Introduction to Motes

Portions adapted from:

Crossbow, Inc.

Bill Maurer
maurer@dsplabs.com
DSP Labs, Livermore Ca., USA

www.intel-research.net/berkeley/

1

Why Smart Dust ?

- Advances in low power wireless communication technology and micro-electromechanical sensors (MEMS) transducers

- "Digital Nervous System"
- "Physical Internet"
- "Ubiquitous Computing"

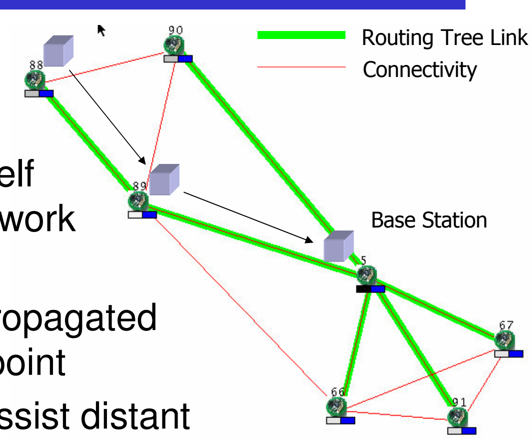
2

NEST Technology

- How do you combine sensing, communication and computation into a complete architecture ?
- What are the requirements of the software?
- Networked-Embedded-Systems-Technology

3

Ad hoc sensing



- Autonomous nodes self assembling into a network of sensors
- Sensor information propagated to central collection point
- Intermediate nodes assist distant nodes to reach the base station

4

Today’s Hardware

- Assembled from off-the-shelf components
- 8bit MCU
- Low-Power Radio
- Sensors
- I/O

5

Key Software Requirements

- Capable of fine grained concurrency
- Small physical size
- Efficient resource utilization
- Highly modular

6

What is TinyOS ?

- An Open-Source Development Environment
- A Simple Operating System
- A Programming Language and Model
- A Set of Services

7

TinyOS – Development Environment

- Windows and Linux
- Multiple Hardware Platforms
  - Not only Crossbow
- Multiple Sensors
  - Not only Crossbow
- Debugging Tools
- Reference Applications

8

TinyOS – A Simple Operating System

- Scheduler
- Concurrency Intensive
- Limited Resources – SW Components for Efficient Modularity

9

TinyOS – A Programming Language and Model

- Separation of construction and composition:
  - programs are built out of *components*
- Specification of component behavior in terms of set of *interfaces*
- Components are statically wired to each other via their interfaces.
  - This increases runtime efficiency

10

TinyOS - Services

- Radio, MAC, Messaging, Routing
- Sensor Interfaces
- Power Management
- Security
- Debug
- Time

11

Why TinyOS ?

- Unix Analog (aka 1969)
  - A Uniform Programming Language
    - C
  - A Uniform Abstraction
    - E.g, device abstraction
  - Open Source
    - Many Different Developers
    - Many Different Needs
  - Many Tools

12

Who Controls TinyOS ?

- UC Berkeley Invented
  - 1st written by Jason Hill in 2000
  - Large portion of development changed to Intel-Berkeley Research Lab

- Intel-Berkeley Research Lab has largest role today in core 'OS' components
  - www.intel-research.net/berkeley/

13

Real-World Deployments

Great Duck Island

http://www.greatduckisland.net/

Center for Embedded Network Sensing

http://www.cens.ucla.edu/

14

Introduction to TinyOS and nesC Programming

- TinyOS Kernel Design and Implementation
- nesC Software Concepts and Basic Syntax
- nesC Code Lab
- TinyOS Packet Networking and PC Base Station Lab

15

TinyOS Design Goals

- Support Networked Embedded Systems
  - asleep but remain vigilant to stimuli
  - bursts of events and operations
- Support Mica Hardware
  - power, sensing, computation, communication
- Support Technological Advances
  - keep scaling down
  - smaller, cheaper, lower power

16

TinyOS Design Options

- Can't Use Existing RTOS's
  - Microkernel Architecture
    - VxWorks, QNX, WinCE, PalmOS
  - Execution Similar to Desktop Systems
    - PDA's, Cell Phones, Embedded PC's
  - More Than a Order of Magnitude Too Heavy & Slow
  - Energy Hog

17

TinyOS Design Conclusion

- **Similar** to Building Networking Interfaces
  - Data Driven Execution
  - Manage Large # of Concurrent Data Flows
  - Manage Large # of Outstanding Events
- **Add:** Managing Application Data Processing
- Conclusion: Need a Multi Threading Engine
  - Extremely Efficient
  - Extremely Simple

18

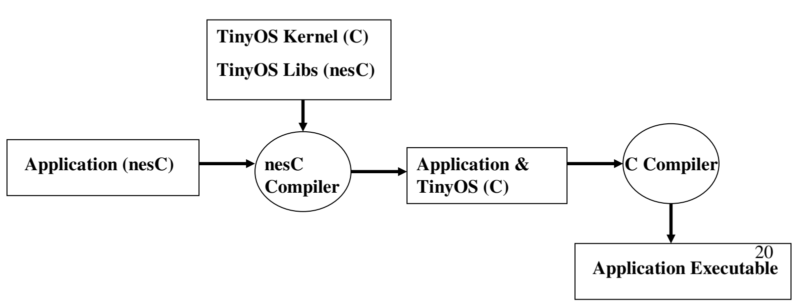
TinyOS Kernel Design

- TinyOS Kernel: 2 Level Scheduling Structure
  - Events
    - Small Amount of Processing
    - E.g. Timer, ADC Interrupts
    - Can Interrupt Longer Running Tasks
  - Tasks
    - Not Time Critical
    - Tasks - Larger Amount of Processing
    - E.g. Computing an Average on an Array
    - Run to Completion WRT other Tasks
      - Implies Only Need a Single Stack

