Embedded Control Systems

ETH – Institute for Dynamic Systems and Control
September 7 to 11 and 14 to 18, 2015

Jim Freudenberg

jfr@umich.edu

Fabian Byland

fbyland@student.ethz.ch

Stijn van Dooren

stijnva@student.ethz.ch

Marianne Schmid

marischm@ethz.ch

Schedule

- Lecture 8:00 – 10:00
  - Sampling and aliasing, signal processing, dynamic systems, integration techniques, etc.
- Assisted Pre-lab: 10:00 – 12:00
  - Material specific to the lab exercise: pulsewidth modulation, quadrature decoding, A/D conversion, etc.
  - I’ll present required information in the lecture room, then we’ll move to the lab

Important Points

- No textbook
  - `www.idsc.ethz.ch/education/lectures/embedded-control-systems.html`
  - Lecture notes, microprocessor reference material, laboratory exercises, and other important information
  - Day to day list of reference materials on website
- No required homework problems
  - Matlab, Simulink, Stateflow

Important Points

- Laboratory exercises
  - 8 laboratory exercises in 10 days using the Freescale MPC5553 microprocessor
    - Most labs are “1-day”
    - First lab will be Monday and Tuesday
    - Schedule posted
  - 33 registered students
  - 11 lab stations with 3 students (“self organize”)

Important Points

- Laboratory exercises have 3 parts:
  - Assisted Pre-lab (10AM-12PM): questions that require you to read the microprocessor reference material and gather the information required to complete the lab exercise
  - Assisted In-lab (1-4PM): the experiment
  - Post-lab (4-5PM) : questions that should reinforce what you learned in the lab exercise (due 10AM the next day)
- You must attend 8 lab sessions and hand in all 8 lab assignments (pre-, in- and post-lab) to receive credit for the course

Everyday Time Schedule

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institute for Dynamic Systems and Control
IDSC
Institut für Dynamische Systeme und Regelungstechnik

151-0593-00

Schedule Embedded Control Systems (Fall 2015)

Prof. L. Guzzella   Prof. R. D'Andrea

Lectures Week 1 in Room ML F 34 (8 to 10 a.m.) | **Lab in Room ML K31**

No.	Date	Topic	No.	Topic	10 a.m.	11	12 a.m.	1 p.m.	2	3	4	5 p.m.
1	2015-09-07	Course introduction. Start on A/D conversion, sampling and aliasing; simple anti-aliasing filter design	1	Familiarization and Digital I/O. Reading in MPC5553 Referense Manual, introduction to hardware (oscilloscope, signal generator, etc.)	Introduction			Pre-Lab		In-Lab		
2	2015-09-08	Finish A/D conversion, sampling and aliasing; simple anti-aliasing filter design; introduction to Matlab and Simulink; demonstrate Simulink by doing "Problem set 1" filter design.	1	Continue with Lab 1	In-Lab			In-Lab		Post-Lab		
3	2015-09-09	Introduction to Stateflow, in particular, demonstrate problem set 2, building a Stateflow quadrature decode model. Introduction to DC motors; derive steady-state motor equations. Present lecture material on optical encoders, quadrature decoding, over/underflow and typecasting.	2	Quadrature Decoding using the enhanced Time Processing Unit (eTPU)	Pre-Lab			In-Lab			Post-Lab	
4	2015-09-10	Discuss motor control (speed control, torque control, power amplifiers); Pulse width modulation; virtual worlds, wall "chatter" and the virtual wall.	3	Queued Analog-To-Digital Conversion (QADC)	Pre-Lab			In-Lab			Post-Lab	
5	2015-09-11	Dynamic systems and transient specifications (review); develop dynamic motor model block diagram and implement in Simulink (domonstrate problem set 3). Develop motor frequency response and demonstrate input PWM attenuation.	4	Pulse Width Modulation (PWM) and Virtual Worlds without Time	Pre-Lab			In-Lab	Post-Lab	Mathwork (location ML F 34)		

Lectures Week 2 in Room HG E 23 (8 to 10 a.m.) | **Lab in Room ML K31**

No.	Date	Topic	No.	Topic	10 a.m.	11	12 a.m.	1 p.m.	2	3	4	5 p.m.
6	2015-09-14	Develop Stateflow model of the virtual wall (demonstrate problem set 5). Develop virtual spring-mass system dynamics (harmonic oscillator). Introduce Euler Integration and pseudo-code for the spring-mass system.	5	Interrupts, Timing, and Frequency Analysis of PWM Signals	Pre-Lab			In-Lab			Post-Lab	
7	2015-09-15	Introduction to z-transforms and numerical instability. Develop the virtual spring-mass-damper (calculate how much damping is required to create a discrete harmonic oscillator using Forward Euler). Introduce state-space notation. Discuss other numerical integration methods; discuss how Matlab does numerical integration	6	Virtual Worlds with Time	Pre-Lab			In-Lab			Post-Lab	
8	2015-09-16	Software architecture, real-time operating systems and scheduling algorithms. Rapid prototyping and automatic code generation.	7	Code Generation with SIMULINK (RAppID Toolbox)	Pre-, In- & Post-Lab			Pre-, In- & Post-Lab				
9	2015-09-17	Software architecture; presentation of MathWorks on Autocode generation with SIMULINK	7	Continue with Lab 7, catch up with other labs	Pre-, In- & Post-Lab			Pre-, In- & Post-Lab				
10	2015-09-18	Introduction to CAN networks.	8	Controller Area Network	Pre-Lab			In-Lab			Post-Lab	

IMPORTANT: You must attend 8 lab sessions and hand in all 8 assignments (pre-, in- and post-lab) to receive credit for the course. Pre-labs are due at the start of the In-labs, Post-labs are due at 5 p.m.

What is an Embedded System?

- Technology containing a microprocessor as a component
  - cell phone
  - digital camera
- Constraints not found in desktop applications
  - cost
  - power
  - memory
  - user interface

⇒ Embedded processor is often the performance and cost limiting component!

What is an Embedded *Control* System?

- Technology containing a microprocessor as a component used for control:
  - automobiles
  - aircraft
  - household appliances
  - copy machines
  - wind turbines
  - hospital beds
  - laser printers
  - civil structures
  - manufacturing
  - energy harvesters
  - medical devices

Characteristics of Embedded Control Systems

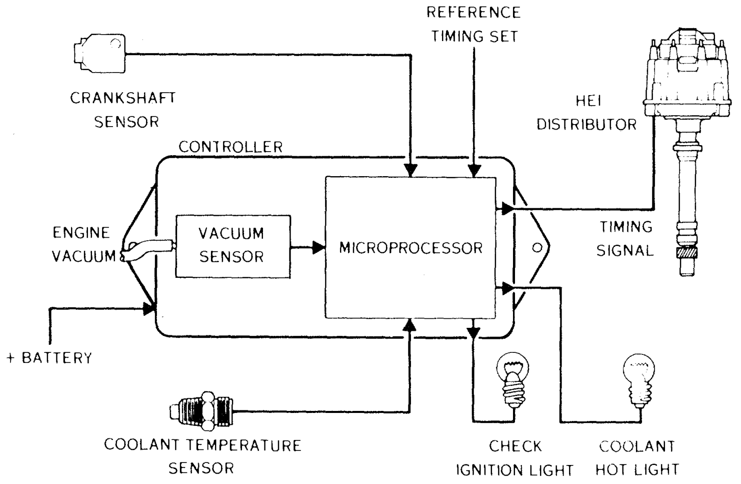
- Interface with external environment
  - sensors and actuators
- Real time critical
  - performance and safety
  - embedded software must execute in synchrony with physical system
- Hybrid behavior
  - continuous dynamics
  - state machines
- Distributed control
  - networks of embedded microprocessors

Prime Example: today's automobile!

The Automobile in 1977

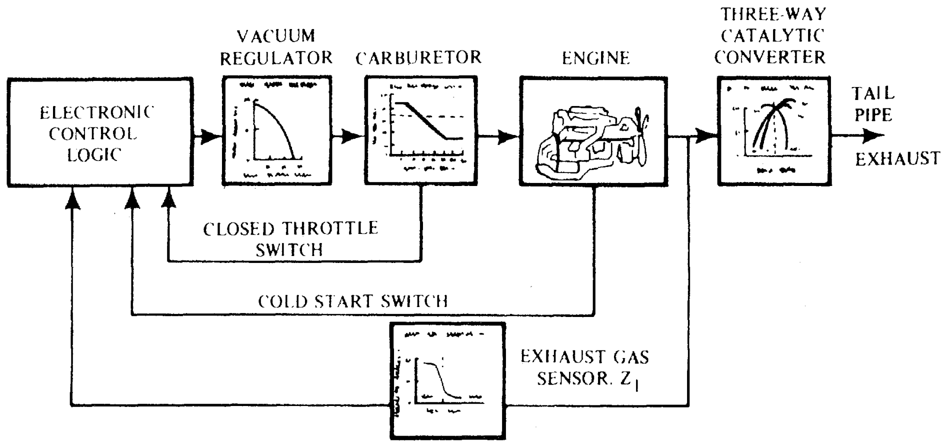
16 electrical systems

- spark timing
- air/fuel



1976 Chrysler

- analog control



1977 GM Olds Toronado
1978 Ford Lincoln Versailles

- microprocessor control

IEEE Spectrum special issue on the Automobile, Nov 1977

The Future in 1977

Gas turbine engines

>100 proposed electrical systems

High end automobiles: as many as 8 microprocessors, one per cylinder (Aston Martin)

10K ROM: plenty unused capacity to control other engine functions

Obstacles:

high cost of sensors and actuators

“the inability of the electrical engineer to characterize the mechanical system for microprocessor programmers”

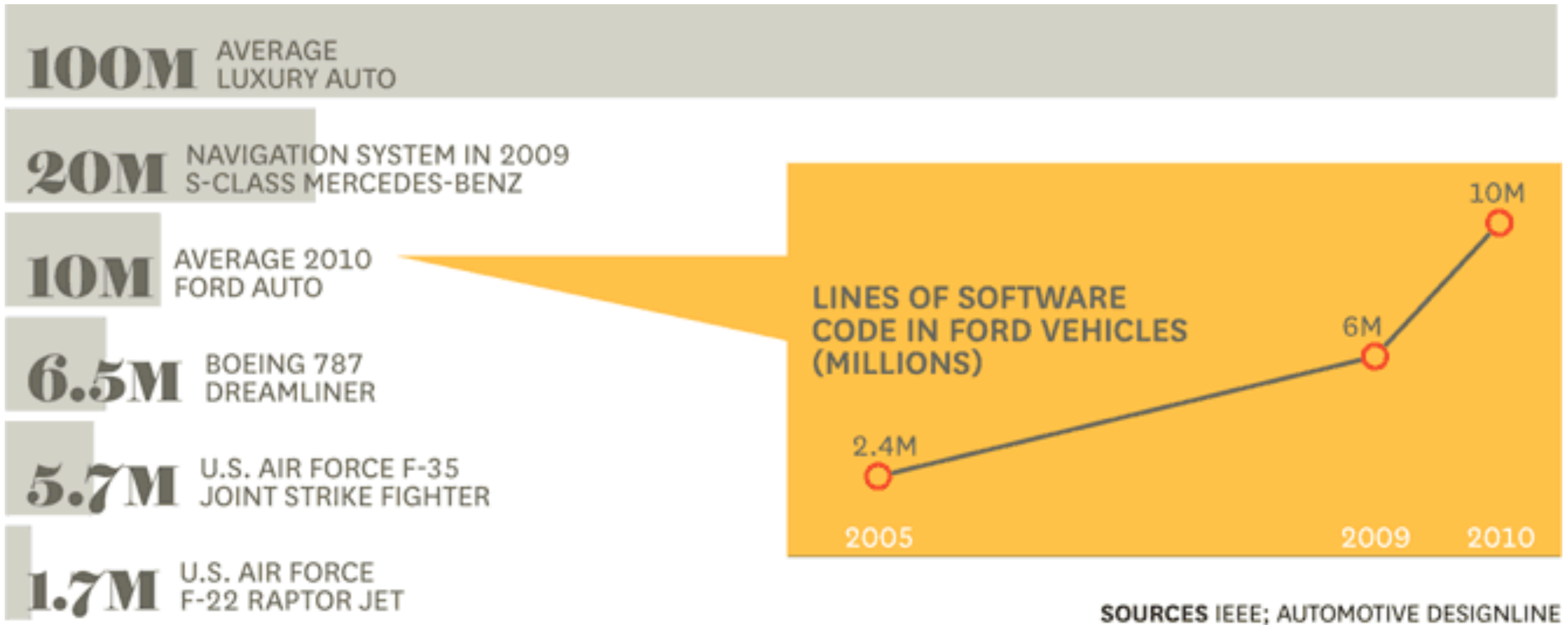
IEEE Spectrum special issue on the Automobile, Nov 1977

The Automobile in 2015

- Drivetrain
  - Variable geometry turbochargers
  - Variable cam timing (intake, exhaust, dual-equal, dual independent)
  - Variable valve timing
  - Variable compression ratio
  - Automatic transmission, continuously variable tranmission
- Chassis control
  - antilock brakes
  - traction control
  - stability control
- Body control
  - seats
  - windows
  - wipers
  - locks
- Infotainment/GPS systems
- Driver assistance & active safety systems

➔ Cars today are safer, less polluting, more fuel efficient, and more c

The Automobile in 2010



Harvard Business Review, 2010

Industry Hiring Needs

- "The auto industry is … hiring a different breed of engineer [to] invent the next generation of complex software [for] m.p.g., clean emissions and crash avoidance technologies."*
- "GM's biggest engineering recruiting challenges are software and controls engineering"*
- Ford: greatest hiring need is for software and electronics skills**
- 2012 SAE salary survey***:
  EEs working in automotive sector earn *$10K/year* more than MEs

*Detroit Free Press, October 2012
** USA Today, July 2013
*** www.sae.org/membership/salarysurvey/

An Industry Request: 1998

Dr. Ken Butts:
- Ford Research (currently Toyota)
- Founding member, MATHWORKS Automotive Advisory Board (1998)

"Why can't I hire students trained to do embedded control software development?"

