The Internet Meets Embedded Systems

http://www.ices.cmu.edu/roses

William Nace

Prof. Philip Koopman

Carnegie Mellon University

Carnegie Mellon

Institute for Complex Engineered Systems

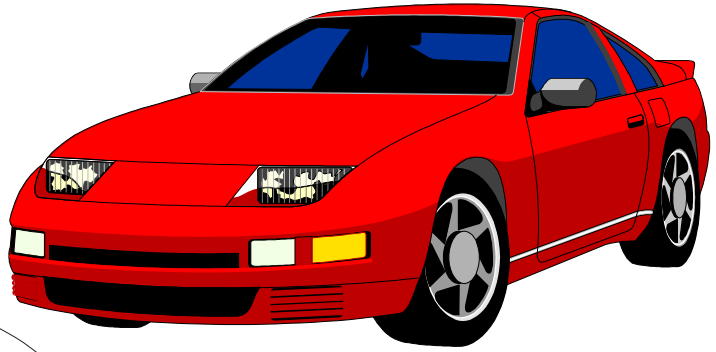
Electrical & Computer ENGINEERING

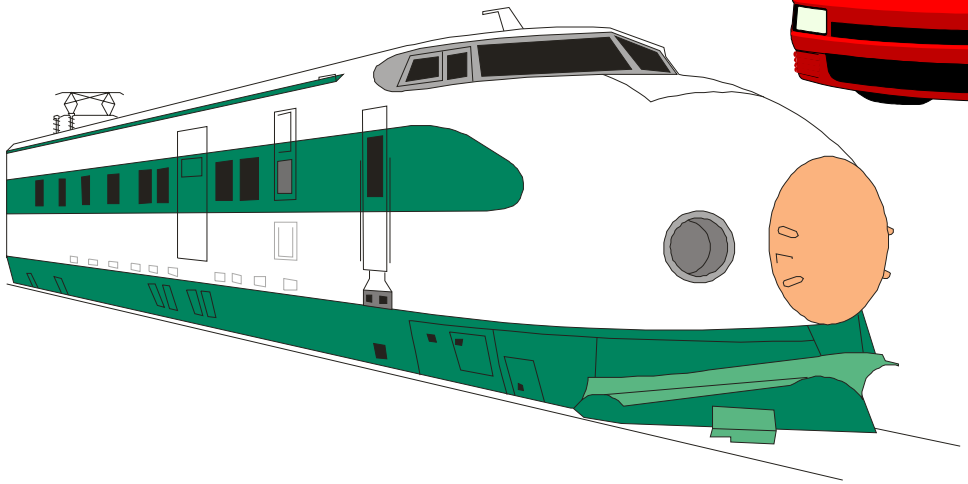
Overview

- **Classical embedded systems**
  - If you learn from them you can stand on their shoulders
- **Some myths**
  - Big CPUs matter
  - Small means trivial
  - Embedded != distributed
  - Security can be solved with airgaps
- **Example: RoSES research project**
  - Automatic graceful degradation on distributed embedded systems
  - Jini on CAN (embedded network)?
  - Embedded education

Embedded System =

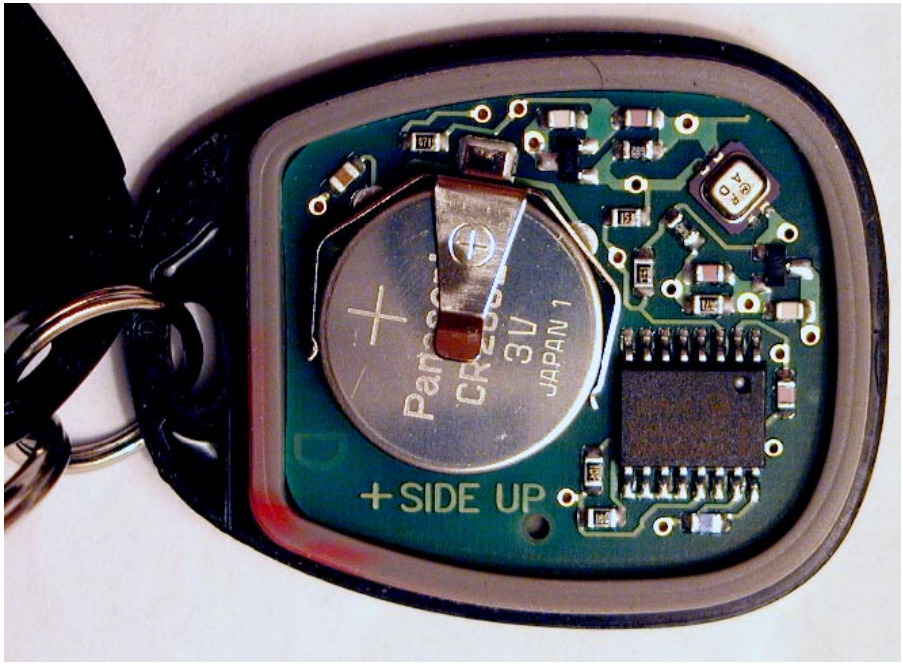
Computers Inside a Product





Typical Embedded System Constraints

- **Small Size, Low Weight**
  - Hand-held electronics
  - Transportation applications -- weight costs money
- **Low Power**
  - Battery power for 8+ hours (laptops often last only 2 hours)
  - Limited cooling may limit power even if AC power available
- **Harsh environment**
  - Power fluctuations, RF interference, lightning
  - Heat, vibration, shock
  - Water, corrosion, physical abuse
- **Safety-critical operation**
  - Must function correctly
  - Must *not* function *in*correctly
- **Extreme cost sensitivity**
  - $.05 adds up over 1,000,000 units