19

TinyOS Applications Under The Hood

- Application is created in the nesC Language
  - Details of nesC Forthcoming
- nesC Programming Language Supports the TinyOS Kernel Design (Events and Tasks)



20

The TinyOS Kernel Under The Hood (nesC Compiler C Code Output)

```
int   main(void) {

  RealMain$hardwareInit();
  TOSH_sched_init();

  RealMain$StdControl$init();
  RealMain$StdControl$start();
  RealMain$Interrupt$enable();

  while (1) {
     TOSH_run_task();
   }
}
```

Hardware and Kernel Initialization

Application Initialization

Infinite Loop

```
static inline void TOSH_run_task(void) {
  while (TOSH_run_next_task())
    ;
  TOSH_sleep();
  TOSH_wait();
}
```

```
bool TOSH_run_next_task(void) {
  uint8_t old_full;
  void (*func)(void );
  if (TOSH_sched_full == TOSH_sched_free) {
     return 0;
   }
  else {
     old_full = TOSH_sched_full;
     TOSH_sched_full++;
     TOSH_sched_full &= TOSH_TASK_BITMASK;
     func = TOSH_queue[(int )old_full].tp;
     TOSH_queue[(int )old_full].tp = 0;
     func();
     return 1;
   }
}
```

1. First Run All Tasks in the Task Queue (Strictly a FIFO)

2. Then Sleep (In Low Power Mode)

3. And Wait for an Interrupt

Task Runs To Completion (But May Be Interrupted By An Event)

21

Overhead of TinyOS Primitive Operations

Operation	**Cost(cycles)**	**Time(uSecs)**	**Normalized to Byte Copy**
Byte Copy	8	2	1
Signal an Event	10	2.5	1.25
Call a Command	10	2.5	1.25
Schedule a Task	46	11.5	6
Context Switch	51	12.75	6
Hardware Interrupt (hw)	9	2.25	1
Hardware Interrupt (sw)	**71**	**17.75**	**9**

	Code Size(bytes)	**Data Size(bytes)**
Processor Init	172	30
Scheduler	178	16
C runtime	82	0
	432	**46**

22

TinyOS/nesC Application Notes

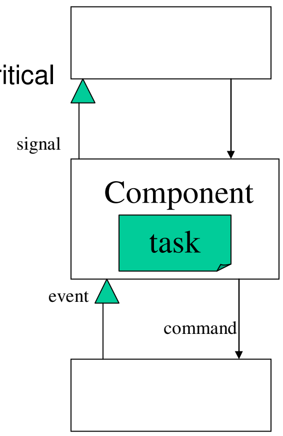
- Everything is Static
  - No Dynamic Memory (no malloc)
  - No Function Pointers
  - No Heap
- nesC Compiler Analysis
  - Data Race Conditions
  - Function Inlining
  - Development Made Easier
  - Robustness Improved

Application Memory Map

- Text/code - Executable Code
  - In the 128K Program Flash
- data – Program Constants
  - In the 128K Program Flash
- bss - Variables
  - In the 4K SRAM
- Free Space - Fixed (No Dynamic Memory)
- stack - Grows Down in the Free Space

TinyOS Concepts Embodied by nesC – Tasks, Events, Commands

- **Tasks**
  - Background computation, non-time critical
- **Events**
  - Time critical
  - External Interrupts
  - Originator gives a '**Signal**'
  - Receiver gets/accepts an '**Event**'
- **Command**
  - Function call to another Component
  - Cannot **Signal**



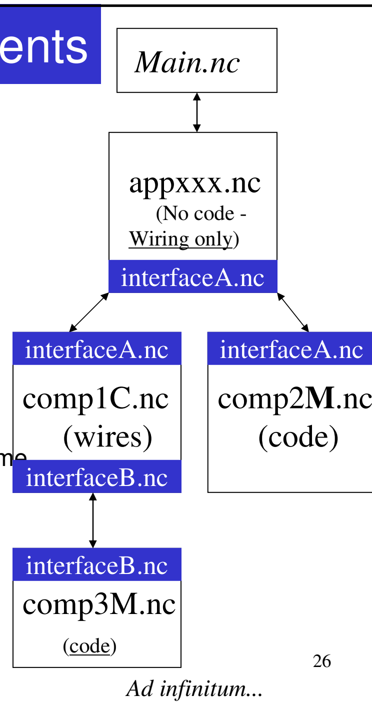
25

Concepts of SW Components

- **Interfaces** (xxx.nc)
  - Specifies functionality to outside world
  - Tell outside world
    - what commands can be called
    - what events need handling
- Software Components
  - **Module** (xxxM.nc)
    - Code file, code implementation
    - It codes the **Interface**
  - **Configuration** (xxxC.nc)
    - Linking/wiring of components
    - When top level app, drop C from filename xxx.nc
    - optional **Module**

TinyOS app Blink – Blinks the Red LED

BlinkM.nc
Blink.nc



26

Blink.nc Application - A top level **configuration** SW component used to form an executable

