Signals, Instruments, and Systems – W11
Field Instruments and Sensor Networks for Environmental Monitoring – Energy Management, Operation Principles, and Deployment
Outline

• Power management in field instruments
  – Power consumption
  – Power generation
  – Power storage

• Sensor networks for environmental monitoring:
  – Basic principles
  – The Sensorscope project
  – Mica-Z – a desktop sensor node
Power Management in Field Instruments
Motivation

• Energy is one of the key bottlenecks for autonomous/unattended operation of embedded systems

• Field instruments as extreme examples
  – Power consumption
  – Power generation
  – Power storage
Power Consumption
Power

\[ P = U \cdot I \]

Examples:
- MICAz:
  2 * 1,5V battery, 25 mA power consumption \(\rightarrow 2 \times 1.5V \times 0.025A = 80mW\)
  (standby: 80 \(\mu\)W)
- Campell Scientific:
  3D ultrasonic anemometer:
  1.2W or 2.4W
- SHT1x temperature and humidity sensor:
  2\(\mu\)W – 3mW
Energy

\[ E = P \cdot t \]

Examples:

- Rechargeable battery (NiMH):
  \[ 1.2V \times 2000mAh = 2400mWh = 2.4Wh \]

- Rechargeable battery (LiPo):
  \[ 3.7V \times 1340mAh = 4958mWh = 4.958Wh \]

- Derived values:
  
  (remember 1 Ws = 1 Joule)
Managing Power Consumption

Consumption vs. capabilities (example 1):

- Disdrometer 1 (“tipping bucket”) (≈ 0W)
  - no snow, sleet
  - no freezing
  - no drop statistics
  - resolution
- Disdrometer 2 (laser) (≈ 10W)
  + snow, sleet
  + freezing
  + drop statistics
  - expensive
  - delicate
- Disdrometer 3 (hot plate) (≈ 100W)
  + snow, sleet
  + freezing
  + simple
  - drop statistics
Managing Power Consumption

Consumption vs. capabilities (example 2):

- **Anemometer 1 (cup) (≈ 0W)**
  - 1D (main direction)
  - no freezing
  - temporal resolution (0.3 Hz)
  - minimal wind speed high
- **Anemometer 2 (ultrasonic) (≈ 10W)**
  + 2D
  + *some* snow, sleet
  + *some* freezing
  + temporal resolution (up to 60Hz)
  - expensive
- **Anemometer 3 (Anemometer 2+heater) (≈ 100W)**
  + 2D
  + snow, sleet
  + freezing
  + temporal resolution
  - expensive
Managing Power Consumption

Consumption vs. processing speed:

- \( P \sim f_{\text{clock}} \)
- Energy/operation = const

Consumption vs. transmission power:

- \( P = f(P_{RF}) \)
- sometimes linear: \( P \sim P_{RF} \)
- often: “sweet spot”

Source: MSP430 data sheet
Sensor Node Energy Roadmap

- Deployed (5W)
- PAC/C Baseline (.5W)
- (50 mW)
- (1mW)

Rehosting to Low Power COTS (10x)
- System-On-Chip
- Adv Power Management Algorithms (50x)

Source: ISI & DARPA PAC/C Program
# Communication/Computation Technology Projection

<table>
<thead>
<tr>
<th></th>
<th>1999 (Bluetooth Technology)</th>
<th>2004</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication</td>
<td>(150nJ/bit)</td>
<td>(5nJ/bit)</td>
</tr>
<tr>
<td></td>
<td>1.5mW*</td>
<td>50uW</td>
</tr>
<tr>
<td>Computation</td>
<td>~ 190 MOPS</td>
<td>(5pJ/OP)</td>
</tr>
</tbody>
</table>

Assume: 10kbit/sec. Radio, 10 m range.

