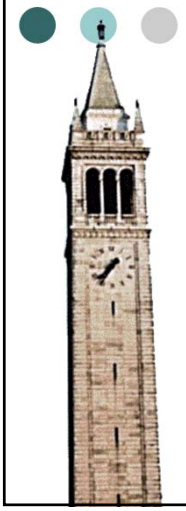


Introduction to Embedded Systems

Sanjit A. Seshia
UC Berkeley
EECS 149/249A
Fall 2015

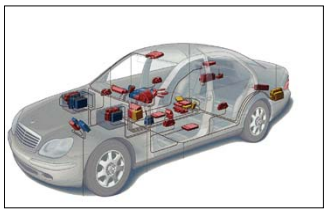
Course Wrap-up

What are Cyber-Physical Systems? (Recap)

Computational systems
- but not first-and-foremost a computer

Integrated with physical processes
- sensors, actuators, physical dynamics


Tomlin et al.


Daimler-Chrysler


Bosch-Rexroth

Play Detective

In the “real world”, it’s more likely you will be working with existing (legacy) embedded systems rather than building your own from scratch.

Let’s go through a few “stories” of large scale design issues or failures with industrial embedded systems and try to guess what topics from the course can help.

The Boeing 777 Problem

The Boeing 777 was Boeing’s first fly-by-wire aircraft, controlled by software. It is deployed, appears to be reliable, and is succeeding in the marketplace. Therefore, it must be a success. However…

Boeing was forced to purchase and store an advance supply of the microprocessors that will run the software, sufficient to last for the estimated 50 year production run of the aircraft and another many years of maintenance.

Why?

Air France 447

"The Airbus A330, operated by Air France from Brazil to Paris, entered an aerodynamic stall from which it did not recover and crashed into the Atlantic Ocean at 02:14 UTC, killing all 228 passengers, aircrew and cabin crew aboard the aircraft."

"The aircraft crashed after temporary inconsistencies between the airspeed measurements – likely due to the aircraft's pitot tubes being obstructed by ice crystals – caused the autopilot to disconnect, after which the crew reacted incorrectly and ultimately led the aircraft to an aerodynamic stall from which they did not recover." [Source: Wikipedia]

What are potential causes?

787 Power System Issue [Identified by FAA]

"A Model 787 airplane that has been powered continuously for 248 days can lose all alternating current (AC) electrical power due to the generator control units (GCUs) simultaneously going into failsafe mode."

This condition is caused by the state of a software counter internal to the GCUs reached after 248 days of continuous power and could lead to loss of all AC electrical power, which could result in loss of control of the airplane.

What might the software bug be? How to catch it?

GM Cadillac bug (circa 2004)

"General Motors Corp will recall 12,329 Cadillac SRXs equipped with all-wheel drive, following two reports of a software anomaly that causes a one-second delay in the anti-lock brakes activating to stop the vehicle -- reportedly only in the first few seconds of driving when the SUV is moving slowly. One owner crashed his SRX into his garage wall following the brake delay, but was uninjured."
[Source: Reuters, 2 Apr 2004; obtained from RISKS digest]

What might the software bug be? How to catch it?

2003 NorthEast Blackout (Power Grid failure)

"Worst outage in North American history."

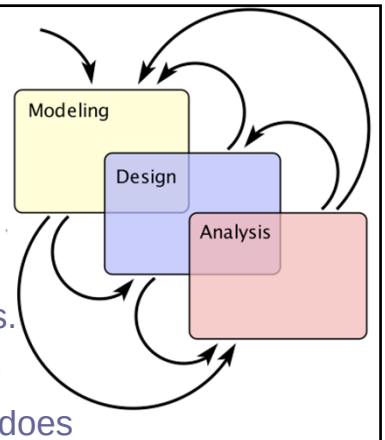
"One [reason] was buried in a massive piece of software compiled from four million lines of C code and running on an energy management computer in Ohio… A silent failure of the alarm function in FirstEnergy's computerized Energy Management System (EMS) is listed in the final report as one of the direct causes of a blackout that eventually cut off electricity to 50 million people in eight states and Canada." [The Register, UK, 8 Apr 2004.]