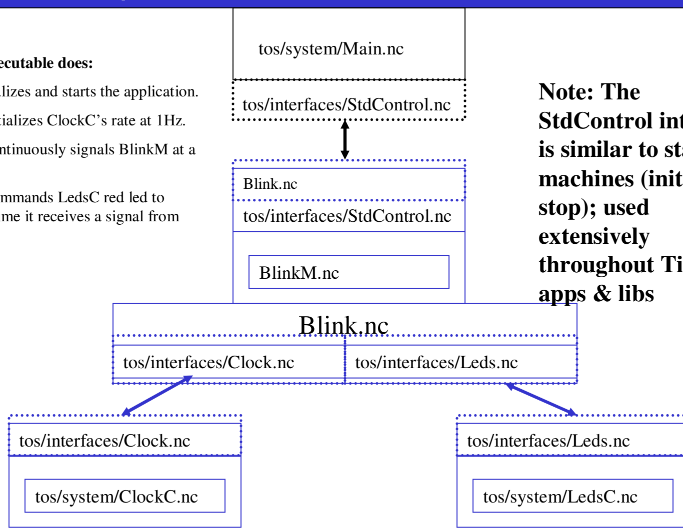
What the executable does:

1. Main initializes and starts the application.

2.BlinkM initializes ClockC’s rate at 1Hz.

3. ClockC continuously signals BlinkM at a rate of 1 Hz.

4. BlinkM commands LedsC red led to toggle each time it receives a signal from ClockC.



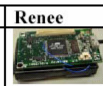
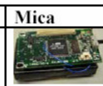
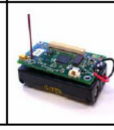
Note: The StdControl interface is similar to state machines (init, start, stop); used extensively throughout TinyOS apps & libs

27

Mote Hardware

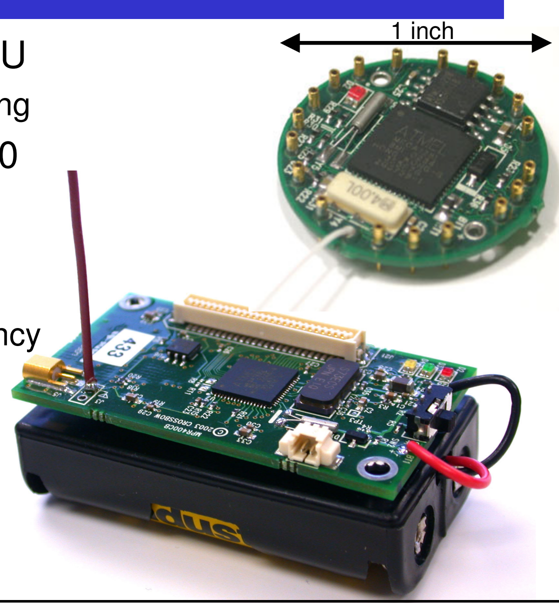
28

Family of Motes

Mote Type	WeC	Renee	Mica	Mica2	Mica2Dot
					
Microcontroller					
Type	AT90LS8535	Atmega163	Atmega128	Atmega128	Atmega128
CPU Clock (Mhz)	4	4	4	7.3827	4
Program Memory (KB)	8	16	128	128	128
Ram (KB)	0.5	1	4	4	4
UARTs	1	1	2 (only 1 used)	2	2
SPI	1	1	1	1	1
I2C	Software	Software	Software	Hardware	Hardware
Nonvolatile storage					
Chip	24LC256		AT45DB041B		
Size (KB)	32		512		
Radio Communication					
Radio	RFM TR1000			Chipcon CC1000	
Frequency	916 (single freq)			916/433 (multiple channels)	
Radio speed (kbps)	OOK		ASK	FSK	
Transmit Power Control	Programmable resistor potentiometer			Programmable via CC1000 registers	
Encoding	SecDed (software)			Manchester (hardware)	

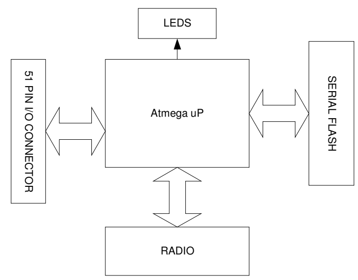
Mica2 and Mica2Dot

- ATmega128 CPU
  - Self-programming
- Chipcon CC1000
  - FSK
  - Manchester encoding
  - Tunable frequency
- Lower power consumption



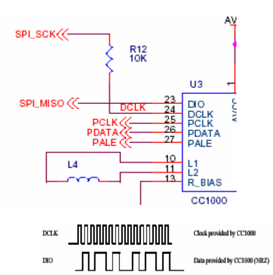
Common Platform Architecture

- Atmega uP
  - 32Khz crystal and 4Mhz (7.3728Mhz Mica2) crystal.
  - 10 bit ADC
  - UART (Mica2/Mica2Dot have 2)
  - SPI bus
  - I2C bus (hardware for mica2/mica2dot)
- Radio (RFM or Chipcon 1000)
- External serial flash memory (512K byte)
- Connectors for interfacing to sensor and programming boards
- 3 programmable leds (1 for Mica2Dot)
- JTAG port (Mica, Mica2, Mica2Dot)



The CC1000 Radio Interface

- Dedicated cpu bus (lines) to configure radio registers for radio frequency, power,…..
- Dedicated SPI bus for data transfer. CC1000 is bus master.
- Radio generates one interrupt every 8 bits when in receive mode.
- Runs usually at 38K or 19K bit rate (default) Manchester (2x bit)
- More in-depth radio discussion later in session.



