Embedded Control Systems

ETH – Institute for Dynamic Systems and Control
September 10 to 14 and 17 to 21, 2018

Jim Freudenberg
jfr@umich.edu

Peter Simon
pesimon@student.ethz.ch

Sebastian Glatz
sglatz@student.ethz.ch

Marianne Schmid
marischm@ethz.ch

Stijn van Dooren
stijnva@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

### Lectures Week 1 (8 to 10 a.m.) in Room HG E 33.1

<table>
<thead>
<tr>
<th>No.</th>
<th>Date</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2018-09-10</td>
<td>Course introduction. Start on A/D conversion, sampling and aliasing; simple and aliasing filter design</td>
</tr>
<tr>
<td>2</td>
<td>2018-09-11</td>
<td>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</td>
</tr>
<tr>
<td>3</td>
<td>2018-09-12</td>
<td>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.</td>
</tr>
<tr>
<td>4</td>
<td>2018-09-13</td>
<td>Discuss motor control (speed control, torque control, power amplifiers); Pulse width modulation; virtual worlds, wall &quot;chatter&quot; and the virtual wall.</td>
</tr>
<tr>
<td>5</td>
<td>2018-09-14</td>
<td>Dynamic systems and transient specifications (review); develop dynamic motor model block diagram and implement in Simulink (demonstrate problem set 3). Develop motor frequency response and demonstrate input PWM attenuation.</td>
</tr>
</tbody>
</table>

### Lab in Room ML E 55

<table>
<thead>
<tr>
<th>Time</th>
<th>10 a.m.</th>
<th>11</th>
<th>12 a.m.</th>
<th>1 p.m.</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5 p.m.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-Lab</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Pre-Lab</td>
<td>In-Lab</td>
<td></td>
<td></td>
</tr>
<tr>
<td>In-Lab</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Pre-Lab</td>
<td>In-Lab</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Post-Lab</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Lectures Week 2 (8 to 10 a.m.) in Room HG E 23

<table>
<thead>
<tr>
<th>No.</th>
<th>Date</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>2018-09-17</td>
<td>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.</td>
</tr>
<tr>
<td>7</td>
<td>2018-09-18</td>
<td>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 software architecture, real-time operating systems and scheduling algorithms. Rapid prototyping and automatic code generation.</td>
</tr>
<tr>
<td>8</td>
<td>2018-09-19</td>
<td>Code Generation with SIMULINK (RApidID Toolbox)</td>
</tr>
<tr>
<td>9</td>
<td>2018-09-20</td>
<td>Software architecture; presentation of MathWorks on Autocode generation with SIMULINK</td>
</tr>
<tr>
<td>10</td>
<td>2018-09-21</td>
<td>Introduction to CAN networks.</td>
</tr>
</tbody>
</table>

### Lab in Room ML E 55

<table>
<thead>
<tr>
<th>Time</th>
<th>10 a.m.</th>
<th>11</th>
<th>12 a.m.</th>
<th>1 p.m.</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5 p.m.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-Lab</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Pre-Lab</td>
<td>In-Lab</td>
<td></td>
<td></td>
</tr>
<tr>
<td>In-Lab</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Post-Lab</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pre-, In- &amp; Post-Lab</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>MathWorks (location: HG E 21)</td>
<td>Pre-, In- &amp; Post-Lab</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 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”
The Automobile in 2018

• Drivetrain
  – Variable geometry turbochargers
  – Variable cam timing (intake, exhaust, dual-equal, dual independent)
  – Variable valve timing
  – Variable compression ratio
  – Automatic transmission, continuously variable transmission

• 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 convenient than in 1977!
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

• Ford: “Across the auto-engineering spectrum right now, there is a war for talent”****

*Detroit Free Press, October 2012
** USA Today, July 2013
*** www.sae.org/membership/salarysurvey/
**** “Detroit Battles for the Soul of Self Driving Machines”, June 2016 Wall Street Journal
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**
  - 19th 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**
  - 11th year as two week block course
  - 33 students/year
  - Mechanical Engineering Graduate Students
Embedded Control Enrollment: UM and ETH

Total Enrollment: 2442
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)
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:

```c
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

**Issue**: 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 $\leftrightarrow$ encoder count
  - torque $\leftrightarrow$ PWM duty cycle
  - 1 degree into wall $\leftrightarrow$ 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 $< 800$Nmm

• 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.*
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
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
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”

\[ T = k(\theta_i - \theta) + k(\theta_j - \theta) \]
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

Diagram showing various control and processing blocks related to control systems.

---

PosCTRL during:
speed_en=0;
pos_en=1;
man_en=0; state = 4;

[(us1>us || si-us>H)]

SpeedCTRL during:
speed_en=1;
pos_en=0;
man_en=0; state = 2;

[enable]

ManCTRL during:
speed_en = 0;
pos_en = 0;
man_en = 1; state = 1;

[enable]
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 ($1 \leftrightarrow 0$)

• “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

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


**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?


“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
Cybersecurity

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 in industry on connected vehicles today!

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

Exhaust system of a Volkswagen Golf

Volkswagen has used two basic types of technology to reduce emissions of nitrogen oxides from diesel engines, by either trapping the pollutants or treating them with urea. The first type is shown here.

Main computer
Engine control module

Diesel oxidation catalytic converter
Oxygen sensor
H2S catalytic converter
Diesel particulate filter
Temperature sensors
Exhaust valve
Nitrogen oxide trap

This system traps nitrogen oxides, reducing toxic emissions. But the engine must regularly use more fuel to allow the trap to work. The car's computer could save fuel by allowing more pollutants to pass through the exhaust system. Saving fuel is one potential reason that Volkswagen's software could have been altered to make cars pollute more, according to researchers at the International Council on Clean Transportation.
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!