"And why don't the students I hire know how to talk to one another?"

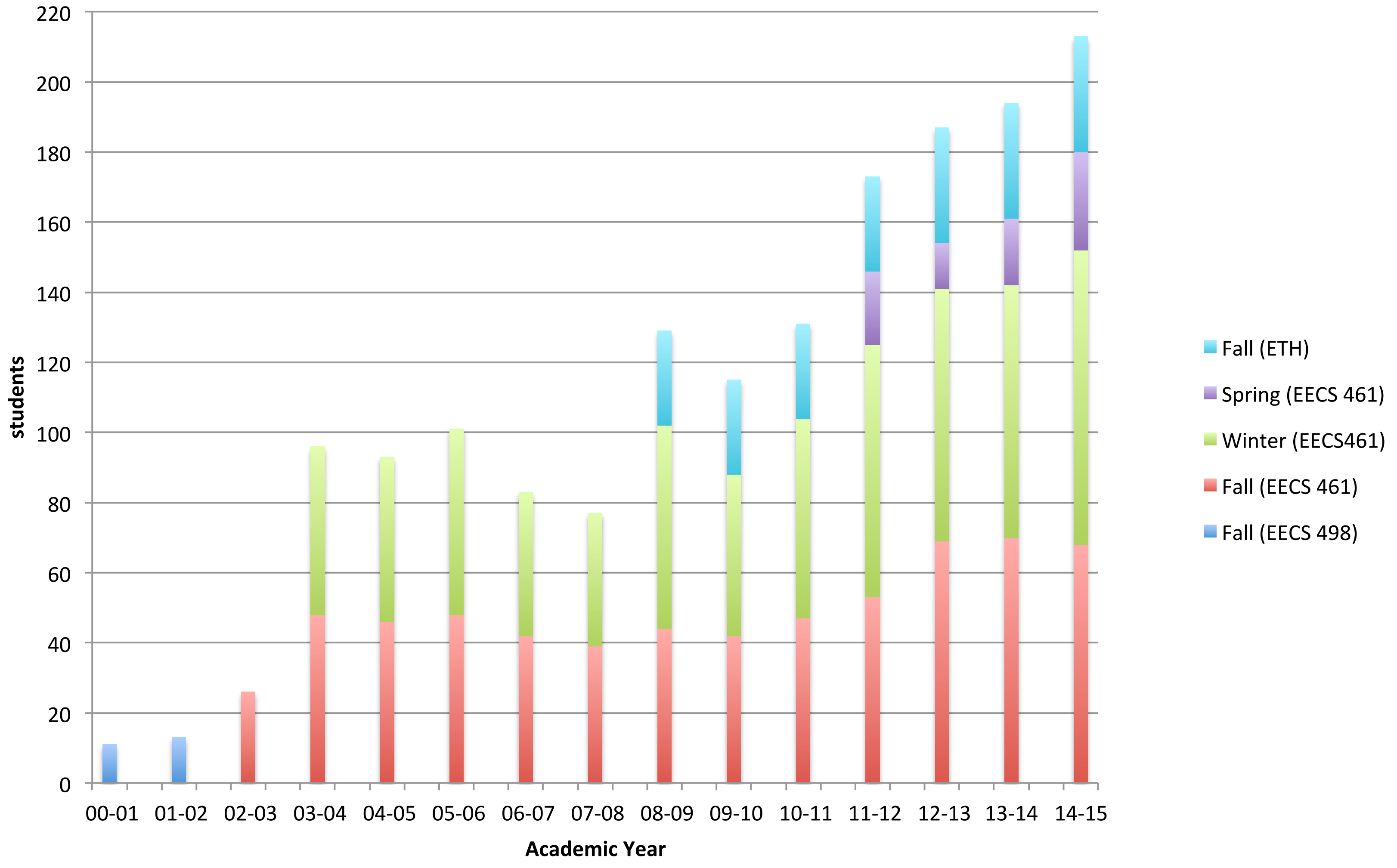
Skills required:
- Control algorithms
- Computer software
- Computer hardware
- Electronics
- Mechanical engineering

Outcome: Two Courses

- UofMichigan: EECS 461, Embedded Control Systems
  - 16th year
  - ~ 200 students/year
  - Jeff Cook, formerly Ford Research
  - Student body:
    - EE and CE, seniors and masters
    - Space permitting, grad students from other departments

- ETH Zurich: 151-0593-00, Embedded Control Systems
  - 8th year as two week block course
  - 33 students/year
  - Mechanical Engineering Graduate Students

Embedded Control Enrollment: UM and ETH



Total Enrollment: 1618

Laboratory Overview

- MPC5553 Microcontroller (Freescale)

- Development Environment
  - Debugger (P&E Micro)
  - Codewarrior C compiler (Freescale)

- Haptic Interface
  - Force feedback system for human/computer interaction

- Rapid Prototyping Tools
  - Matlab/Simulink/Stateflow, Embedded Coder (The Mathworks)
  - RAppID Toolbox (Freescale)

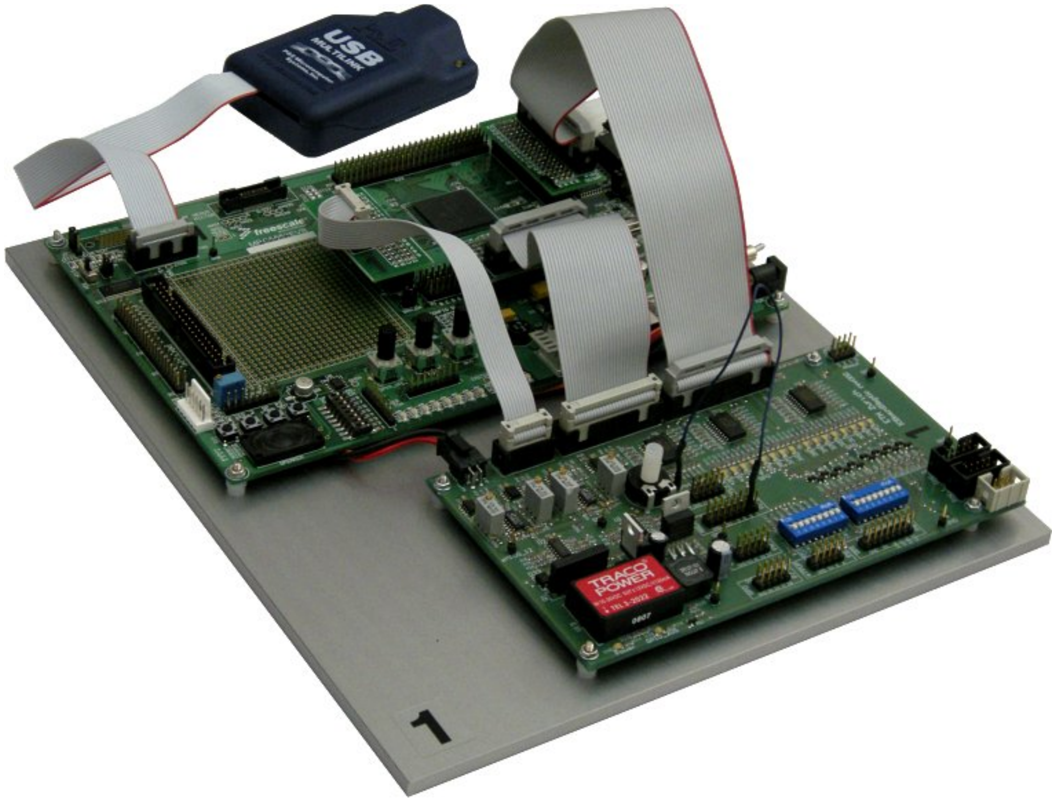
Freescale MPC5553 Microcontroller

- 32 bit PPC core
  - floating point
  - 132 MHz
  - -40 to +125 °C temperature range
- Programmable Time Processing Unit (eTPU)
  - Additional, special purpose processor handles I/O that would otherwise require CPU interrupt service (or separate chip)
  - Quadrature decoding
  - Pulse Width Modulation
- Control Area Networking (CAN) modules
- 2nd member of the MPC55xx family
  - real time control requiring computationally complex algorithms
  - MPC5554 replaces MPC555 for powertrain control
  - MPC5553 has on-chip Ethernet for manufacturing applications

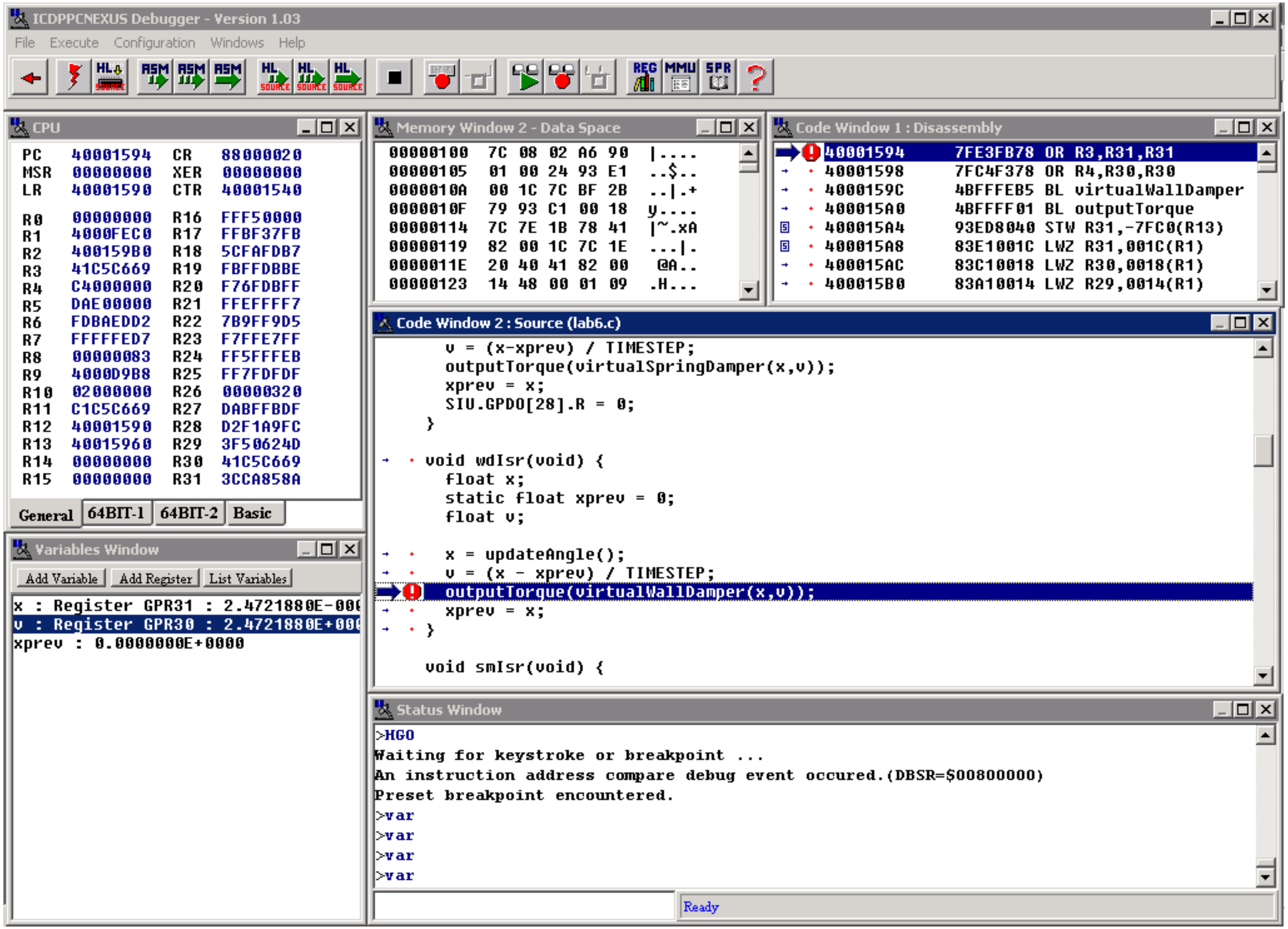
MPC5553 EVB

- Evaluation board (Freescale)
  - 32 bit PPC core
  - floating point
  - 128 MHz

