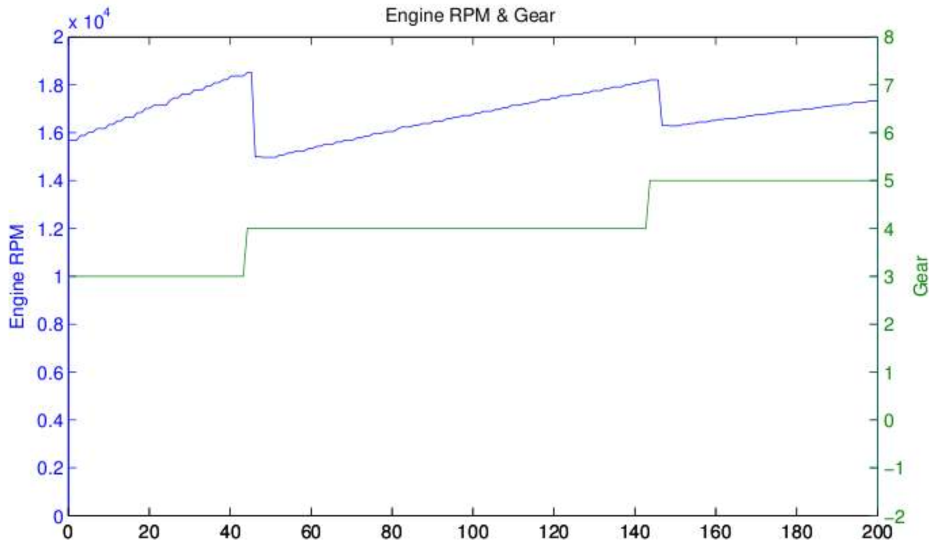
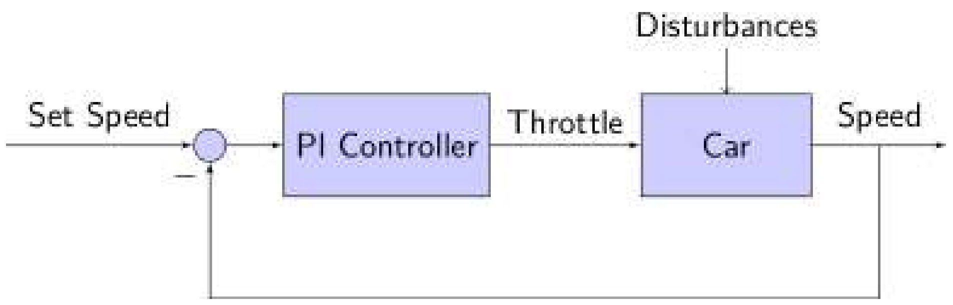


Fig 5. Automatic Transmission

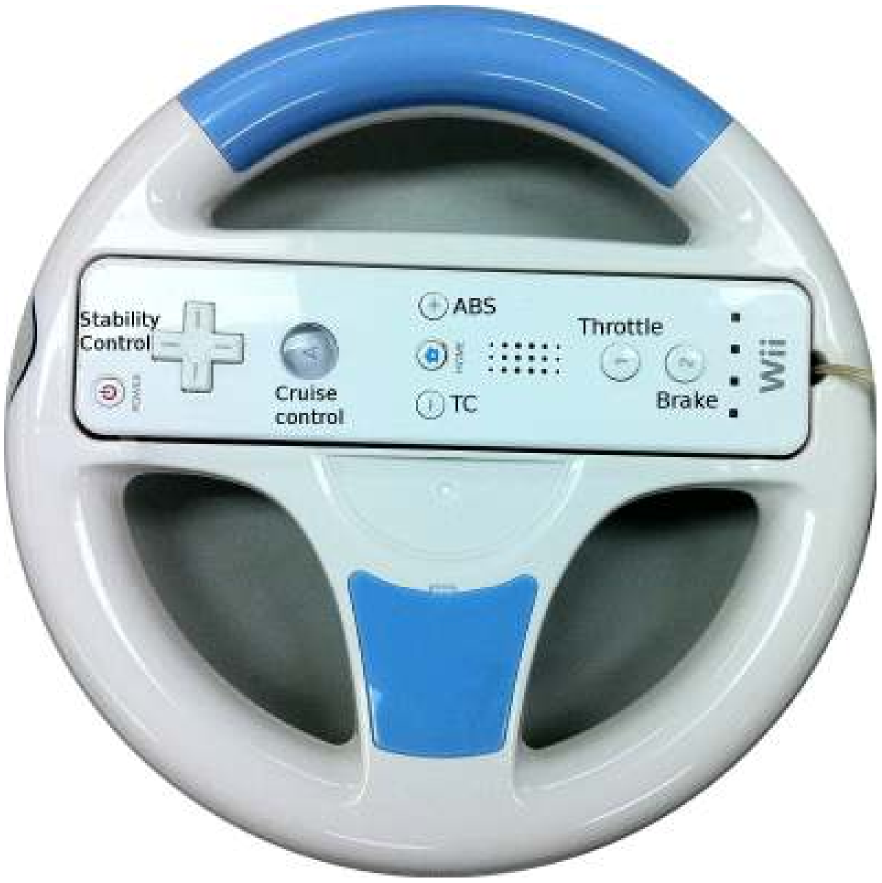
Cruise control



When the user wants the car to maintain a certain speed he can turn on cruise control when he reaches the speed. We use a proportional-integral controller with Kp = 5 and Ki = 0.05.

User Input

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



The user inputs would go over the CAN bus through a PCAN adapter. We wrote a program in C using the wiiuse and PCAN libraries to send the user inputs over the CAN bus.

SVN repository

We have created an online SVN repository which contains all our code for the project. The repository has been made public with read only access. https://www.assembla.com/spaces/ese519autonet