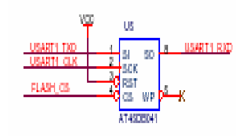
Baud Rate	Xmt or Rcv Time(*)
19K	~40msec
38K	~20msec

(*) Does not include random delay

The Flash Memory Interface

• 512 K bytes of flash (non-volatile) storage

• Useful for data logging.



• Used by GSK (Generic Sensor Kit) and TinyDB for data logging.

• Used by XNP for code download.

• Serial interface to Atmega uP

• TinyOS driver (Logger..) bit bangs interface

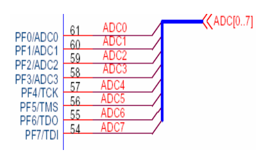
• Attached to 2nd uart port on Mica2. Another driver (UCB) uses synchonous usart for high speed data transfer.(5KB/Sec driver)

•Beware, device consumes 15 ma when storing to memory

33

The ADC Interface

• Eight channels of 10 bit ADC, multiplexed.



• Dedicated channels (Mica2):

  • ADC0 : Radio's RSSI

• Shared Mica2 Channels

  • ADC7 : Battery monitor (can be shared with another channel but will have ~10K ohm impedance.

  • ADC4..ADC7: JTAG. If using JTAG debug these channels won't work as ADC inputs.

• Shared Mica2Dot Channels

  • ADC1: Shared for both thermistor and battery voltage

  • ADC4..ADC7: JTAG. If using JTAG debug these channels won't work as ADC inputs

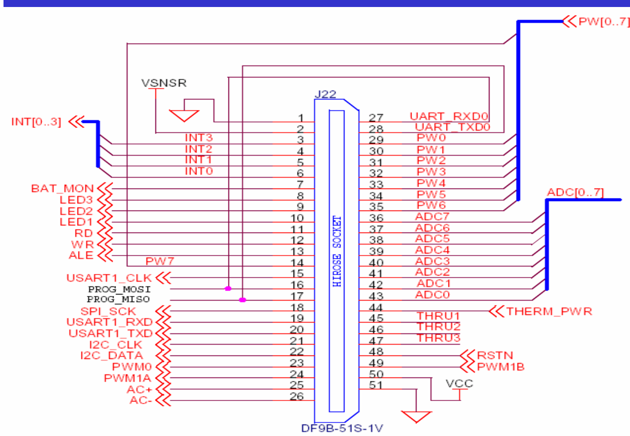
34

Mica2 Sensor Interface

Pin	Name	Description	Pin	Name	Description
1	GND	Ground	27	UART_RXDO	Uart0 Rcv
2	VSNR	Voltage(battery)	28	UART_TXDO	Uart0 Tx
3	INT3	GPIO	29	PWO	GPIO/PWM
4	INT2	GPIO	30	PW1	GPIO/PWM
5	INT1	GPIO	31	PW2	GPIO/PWM
6	INT0	GPIO	32	PW3	GPIO/PWM
7	BAT_MON	Battery Monitor Voltage	33	PW4	GPIO/PWM
8	LED3	Green Led	34	PW5	GPIO/PWM
9	LED2	Yellow Led	35	PW6	GPIO/PWM
10	LED1	Red Led	36	ADC7	GPIO/ADC CH7, JTAG
11	RD	GPIO	37	ADC6	GPIO/ADC CH6, JTAG
12	WR	GPIO	38	ADC5	GPIO/ADC CH5, JTAG
13	ALE	GPIO	39	ADC4	GPIO/ADC CH, JTAG
14	PW7	GPIO	40	ADC3	GPIO/ADC CH3
15	USART1_CLK	Usart clock	41	ADC2	GPIO/ADC CH2
16	PROG_MOSI	Programmer Pin	42	ADC1	GPIO/ADC CH1
17	PROG_MISO	Programmer Pin	43	ADC0	GPIO/ADC CH0
18	SPI_CLK	Radio Clock	44	THERM_PWR	GPIO
19	USART1_RXD	Usart1 receive	45	THRU1	Thru user connect
20	USART1_TXD	Usart1 xmit	46	THRU2	Thru user connect
21	I2C_CLK	I2C bus clock	47	THRU3	Thru user connect
22	I2C_DATA	I2C bus data	48	RSTN	uP reset
23	PWMO	GPIO	49	PWM1B	GPIO
24	PWM1A	GPIO	50	VCC	Voltage (battery)
25	AC+	GPIO	51	GND	ground
26	AC-	GPIO			

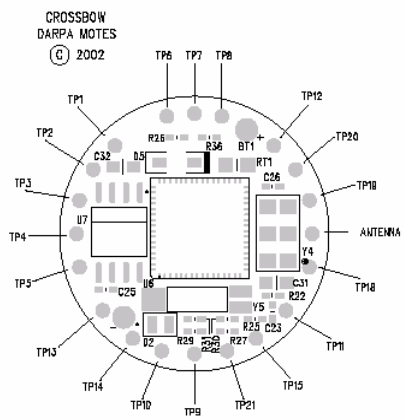
Blue: OK to use
Yellow: OK to use but has shared functionality
Red: Do no use
See Atmega128 specification for more information regarding signal functionality.

