SYSTEM ARCHITECTURE DIRECTIONS FOR NETWORKED SENSORS

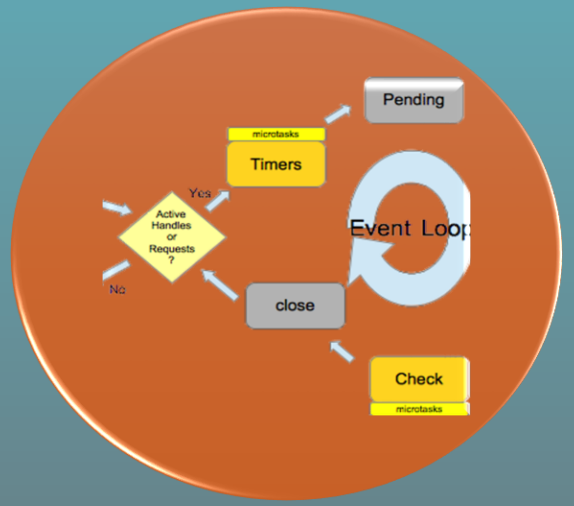
Χάρης Φίλης ΑΕΜ : 9449 ERTS2021

System Architecture Directions for Networked Sensors~Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, Kristofer Pister[{jhill, szewczyk, awoo, culler}@cs.berkeley.edu,{shollar, pister}@eecs.berkeley.edu].

23/3/2021

TO THIS END

Design & Develop Small Device – Class



Tiny Event-Driven OS
TinyOS

Supports Modularity and Concurrency-Intesive Operations

TYPES OF SENSORS/DETECTORS/TRANDUCERS

VISION AND IMAGING SENSORS
TEMPERATURE SENSORS
RADIATION SENSORS
PROXIMITY SENSORS
PRESSURE SENSORS
POSITION SENSORS
PHOTOELECTRIC SENSORS
PARTICLE SENSORS
MOTION SENSORS
METAL SENSORS
LEVEL SENSORS
LEAK SENSORS
HUMIDITY SENSORS
GAS AND CHEMICAL SENSORS
FORCE SENSORS
FLOW SENSORS
FLAW SENSORS
FLAME SENSORS
ELECTRICAL SENSORS
CONTACT SENSORS
NON-CONTACT SENSORS

Quantity being Measured	Input Device (Sensor)	Output Device (Actuator)
Light Level	Light Dependant Resistor (LDR) Photodiode Photo-transistor Solar Cell	Lights & Lamps LED's & Displays Fibre Optics
Temperature	Thermocouple Thermistor Thermostat Resistive Temperature Detectors	Heater Fan
Force/Pressure	Strain Gauge Pressure Switch Load Cells	Lifts & Jacks Electromagnet Vibration
Position	Potentiometer Encoders Reflective/Slotted Opto-switch LVDT	Motor Solenoid Panel Meters
Speed	Tacho-generator Reflective/Slotted Opto-coupler Doppler Effect Sensors	AC and DC Motors Stepper Motor Brake
Sound	Carbon Microphone Piezo-electric Crystal	Bell Buzzer Loudspeaker

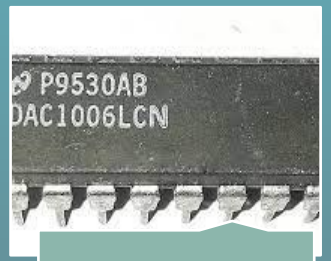
Avg. Component Size : 1 in^2
Avg. Component Power : 1 watt

Light	Heat | Temperature	Position	Force | Pressure
Speed	Sound	Movement	Chemical Presence

In each of these areas, the technology
is crossing a critical threshold that makes
networked sensors
an exciting regime to apply systematic design methods

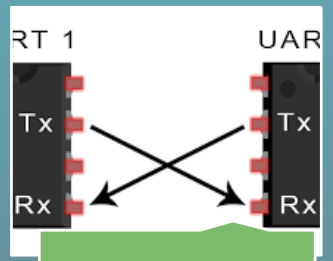
PERIPHERALS OF A MODERN MCU

Non – volatile memory



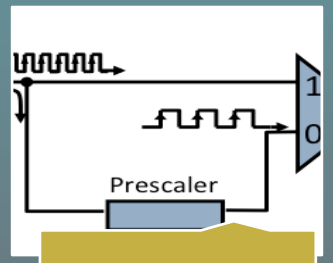
DAC

ADC



UART

Interrupt Controller(WIC,PIC etc.)



Counters|Timers

RF-CMOS components

THERE ARE TWO ISSUES FOR THE PROTOTYPE SYSTEM

- *these devices are concurrency intensive* –> several different flows of data must be kept moving simultaneously
- *The system must provide efficient modularity* -> hardware specific and application specific components must snap together with little processing and storage overhead.

CONTENTS

NETWORKED SENSOR CHARACTERISTICS

System Architecture Directions for Networked Sensors~Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, Kristofer Pister[{jhill, szewczyk, awoo, culler}@cs.berkeley.edu,{shollar, pister}@eecs.berkeley.edu].