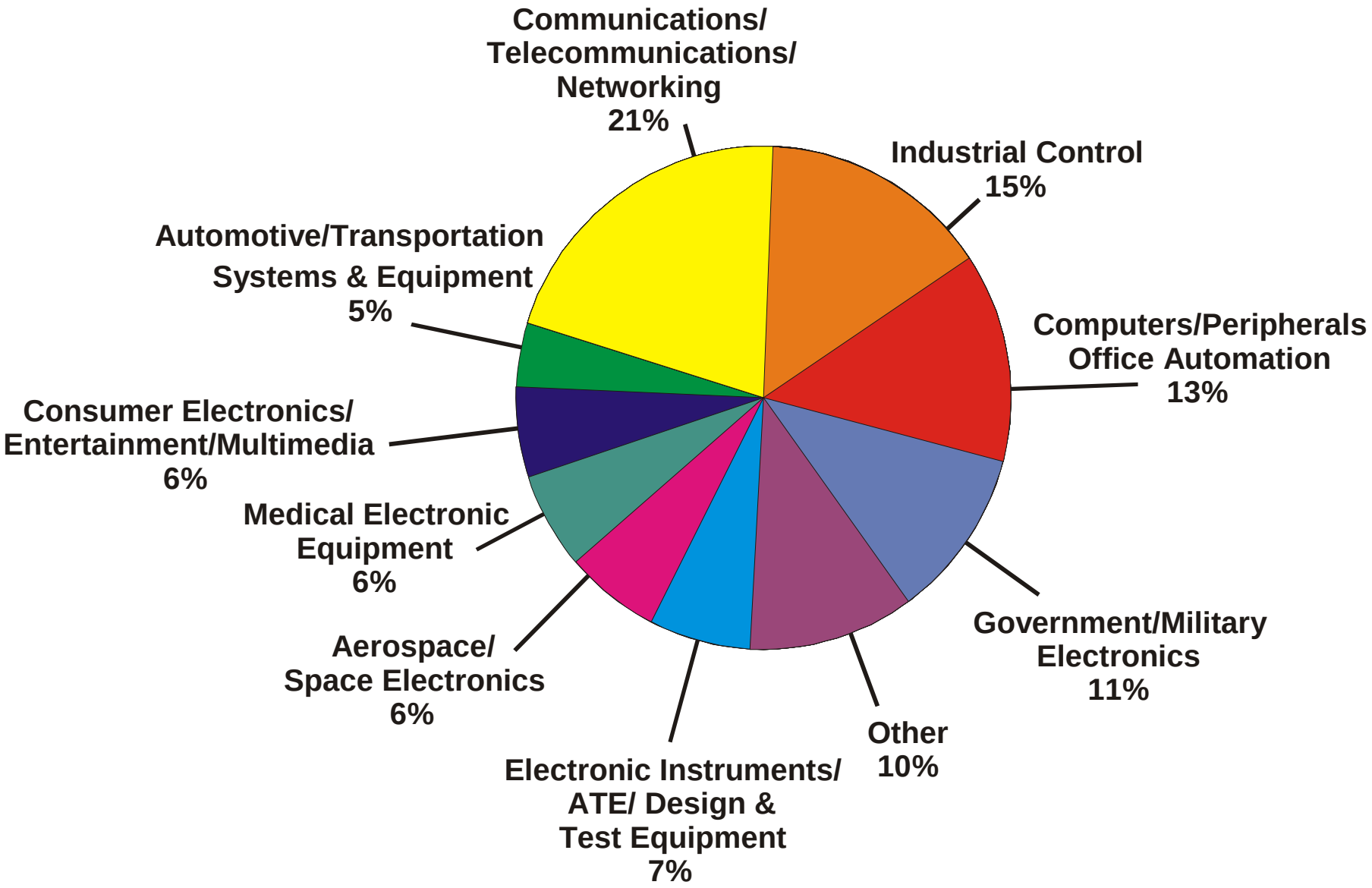


Why Are Embedded Systems Different?