- Interface board (UofM)
  - buffering
  - dipswitches
  - LEDs
  - rotary potentiometer



Nexus Compliant Debugger (P&E Micro)



ICDPPCNEXUS Debugger - Version 1.03
File Execute Configuration Windows Help
REG MMU SPR
CPU
Memory Window 2 - Data Space
Code Window 1 : Disassembly
Code Window 2 : Source (lab6.c)
General 64BIT-1 64BIT-2 Basic
Variables Window
Add Variable Add Register List Variables
Status Window
Ready

```
PC   40001594   CR   88000020
MSR  00000000   XER  00000000
LR   40001590   CTR  40001540

R0   00000000   R16  FFF50000
R1   4000FEC0   R17  FFBF37FB
R2   400159B0   R18  5CFAFDB7
R3   41C5C669   R19  FBFFDBBE
R4   C4000000   R20  F76FDBFF
R5   DAE00000   R21  FFEFFFF7
R6   FDBAEDD2   R22  7B9FF9D5
R7   FFFFFED7   R23  F7FFE7FF
R8   00000083   R24  FF5FFFEB
R9   4000D9B8   R25  FF7FDFDF
R10  02000000   R26  00000320
R11  C1C5C669   R27  DABFFBDF
R12  40001590   R28  D2F1A9FC
R13  40015960   R29  3F50624D
R14  00000000   R30  41C5C669
R15  00000000   R31  3CCA858A
```

```
00000100  7C 08 02 A6 90   |....
00000105  01 00 24 93 E1   ..$..
0000010A  00 1C 7C BF 2B   ..|.+
0000010F  79 93 C1 00 18   y....
00000114  7C 7E 1B 78 41   |~.xA
00000119  82 00 1C 7C 1E   ...|.
0000011E  20 40 41 82 00    @A..
00000123  14 48 00 01 09   .H...
```

```
40001594   7FE3FB78 OR R3,R31,R31
40001598   7FC4F378 OR R4,R30,R30
4000159C   4BFFFEB5 BL virtualWallDamper
400015A0   4BFFFF01 BL outputTorque
400015A4   93ED8040 STW R31,-7FC0(R13)
400015A8   83E1001C LWZ R31,001C(R1)
400015AC   83C10018 LWZ R30,0018(R1)
400015B0   83A10014 LWZ R29,0014(R1)
```

```
    v = (x-xprev) / TIMESTEP;
    outputTorque(virtualSpringDamper(x,v));
    xprev = x;
    SIU.GPDO[28].R = 0;
  }

  void wdIsr(void) {
    float x;
    static float xprev = 0;
    float v;

    x = updateAngle();
    v = (x - xprev) / TIMESTEP;
    outputTorque(virtualWallDamper(x,v));
    xprev = x;
  }

  void smIsr(void) {
```

```
x : Register GPR31 : 2.4721880E-000
v : Register GPR30 : 2.4721880E+000
xprev : 0.0000000E+0000
```

```
>HGO
Waiting for keystroke or breakpoint ...
An instruction address compare debug event occured.(DBSR=$00800000)
Preset breakpoint encountered.
>var
>var
>var
>var
```

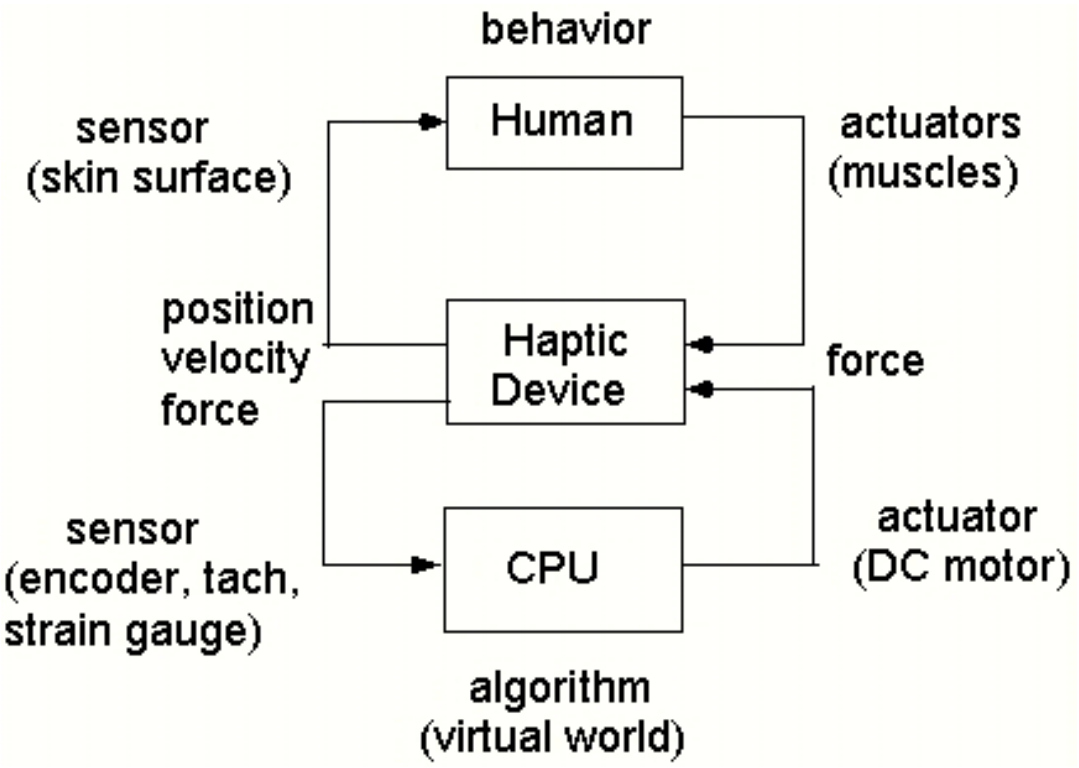
Haptic Interface

- Enables human/computer interaction through sense of touch
  - force feedback
  - virtual reality simulators (flight, driving)
  - training (surgery, assembly)
  - teleoperation (manufacturing, surgery)
  - X-by-wire cars

- Human visual sensor: 30 Hz
- Human haptic sensor: 500Hz-1kHz

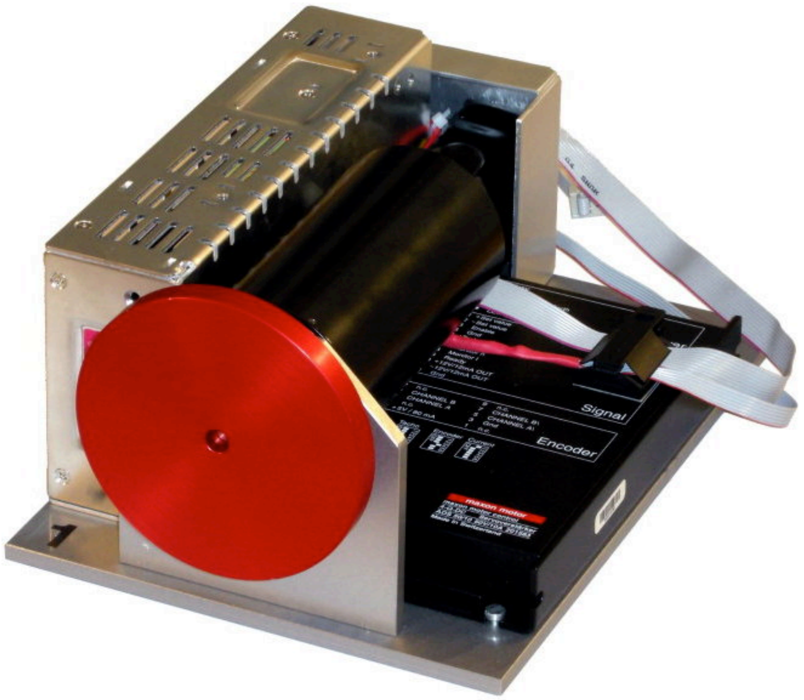
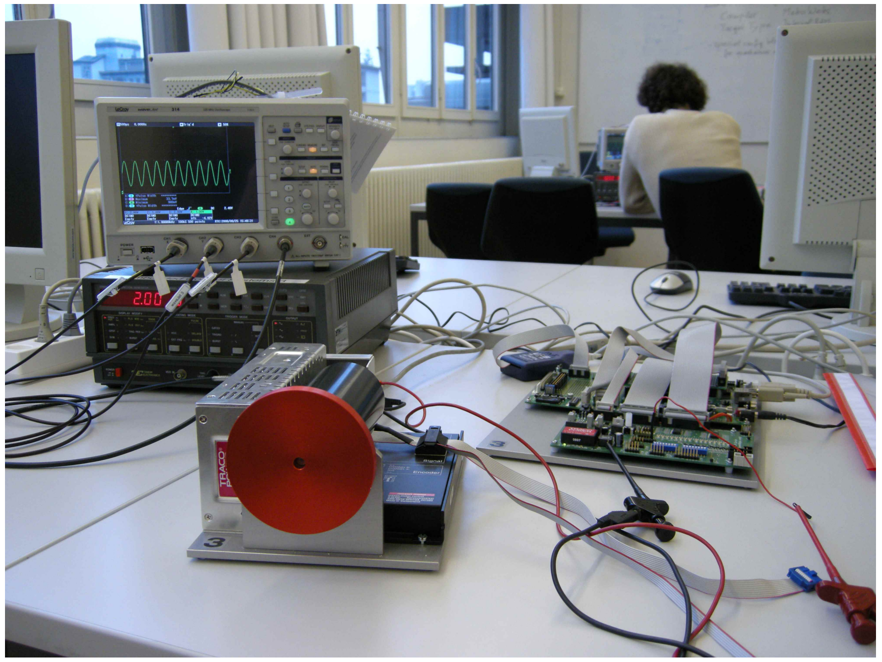
- Ideal pedagogical tool….
  - student satisfaction
  - virtual reality algorithms easy to understand
  - tricky to get right

Force Feedback



Haptic Wheel & Lab Station

- Haptic Interface
  - DC motor
  - PWM amplifier w/ current controller
  - optical encoder
  - 3rd generation

Lectures (I)

- Quantization
- Sampling
- Linear filtering
- Quadrature decoding
- DC motors
- Pulse Width Modulation (PWM) amplifiers
- Motor control: current (torque) vs. speed
- MPC5553 architecture. Peripherals: eMIOS, eTPU…
- Haptic interfaces.
  - virtual wall
  - virtual spring/mass/damper
- Simulink/Stateflow modeling of hybrid dynamical systems
- Numerical integration.