NETWORKED SENSOR CHARACTERISTICS

Small physical size and low power consumption

- size and power constrain the processing, storage, and interconnect capability of the basic device
- the software must make efficient use of processor and memory while enabling low power communication

Concurrency-intensive operation

- Main mode -> information flow form node to node with modest processing on-the-fly
- This should happen simultaneously (e.g., in notes multi-hop and bridging problem)
- Each of the flow generally involve many low-level events interleaved with higher-level processing
- This high-level processing will extend over multiple *real-time events*

Limited Physical Parallelism and Controller Hierarchy

- Less Independent Controllers
- Less capabilities of the Controllers
- Lower level of processor-memory-switch level interconnect

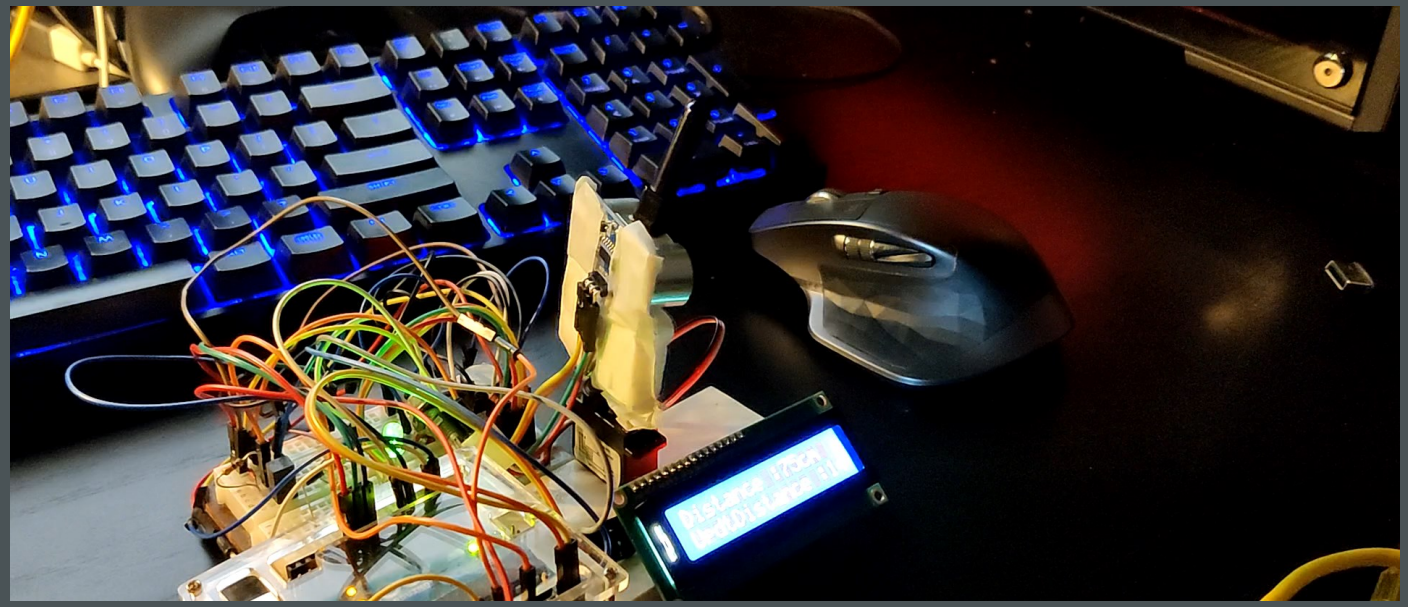
Diversity in Design and Usage

- Networked sensor devices -> application specific(!general purpose)
- Physical differences between devices (due to wide range of applications)
- Easily assemble just the software components (required for the application) from the hardware components
- Generic development environment needed
- Component migration across hardware/software should be natural

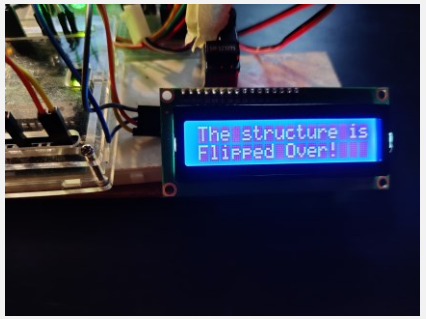
Robust Operation

- Many devices, no supervision, long time operation application
- The communication cost for cross device failover(ανακατεύθυνση πληροφορίας) is prohibitive
- => Enhance Reliability of each device
- => Increase Application Reliability(tolerate individual device failures)

