Wireless Sensor Networks

Lecture 8 – CS252

Sensor Networks: The Vision

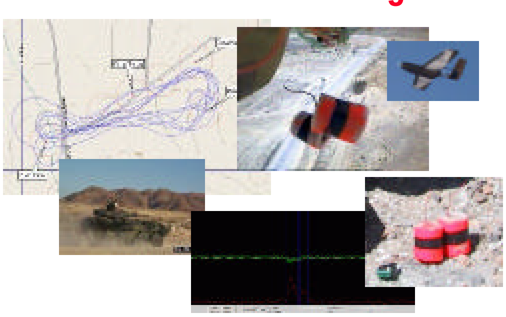
- Push connectivity out of the PC and into the real world
- Billions of sensors and actuators **EVERYWHERE!!!**
- Zero configuration
- Build everything out of CMOS so that each device costs pennies
- Enable wild new sensing paradigms

Why Now?

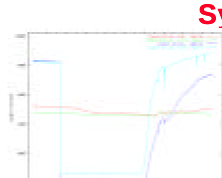
Combination of:
- Breakthroughs in MEMS technology
- Development of low power radio technologies
- Advances in low-power embedded microcontrollers

Real World Apps…

Vehicle Tracking

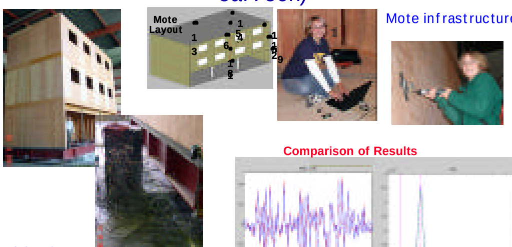


Cory Energy Monitoring/Mgmt System



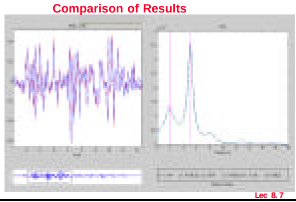
- 50 nodes on 4th floor
- 5 level ad hoc net
- 30 sec sampling
- 250K samples to database over 6 weeks

Structural performance due to multi-directional ground motions (Glaser & CalTech)

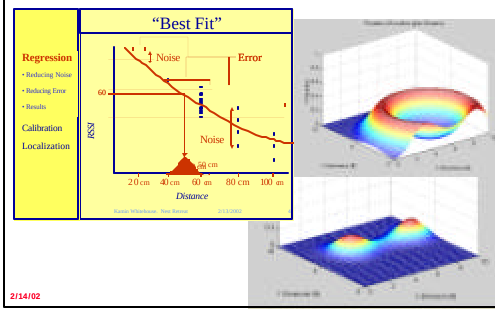


Wiring for traditional structural instrumentation + truckload of equipment

Comparison of Results



Node Localization



Sensor Network Algorithms

- Directed Diffusion – Data centric routing (Estrin, UCLA)
- Sensor Network Query Processing (Madden, UCB)
- Distributed Data Aggregation
- Localization in sensor networks (UCLA, UW, USC, UCB)
- Multi-object tracking/Pursuer Evader (UCB, NEST)
- Security

Recipe For Architectural Research

1. Take known workload
2. Analyze performance on current systems
3. Form hypothesis on ways of improving "performance"
4. Build new system based on hypothesis
5. Re-analyze same workload on new system
6. Publish results

Our Approach….

1. Hypothesize about requirements based on potential applications
2. Explore design space based on these requirements
3. Develop hardware platform for experimentation
4. Build test applications on top of hardware platform
5. Evaluate performance characteristics of applications
6. GOTO step 1 (hopefully you'll come up with a better set of requirements)

Sensor Node Requirements

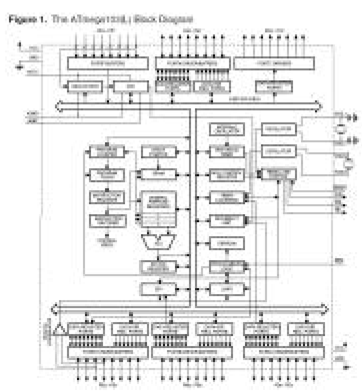
- Low Power, Low Power, Low Power…
- Support Multi-hop Wireless Communication
- Self Configuring
- Small Physical Size
- Can Reprogram over Network
- Meets Research Goals
  - Operating system exploration
  - Enables exploration of algorithm space
  - Instrumentation
  - Network architecture exploration