What could have gone wrong? [Hint: multi-threaded software]

Some Characteristics of Cyber-Physical Systems

- Reactive
  - operates at the speed of the environment
- Real-time
  - timing of events matters!
- Concurrent
  - system + environment, at a minimum
- Heterogeneous
  - hardware/software/networks, physical processes
- (increasingly) Networked
  - distributed, exposed to attacks

Modeling, Design, Analysis



Modeling is the process of gaining a deeper understanding of a system through imitation. Models specify **what** a system does.

Design is the structured creation of artifacts. It specifies **how** a system does what it does. This includes optimization.

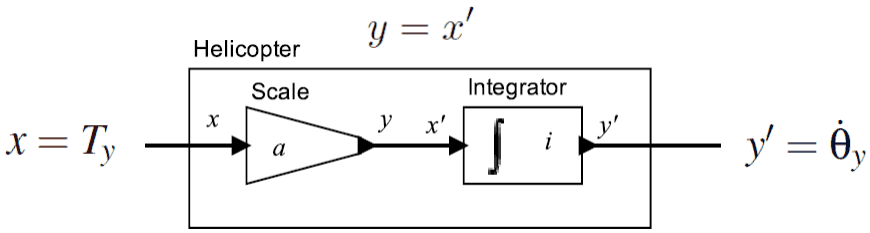
Analysis is the process of gaining a deeper understanding of a system through dissection. It specifies **why** a system does what it does (or fails to do what a model says it should do).

Model-Based Design

1. Create a **mathematical model** of all the parts of the cyber-physical system
   - Physical processes
   - Controllers: software, hardware, etc.
   - Software environment
   - Hardware platform
   - Network
   - Sensors and actuators
2. Construct the implementation from the model

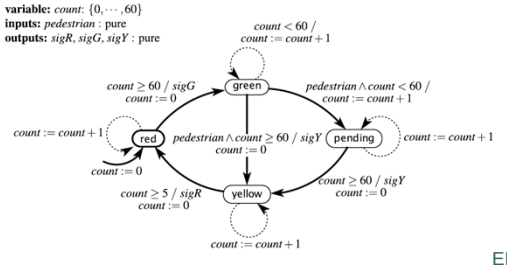
Modeling Techniques covered in the course (1)

- Differential Equations → Physical processes
- Actor Models
- Time-domain modeling
- Feedback control



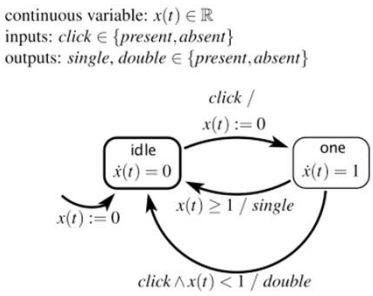
Modeling Techniques covered in the course (2)

- Finite-State Machines → for Modal Behavior, as in a controller, software
- Determinism, Receptiveness
- Trace – modeling the input/output behavior of an FSM
- Composition and Hierarchy
  - Synchronous/Asynchronous composition, StateCharts



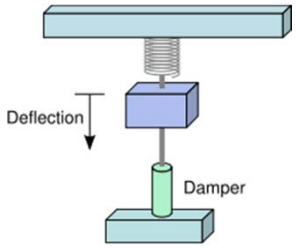
Modeling Techniques covered in the course (3)

- Timed/Hybrid Automata → for Modal Behavior + continuous dynamics
- Jumps and flows



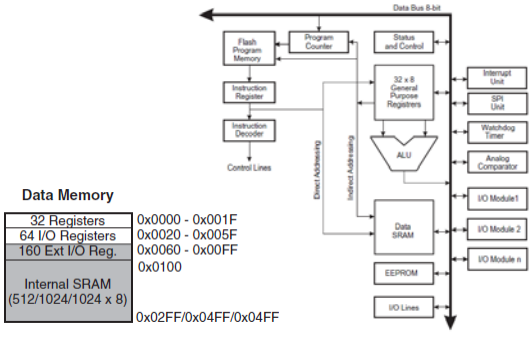
Modeling & Design: Sensors and Actuators

