Embedded Control Systems

Lecture: 12:00-1:30 Tue./Thu. 1005 Dow
Lab: 4342 EECS

J. S. Freudenberg
OFFICE: 4425 EECS bldg
PHONE: (734) 763-0586
EMAIL: jfr@eecs.umich.edu

Temporarily, Jeff Cook
EMAIL: jeffcook@eecs.umich.edu
OFFICE: 4235 EECS bldg

GSIs:
Zhaori Cong (Tuesday and Thursday 3:00-6:00)
zcong@umich.edu
Jessica Szemraj (Monday 2:00-5:00 and Wednesday 9:30-12:30)

Embedded Control Systems

- Background:
  - University of Michigan and Ford Motor Company, 2004
  - Control theorists and computer scientists: why do we have to hire one of each to develop embedded controls?
  - Teach a little computer engineering to control theorists, and a little signal processing and control to computer engineers
  - Also taught at ETH (2008, 2009)

Important Points

- No textbook
  - www.eecs.umich.edu/courses/eecs461
  - Lecture notes, microprocessor reference material, laboratory exercises, homework problems and lots of other important information will be posted
  - Syllabus lists some useful (but not required) books on embedded systems programming
  - I'll mention during lecture what you should be reading
- Homework will be Matlab, Simulink, Stateflow
  - Problem sets will be posted on the website
  - Typically have one week per problem set. Homework is due at the beginning of class. Late homework will not be accepted. The *Homework Policy is posted on the course website, and included in the syllabus.*

Important Points

- Laboratory exercises
  - 8 laboratory exercises plus a project using the Freescale MPC5553 microprocessor
    - Most labs are "1-day" (1 lab per week)
    - First lab will be Monday, 14 September
    - Lab instructors will typically have "open hours" on Fridays Check with your lab instructor for times
  - 6 lab stations with 2 students ("self organize")

Important Points

- Laboratory exercises have 3 parts:
  - **Pre-lab:** questions that require you to read the microprocessor reference material and gather the information required to complete the lab exercise
  - **In-lab:** the experiment
  - **Post-lab:** questions that should reinforce what you learned in the lab exercise
  - Read the "lab policy" in the syllabus

Other Useful Information

- Grading:
  - Homework: 25%
  - Laboratory Assignments: 25%
  - Quizzes (tentatively scheduled October 29th and December 1st): 30%
  - Project: 20%
- Office Hours (Freudenberg, when he returns): 2:00-3:00 Monday and Wednesday, but feel free to stop by or email to set up an appointment
- Office Hours (Cook): I'll be in my office by 9:00 pretty much every day; stop by or email.
- Email alias: `eecs461@eecs.umich.edu`
  - See syllabus for instructions

Outline

- Embedded systems and embedded *control* systems
- Laboratory description
  - Freescale MPC5553 microcontroller
  - Software development environment
  - Haptic interface
- Lecture Topics
- Laboratory Exercises

What is an Embedded System?

- Technology containing a microprocessor as a component
  - cell phone
  - PDA
  - digital camera
- Constraints not found in desktop applications
  - cost
  - power
  - memory
  - 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:*
  - Automobile
  - Aircraft and UAV
  - Active control of civil structures
  - Manufacturing tools
  - Household appliances
  - Many others …

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
- Distributed control
  - networks of embedded microprocessors

Skills Required for Embedded Controls

- Algorithms (control, signal processing, communications)
- Computer software (real time, multitasking)
- Computer hardware (interfacing, memory constraints)
- Digital electronics
- Sensors and actuators
- Mechanical design

- Multi-disciplinary!

An Embedded Design Team

- May consist of:
  - Applications engineers
    - Model the systems to be controlled, design control algorithms
  - Hardware specialists
    - Low-lever drivers and other hardware specific design
  - Software engineers
    - Write C code from specifications given to them by applications engineers
- Applications engineers, hardware engineers and software engineers have to communicate!

Languages

- Some assembly language
  - device drivers, highly optimized code
- Most coding done in C
  - interest in C++ and Java, but too much overhead for highly constrained applications
- Automatic code generation
  - automatically generate C code from a Matlab/Simulink model used to design and test control algorithm
  - currently useful for rapid prototyping on non-production processor
  - also used for high end applications (NASA)

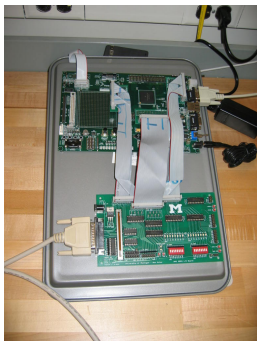
Laboratory Overview

- MPC5553 Microcontroller (Freescale)
  - Originally automotive control, now used in many applications
- 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, Real Time Workshop (The Mathworks)
  - RAppID Toolbox (Freescale)
- Real Time Operating System
  - OSEKturbo RTOS (Freescale)

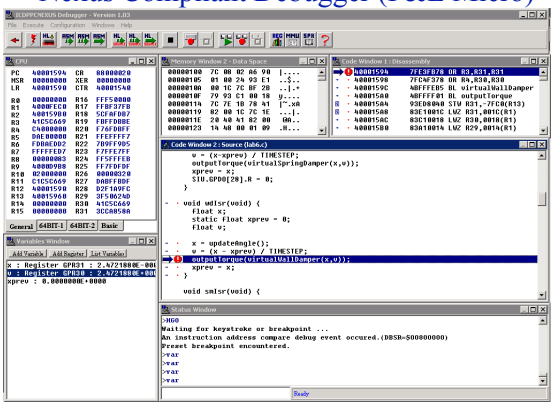
MPC5553 EVB

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

• Interface board (UofM)
- – buffering
- – dipswitches
- – LEDs
- – sliding potentiometer



