



# Presentation on Tiny OS



Presented by : Srilekha Chitti  
20<sup>th</sup> March 2006

# About Tiny OS

- n Authored by 3 students\* of UC Berkeley.
- n Specifically designed for real time embedded sensors.
- n It is an OS for wireless network embedded systems .
- n It is a very small Operating system.
- n Focuses on saving power.
- n The Tiny OS software is implemented on NesC platform which has a syntax like C.





- n TinyOS is an event-driven operating system useful for behavioral applications.
- n The basic aim is to design a s/w that will abstract power management, networking and sensor measurements from the core application development.



- n Maintain flexibility by keeping component-based design.
- n TinyOS provides an open-source software platform which promotes abstraction, code re-use, and flexibility.

# DATA MEMORY MODEL

- n The stack size is 4KB.
- n It has a static memory allocation.No heap ,no function pointers.
- n Global variables are available on per frame basis.
- n Local variables are saved on the stack and declared within methods .



# Structure Of Tiny OS

- n This OS consists of a tiny scheduler and a graph of components.
- n There are 2 threads of application tasks and hardware event handlers.
- n Scheduler
  1. The task scheduler is a simple FIFO queue.





1. The scheduler uses a bounded size scheduled data structure. Tiny Os 1.1 supports up to 7 pending tasks.i.e Queue size=8.
2. Puts the processor to sleep when the tasks are complete, due to severe power constraints.

# Tasks And Hardware Event Handlers

- n Tasks.
- n Non preemptive FIFO.
  1. Atomic with respect to other tasks (No multithreading).
  2. Run in background while events are being processed.
  3. Can be preempted by events.
  4. Perform computationally intensive work.





# Events

- n Event handlers deal with the hardware events.
- n They are time critical and are of shorter duration as compared to tasks.
- n They interrupt tasks.
- n Priority doesn't exist among events. (LIFO)



- n nesC applications are built out of **components** with well-defined, bidirectional **interfaces**. Second, nesC defines a concurrency model, based on **tasks** and **hardware event handlers**, and detects **data races** at compile time.
- n Modules provide application code, implementing one or more interface.



## Contd..

- n Configurations are used to assemble other components together, connecting interfaces used by components to interfaces provided by others. This is called **wiring**.
- n Every nesC application is described by a **top-level configuration** that wires together the components inside.

# Naming Conventions In Tiny OS

- n The naming conventions in Tiny OS are strongly influenced by Java coding.
- n Interfaces
  1. Modules and components are connected using interfaces.
  2. Interfaces should be verbs or nouns in mixed case with first letter of each word capitalized. E.g. SendMsg.





## n Components

1. These are the set of command and event handlers.
2. Each component declares the commands and events it uses.
3. The event handlers are invoked to deal with hardware events.
4. To name a Component :



- n There are 2 cases ,those terminating with an M and those with a C.
- n The upper case C stands for Component ,used in order to distinguish between interface and a component providing the interface. E.g. TimerC



- n The upper case M stands for Module. This is used when a single component has both a configuration as well as a module. E.g. TimerM



- n Applications should be name of the top level component with the trailing C removed.
- n If an application tests TINYOS hardware then the first word should be “Test”.

# Naming Conventions In Tiny OS Contd..

- n Commands events and tasks should be verbs in mixed case with first letter of each internal word capitalized and with the first letter lower cased. E.g. putDone
- n In case of direct access from hardware the command should be prefixed with “TOSH\_” followed by the needed name.





# Issues in TinyOS

- n Severe memory constraints.
- n Battery powered therefore power constraints do exist, so proper power management is needed.



- n Totally flexible, its event-driven model enables fine-grained power management but allows the scheduling flexibility made necessary by the unpredictable nature of wireless communication and physical world interfaces.
- n It is an Open source platform.
- n So there are going to be a lots of developments or “Work In Progress”.

# References

- n “TinyOS Tutorial”, <http://www.tinyos.net/tinyos-1.x/doc/tutorial/>
- n Phil Levis, Sam Madden, David Gay, Joseph Polastre, Alec Woo, David Culler, “The Emergence of Networking Abstractions and Techniques in TinyOS”, <http://www.tinyos.net/papers/tinyos-nsdi04.pdf>
- n David Gay, Phil Levis, David Culler, “Software Pattern Designs For TinyOS”  
<http://www.tinyos.net/papers/lctes05.pdf>



Questions???





THANK YOU