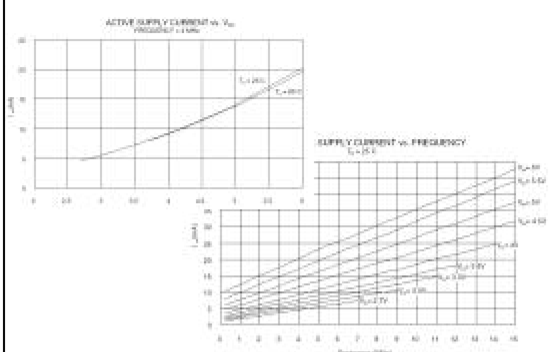
First Decision: The central controller

- What will control the device?
- Modern Microcontroller Sidebar
  - What's inside a microcontroller today?
  - What peripheral equipment do you need to support one?
  - How do you program one?

Major Axes of Microcontroller Diversity

- Flash based vs. SRAM based
  - Combination of FLASH and CMOS logic is difficult
- Internal vs. External Memory
- Memory Size
- Digital Only vs. On-chip ADC
- Operating Voltage Range
- Operating Current, Power States and wake-up times
- Physical Size
- Support Circuitry Required
  - External Clocks, Voltage References, RAM
- Peripheral Support
  - SPI, USART, I2C, One-wire
- Cycle Counters
- Capture and Analog Compare
- Tool Chain



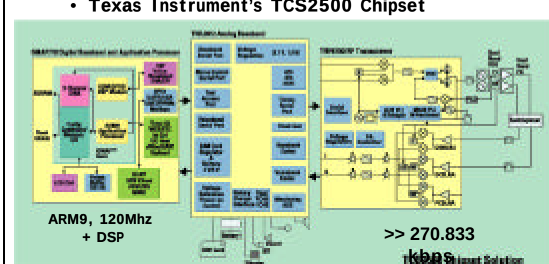


Second Decision: Radio Technologies

- Major RF Devices
  - Cordless Phones Digital/Analog
    - » Single Channel
  - Cellular Phones
    - » Multi-channel, Base station controlled
  - 802.11
    - » "wireless Ethernet"
  - Bluetooth
    - » Emerging, low-power frequency hopping

What is in your cell phone?

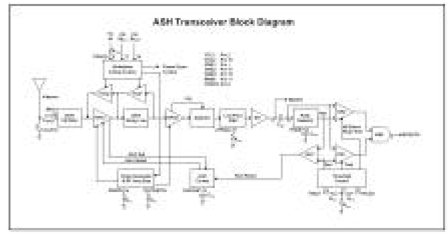
- Texas Instrument's TCS2500 Chipset



RFM TR1000 Radio

- 916.5 Mhz fixed carrier frequency
- No bit timing provided by radio
- 5 mA RX, 10 mA TX
- Receive signal digitized based on analog thresholds
- Able to operate in OOK (10 kb/s) or ASK (115 kb/s) mode
- 10 Kbps design using programmed I/O
- Design SPI-based circuit to drive radio at full speed
  - full speed on TI MSP, 50 kb/s on ATMEGA
- Improved Digitally controlled TX strength DS1804
  - 1 ft to 300 ft transmission range, 100 steps
- Receive signal strength detector

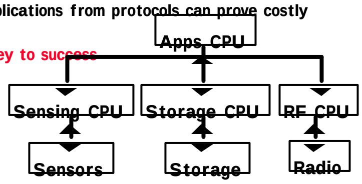
TR 1000 internals



Why not use federation of CPUs?

- Divide App, RF, Storage and Sensing
- Reproduce PC I/O hierarchy
- Dedicated communications processor could greatly reduce protocol stack overhead and complexity
- Providing physical parallelism would create a partition between applications and communication protocols
- Isolating applications from protocols can prove costly

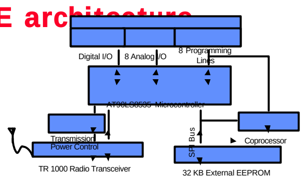
Flexibility is Key to success



Can you do this with a single CPU?