**Large cost of communications relative to computation continues**
Free Space Path Loss (Friis Law)

From Week 7 slides

• Signal power decay in air:

\[ L = \left( \frac{4\pi df}{c} \right)^2 \]

\[ L_{dB} = 20 \log_{10}(d) + 20 \log_{10}(f) - 147.56 \]

• Proportional to the square of the distance \( d \)
• Proportional to the square of the frequency \( f \)
  – high frequency = high loss
  – low frequency = low bandwidth
Ex.: Mica-Z vs. TinyNode

From Week 7 slides

Mica-Z (Crossbow)
- Microcontroller:
  - ATmega128L
  - TinyOs
- Transceiver:
  - Chipcon CC2420
  - 2.4 GHz carrier
  - Throughput: up to 250k bps
  - Range: up to 75 m

TinyNode (Shockfish)
- Microcontroller:
  - TI MSP430
  - TinyOs
- Transceiver:
  - Semtech XE 1205
  - 868 and 915 MHz carriers
  - Throughput: up to 153k bps
  - Range: up to 2 km
Power Generation
Power generation methods

- Solar
- Wind
- Temperature difference (Seebeck Effect)
- Vibration
- Hydro
Solar power generation

1m²

1000W → 150W → 135W → 95W

100W/m²  Efficiency: 10-20%  Efficiency: 90%  Efficiency: 70%  Efficiency: 90%

0.1m² per W (e.g., 32x32 cm²)
Power Storage
Batteries

- primary (non rechargeable, red)
  “often a good idea”
- secondary (rechargeable, blue)

Other important parameters:
- number of cycles (few 100 – few 1000) optimal conditions
- cold temperature behaviour

Interesting alternative: super capacitor
- 30 Wh/kg
- expensive
- very high number of cycles (> 100’000)
From a Single Instrument to a Network
Motivation for Sensor Networks

What if we could monitor events which …

– have a large *spatial* and *temporal* distribution
– require *in-situ* measurements
– take place in hard to access places
– generate data which need to be available in *real-time*
Motivation for Sensor Networks

What would we need for that?

A device which …

– is cheap – so we can distribute *many* of it
– is reliable – so we can measure for a long *time*
– uses little power – battery/solar cell powered
– has a radio – so it can *communicate*
– can potentially move – so it can potentially *relocate*
Building a Sensor Network: Key Concepts
Introduction

Temperature
Humidity
Light
Topology

?
Topology
Topology
Topology

GPRS
Topology

Pros
• Very simple!
• No restrictions in sensor locations

Cons
• The server may be quite far from the stations
• A long-range link may consume a lot of energy!
Topology (TinyNode Example)

- Sensor: 0.5 mA
- MSP430: 3 mA
- XE1205: 50 mA
Topology (SensorScope Example)

Power Consumption [mA]

- Sensor: 0.5 mA
- MSP430: 3 mA
- XE1205: 50 mA
- GPRS: 700 mA

14x times the XE1205!
Topology

Assuming four AA batteries, 1.2 V, 2000 mAh

- Sensor: **167 days**
- MSP430: **28 days**
- Short range radio: **1.7 days**
- Long range radio: **8 hours**
Topology

But makes sense on projects such as OpenSense (see next week)!
Topology

Short range

Sink

GPRS
Topology

Recall Friis law again:

\[ L = \left( \frac{4\pi df}{c} \right)^2 \]

Example: To transmit over 5 Km we can using 868 MHz we can:

- One hop of 5 km: \( L = 106 \text{ dB} \)
- Two hops of 2.5 km: \( L = 99 \text{ dB} \)
- Five hops of 1 km: \( L = 92 \text{ dB} \)

Energy is the main issue !!!
Multi-hop Sensor Network
Multi-hop Sensor Network

Pros

• Only one car battery in the network
• The sensor network has extended monitoring coverage
• Multiple routes for stations to communicate with the sink
• Auto configurable network (robustness)

Cons

• Significantly more complicated
• Data rate is not increased
• Unable to use directional antennas
Multi-hop WSNs