DEMO RADAR | PARKING ASSISTANT | TRESPASSER DETECTOR

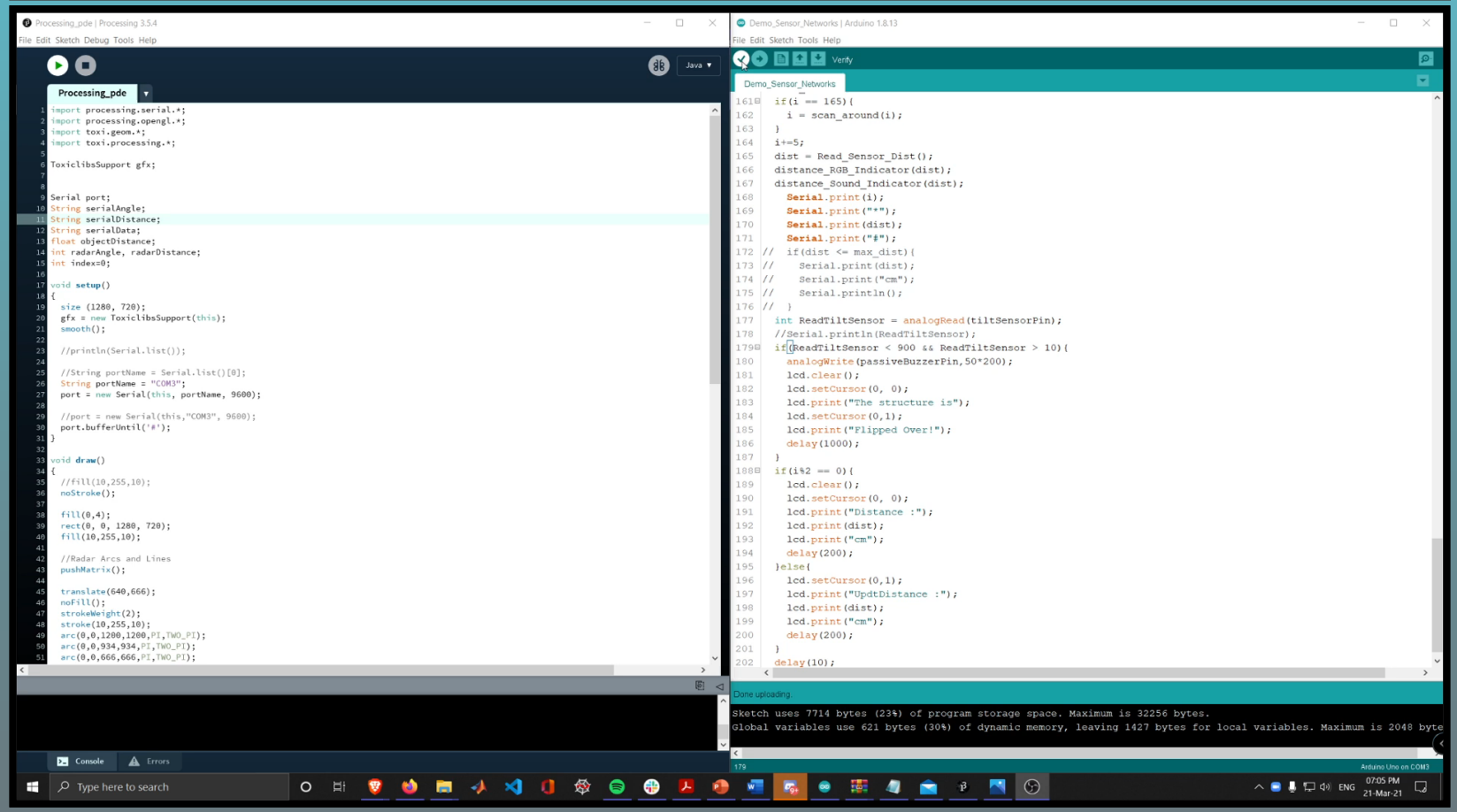


MY *DEMO*



System Architecture Directions for Networked Sensors~*Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, Kristofer Pister[{jhill, szewczyk, awoo, culler}@cs.berkeley.edu,{shollar, pister}@eecs.berkeley.edu].*

DEMO SIGNAL PROCESSING



Credits(Processing lib) : **Karsten Schmidt**|postspectacular

EXAMPLE DESIGN POINT

- networked sensor platform used based on *James McLurkin. Algorithms for distributed sensor networks. In Masters Thesis for Electrical Engineering at the Univeristy of California, Berkeley, December 1999.*

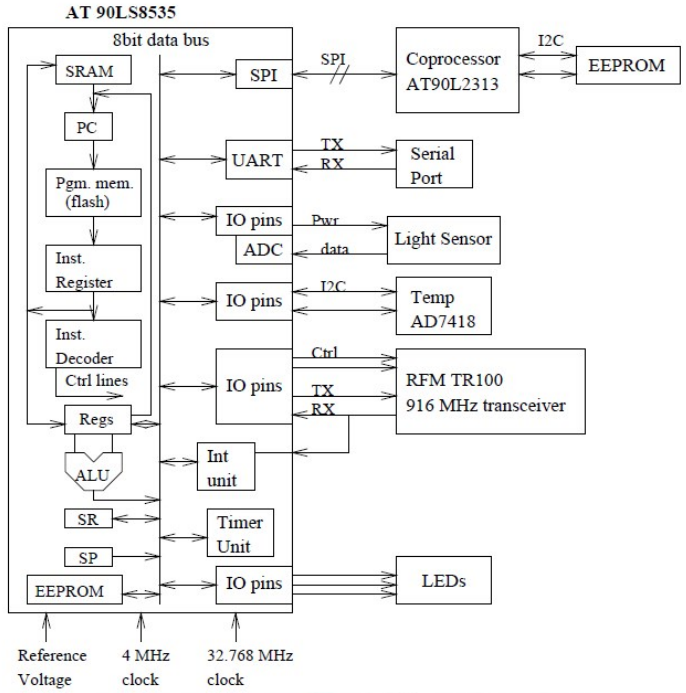


Figure 1: Photograph and schematic for representative network sensor platform

HARDWARE ORGANIZATION

- Atmel AVR 8-Bit RISC processor (ATMEL 90LS8535) – (not noteworthy in the paper).
- Memory Constrained System [8 KB of flash(Program Memory), 512 bytes of SRAM (data memory),]
- (timers and interrupt controllers)More noteworthy are the three sleep modes: ***idle***, which just shuts o the processor, ***power down***, which shuts off everything but the watchdog and asynchronous interrupt logic necessary for wake up, and ***power save***, which is similar to the power down mode, but leaves an asynchronous timer running.