- **Classical Embedded**
  - 5-50 year life cycle
  - Small, multidisciplinary design team
  - Real-time control of the physical world
  - Safety/mission critical
  - Synchronized, bursty, short network messages
  - School of hard knocks
- **Classical Internet**
  - 3 month – 3 year life cycle
  - Mostly software with a little hardware
  - Data processing
  - Usually not perceived as critical
  - Ethernet; TCP/IP
  - University

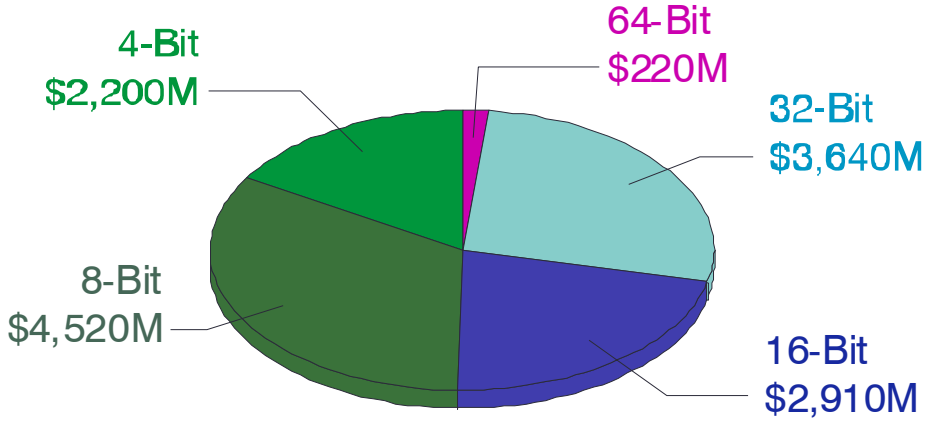
There Are Many Application Areas

Primary End Product of
Embedded Systems Programming
Subscribers (Dec. 1998)



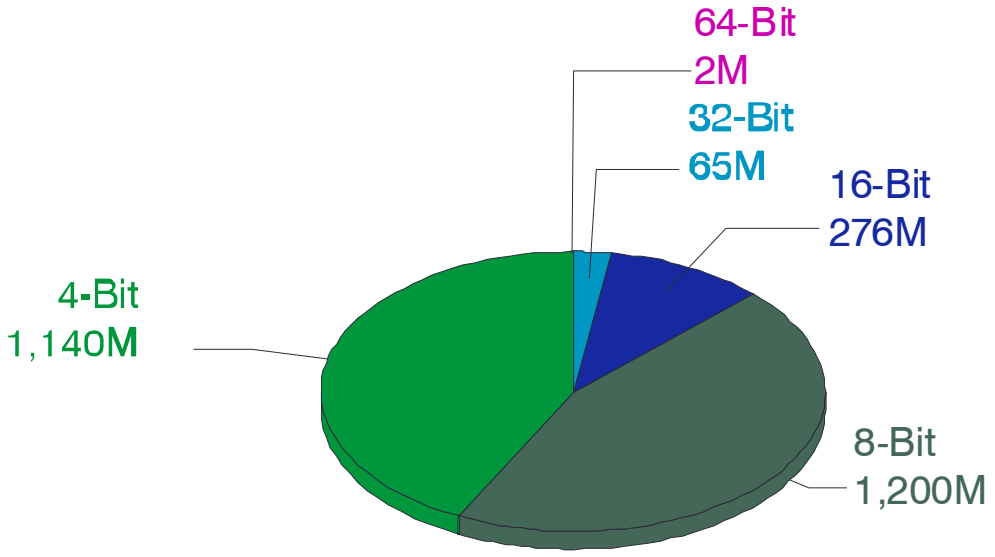
Myth: 32-bit+ CPUs Are What Matter

- ***Reality:* 32-bit+ CPUs are a small fraction of the market**
  - Nearly 100% by hype and academic research measures
  - About 25% by dollar amount
  - 2% to 3% by volume
  - 150 Million PCs *vs.* 7.5 Billion embedded CPUs + in 2000