Implementation:

- Neighborhood discovery
- Data routing
- Time synchronization
- Duty-cycling (radio management)
Neighborhood

Hello messages (Beacons) are one common method:
1. Node A sends a HELLO message to its neighbors (B, C, and D).
3. Node B sends a HELLO message to its neighbors (A, C, and D).
4. …
Neighborhood

Hello messages (Beacons) are one common method:
1. Node A sends a HELLO message to its neighbors (B, C, and D).
3. Node B sends a HELLO message to its neighbors (A, C, and D).
4. ...
Neighborhood

Hello messages (Beacons) are one common method:
1. Node A sends a HELLO message to its neighbors (B, C, and D).
3. Node B sends a HELLO message to its neighbors (A, C, and D).
4. …
Hello messages (Beacons) are one common method:
1. Node A sends a HELLO message to its neighbors (B, C, and D).
3. Node B sends a HELLO message to its neighbors (A, C, and D).
4. …
Neighborhood

What information do we need about our neighbors?

• Distance to sink
• Last time heard
• Link quality
Neighborhood

Node E’s neighborhood table

<table>
<thead>
<tr>
<th>Id</th>
<th>Age</th>
<th>Distance</th>
<th>Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>2 min</td>
<td>1 hop</td>
<td>87%</td>
</tr>
<tr>
<td>C</td>
<td>2 min</td>
<td>1 hop</td>
<td>98%</td>
</tr>
<tr>
<td>F</td>
<td>4 min</td>
<td>2 hops</td>
<td>74%</td>
</tr>
<tr>
<td>G</td>
<td>1 min</td>
<td>2 hops</td>
<td>93%</td>
</tr>
</tbody>
</table>

A few remarks:

• Only the distance to the sink is stored.
• Neighborhood discovery can’t be done only once!
• We need to estimate link qualities!
Neighborhood

Variations of simple schema:

- Each node sends $X$ beacons per minute.
- Number of beacons received per minute are stored.
- Quality is estimated over the past $Y$ minutes by counting losses.

Example ($X = 10; Y = 4$):

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$t-4$</td>
<td>$t-3$</td>
<td>$t-2$</td>
<td>$t-1$</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>7</td>
<td>8</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

Quality $= 0.8$

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$t-4$</td>
<td>$t-3$</td>
<td>$t-2$</td>
<td>$t-1$</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>8</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

Quality $= 0.71$

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$t-4$</td>
<td>$t-3$</td>
<td>$t-2$</td>
<td>$t-1$</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>6</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

Quality $= 0.64$
Time Synchronization

Weather conditions, especially temperature and humidity, may have a significant effect on hardware.

Crystal oscillators are highly impacted by temperature!
Time Synchronization

![Graph showing temperature and time drift over days for Indoor, Outdoor, and Freezer conditions.](image-url)
Time Synchronization

Nodes need to know the time to:
• Timestamp packets
• Synchronize actions (e.g., taking samples, transmitting data)

How do we get time:
• Fully decentralized: Every node gets the time itself
• Partially centralized: Time is propagated from reference nodes
Time Synchronization

Every node gets the time:

• Atomic clock receivers:
  • Cheap (both energy and $)
  • Complexity
  • Limited coverage

• GPS:
  • Coverage
  • Complexity
  • High cost (energy and $)

• GPRS: same as GPS with less coverage

What about a partially centralized approach?
Time Synchronization

For instance, the sink serve as time reference node
Field Deployment of Sensor Networks

- The SensorScope Project
SensorScope

Microcontrollers are inside hermitically sealed boxes, attached on a mounting pole with up to seven external sensors.