Mica2 Sensor Interface



Mica2Dot Sensor Interface

PIN	DESCRIPTION
TP1	GND
TP2	ADC7
TP3	ADC6
TP4	ADC5
TP5	ADC4
TP6	VCC
TP7	PW1
TP8	PW0
TP9	UART_TXD
TP10	UART_RXD
TP11	RESETN
TP12	SPI_CK
TP13	ADC3
TP14	ADC2
TP15	PWM1B
TP18	GND
TP19	INT1
TP20	INT0
TP21	THERM_PWR



- 6 ADC Channels
- 6 I/O Channels

Power Budgets

SYSTEM SPECIFICATIONS		
Currents		
	value	units
Micro Processor (Atmega128L)		
current (full operation)	6	ma
current sleep	8	ua
Radio (Chipconn 1000)		
current in receive	8	ma
current xmit	12	ma
current sleep	2	ua
Flash Serial Memory (AT45DB041)		
write	15	ma
read	4	ma
sleep	2	ua
Sensor Board		
current (full operation)	5	ma

Average, full operation, current: ~15 ma

AA Batteries are ~1800ma which mean ~ 120hrs (5 days)

Batteries

- AA – standard, 1800ma hours. Slow decay. Best price.
- Lithium – 3.6, fast decay, more expensive.
- Beware of low battery voltage (adc, flash programming…..)
- DC Booster may/may not help
- UCB Mica2Dot NiMH 3AH, single cell, with booster and recharge.

Crossbow Sensor Boards

Part #	Mote Support	Sensors
MTS101CA	MICA,MICA2	Light (photo resistor) Temperature (Thermistor) Prototyping area
MDA300CA	MICA2DOT	Protoyping
MTS300CA	MICA, MICA2	Light, Temperature, Acoustic, Sounder, 2-Axis Accelerometer (ADXL202), and 2-Axis Magnetometer
MTS500CA	Mica2Dot	Prototyping
MDA300CA	Mica2	On board humidity/temp. External sensors.
MTS400/420	Mica2	GPS weatherboard
Not released:	Mica2Dot	Weatherboards

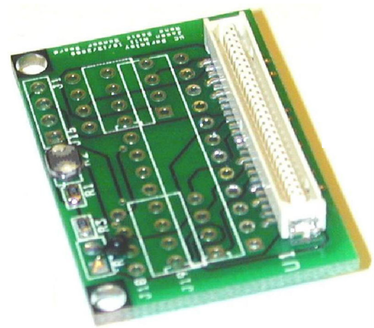
See **MTS/MDA Sensor and Data Acquisition Boards User's Manual**

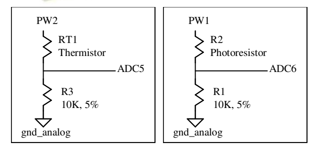
MTS101CA

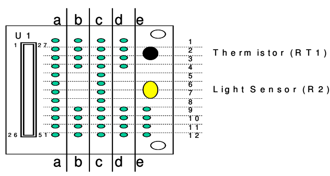
- Light photo resistor-Clairex CL94L
- Thermistor - YSI 44006,
- Both sensor are highly non-linear.
- Good prototyping area.

To use this sensor board add (modify) the apps/app/makefile for:

SENSORBOARD = basicsb



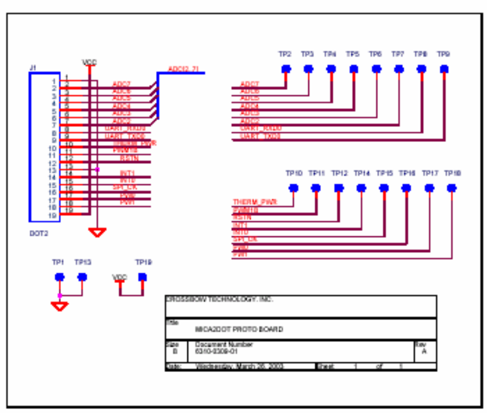




41

MDA500CA

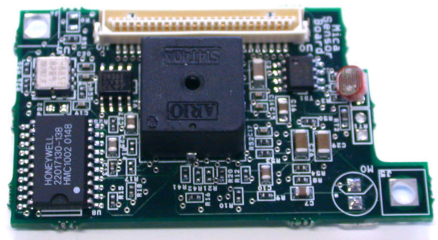
- Prototyping board for mica2dots



42

MTS300CA/MTS310CA

- Light (Photo)-Clairex CL94L
- Temperature-Panasonic ERT-J1VR103J
- Acceleration-ADI ADXL202
  - 2 axis
  - Resolution: ±2m*g*
- Magnetometer-Honeywell HMC1002
  - Resolution: 134µG
- Microphone
- Tone Detector
- Sounder
  - 4.5kHz



SENSORBOARD = micasb

43

MTS400/420 – GPS/Weather

- Gps (LeadTek 9546)   - optional
  - SiRFstartII LP chipset (60ma)
  - External active antenna.
  - 12 channels
  - 15 Meter (SA off); 7 Meter (WAAS corrected)
  - DC Booster to maintain required voltage
- Temperature & Humidity (Sensirion SHT11).
  - All digital (14 bits)
  - 3.5% RH accuracy, 0.5degC Temperature accuracy

44

MTS400/420 – GPS/Weather

•Barometric Pressure and Temperature (Intersema MS5534A)

- All digital
- 300 to1100 mbar, 3% accuracy
- -10 to +60 degC, 3% accuracy

• Ambient Light (TAOS TSL2250)

- All digital
- 400-1000nm response

•Acceleration-ADI ADXL202

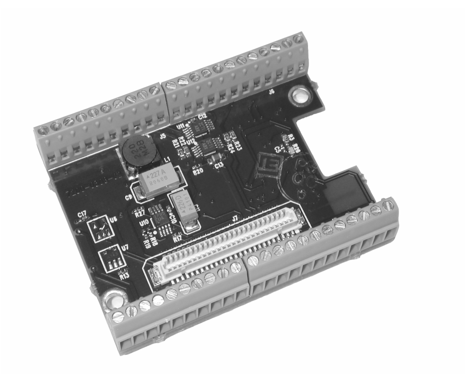
- –2 axis
- –Resolution: ±2m*g*

•2 K EEPROM for user configuration info.