$13,490M Total

1994 Worldwide
Microcontroller Revenue
($Million U.S.)



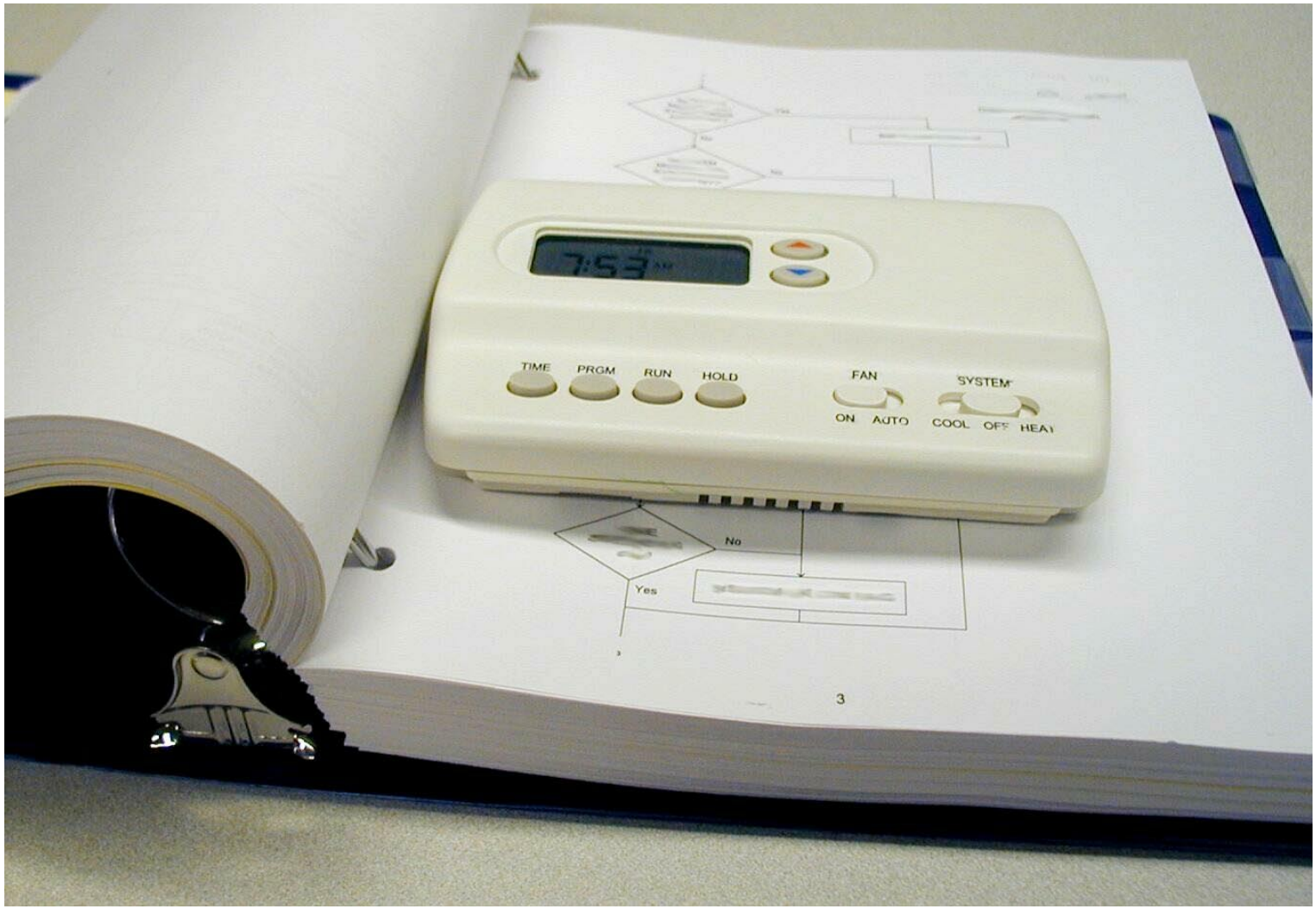
2,683M Total

1994 Worldwide
Microcontroller Units
(Million Devices)

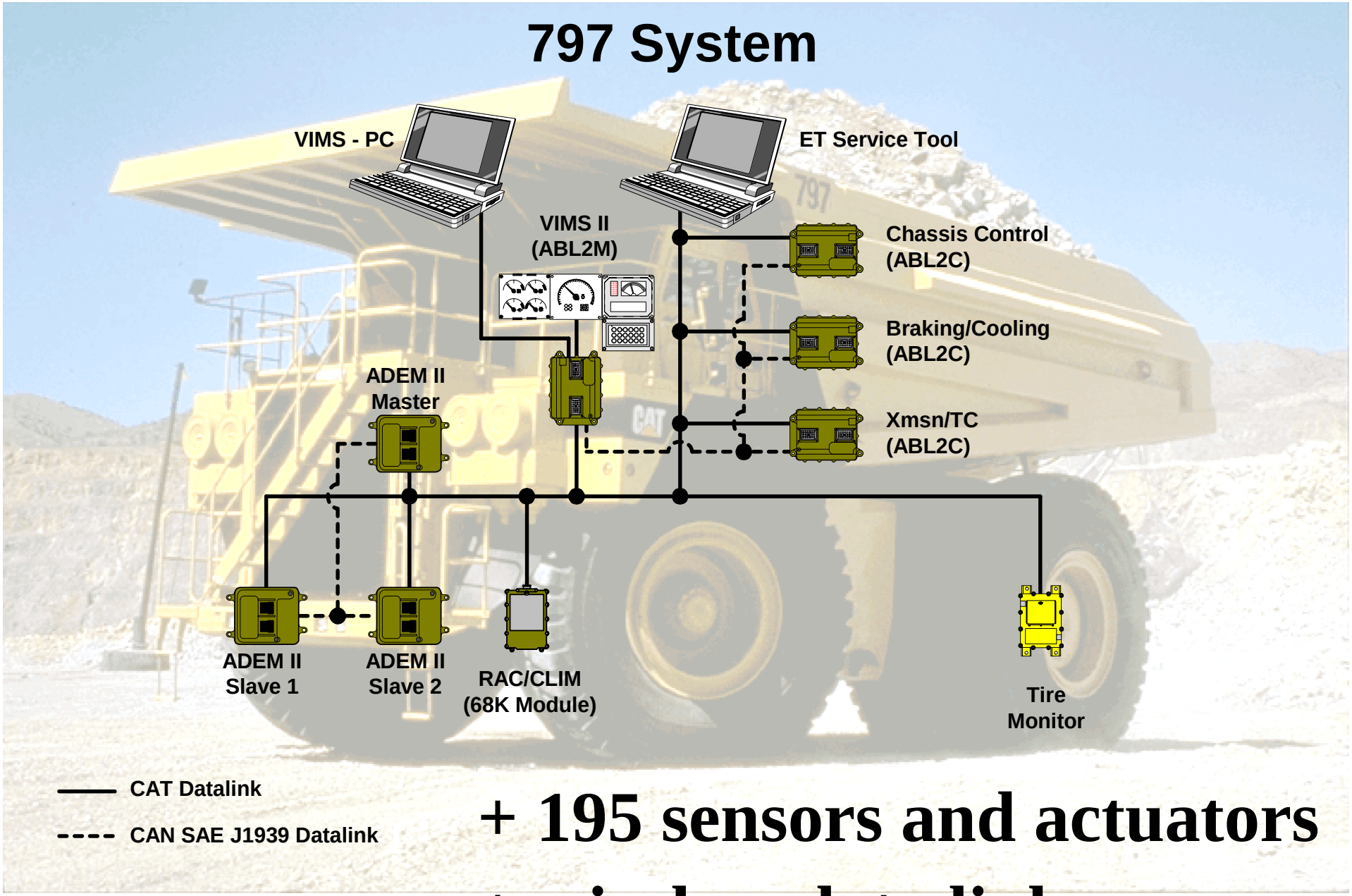
Approximated from EE Times,
March 20, 1995
Source: The Information Architects