Lectures (II)

- Networking:
  - Control Area Network (CAN) protocol.
  - Distributed control
- Interrupt routines: timing and shared data
- Software architecture
  - Round robin
  - Round robin with interrupts
  - Real time operating systems (RTOS)
  - Multitasking
- Shared data: semaphores, priority inheritance, priority ceiling
- Real time computation. Rate monotonic scheduling.
- Rapid prototyping. Autocode generation.
- Model based embedded control software development
- PID control design

Laboratory Exercises

- Each teaches
  - a peripheral on the MPC5553
  - a signals and systems concept
  - Labs 1-6, 8: program in C
  - Lab 7: autocode generation
  - Each lab reuses concepts (and code!) from the previous labs

- Lab 1: Familiarization and digital I/O
- Lab 2: Quadrature decoding using the eTimer
- Lab 3: Queued A-D conversion
- Lab 4: Pulse Width Modulation and simple virtual worlds
- Lab 5: Interrupt timing and frequency analysis of PWM signals
- Lab 6: Virtual worlds with dynamics
- Lab 7: Rapid Prototyping
- Lab 8: Controller Area Network (CAN)

Lab 1: Familiarization and Digital I/O

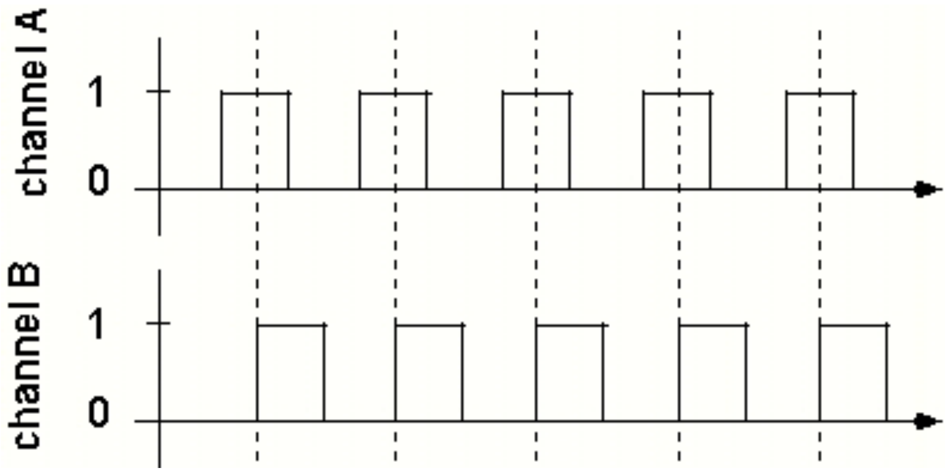
- Use General Purpose Input/Output (GPIO) on MPC5553
- Use "union" command to write C code to perform bit manipulations
- Read two 4-bit numbers set by dipswitches
  - add and display on LEDS
- Write C header file to access various bits in a 16 bit register:

```
typedef union SIU_CONFIG_UNION {
    /* This allows access to all 16-bits in the register */
    unsigned short REG;
    /* This structure allows access to the individual bytes of the register */
    struct {
        unsigned short UPPER:8; /* access to the top 8 bits */
        unsigned short LOWER:8; /* access to the bottom 8 bits */
    } BYTE;
    /* This structure splits apart the different fields of the register */
   struct {
        unsigned short :2; /* indicates 2 unused bits in the register */
        unsigned short FIELD1:8; /* access to the 8-bit field named FIELD1 */
        unsigned short FIELD2:6; /* access to the next 6-bit field */
   } FIELDS;
} EXAMPLE_REGISTER;
```

- Remaining labs use Freescale supplied header files

Lab 2: Quadrature Decoding

- Optical encoder attached to motor generates two 90° out of phase square waves:



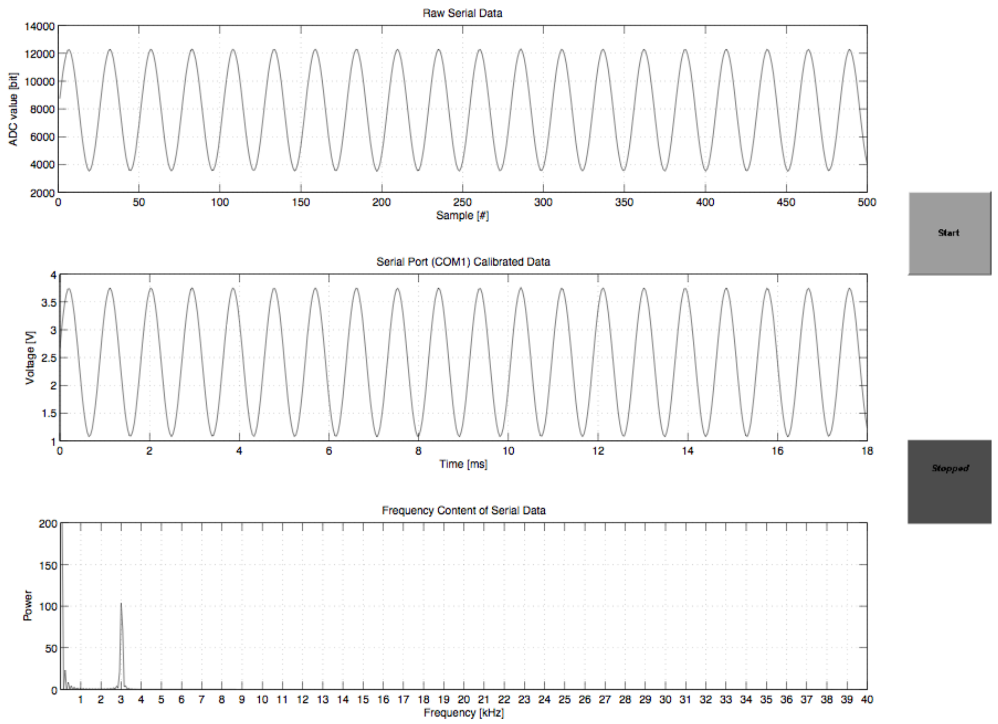
- QD function on MPC5553 eTPU:

  decodes quadrature signal into counter
- CPU must read counter before overflow

<u>Issue</u>: How fast can wheel turn before counter overflows?

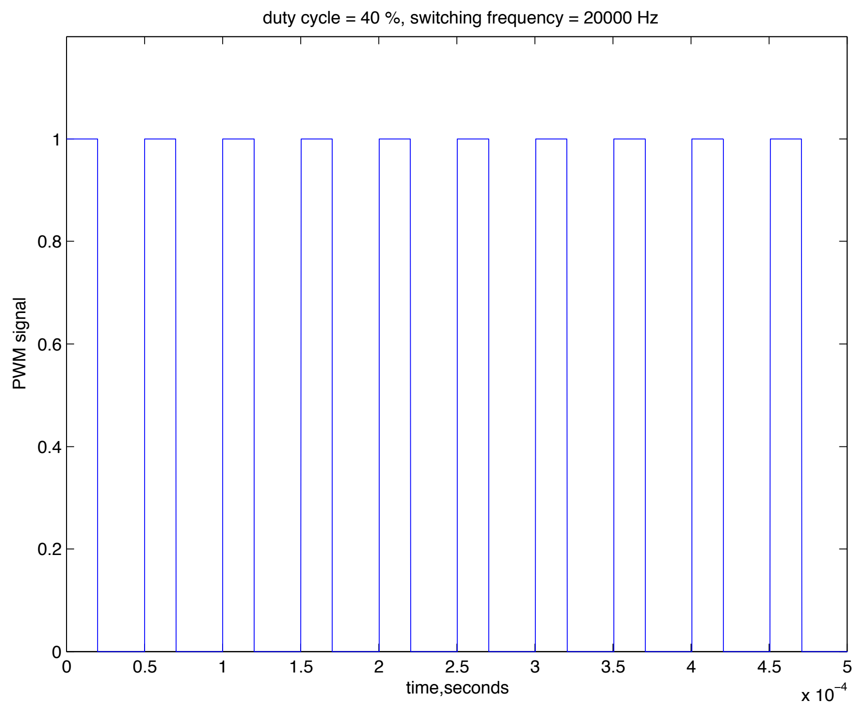
Lab 3: A/D Conversion

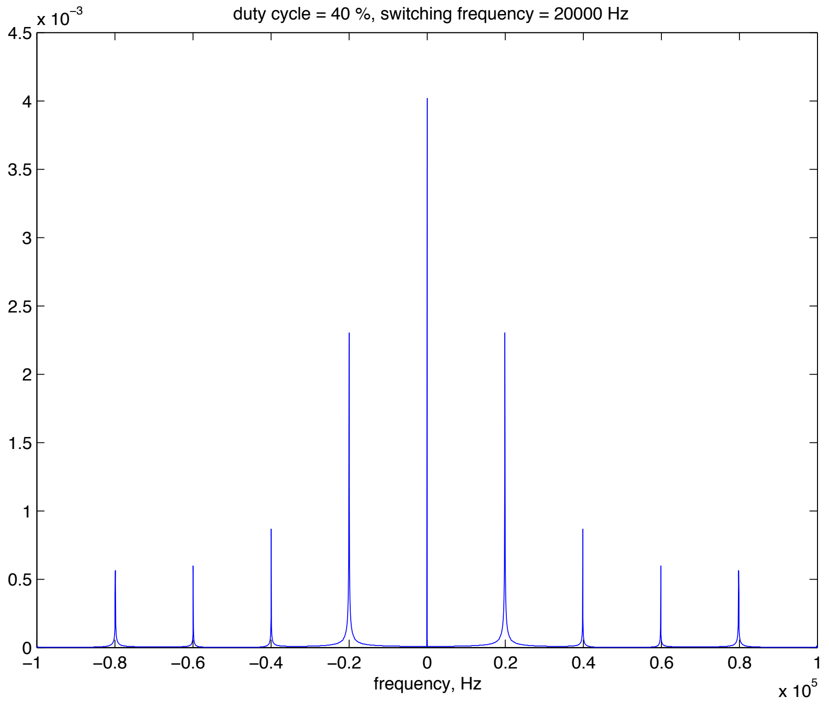
- Uses QADC on the MPC5553
- Acquire analog input from potentiometer or signal generator
- Measure time required for one conversion by toggling bit
- Investigate aliasing
- Software oscilloscope:



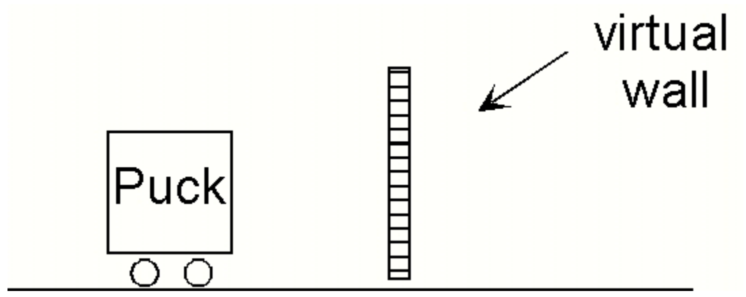
Lab 4: Pulse Width Modulation

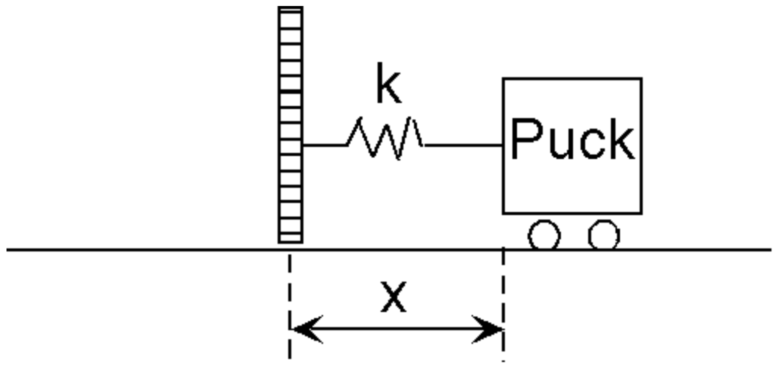
- Drive DC motor with a PWM signal
  - Switching frequency 20 kHz
  - Duty cycle 40%
  - eMIOS peripheral on MPC5553