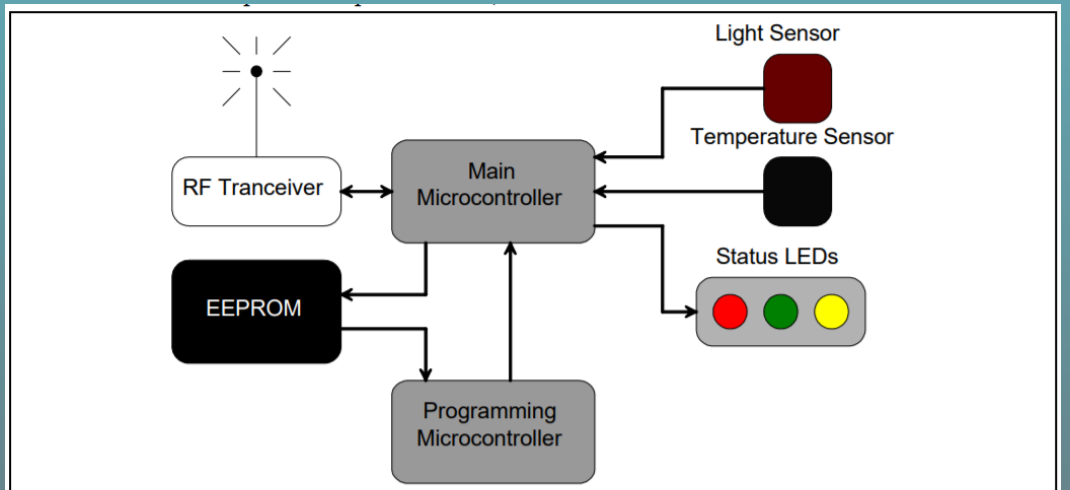


Figure 29: Block diagram of the macromote hardware. The main microcontroller is responsible for almost all the computational duties. To download new software, the user needs to download new software. This processor then puts the new code into the EEPROM, programs the main processor, and resets the system. This enables remote

HARDWARE ORGANIZATION

- Three LEDs represent outputs connected through general I/O port (ex. display digital values or status)
- The Photo-Sensor represents an analog input device with simple control lines.(directed to power save). Signal is directed to ADC.
- The **radio** represents an asynchronous input/output device with **hard real time constraints.**

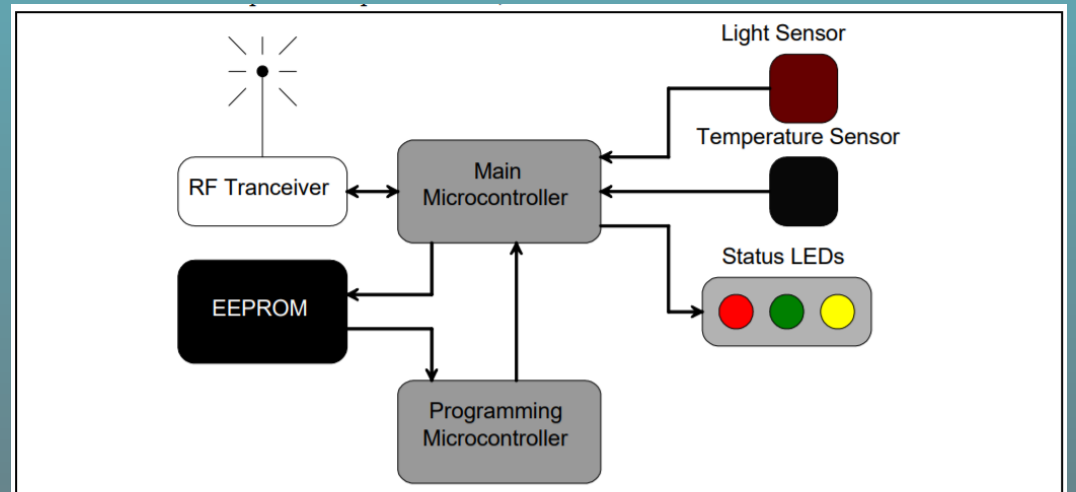


Figure 29: Block diagram of the macromote hardware. The main microcontroller is responsible for almost all the computational duties. To download new software, the user needs to download new software. This processor then puts the new code into the EEPROM, programs the main processor, and resets the system. This enables remote

HARDWARE ORGANIZATION

- Temperature Sensor: (analog device AD7418) digital sensor which has A/D converter integrated. Chip-to-chip protocol used is I2C (two-wire protocol as used in Demo lcd i2c)*
- The Serial Port: Important asynchronous bit level device with byte-level controller support. I/O pins connected to UART.(more in notes)
- **Coprocessor :** 1)The coprocessor is connected in a way that allows it to reprogram the main microcontroller. 2)Sensor reprogramming by transferring data from the network into the coprocessors EEPROM. 3)Alternative use of co-processor is extra storage.