Myth: Embedded Systems Are Trivial

- ***Reality:*** **Winning the game requires shoving 20 pounds into an 3 ounce sack**
  - Here's the design package for a household setback thermostat



Myth: Embedded Networking Is Novel



+ 195 sensors and actuators

+ wireless data link

797sys.vsd
6-18-98
dab/jwf
Warning: All paper copies of this document are uncontrolled

Myth: Discipline Will Solve Security Worries

- **Hacker's can't hurt your car if the infotainment system doesn't "talk" to the braking system**
  - Solution: don't put a connection between radio and brakes
- **Product idea: radio volume to achieve constant SNR**
  - Road noise based on wheel speed, tire pressure, road surface
  - Which sensor has the best information about this?
  - Anti-lock brake system
    - "Well, we'll just put in a fire-wall… surely that will be OK"
  - ***Reality:*** the connectivity will happen; denial is counterproductive
  - Prototype vehicle of a Big-3 manufacturer suffered failure when the radio speaker caused an engine controller malfunction

Other Security Concerns

- **Denial of Service Attacks?**
  - Will a SYN flood against your house's door lock keep you out?
- **"Regular" Hacker attacks?**
  - Will you get divorced because a script kiddie stored the Playboy channel on your TIVO?
  - Will malicious data mangling make your refrigerator order 500 gallons of milk?
- **Who is the sysadmin for your car?**
  - Will CERT point you to firmware patches for airbag?

Would *You* Drive A Car In Which:

"THE SOFTWARE is provided 'AS IS' and with all faults. THE ENTIRE RISK AS TO SATISFACTORY QUALITY, PERFORMANCE, ACCURACY, AND EFFORT (INCLUDING LACK OF NEGLIGENCE) IS WITH YOU."

(You will.)

Embedded Internet Challenges

- **Embedded systems actually have to work!**
  - When was the last time you rebooted your car?
  - They must degrade gracefully when components fail
  - They must be self-stabilizing in exceptional operating situations
- **Real-time control systems have to work in real time**
  - Closing control loops over Internet?
- **Configuration management has to be a non-issue**
  - Do you want to have to resolve device driver conflicts for your house?
- **Diverse devices have to talk to each other**
  - Need for common data representations & communication

RoSES Project As An Example

- **Robust Self-Configuring Embedded Systems**
- **Product families + automatic reconfiguration =**
  - Operation with failed components
  - Automatic integration of inexact spares
  - Automatic integration of upgrades
  - Fine-grain product family capability
- **Potential Impact:**
  - Logical component interfaces + configuration mgr.
  - Fine-grain software component run-time support
  - Architectures that are naturally resilient

RoSES = Product Families + Reconfiguration

- **Product Families:**
  - Different variations of components define products in a family
  - Each particular product has HW components with SW to provide features
  - With many possible HW components, there are many HW/SW combinations

- **Reconfiguration:**
  - RoSES is “Plug and play” for embedded systems – in factory and in the field

- **RoSES doesn’t care *why* it is doing reconfiguration!**
  - Component fails –
    triggers reconfiguration for degraded operation
  - Component replaced –
    reconfiguration to integrate repair part
  - New HW *or* SW component added (mid-life upgrade) –
    reconfiguration to upgrade system
  - New system built in factory –
    perform “re”-configuration for first time

