Embedded Control Systems

Lecture: MW 130-3PM 1311 EECS
Labs: 4342 EECS

Jeff Cook
jeffcook@eecs.umich.edu
Office: 4238 EECS

Zhaori Cong (Thursday 9:30)
zcong@umich.edu

Jeff Roder (Tuesday, Thursday 1:30)
roderjef@umich.edu

John Schmotzer (Monday 3:30, Wednesday 10:00)
jwschmo@umich.edu
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)
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 two weeks beginning Monday, 12 January – BUT MLK day on 19 January means Monday section has only one scheduled lab
  • Lab instructors will have “open hours” on Friday, 16 January and/or Friday 23 January for Monday students. Check with your lab instructor for times
  – 6 lab stations with 2 students (“self organize”)
Important Points

• Special lecture on embedded system programming
  – Important information for lab #1
  – When to do this lecture?
    • Monday? … but lab starts at 3:30
    • Special lecture on Friday?
      – Same time and place, if I can get the room
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 for February 18th and April 1st): 30%
  – Project: 20%

• Office Hours: 10:00 - Noon, Monday and Wednesday, but feel free to stop by or email me to set up an appointment

• 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!
Industry Trends

• Increasing complexity of embedded control systems and software
  – Actuators, sensors, processors, networks
  – Typical small car contains ~70 microprocessors

• Model based embedded control software design
  – Matlab/Simulink/Stateflow
  – Autocode generation
  – Rapid prototyping
  – Hardware in the loop (HIL) testing

• “Separation between control design and controller implementation is not sustainable in embedded market”*

* Industry Needs for Embedded Control Education, Tutorial Session 2005 ACC
  J. Freudenberg (UM), B. Krogh (CMU), J. Cook (Ford), K. Butts (Toyota), J. Ward (Eaton)
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)
MPC5553/5554 Examples: Automotive Applications

• Powertrain
  – Fuel and ignition control
  – Aftertreatment control for diesels
  – Valve control, turbocharger control, transmission control including CVT
  – Control of hybrid-electric powertrains
• Safety
  – ABS, traction control, electronic stability control, rollover control
• Lots of I/O: sensors & actuators
  – Real time critical: performance & safety
  – Harsh environment (EMI, noise, vibration, temperature)
Automotive Distributed Systems: Mobile Networking

- High-speed CAN
- Low-speed CAN
- Local Interconnect Network (LIN)
- Media Oriented Systems Transport (MOST)
- Bluetooth
- Intelligent Transportation System Data Bus (IDB 1394)
- FlexRay, Time-triggered CAN …
Application of the MPC555 (predecessor of the MPC5553)

- SeaScan transoceanic pilotless aircraft
- ScanEagle Intelligence, surveillance and reconnaissance support; USS Oscar Austin (DDG 79) Guided Missle Destroyer
- The Insitu Group: www.insitu.com
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)
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
  – 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-1kHz

---

Force Feedback

- Sensor (skin surface)
- Position, velocity, force
- Haptic Device
- Algorithm (virtual world)
- Human
- Actuators (muscles)
- CPU
- Actuator (DC motor)
Haptic Wheel

- Prof. Brent Gillespie, Mech Eng Dept, UofM
  - DC motor
  - PWM amplifier w/ current controller
  - optical encoder
  - 128/18 gear ratio
Haptic Wheel
(New and Improved for 2009)
Virtual Environments

- Virtual wall
- Virtual spring-mass
Steer-by-wire Automobiles

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:

![Diagram of quadrature encoder outputs]

- 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 $\Leftrightarrow$ 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
    * 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

\[
\ddot{w} + \frac{k}{m} w = \frac{k}{m} z
\]

\[
\ddot{\theta}_w + \frac{k}{J_w} \theta_z = \frac{k}{J_w} \theta_z
\]
Lab 6: 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
Lab 7: Controller Area Networking (CAN)

- Networking protocol used in time-critical applications
  - automotive
  - manufacturing
- Messages have unique identifiers: priorities
- Allows computation of worst case response time
- Lab exercises:
  - a wall that is chatter free when wall implemented locally can chatter due to delay when implemented remotely
  - connect each wheel to its virtual neighbors with virtual springs to create a virtual chain of 6 labstations.
  - estimate network utilization.
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
Model Based Embedded Control Software Development
Rapid Prototyping (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
OSEKturbo RTOS (Freescale)

- OSEK/VDX compliant
- Scalable
- Task scheduler
- Priority ceiling protocol
- Eliminates
  - deadlock
  - priority inversion
RAppID Toolbox (Freescale)

- Processor and peripheral initialization blocks
- Device driver blocks
- Enables multitasking with OSEKturbo RTOS or nested interrupts
Lab 8: Two virtual wheels

• Two subsystems:
  – High priority fast subsystem
  – Low priority slow subsystem
• Model the multi-rate system in Simulink
• Demonstrate real-time operating system (RTOS)
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

![Diagram of Adaptive Cruise Control](image.png)

- Headway Sensor
- Path determination algorithm
- Adaptive Cruise Control Algorithm
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