45

MDA300

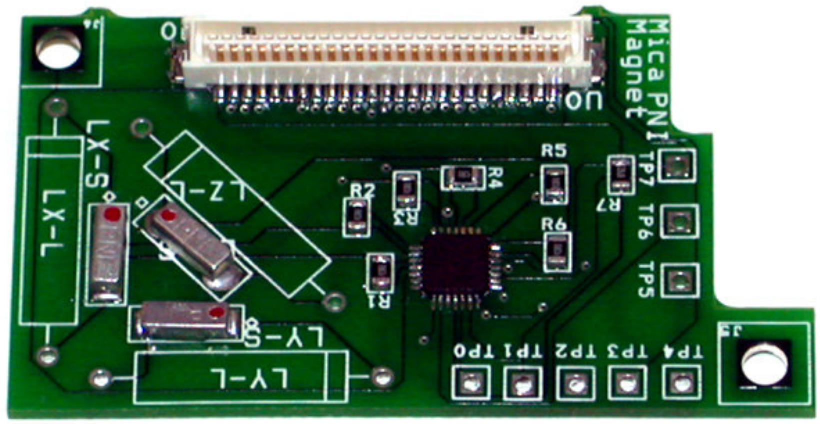
- *8* External Analog Inputs
  - External Sensors
  - Hi and low level signals
  - Block Screw Terminal
- *8* channel digital I/O
- *2* relays
- On board *12*-bit ADC
  - 0-2.5V, 0-3V, 0-5V Ranges
- Stable 2.5V Reference
- 3V and 5V power
- Designed by UCLA CENS w/ Crossbow and UCB



http://www.cens.ucla.edu/~mhr/daq/ 46

PNI- Magnetometer/Compass

- Resolution: 400 µGauss
- Very low power
- Three axis



Ultrasonic Transceiver

- Used for ranging
- Up to 2.5m range
- 6cm accuracy
- Dedicated microprocessor
- 25kHz element
- Mica2 and Mica2Dot versions

48

Mica2Dot WB

• UCB environmentally packaged weatherboards for GDI

•Temperature & Humidity (Sensirion SHT11).

  • All digital (14 bits)

  • 3.5% RH accuracy, 0.5degC Temperature accuracy

•Barometric Pressure and Temperature (Intersema MS5534A)

  • All digital

  • 300 to1100 mbar, 3% accuracy

  • -10 to +60 degC, 3% accuracy

•Ambient Light (TAOS TSL2250)

  •All digital

  •400-1000nm response

• Photosensitive Light Sensor..

49

Mote In Tires

• Real time control of vehicle dynamics.

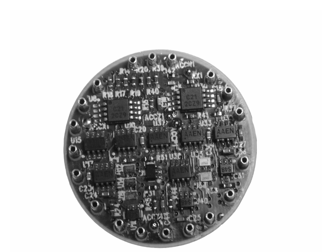
•3 bridge accelerometers (500g-1000g) mounted in tire.

• Sensor board has 3 channels of amplifiers, filters, programmable D/As for bridge balancing.

•Monitor and analyzed acceleration forces when tire is in contact with ground.

• Transmit results every revolution.

• 3 motes, 1 master, 2 slaves.



50

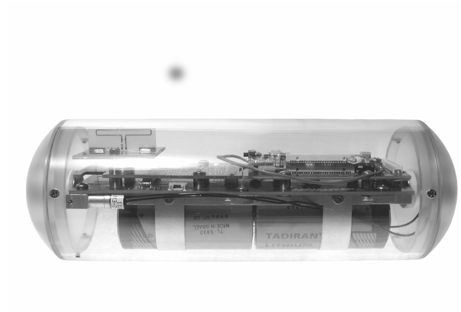
Micro Radar

•Darpa project: Detect intruders with micro-powered radar detectors and relay data through mote network.

•Drop detectors from UAV (ex: Predator)

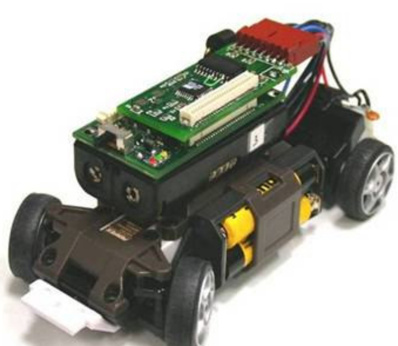
•Ghz Doppler radar detector.