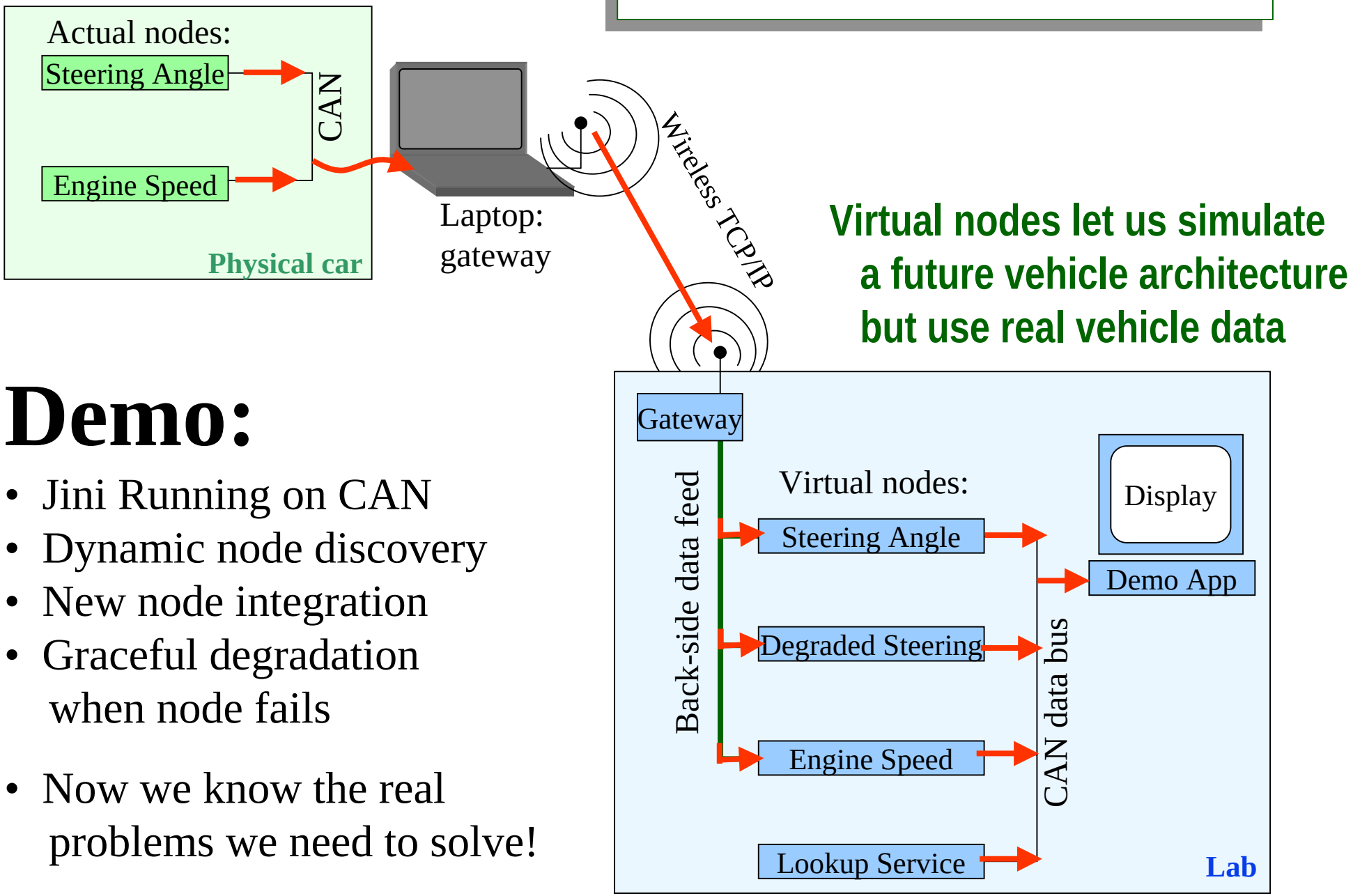
Why Does RoSES Matter?

- **Current approaches require specific engineering effort**
  - Every failure mode must be considered by design engineers
  - More components means exponentially more combinations
  - Soon there will be too many combinations to consider by hand

- **Enables shift to software-driven architectures**
  - Sensors, actuators, and computers are hardware components
  - Software can be treated as components too (not tied to HW)
  - Optimization problem is then to **automatically, in the field:**
    - Select which SW components make best use of limited resources
    - Map those SW components to available HW components
    - Ensure correct real-time operation

- **RoSES Goal:**
  Self-organizing software systems that make best possible use of available hardware resources
- **Maybe someday this will generalize to the Internet**

End-to-end Testbed Data Flow



Virtual nodes let us simulate a future vehicle architecture but use real vehicle data