Lab 4: Virtual Wall





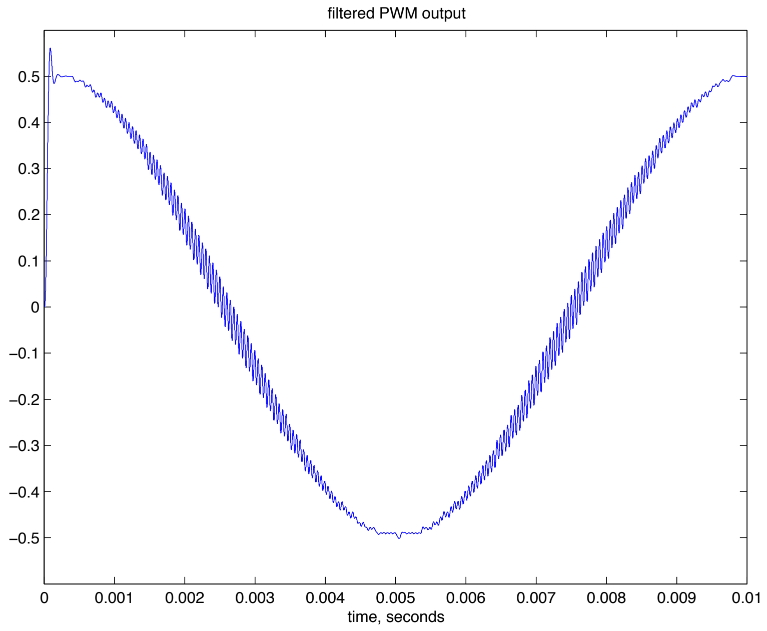
- Software loop
  - read position from encoder
  - compute force *F = 0* or *F = kx*
  - set PWM duty cycle
- Rotary motion
  - degrees ⇔ encoder count
  - torque ⇔ PWM duty cycle
  - 1 degree into wall ⇔ 400 N-mm torque

- Wall chatter
  - large *k* required to make stiff wall
  - limit cycle due to sampling and quantization

Lab 5: Interrupt Timing and PWM Frequency Analysis

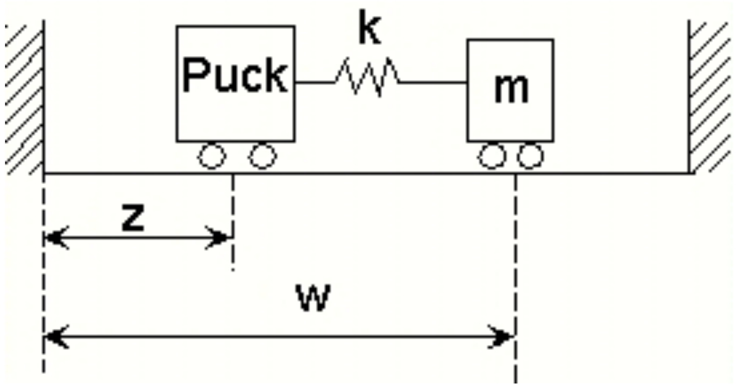
- Use interrupt timer to generate a time step for numerical differentiation and integration
- Periodically modulate duty cycle of a 20kHz PWM signal by writing an ISR that either
  - Samples 100 hz sine wave.
  - Calls C sine function
  - Uses lookup table
- Time ISR by toggling a bit
- Filter PWM signal to remove

20kHz switching frequency.

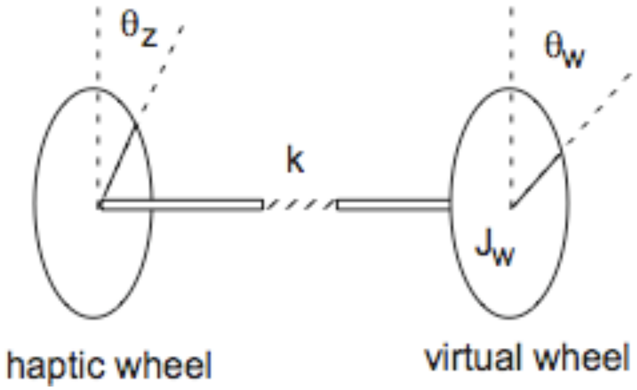


Lab 6: Virtual Spring-Mass System

- Virtual spring-mass system: reaction force $F = k(w-z)$
- Measure z, must obtain w by numerical integration
- Use interrupt timer to generate a time step



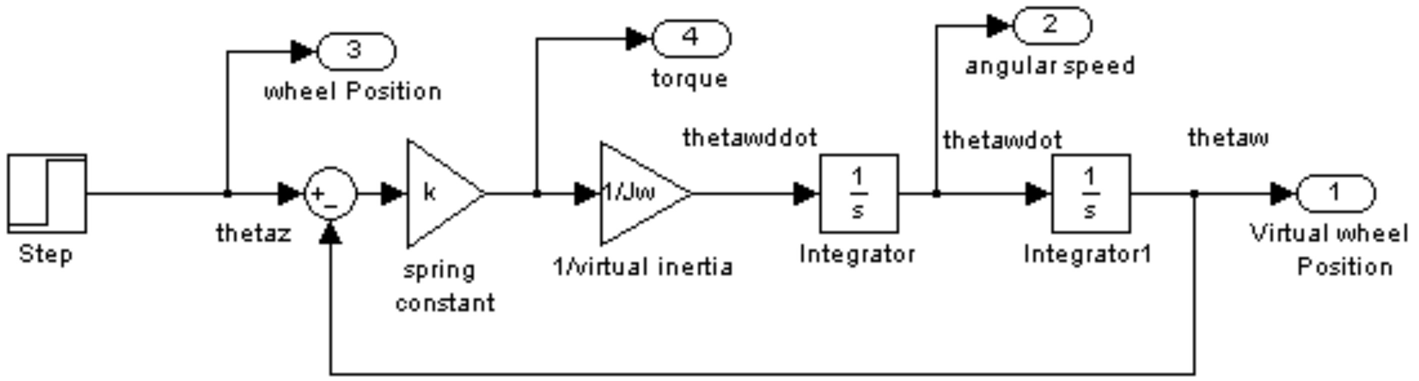
$$\ddot{w} + \frac{k}{m} w = \frac{k}{m} z$$



$$\ddot{\theta}_w + \frac{k}{J_w} \theta_w = \frac{k}{J_w} \theta_z$$

Design Specifications

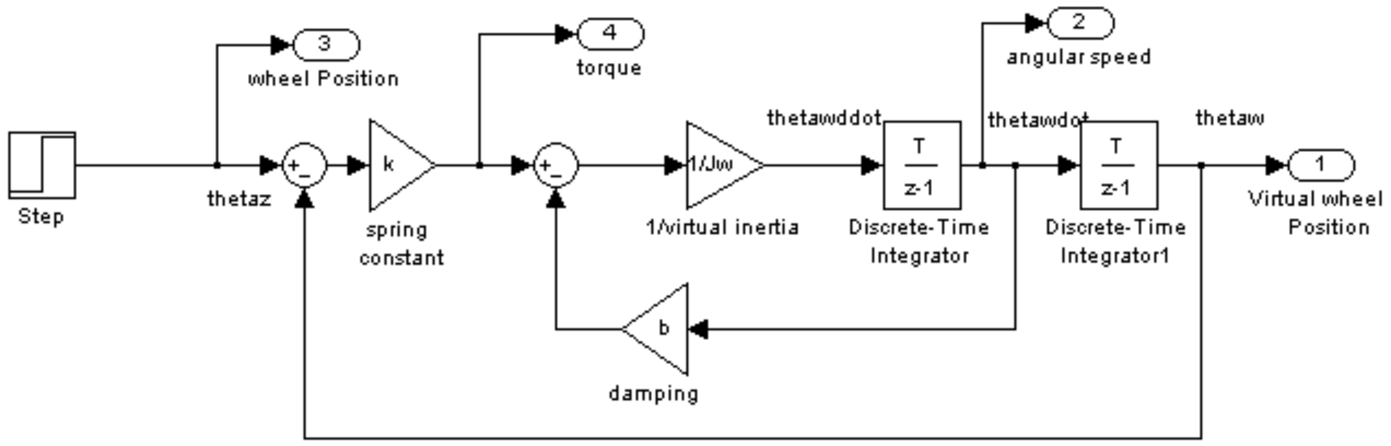
- Choose k and J_w so that
  - virtual wheel oscillates at 1Hz
  - maximum torque in response to 45 degree step in wheel position is < 800Nmm



- Verify design in Simulink before testing on hardware

Numerical Integration

- Forward Euler:
  - easy to program in real time
  - no direct feedthrough, no algebraic loops
  - numerically unstable!

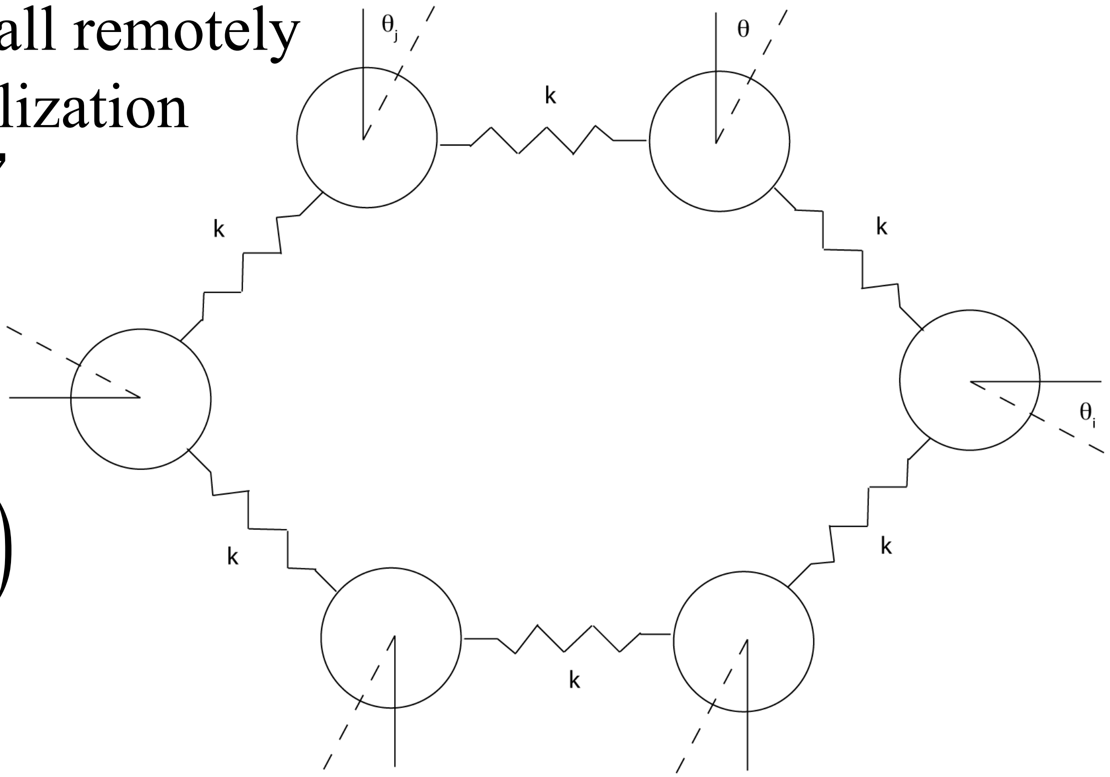


- Question: Can we restore stability by adding *virtual damping*?

  Yes! *Can compute b mathematically.*

Lab 8: Controller Area Networking (CAN)

- Networking protocol used in time-critical applications
- Messages have unique identifiers: priorities
- Allows computation of worst case response time
- Lab exercises:
  - implement virtual wall remotely
  - estimate network utilization
  - virtual “daisy chain”



$$T = k\left(\theta_i - \theta\right) + k\left(\theta_j - \theta\right)$$

Autocode Generation (I)

- Derive a mathematical model of system to be controlled
- Develop a Simulink/Stateflow model of the system.
- Design and test a control algorithm using this model.
- Use Simulink Coder to generate C-code.
- Eliminates coding errors.

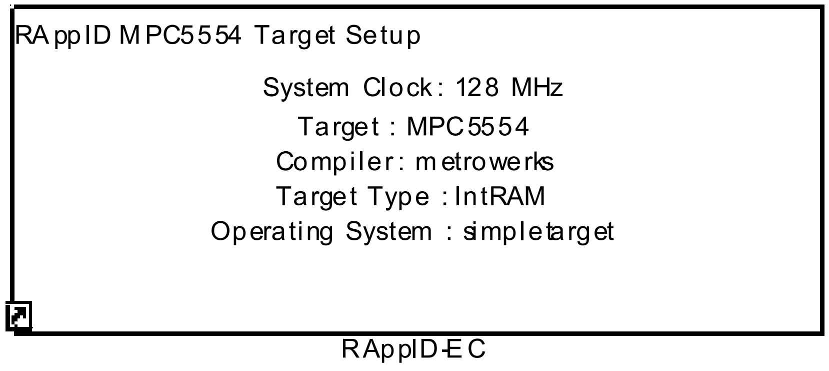
- **Rapid prototyping**: Speeds product development as generated code can be tested in many design cycles
- Autocode in production:
  - Nonconsumer market: NASA, aerospace
  - Automotive: body control
    powertrain control??