Nexus Compliant Debugger (P&E Micro)



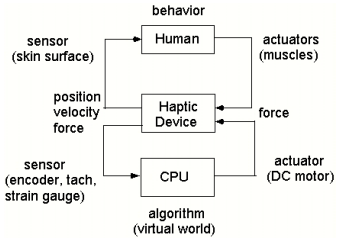
Haptic Interface

- Enables human/computer interaction through sense of touch
  - force feedback joystick
  - virtual reality simulators (flight, driving)
  - training (surgery*, assembly)
  - teleoperation (manufacturing, surgery**)
  - X-by-wire cars
- Human visual sensor: 30 Hz
- Human haptic sensor: 500Hz-1kH

---

* D. Sordid and S. K. Moore, "The Virtual Surgeon", IEEE Spectrum, July 2000.
** J. Rosen and B. Hannaford, "Doc at a Distance", IEEE Spectrum, October 2006.

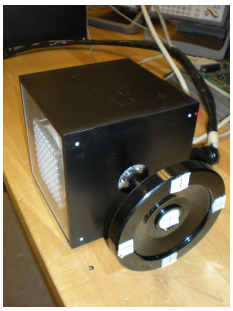
Force Feedback



Haptic Wheel

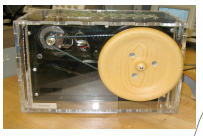
(Thanks to Prof. Brent Gillespie and students, ME Dept)

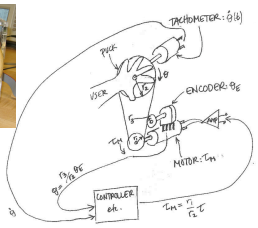
DC motor and PWM amplifier with current controller



Virtual Environments

- Virtual wall
- Virtual spring-mass





Lab Station

Lectures (I)

- Quantization
- Sampling
- Linear filtering
- Quadrature decoding
- DC motors
- Pulse Width Modulation (PWM) amplifiers
- Motor control: current (torque) vs. speed
- MPC5553 architecture. Peripherals: eTPUs, eMIOS, eDMA,…
- 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
- Each uses concepts (and code!) from the previous labs

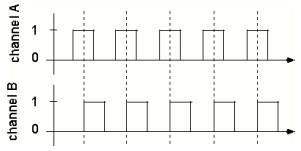
- Lab 1: Familiarization and digital I/O
- Lab 2: Quadrature decoding using the eTPU
- Lab 3: Queued A-D conversion
- Lab 4: Pulse Width Modulation and virtual worlds without time
- Lab 5: Interrupt timing and frequency analysis of PWM signals
- Lab 6: Virtual worlds with time.
- Lab 7: Controller Area Network (CAN)
- Lab 8: Rapid Prototyping

Lab 1: Familiarization and Digital I/O

- Use General Purpose Input/Output (GPIO) on MPC5553
- Read two 4-bit numbers set by dipswitches, add the values and display the results on LEDS

Lab 2: Fast Quadrature Decoding

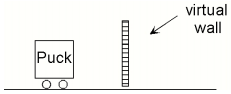
- Position measurement using an optical encoder
- Optical encoder attached to motor generates two 90° out of phase square waves:

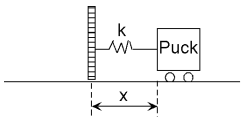


- QD function on MPC5553 eTPU:
  decodes quadrature signal into counter
- CPU must read counter before overflow

Issue: How fast can wheel turn before counter overflows?

Lab 4: Virtual Wall

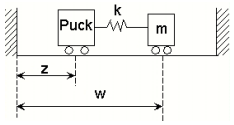




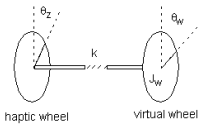
- Software loop
  - read position from encoder
  - compute force $F = 0$ or $F = kx$
  - set PWM duty cycle
- Rotary motion
  - degrees ⇔ encoder count
  - torque ⇔ PWM duty cycle
  - 1 degree into wall ⇔ 400 N-mm torque
- Wall chatter
  - large k required to make stiff wall
  - limit cycle due to
    - * sampling
    - * computation delay
    - * quantization
    - * synchronization

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



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



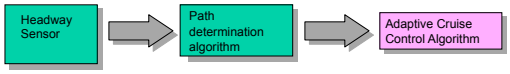
$$\ddot{\theta}_w + \frac{k}{J_w}\theta_z = \frac{k}{J_w}\theta_z$$

Rapid Prototyping (I)

- Lab 8 involves automatic code generation from Simulink models:
  - 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 Real Time Workshop (RTW) to generate C-code.
  - Eliminates coding errors.
  - Speeds product development: generated code can be tested in many design cycles
  - Hand coding still required for production

Project (at UM): Adaptive Cruise Control

- Distance Control
  - Follows target at timed headway in ACC mode by use of throttle and brakes
- Speed Control
  - Automatically returns to cruise set speed when target clears



Project: Adaptive Cruise Control

- Driving simulator
- Bicycle model of vehicle
- 6 vehicles interacting over CAN network
- ACC algorithm: 3 states
  - manual (sliding pot)
  - constant speed
  - constant distance
- Takes 3+ weeks, all done with Simulink, Stateflow, and autocode generation