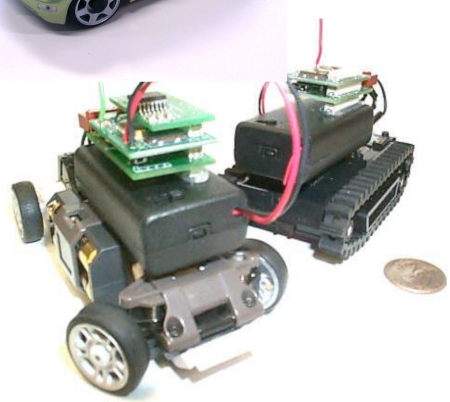
• Done with LLL and Advantaca



COTS-BOTS (UCB)

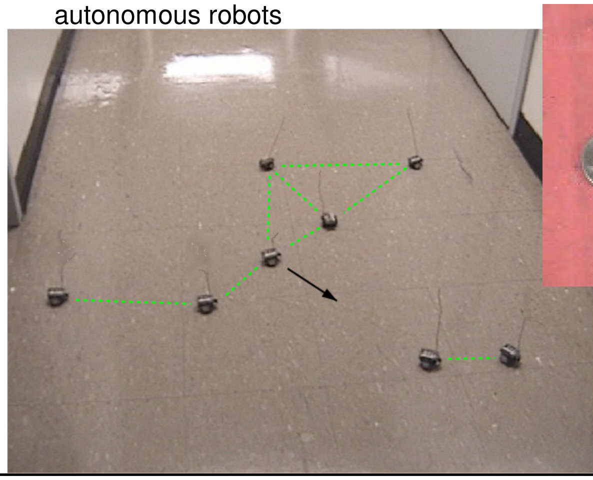
- 5” x 2.5” x 3” size
- <$250 total
- 2-axis accelerometer





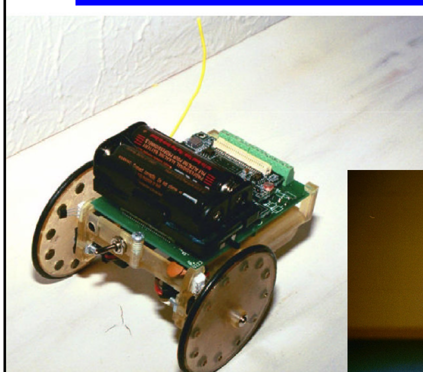
Robomote (USC)

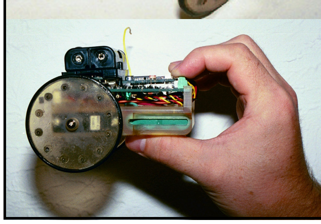
- Less than 0.000047m^3
- \$150 each
- Platform to test algorithms for adaptive wireless networks with autonomous robots



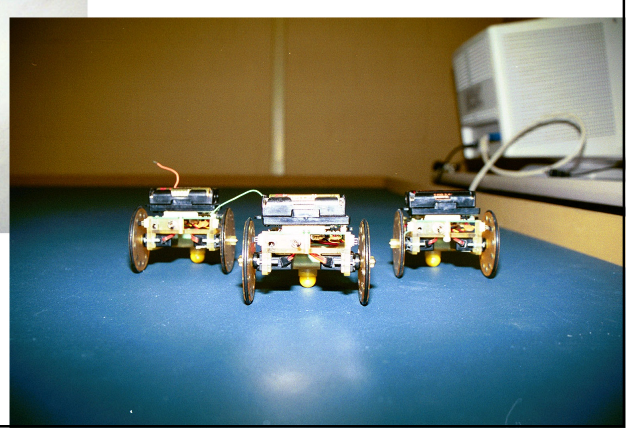
53

MICAbot (Notre Dame)





- Designed for large-scale research in distributed robotics and ad-hoc wireless networking.
- \$300 each



Ratiometric Adcs & Sensors

• Atmega128 is 10 bit (1024) ratio metric ADC

• If sensor is ratio metric then don't have to measure battery voltage. (Sensor's FS changes with battery voltage).

•Ratio metric sensors may not work over full range of battery voltage.

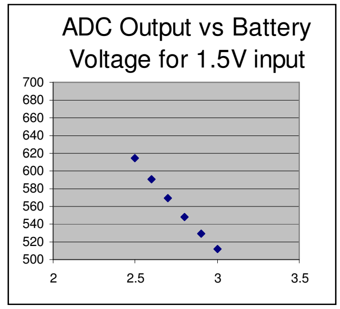
•ADC full scale is proportional to battery voltage.

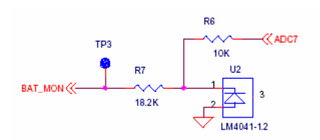
• Must measure battery voltage to get accurate sensor readings:

Battery Volts = RefVolt*ADC_FS/data

•Mica2 and Mica2Dot have on-board voltage references to calibrate the ADC full scale.