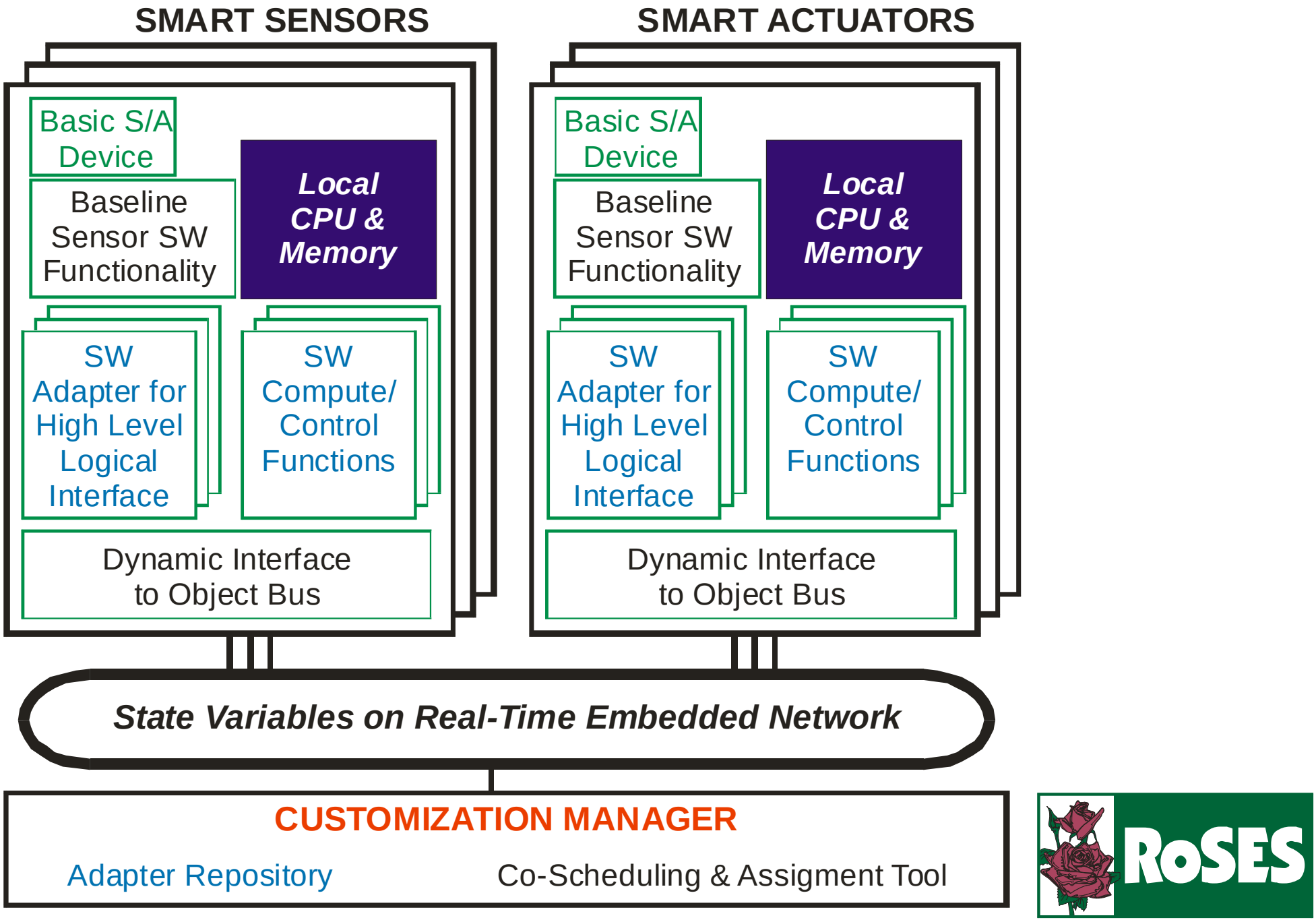
Demo:

- Jini Running on CAN
- Dynamic node discovery
- New node integration
- Graceful degradation when node fails

- Now we know the real problems we need to solve!

RoSES

Generic RoSES System Architecture



Jini Meets Embedded Networks (CAN)

- **Jini designed to be portable to “any” system**
  - Original implementation on TCP/IP
- **CAN (Control Area Network) is de facto automotive standard**
  - Global priority; short messages; periodic synchronized transmissions
  - It’s about as far away from Ethernet as you can get
- **What did we learn?**
  - Jini is portable to “anything” ***as long as it runs TCP/IP and RMI***
  - Reconfiguration time took many minutes without tricks
  - Plus all the problems with attempting “real time” Java
  - *Conclusion:* Many engineers used to desktop computing have not been exposed to the way the embedded world works

RoSES Research Questions

- **What is the best way to do embedded plug & play?**
  - We think RoSES will provide a reasonable alternative
- **What software architectures work best with RoSES?**
  - Is there such a thing as an architectural style that is naturally robust? (*"we think so"*)
- **Can we quantify robustness?**
  - Can we understand how to partially automate things like failure analysis? (*"we think so"*)
- **What design methodologies work for these systems?**
  - Can we represent all the special needs of distributed embedded real-time systems in UML? (*"perhaps in UML+++"*)
  - Can we teach people methodical design? (*"yes"*)