Autocode Generation (II)

- Need Simulink blocks:
  - device drivers
  - processor and peripheral initialization
- Issues:
  - efficiency of generated code
  - structure of code
- Multitasking
  - with RTOS, task states
  - without RTOS, nested interrupts

RAppID Toolbox (Freescale)

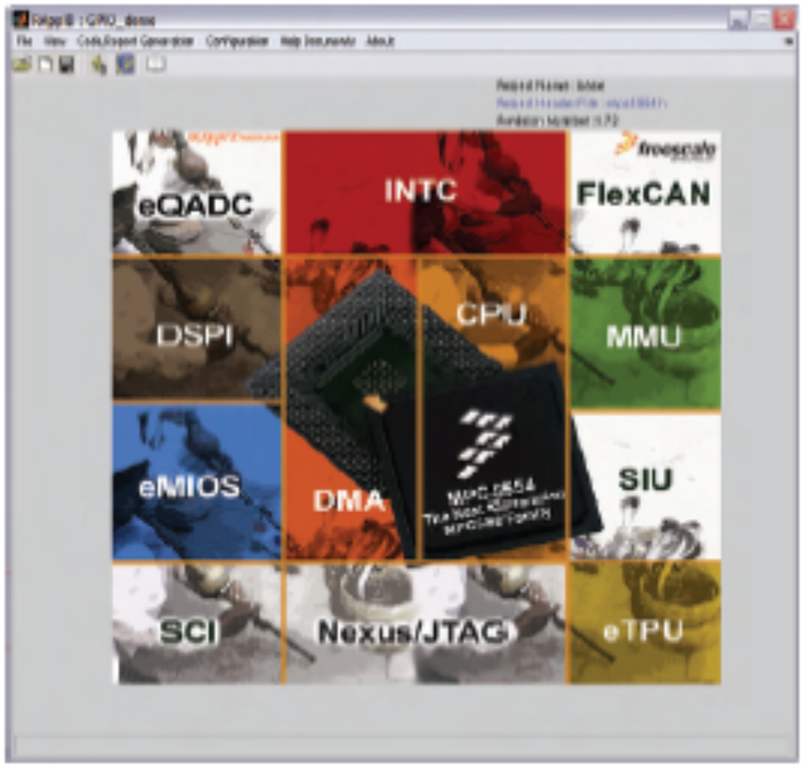
- Processor and peripheral initialization blocks
- Device driver blocks
- Enables multitasking with nested interrupts



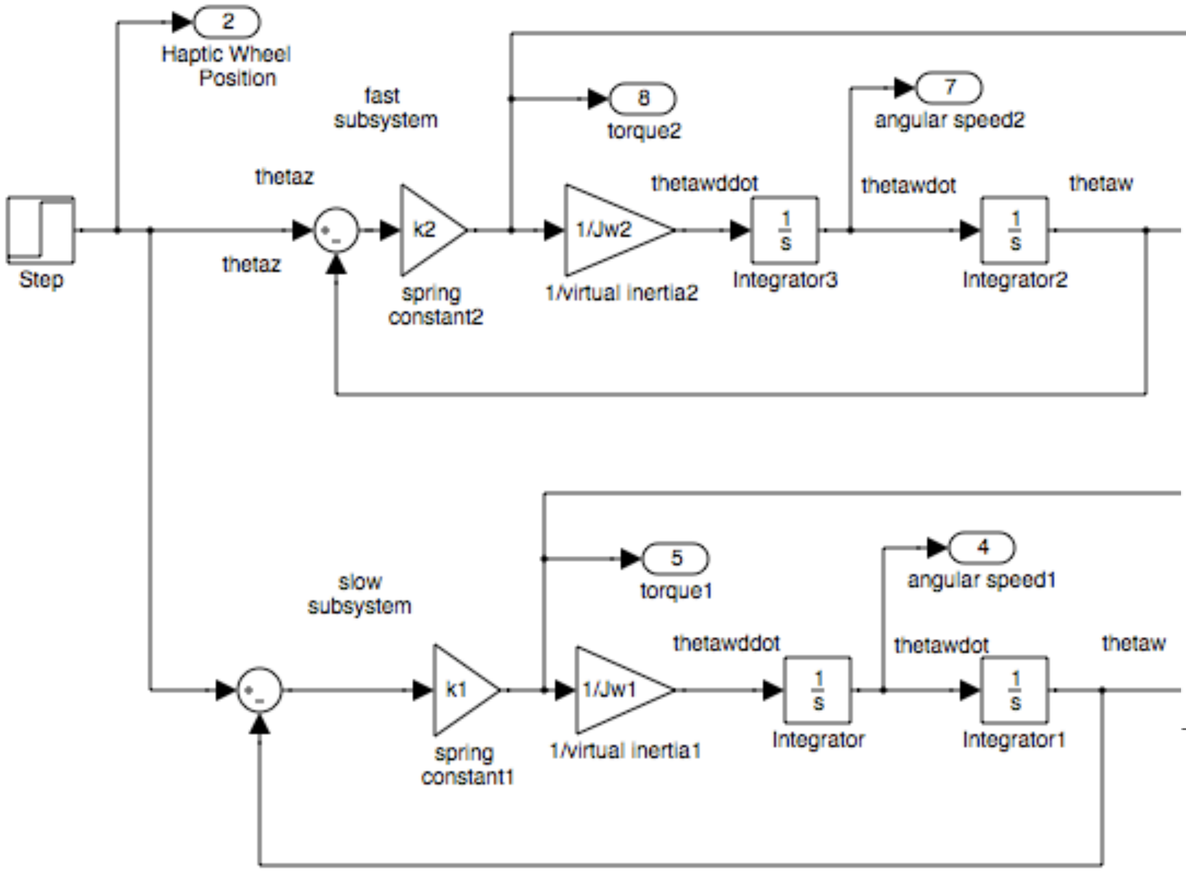
RAppID MPC5554 Target Setup

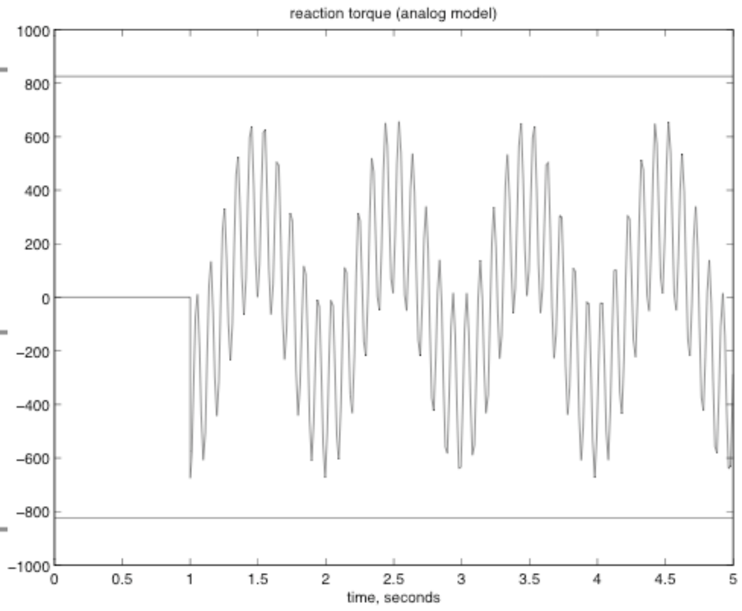
System Clock: 128 MHz
Target : MPC5554
Compiler: metrowerks
Target Type : IntRAM
Operating System : simpletarget