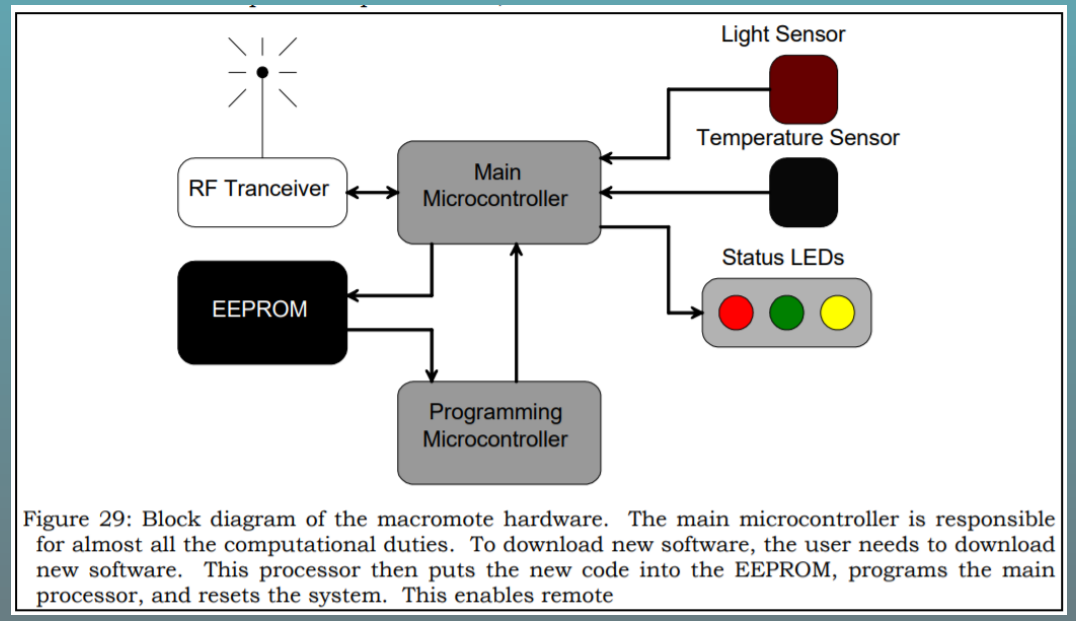
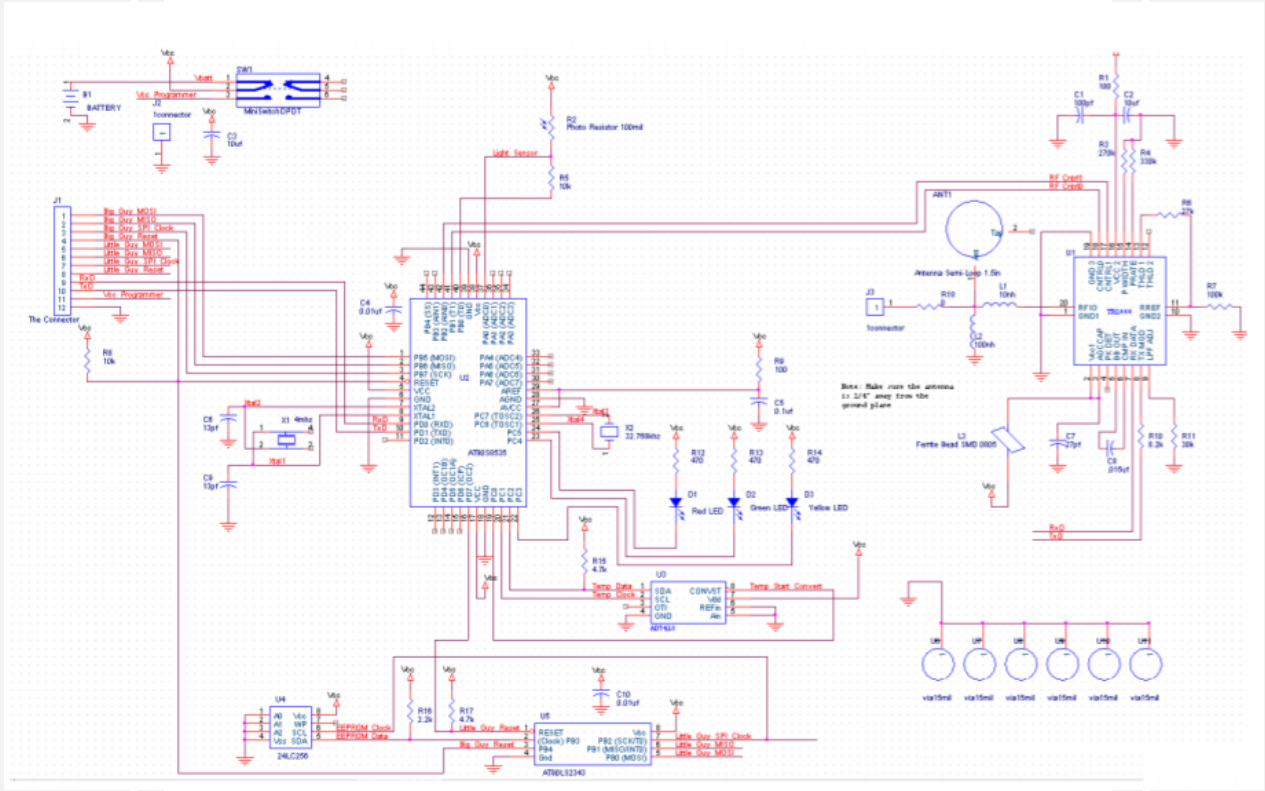


Figure 29: Block diagram of the macromote hardware. The main microcontroller is responsible for almost all the computational duties. To download new software, the user needs to download new software. This processor then puts the new code into the EEPROM, programs the main processor, and resets the system. This enables remote