Price: 4000-5000 CHF
SensorScope
SensorScope

Shockfish TinyNode with TinyOS 2.x
MSP430 16-bit microcontroller @ 8MHz
48KB ROM, 10KB RAM, and 512KB flash memory
Semtech XE1205 radio transceiver @ 868MHz, 76Kbps
SensorScope

162x140mm solar panel
12Ah NiMH rechargeable
Visualizing Filter "Public Stations"

Local time: 2:59 (GMT+1)
SensorScope

Many previous successful deployments

97 stations deployed at EPFL (one year)
SensorScope

Many previous successful deployments

16 stations deployed at Le Génépi to monitor conditions leading to dangerous mudslides (two months)
MicaZ – An Example of a Sensor Node and Embedded System
MICA mote family

- designed in EECS at UC Berkeley
- manufactured/marketed by Crossbow
  - several thousand produced
  - used by several hundred research groups
  - about CHF 250/piece
- variety of available sensors
MICAz

- **Atmel ATmega128L**
  - 8 bit microcontroller, ~8MHz
  - 128kB program memory, 4kB SRAM
  - 512kB external flash (data logger)
- **Chipcon CC2420**
  - 802.15.4 (Zigbee)
- **2 AA batteries**
  - about 5 days active (15-20 mA)
  - about 20 years sleeping (15-20 µA)
- **TinyOS**
Perception - Sensor Board

• MTS 300 CA
  – Light (Clairex CL94L)
  – Temp (Panasonic ERT-J1VR103J)
  – Acoustic (WM-62A Microphone)
  – Sounder (4 kHz Resonator)
An operating system (OS) is an interface between hardware and user applications.

It is responsible for **the management and coordination of tasks** and **the sharing of the limited resources** of the computer system.

A typical OS can be decomposed into the following entities:

- **Scheduler**, which is responsible for the sharing of the processing unit (microprocessor or microcontroller).
- **Device drivers**, which are low-level programs that manage the various devices (sensors, actuators, secondary memory storage devices, etc.).
- **Memory management unit**, which is responsible for the sharing of the memory (virtual memory).
- **Optional**: Graphical User Interface, File System, Security, etc.

Most “OS” for embedded systems include these two entities only!
Computation - TinyOS

- Minimal OS designed for Sensor Networks
- Event-driven execution
- Programming language: nesC (C-like syntax but supports TinyOS concurrency model)
- Widespread usage on motes
  - MICA (ATmega128L)
  - TELOS (TI MSP430)
- Provided simulator: TosSim
Communication - 802.15.4 / Zigbee

• Emerging standard for low-power wireless monitoring and control
  – 2.4 GHz ISM band (84 channels), 250 kbps data rate

• Chipcon/Ember CC2420: Single-chip transceiver
  – 1.8V supply
    • 19.7 mA receiving
    • 17.4 mA transmitting
  – Easy to integrate: Open source drivers
  – O-QPSK modulation (Code Division Multiple Access, CDMA); “plays nice” with 802.11 and Bluetooth
Communication - Standards

Complexity/power/cost

Bluetooth
802.15.4 Zigbee
CC1000

802.11b
802.11a
802.11g

11 Mbps
54 Mbps

720 kbps
250 kbps
38.4 kbps

Data rate
Conclusion
Take Home Messages

• Power is most often the key design constraint in embedded systems
• Efficient power management strategies can decrease the consumption by several orders of magnitude
• Power is comparably difficult to generate and store
• Sensor networks enable environmental monitoring in remote locations and of difficult-to-measure processes
• Real-world deployments may be highly unpredictable!
• The Mica-Z desktop node is the result of a pioneering open source (both hardware and software) project at UC Berkeley; it is still widely used in educational settings and has stimulated significantly the creation of the research community in wireless sensor networks
Additional Literature – Week 11

Books

Pointers
• Sensorscope: http://sensorscope.epfl.ch/
• Swiss Experiment: http://www.swiss-experiment.ch/
• Permasense: http://www.permasense.ch/
• GITEWS: http://www.gitews.de
• WiSARD network: http://wisardnet.nau.edu/
• CENS: http://research.cens.ucla.edu/
• TinyOS: http://www.tinyos.net/