- How Sensors and Actuators Work: Basics
- Interfacing to Sensors
- Modeling Sensors and Actuators



Design: Memory Architectures

- Types of Memory
- Memory Maps and Organization
- Memory Model for C programs
- Memory Hierarchy and Protection



Source: ATmega168 Reference Manual

Design: Concurrent Programming with Interrupts

- I/O Mechanisms in Software: Polling vs. Interrupts
- Setting up Interrupts
- Reasoning about Interrupt-Driven Programs

```
volatile uint timer_count = 0;
void ISR(void) {
  if(timer_count != 0) {
    timer_count--;
  }
}
int main(void) {
  // initialization code
  SysTickIntRegister(&ISR);
  ... // other init
  timer_count = 2000;
  while(timer_count != 0) {
    ... code to run for 2 seconds
  }
}
```


Concurrency: Modeling and Design

- Threads
- Processes
- Multi-Tasking and Priorities
- Synchronous/Reactive Languages
- Dataflow

Real-Time: Design and Analysis

- Scheduling
  - Pre-emptive and non-preemptive
  - RMS vs EDF
  - Priority inversion, protocols: PIP, PCP
  - Anomalies in multiprocessor scheduling
- Execution Time Analysis
  - Blending measurements, platform modeling, and static analysis of code

Modeling & Analysis: Specification & Temporal Logic

- The Need for Formal Specification
- Linear Temporal Logic

8.5.2.2 ErrorReset

a. The *ErrorReset* state shall be entered after a system reset, after link operation is terminated for any reason or if there is an error during link initialization.

b. In the *ErrorReset* state the Transmitter and Receiver shall all be reset.

c. When the reset signal is de-asserted the *ErrorReset* state shall be left unconditionally after a delay of 6,4 µs (nominal) and the state machine shall move to the *ErrorWait* state.

d. Whenever the reset signal is asserted the state machine shall move immediately to the *ErrorReset* state and remain there until the reset signal is de-asserted.

Analysis and Verification

- Reachability Analysis
  - Compute the set of all states of the system reachable from any initial state
- Model Checking
  - Does the (closed-loop) system satisfy a temporal logic property?
- Equivalence and Refinement
  - When are two state machines equivalent?
  - When does one model refine another?

Fault Tolerance and Security

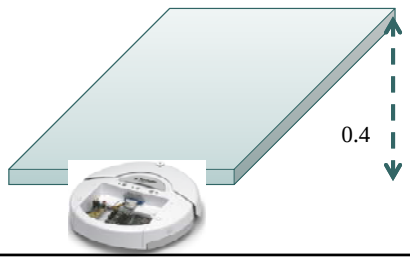
- Tolerating faults in sensors, computation, actuators
  - Self-checking, N-modular redundancy, interval readings for sensors, etc.
- Security & Privacy
  - Integrity, Confidentiality, Availability under attacks
  - Besides traditional issues, need to worry about physical properties and constraints (e.g. power)
  - Privacy properties and enforcement

Distributed Systems and Networking

- Proprietary protocols: CAN and FlexRay
- Time synchronization (IEEE 1588)
  - Leader election protocol
- Wireless protocols: ZigBee, OpenWSN, …
- Time-Triggered Ethernet, …

The Lab

- CPS Programming in C (low-level language)
- CPS Programming in LabVIEW (high-level modeling language)
- Modeling Physical Processes and Interfacing to Sensors and Actuators [provided]
- Specification & Temporal Logic [auto-grader]



$+ \quad \mathbf{F}_{[0,40]} \; z \geq 0.4$

Other Relevant Topics we didn't cover in-depth

- Architecture for embedded systems
  - E.g. low power, predictable timing, etc.
- Programming languages and compilers
- Testing and debugging
- Controller synthesis
- Simulation strategies
- Hybrid systems (more than timed automata)
- …

Future of CPS Design

Rising trend: combine model-based design with data-driven methods (learning from data)

This course discussed how design is done today, but you can be sure that *the technology will change*!

Our goal has been to give you what you need to think *critically* about the technology.