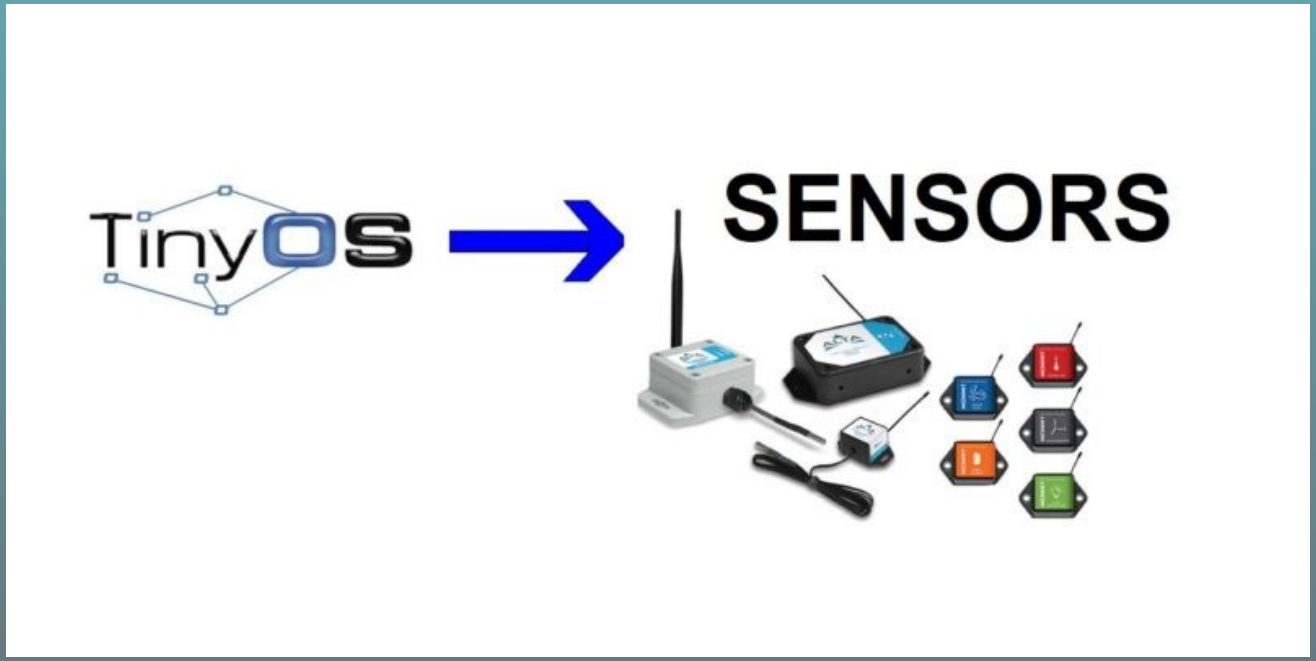
MACROMOTE SCHEMATIC



POWER CHARACTERISTICS TABLE

Component	Active (mA)	Idle (mA)	Inactive (μA)
MCU core (AT90S8535)	5	2	1
MCU pins	1.5	-	-
LED	4.6 each	-	-
Photocell	.3	-	-
Radio (RFM TR1000)	12 tx	-	5
Radio (RFM TR1000)	4.5 rx	-	5
Temp (AD7416)	1	0.6	1.5
Co-proc (AT90LS2343)	2.4	.5	1
EEPROM (24LC256)	3	-	1

TINY MICRO-THREADING OPERATING SYSTEM (**TINYOS**)



ISSUES ADDRESSED AND SOLUTIONS

1. Large numbers of concurrent flows
2. Numerous outstanding events

Solutions Given :

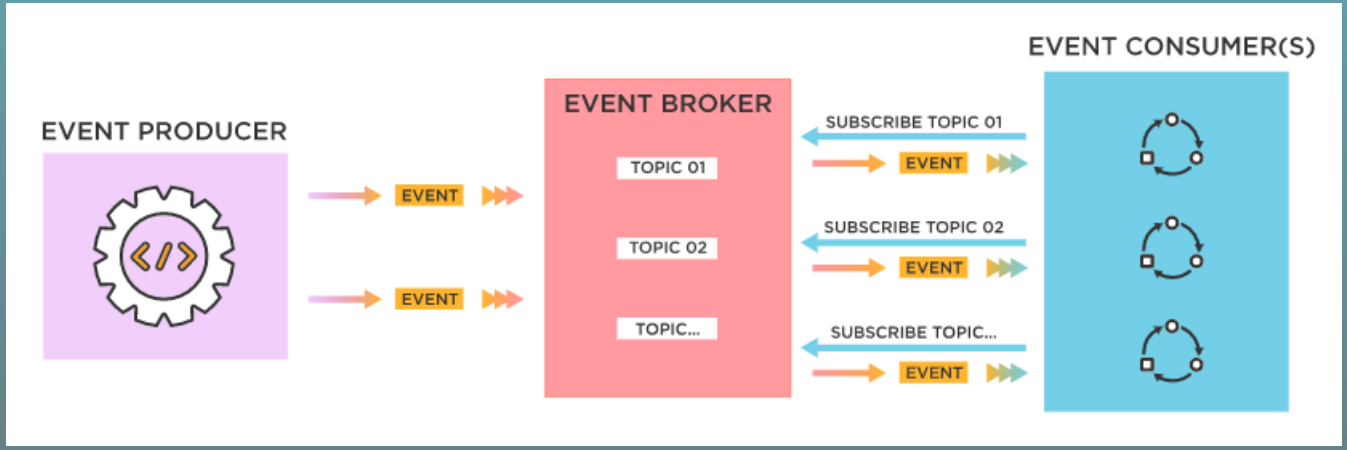
- Physical Parallelism
- Virtual Machines

} tackle this by building an extremely efficient multithreading engine

This engine

- Has 2 levels of scheduling structure (TAM,CILK)
- Finite State Machine execution model but more programmable.
- Scales with the current technology trends supporting both smaller, tightly integrate designs as well as the crossover of software components into hardware.