The RENE architecture

- Atmel AT90LS8535
  - 4 Mhz 8-bit CPU
  - 8KB Instruction Memory
  - 512B RAM
  - 5mA active, 3mA idle, <5uA powered down
- 32 KB EEPROM
  - 1-4 uj/bit r/w
- RFM TR1000 radio
  - Programmed I/O
  - 10 kb/s - OOK
- Network programmable
- 51-pin expansion connector
- GCC based tool/programming chain



1.5"x1" form factor

What is the software environment?

- Do I run JINI? Java?
- What about a real time OS?
- IP? Sockets? Threads?
- Why not?

TinyOS

- OS/Runtime model designed to manage the high levels of concurrency required
- Gives up IP, sockets, threads
- Uses state-machine based programming concepts to allow for fine grained concurrency
- Provides the primitive of low-level message delivery and dispatching as building block for all distributed algorithms

Key Software Requirements

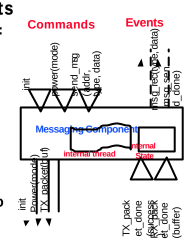
- Capable of fine grained concurrency
- Small physical size
- Efficient Resource Utilization
- Highly Modular
- Self Configuring

State Machine Programming Model

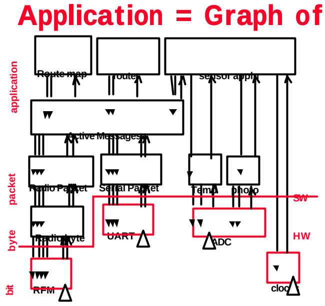
- System composed of state machines
- Command and event handlers transition modules from one state to another
  - Quick, low overhead, non-blocking state transitions
- Many independent modules allowed to efficiently share a single execution context

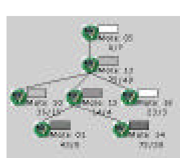
Tiny OS Concepts

- Scheduler + Graph of Components
  - constrained two-level scheduling model: threads + events
- Component:
  - Commands,
  - Event Handlers
  - Frame (storage)
  - Tasks (concurrency)
- Constrained Storage Model
  - frame per component, shared stack, no heap
- Very lean multithreading
- Efficient Layering



Application = Graph of Components





Example: ad hoc, multi-hop routing of photo sensor readings

3450 B code
226 B data

Graph of cooperating state machines on shared stack

System Analysis

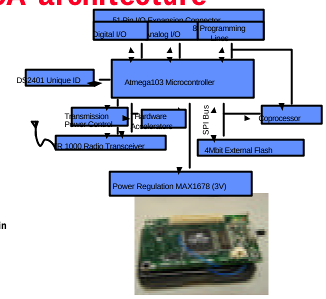
- After building apps, system is highly memory constrained
- Communication bandwidth is limited by CPU overhead at key times. Communication has bursty phases.
- Where did the Energy/Time go?
  - 50% of CPU used when searching for packets
  - With 1 packet per second, >90% of energy goes to RX!

Architectural Challenges

- Imbalance between memory, I/O and CPU
  - Increase memory (Program and Data) by selecting different CPU
- Time/energy spent waiting for reception
  - Solution: Low-power listening software protocols
- Peak CPU usage during transmission
  - Solution: Hardware based communication accelerator

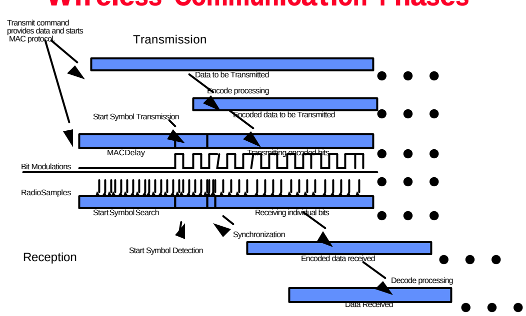
The MICA architecture

- Atmel ATMEGA103
  - 4 Mhz 8-bit CPU
  - 128KB Instruction Memory
  - 4KB RAM
  - 5.5mA active, 1.6mA idle, <1uA powered down
- 4 Mbit flash (AT45DB041B)
  - SPI interface
  - 1-4 uj/bit r/w
- RFM TR1000 radio
  - 50 kb/s – ASK
  - Communication focused hardware acceleration
- Network programmable
- 51-pin expansion connector
  - Analog compare + interrupts
- GCC based tool/programming chain