Embedded System Educational Issues

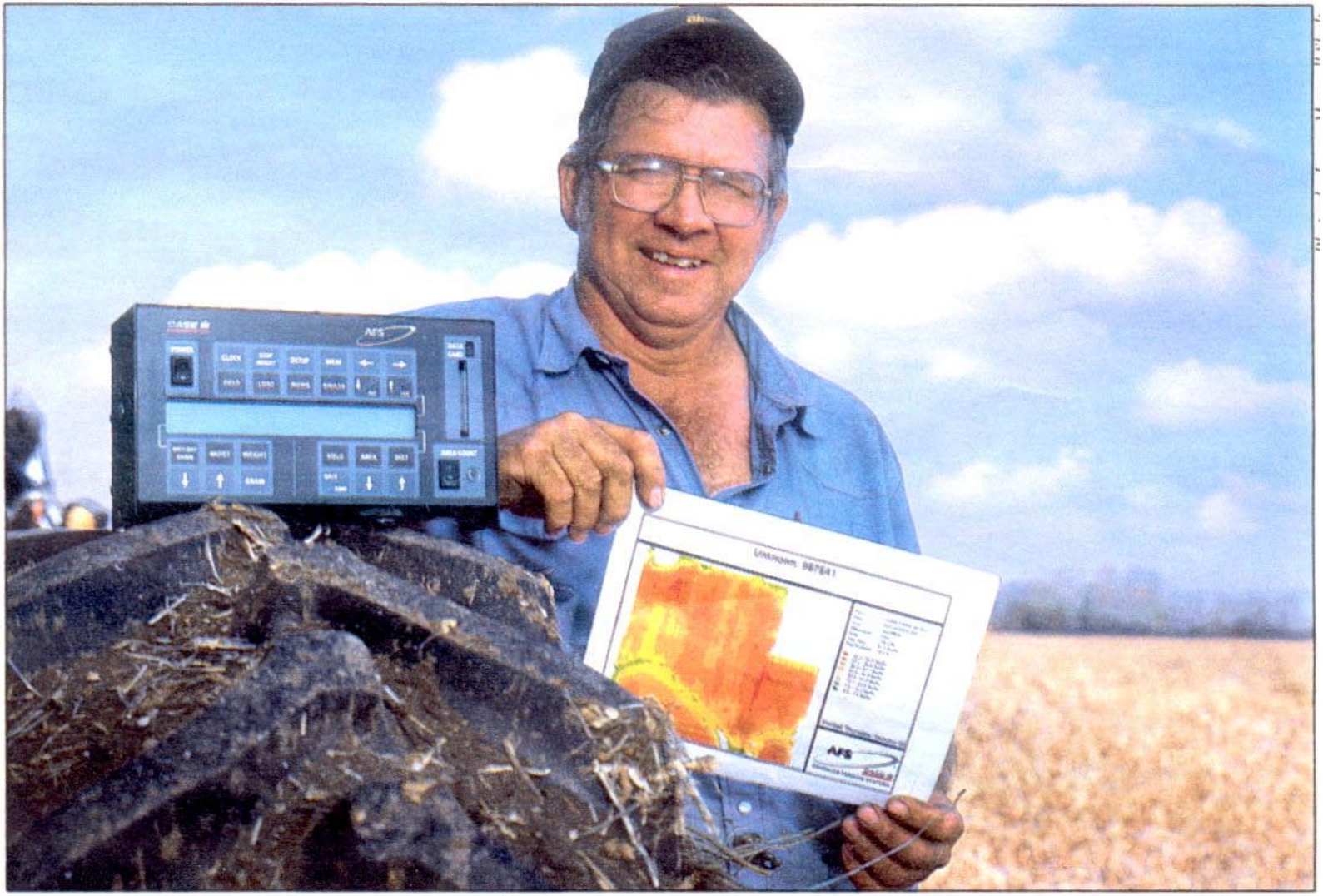
- **Embedded *system* engineers are generalists in an age of specialization**
  - Multi-disciplinary tradeoffs, often with design team size of one\
- **Need education way beyond traditional A/D, D/A, and assembly:**
  - Real time operating systems & scheduling
  - System design methodologies (requirements / design / test / *etc.*)
    - Many engineers need software/system engineering literacy
  - Distributed systems & distributed networks
    - Entirely different set of tradeoffs for embedded than for “regular” networks
  - Architectural approaches to distributed systems
  - Critical system design (dependability, safety)
  - Human/computer interfaces
  - Specialty skills: low power, design for particular constraints

Challenge Areas

- **Increase integration levels (including Analog)**
  - Hardware + Software + I/O + Storage + Human + Mechanical + logistics co-design
    - Ultra-fast CPUs or programmable logic are part of the equation
    - So is verification/certification of self-configuring systems
  - Optimizing for *System* (big picture) life cycle is ultimately what counts
- **How do you get ultra-dependability for only a buck?**
  - Dependability = Reliability + Security + . . .
  - Multi-vendor Integration without a single big OS vendor?
  - Would you trust your life to software on a $1 micro? (You will.)
- **Biggest opportunity**
  - Nobody cares if their car engine controller is “Intel Inside” (yet)

In The End, Being Useful Is What Matters

"IT SURE WOULD BE MORE WORK WITHOUT COMPUTERS," SAYS A SOYBEAN FARMER WHO RELIES ON HIGH-TECH HELP FOR HARVESTING.



HARVESTING BEANS AND DATA. Ted Sander, 52, a farmer from Moberly, Mo., uses an onboard computer to create maps that show which plots need more fertilizer, herbicide or pesticide.

[Parade Magazine]