TINY OS IS BASED ON AN *EVENT DRIVEN* MODEL



TINY OS IS BASED ON AN *EVENT DRIVEN* MODEL

The collection of tasks associated with an event are handled rapidly, and no blocking or polling is permitted. Unused CPU cycles are spent in the sleep state as opposed to actively looking for an interesting event. Additionally, with real-time constraints the calculation of CPU utilization becomes simple { allowing for algorithms that adjust processor speed and voltage accordingly

TinyOS Design

- A)Tiny scheduler (FIFO)
- B)Graph of components

- B1) command handlers
- B2) set of event handlers
- B3) encapsulated fixed-size frame
- B4) bundle of simple tasks(preempted by events)

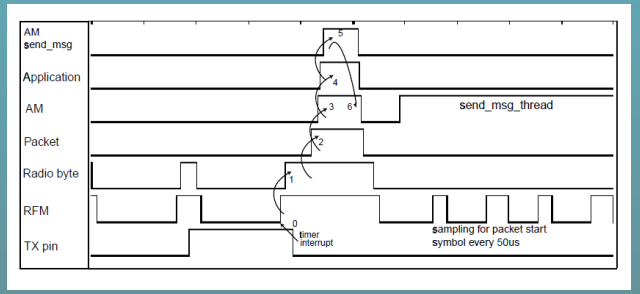
TinyOS Design/Example Component

Component Name	Code Size (bytes)	Data Size (bytes)
Multihop router	88	0
AM_dispatch	40	0
AM_temperature	78	32
AM_light	146	8
AM	356	40
Packet	334	40
RADIO_byte	810	8
RFM	310	1
Photo	84	1
Temperature	64	1
UART	196	1
UART_packet	314	40
I2C_bus	198	8
Procesor_init	172	30
TinyOS scheduler	178	16
C runtime	82	0
Total	3450	226

Table 2: Code and data size breakdown for our complete system. Only the processor init, the TinyOS scheduler, and the C runtime are required for every application, the other components are included as needed.

COMPONENT TYPES (COMPONENT MODELS)

- Hardware Abstraction Components(RFM)
- Synthetic Hardware Components(Radio Byte)
- High Level Software Components(Messaging Module)



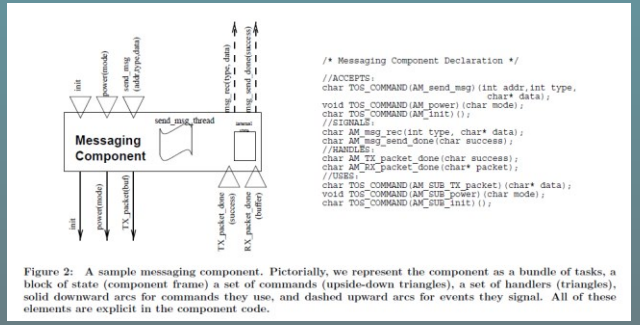


Figure 2: A sample messaging component. Pictorially, we represent the component as a bundle of tasks, a block of state (component frame) a set of commands (upside-down triangles), a set of handlers (triangles), solid downward arcs for commands they use, and dashed upward arcs for events they signal. All of these elements are explicit in the component code.

PUTTING ALL TOGETHER

- *A sample configuration of a networked Sensor + Routing topology.*
- *Details breakdown for energy consumption of each component across each layer for Transmission and Reception of packet.*

Components	Packet reception breakdown	Percent CPU Utilization	Energy (nJ/bit)
AM	0.05%	0.02%	0.33
Packet	1.12%	0.51%	7.58
Radio handler	26.87%	12.16%	182.38
Radio decode task	5.48%	2.48%	37.2
RFM	66.48%	30.08%	451.17
Radio Reception	-	-	1350
Idle	-	54.75%	-
Total	100.00%	100.00%	2028.66
Components	Packet transmission breakdown	Percent CPU Utilization	Energy (nJ/bit)
AM	0.03%	0.01%	0.18
Packet	3.33%	1.59%	23.89
Radio handler	35.32%	16.90%	253.55
Radio encode task	4.53%	2.17%	32.52
RFM	56.80%	27.18%	407.17
Radio Transmission	-	-	1800
Idle	-	52.14%	-
Total	100.00%	100.00%	4317.89

Table 4: Details breakdown of work distribution and energy consumption across each layer for packet transmission and reception. For example, 66.48% of the work in receiving packets is done in the RFM bit-level component and it utilizes 30.08% of the CPU time during the entire period of receiving the packet. It also consumes 451.17nJ per bit it processes. Note that these measurements are done with respect to raw bits at the physical layer with the bit rate of the radio set to 100 μs/bit using DC-balanced ON-OFF keying.