RAppID-EC



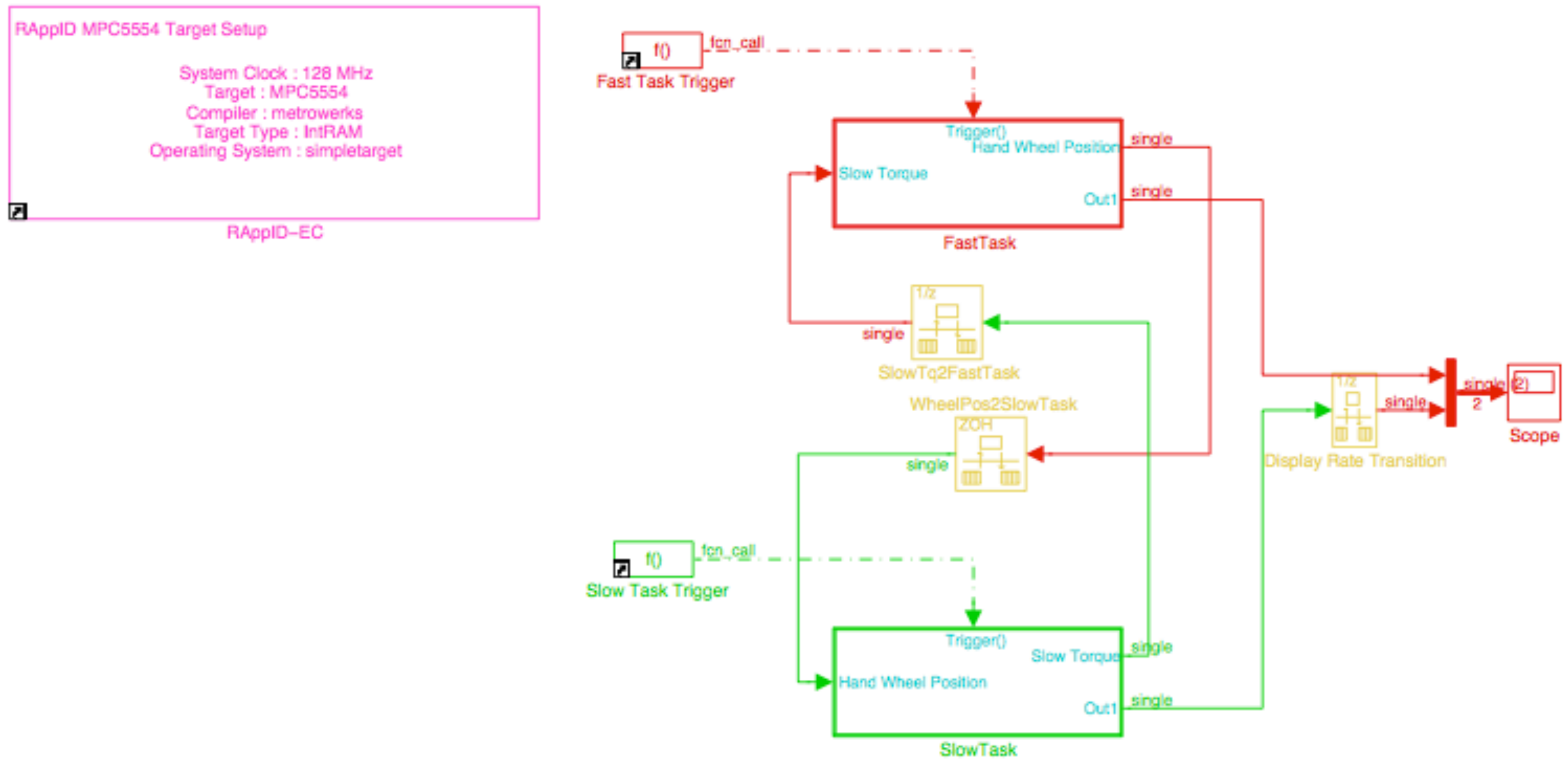
Lab 7: Two virtual wheels



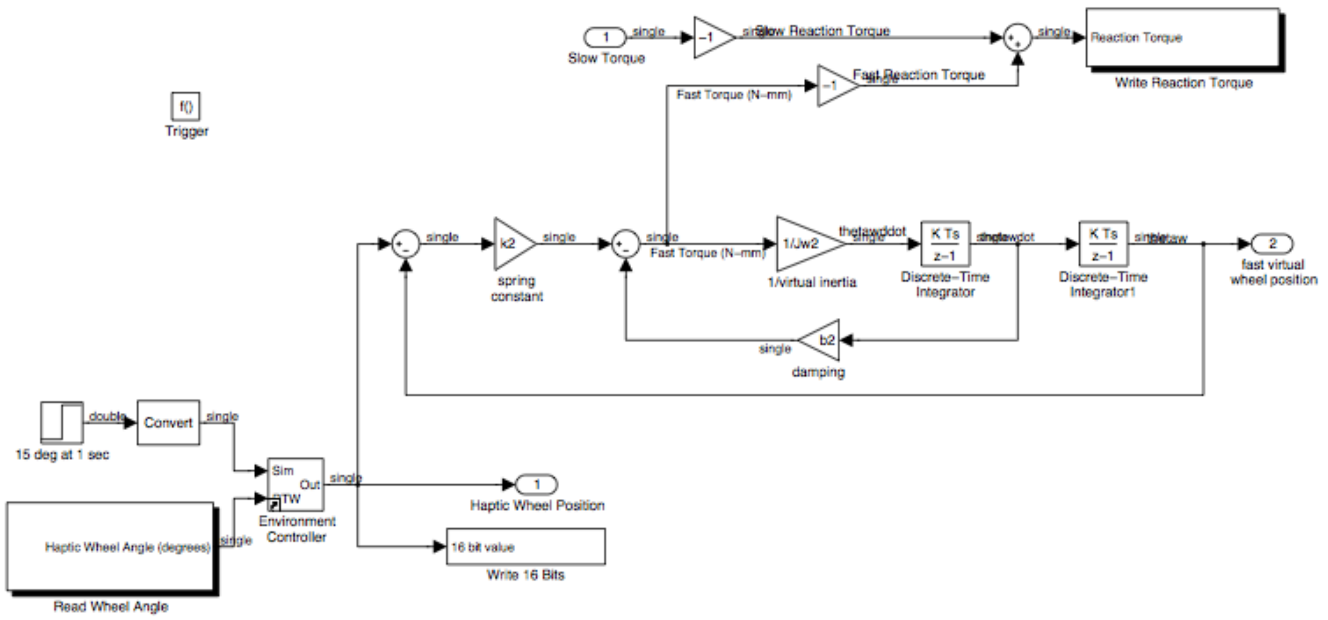


Total reaction torque

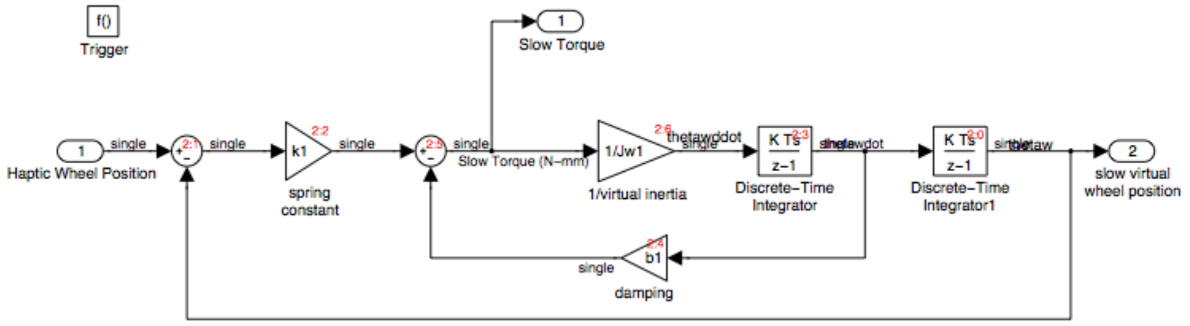
Multirate Simulation for Code Generation