2xAA form factor Cost-effective power source

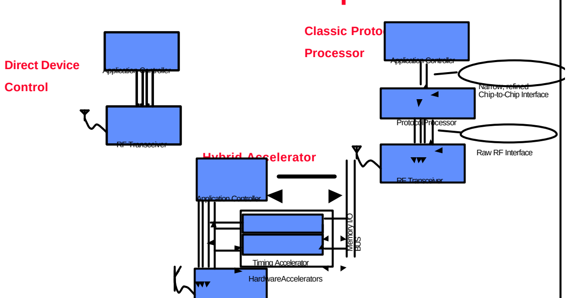
Wireless Communication Phases



Radio Interface

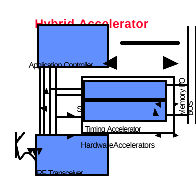
- Highly CPU intensive
- CPU limited, not RF limited in low power systems
- Example implementations
  - RENE node:
    - » 19,200 bps RF capability
    - » 10,000 bps implementation, 4Mhz Atmel AVR
  - Chipcon application note example:
    - » 9,600 bps RF capability
    - » Example implementation 1,200bps with 8x over sampling on 16 Mhz Microchip PICmicro (chipcon application note AN008)

Node Communication Architecture Options



Accelerator Approach

- Standard Interrupt based I/O perform start symbol detection
- Timing accelerator employed to capture precise transmission timing
  - Edge capture performed to +/- 1/4 us
- Timing information fed into data serializer
  - Exact bit timing performed without using data path
  - CPU handles data byte-by-byte



Results from accelerator approach

- Bit Clocking Accelerator
  - 50 Kbps transmission rate
    - » 5x over Rene implementation
  - >8x reduction in peak CPU overhead
- Timing Accelerator
  - Edge captured to +/- ¼ us
    - » Rene implementation = +/- 50 us
  - CPU data path not involved

Power Optimization Challenge

Scenario:
- 1000 node multi-hop network
- Deployed network should be "dormant" until RF wake-up signal is heard
- After sleeping for hours, network must wake-up with-in 20 seconds

Goal:
- Minimize Power consumption

What are the important characteristics?

- Transmit Power?
- Receive Power consumption of the radio?
- Clock Skew?
- **Radio turn-on time?**

Solutions

- Minimize the time to check for "wake-up" message
- "check" time must be greater than length of wake-up message
- If data packets are used for wake up signal, then "check" time must exceed packet transmission time
- Instead use long wake-up tone

Tone-based wake-up protocol

- Each node turns on radio for 200us and checks for RF noise
- If present, then node continues to listen to confirm the tone
- If not, node goes back to sleep for 4 seconds
- Resulting duty cycle? .0002/4 = .005 %.
- 200us due to wake-up time of the radio

Project Ideas

- Tos_sim ++
  - RF usage modeling
  - Cycle-accurate simulation
  - Nono-joule-accurate simulation
- Tiny application specific VM
  - Source program lang
  - Intermediate representation
  - Mobile code story
  - Communication model
- Analysis of CPU Multithreading/Radical core architectures
- Federated Architecture Alternative

Project Ideas (2)

- Closed loop system analysis
  - Simulation of closed loop systems
  - Impact of design decisions on latency
- Channel characterization, Error Correction
- Stable, energy efficient, multi-hop communication implementation
- Scalable Reliable Multicast Analog
- Sensor network specific CPU design
- “Passive Vigilance” Circuits
- Power Harvesting
- Correct Architectural Balance (Memory:I/O:CPU)
- Self-diagnosis/watchdog architecture
- Cryptographic Support
- Alternate Scheduling Models – Perhaps periodic real-time
- Explore query processing/content based routing
- Design and build your own X

Microcontroller Alternatives

- Atmega 163
  - same pin out as RENE
  - 2x memory
  - Can self-reprogram

  Not enough memory
- ARM Thumb
  - lower power consumption, lower voltage
  - greater performance
  - poor integration → slow radio

  Peripheral support missing
- TI MSP340
  - Superior performance
  - 1/10 power consumption
  - Better integration

  No GCC, tool chain missing