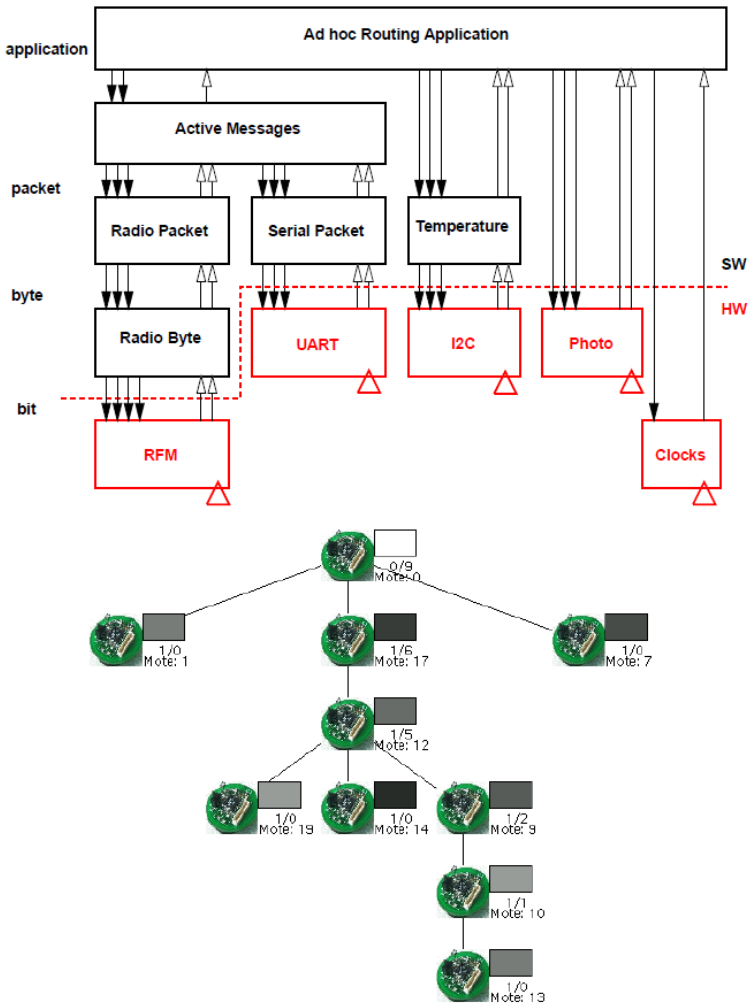


Figure 3: A sample configuration of a networked sensor, and the routing topology created by a collection of distributed sensors.

System Architecture Directions for Networked Sensors~*Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, Kristofer Pister[{jhill, szewczyk, awoo, culler}@cs.berkeley.edu,{shollar, pister}@eecs.berkeley.edu].*

Component Name	Code Size (bytes)	Data Size (bytes)
Multihop router	88	0
AM_dispatch	40	0
AM_temperature	78	32
AM_light	146	8
AM	356	40
Packet	334	40
RADIO_byte	810	8
RFM	310	1
Photo	84	1
Temperature	64	1
UART	196	1
UART_packet	314	40
I2C_bus	198	8
Procesor_init	172	30
TinyOS scheduler	178	16
C runtime	82	0
Total	3450	226

Table 2: Code and data size breakdown for our complete system. Only the processor init, the TinyOS scheduler, and the C runtime are required for every application, the other components are included as needed.

Operations	Cost (cycles)	Time (μs)	Normalized to byte copy
Byte copy	8	2	1
Post an Event	10	2.5	1.25
Call a Command	10	2.5	1.25
Post a task to scheduler	46	11.5	6
Context switch overhead	51	12.75	6
Interrupt (hardware cost)	9	2.25	1
Interrupt (software cost)	71	17.75	9

Table 3: Overhead of primitive operations in TinyOS

EVALUATION

RELATED WORK/OS

Name	Preemption	Protection	ROM Size	Configurable	Targets
pOSEK	Tasks	No	2K	Static	Microcontrollers
pSOSystem	POSIX	Optional		Dynamic	PII → ARM Thumb
VxWorks	POSIX	Yes	≈ 286K	Dynamic	Pentium → Strong ARM
QNX Neutrino	POSIX	Yes	$> 100K$	Dynamic	Pentium II → NEC chips
QNX Realtime	POSIX	Yes	100K	Dynamic	Pentium II → 386's
OS-9	Process	Yes		Dynamic	Pentium → SH4
Chorus OS	POSIX	Optional	10K	Dynamic	Pentium → Strong ARM
Ariel	Tasks	No	19K	Static	SH2, ARM Thumb
CREEM	data-flow	No	560 bytes	Static	ATMEL 8051

ARCHITECTURAL IMPLICATIONS

- The utilization of a microcontroller per device(networked sensor device) is an architectural option(not a requirement)
- The interconnection of such systems will need to support an event-based communication model.
- Tradeoffs arise between power consumption and communication, flexibility and functionality
- the Radio Byte component needs to become a hardware abstraction rather than synthetic hardware

“It is clear that there is a strong tie between the software execution model and the hardware architecture that supports it. Just as SPEC benchmarks attempted to evaluate the impact of architectural changes on the entire system in the workstation regime, we have attempted to begin the systematic analysis architectural alternatives in the network sensor regime.”


Dziękuję
Ευχαριστω
Kiitos
有り難う
Obrigado
谢谢
Hvala
Tack
תודה
Merci
Danke
Gracias
Thank you
Grazie
شكرا
謝謝
ありがとう
감사합니다
Mulţumesc
Спасибо
Asante

Ερωτήσεις;