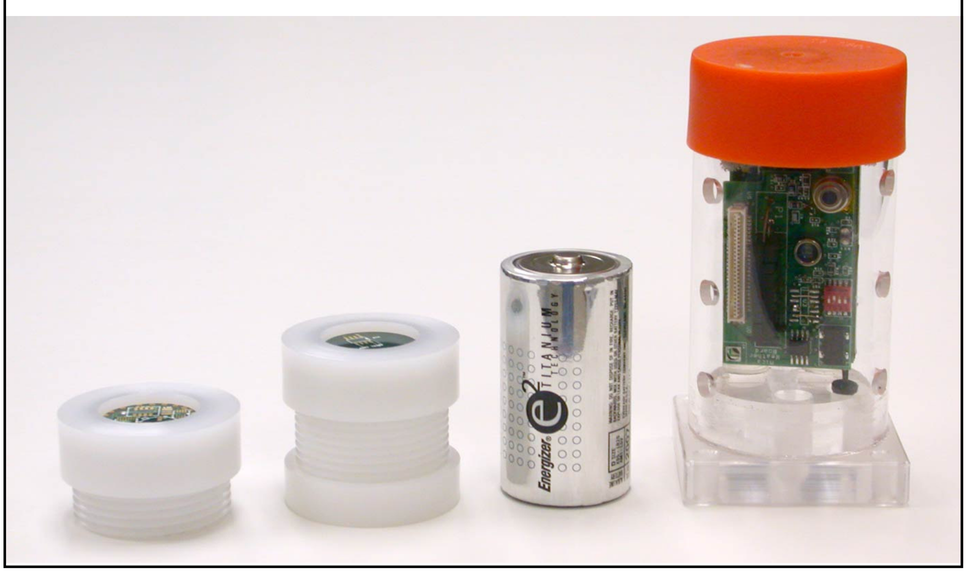
/contrib/xbow/apps/XSensorMica2



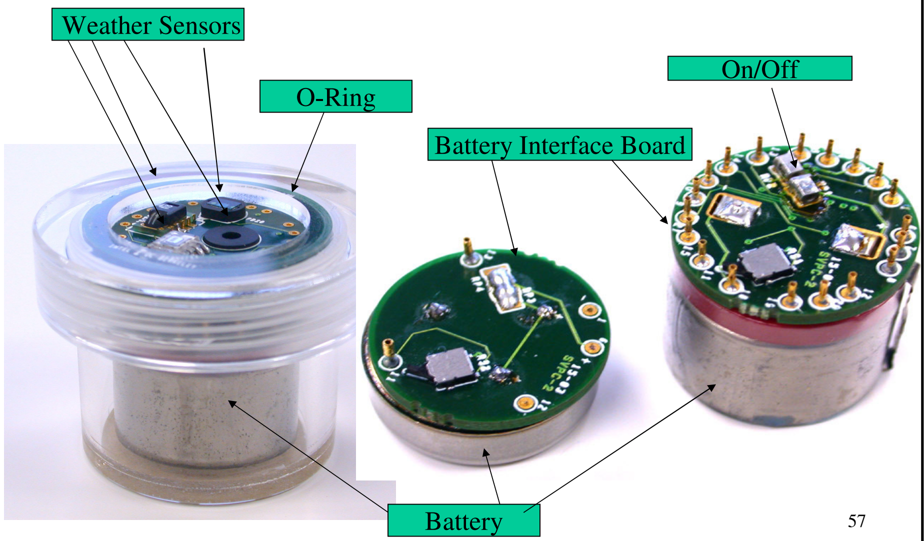


55

Enclosures for Environmental Monitoring

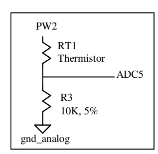


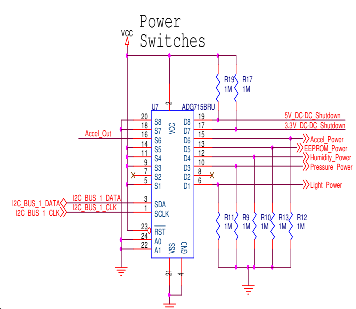
Mica2Dot Enclosures



Sensor Power Management

- Simple Strategy for Low Power Sensors:
  - Use Atmega output pins to source sensor power.
  - Will source ~5-10ma of current per pin.
- Analog Switch Strategy
  - Use hardware I2C (mica2) or software I2C (mica2dot) (in Sourceforge)
  - Switch connects sensor power to VCC.
  - ADG714 switch has 2.5 ohm on resistance
- DC-DC Booster Strategy
  - Create battery independent, constant supply voltage.
  - Create +5 V or more
  - Turn on booster from analog switch or Atmega
  - Boosters are ~80%-90% efficient. Need good layout and decoupling. Not ratiometric





Mote Programming and Base Station Boards

Overview:

- MIB500 Parallel Port Programmer
- MIB510 Serial Port Programmer
- eMote
- USB

59

MIB500

•Programs mote through the PC's parallel port

•Supports Mica, Mica2, Mica2Dot

•Voltage monitor to protect from low battery voltage. Low battery voltage can cause fuse errors.

•Serial port for base station operation

•Parallel port can cause flash corruption on some computers due to uisp parallel port drivers. THESE MAY BE IRRECOVERABLE

• Crossbow application note at www.xbow.com to help fix uisp problems.

• JTAG connector: AVRStudio and JTAG pod allows viewing and setting all fuses.

60

MIB510

• Q3 release

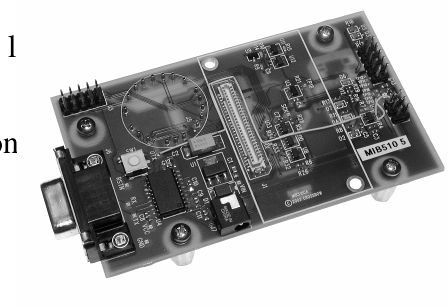
•Programming through the serial port. On board ISP uP is 3x faster than parallel port.

•Shares serial port with mote for base station operation.

•Voltage monitor to protect from l battery voltage

• Suports Mica (Atmega128 uP on Mica2, Mica2Dot

• JTAG port powered directly.



USB

• Q4 release

•USB interface for programming and base station operation .

•Power supplied thru USB.

62

eMote

Ethernet connection as serial forwarder.

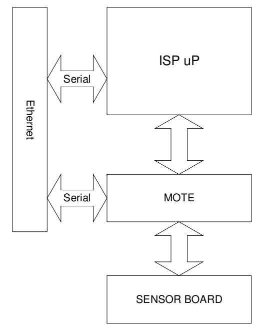
•Programming through ethernet.

• Remote base station operation through ethernet.

• Remote powered ethernet sensor.

• Remote code debugging through ethernet. Ideal for mote network debug.

• Similar configuration (eprb) used extensively at UCB for mote development.



63