Fast and slow subsystems

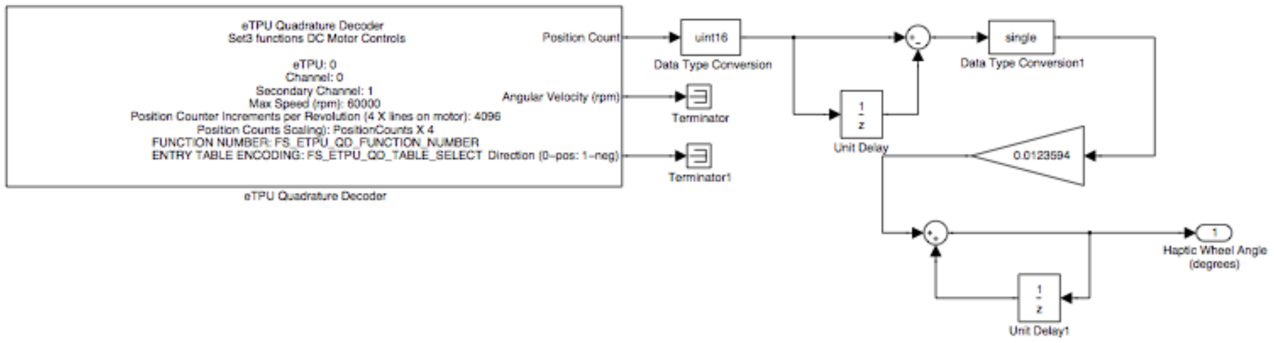


Fast subsystem

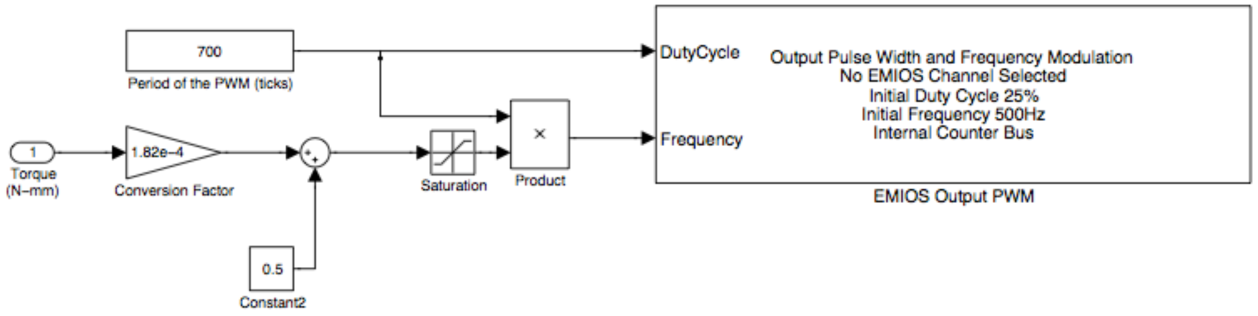


Slow subsystem

Device Drivers



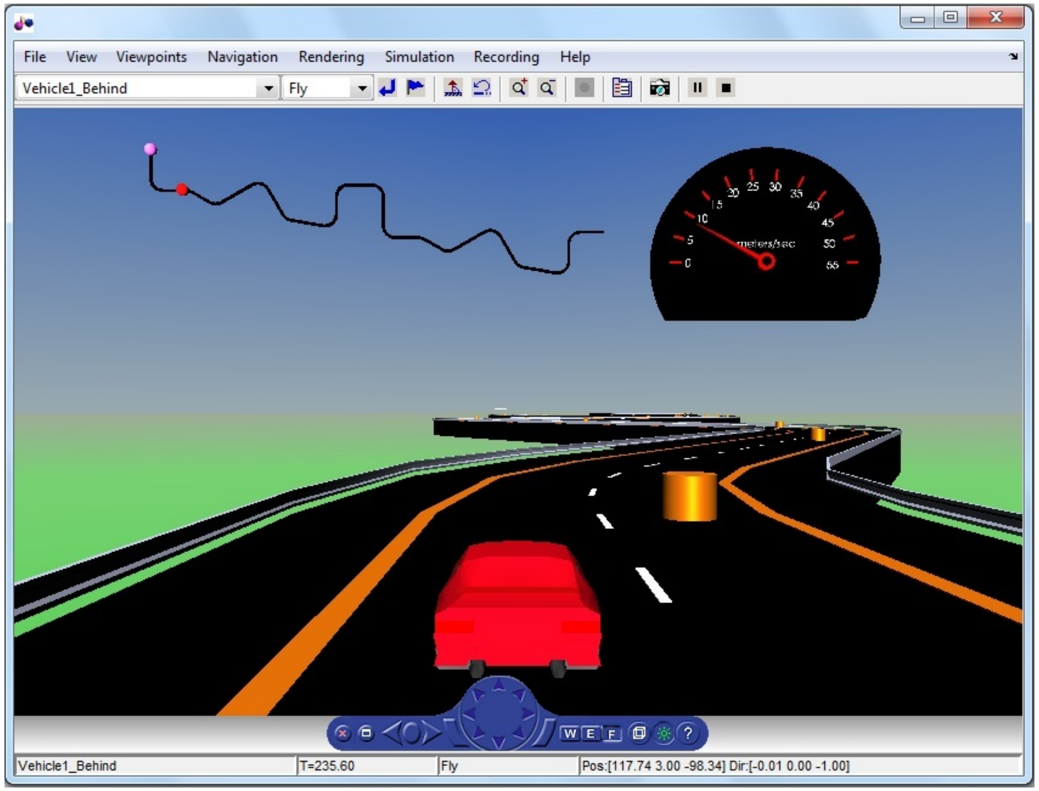
Read encoder and translate to degrees

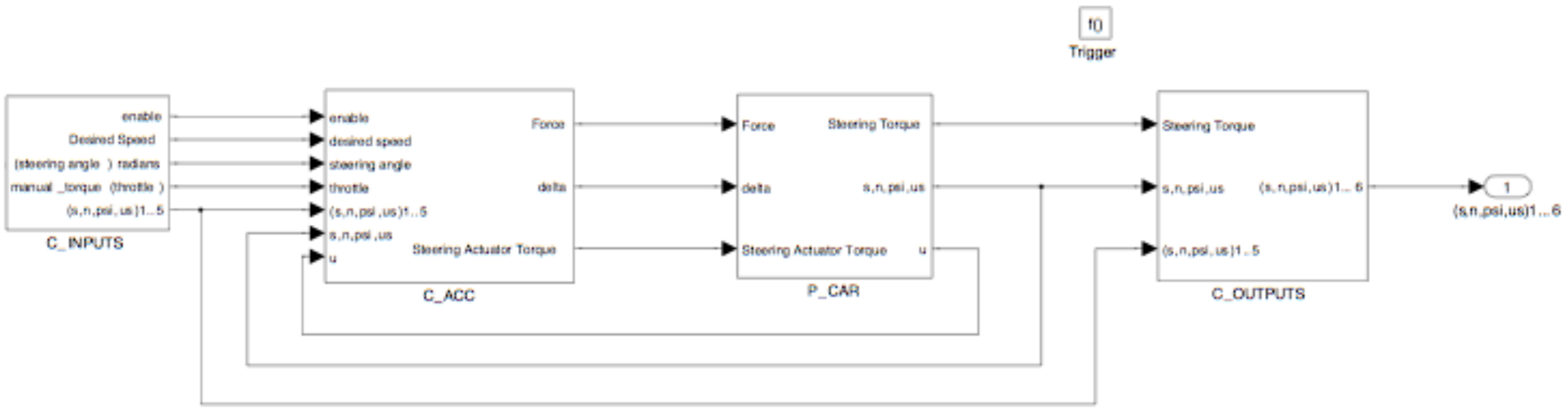


Convert torque to duty cycle and write to PWM

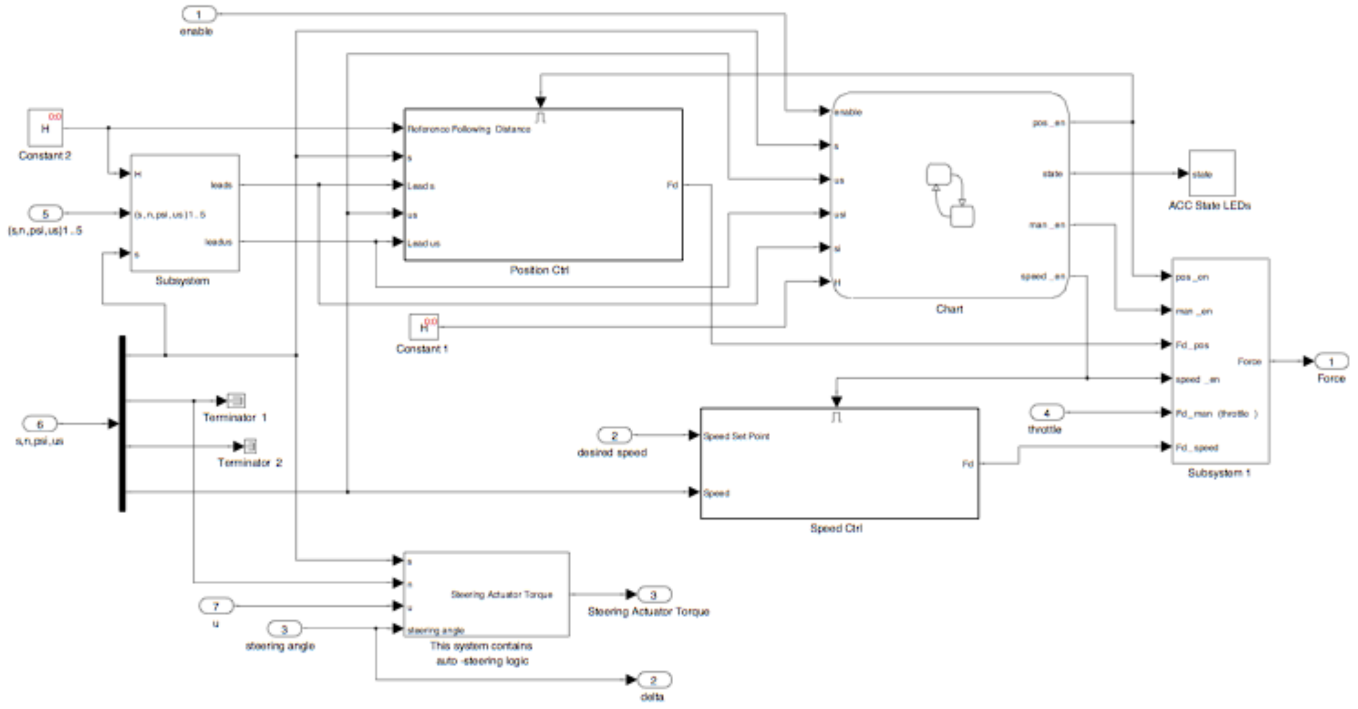
UM Project: Adaptive Cruise Control

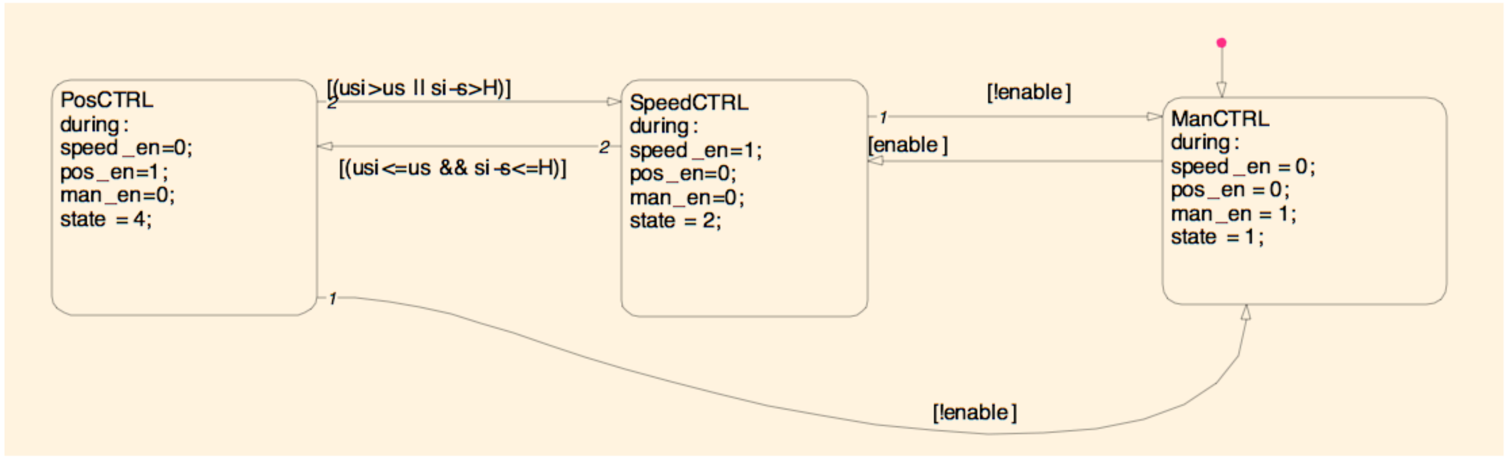
- Driving simulator
- Bicycle model of vehicle
- 6 vehicles interacting over CAN network
- “Lane centering”
- ACC algorithm: 3 states
  - manual (sliding pot)
  - constant speed
  - constant distance





Controller Block Diagram





Observations

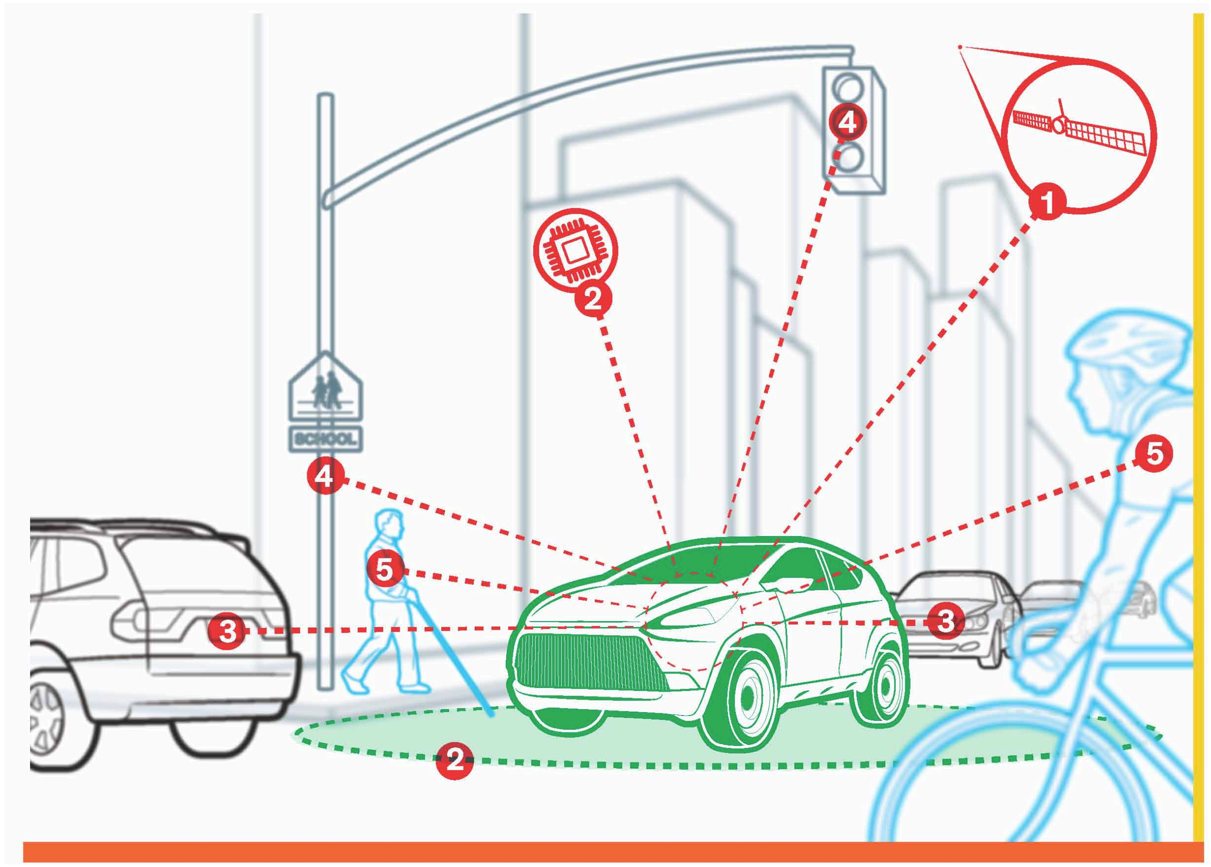
- Multidisciplinary
- Multiple layers of abstraction

- Successful embedded engineers understand *time*
  - Mechanical/electrical engineers: time in the application domain (physics)
  - Computer engineers: time on the microprocessor ($1 \rightleftarrows 0$)

- "pure" software engineers lack necessary background.

- Applications in many areas
  - aerospace
  - household appliances
  - robotics
  - civil engineering
  - defense
  - medical devices

The Automobile and the Future



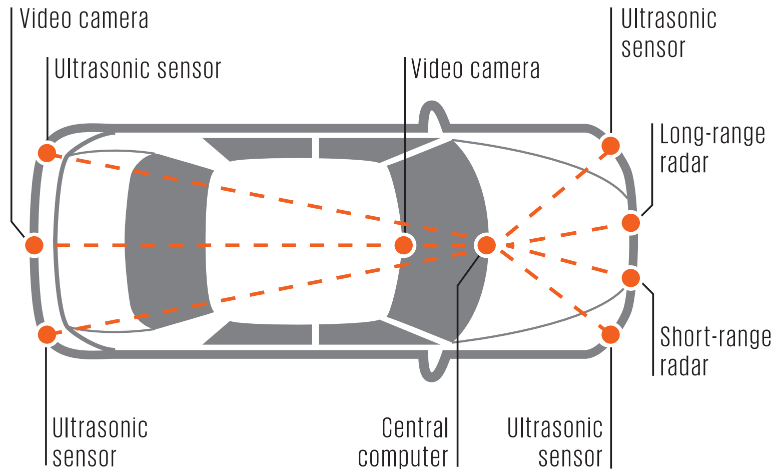
More fuel efficient vehicles = lighter & less survivable in an accident.

Solution: avoid accidents by eliminating driver error.

➔More jobs for embedded control engineers

Lino Guzzella ETH, IEEE Spectrum May 2014

Active Safety Systems



Adaptive cruise control
Collision avoidance
Lane departure warning
Lane following

IEEE Spectrum May 2014, “The Rise of the Crash Proof Car”

Fully Autonomous Vehicles?

May 27, 2014

A Cautionary Note

October 2013: $3 million settlement
Bookout vs Toyota
unintended acceleration

March 2014: $1.2 billion settlement

Wall Street Journal, December 2013: “Will tort law kill driverless cars?”
Fox News, March 2014: “Justice Department announces $1.2 billion settlement with Toyota”

Testimony

Expert witnesses:

Phil Koopman, CMU - “Code had >10,000 global variables*”

Michael Barr, Barr Group – “Code had bugs that could cause unintended acceleration**”

Wall Street Journal: “how is a car maker supposed to defend itself when it can't prove that its software behaves safely under all circumstances?”

The Google car has been driven 500K miles with no accident

Toyota Camrys were driven *billions of miles* before software error (if it was that) emerged

---

*Jack Ganssle “A Pox on Globals”, embedded.com, Oct. 2006

**www.safetyresearch.net/2013/11/07/

National Science Foundation Research

Cyber-physical systems (CPS):

interaction between computational elements and physical world.

~ networks of embedded control systems

Since comprehensive testing is not feasible…

how to write software that works because it is written correctly?

NSF CPS Frontier Project (UofM, http://www.dynamiccps.org):

"Correct-by-Design Control Software Synthesis for Highly Dynamic Systems"

Pedagogical Challenge: CPS requires students to be educated

"outside the traditional academic stovepipes"

Impact on Pedagogy

Michael Barr: Top 10 embedded software bugs

The ones we learn about in EECS 461 are underlined!

Race Condition
Non-reentrant function
Missing volatile keyword
Stack Overflow
Heap Fragmentation

Memory leak
Deadlock
Priority inversion
Incorrect priority assignment
Jitter

www.embedded.com 2010

Another Issue

Wired Magazine, July 21, 2015:

Hackers Remotely Kill a Jeep on the Highway—With Me in It

After the brakes were remotely disabled:

Cybersecurity is beyond the scope of EECS 461, yet former EECS 461 students are working in industry on connected vehicles today!

www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/

Conclusions

Electronics and software in automobiles has been a roaring success!

cleaner
safer
more sustainable

Many other application areas:

aerospace
defense
medical
appliances

The future will *require* more embedded control systems!

Big questions:

are we creating technology too complex to understand and maintain?
how do we train the workforce?

First step: Take Embedded Control Systems!