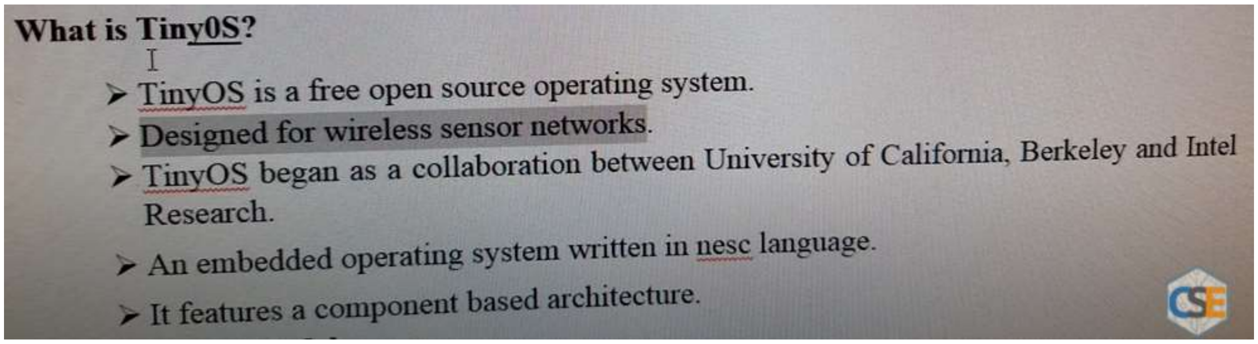
Practical No: 2

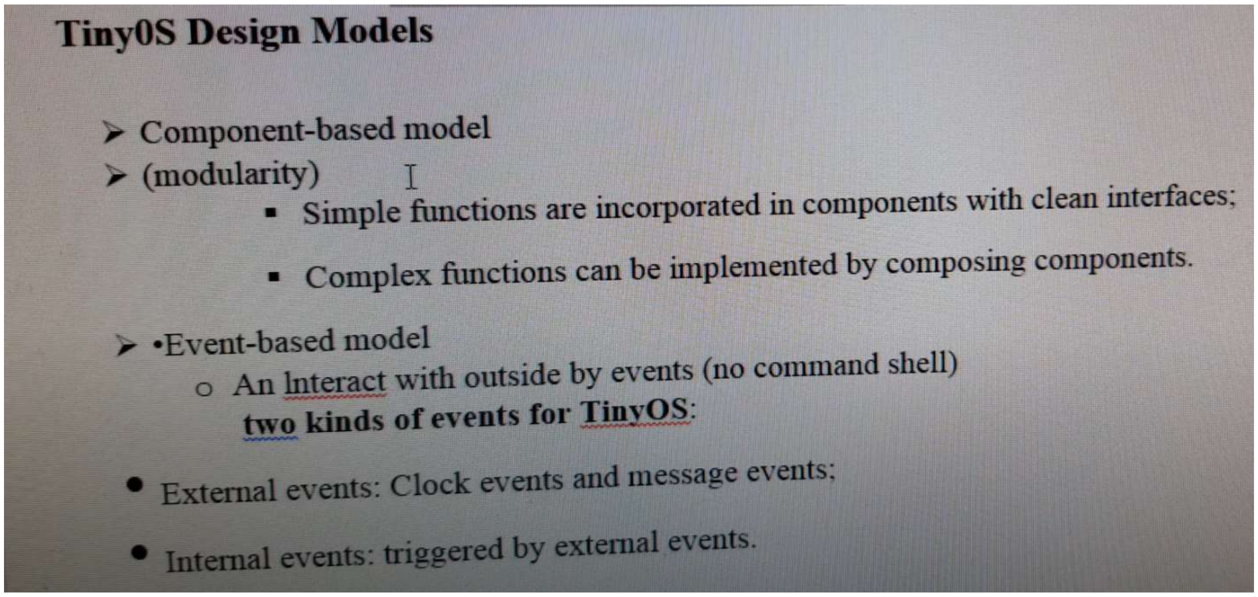
Aim: Exploring and understanding TinyOS computational concepts:- Events, Commands ,Task and NesC model and NesC Components

Tiny OS:-



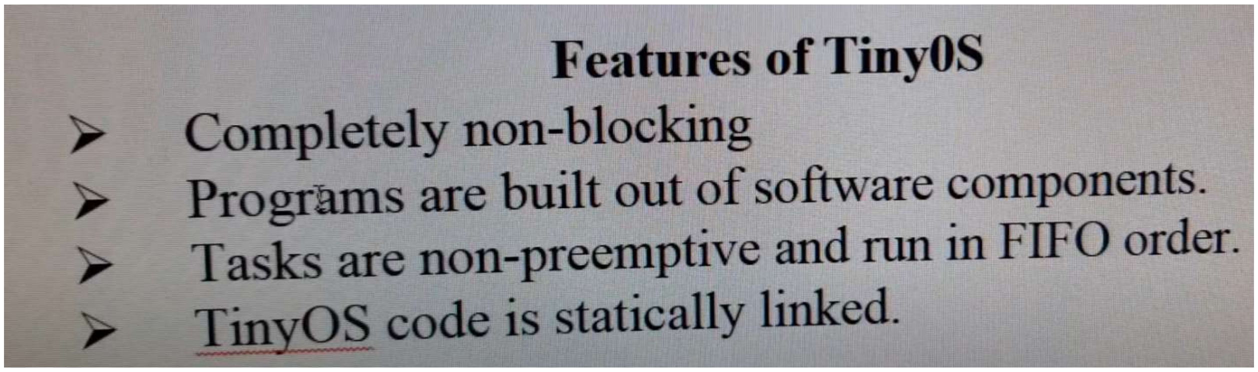
What is Tiny0S?

- TinyOS is a free open source operating system.
- Designed for wireless sensor networks.
- TinyOS began as a collaboration between University of California, Berkeley and Intel Research.
- An embedded operating system written in nesc language.
- It features a component based architecture.



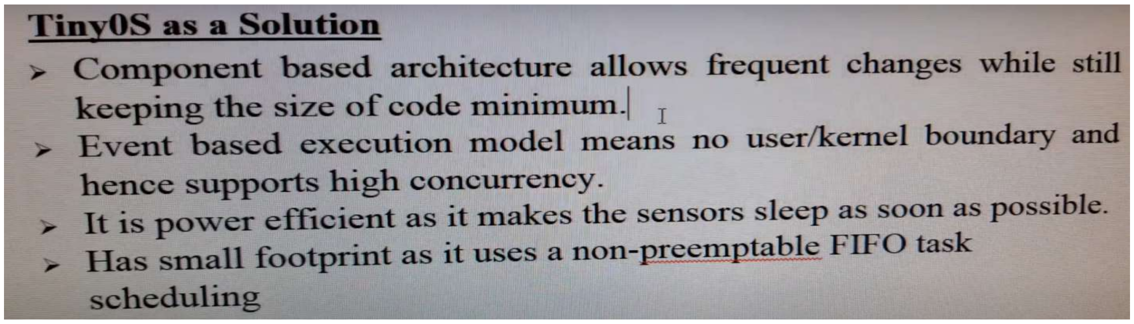
Tiny0S Design Models

- Component-based model
- (modularity)
  - Simple functions are incorporated in components with clean interfaces;
  - Complex functions can be implemented by composing components.
- •Event-based model
  - An Interact with outside by events (no command shell)
    two kinds of events for TinyOS:
- External events: Clock events and message events;
- Internal events: triggered by external events.

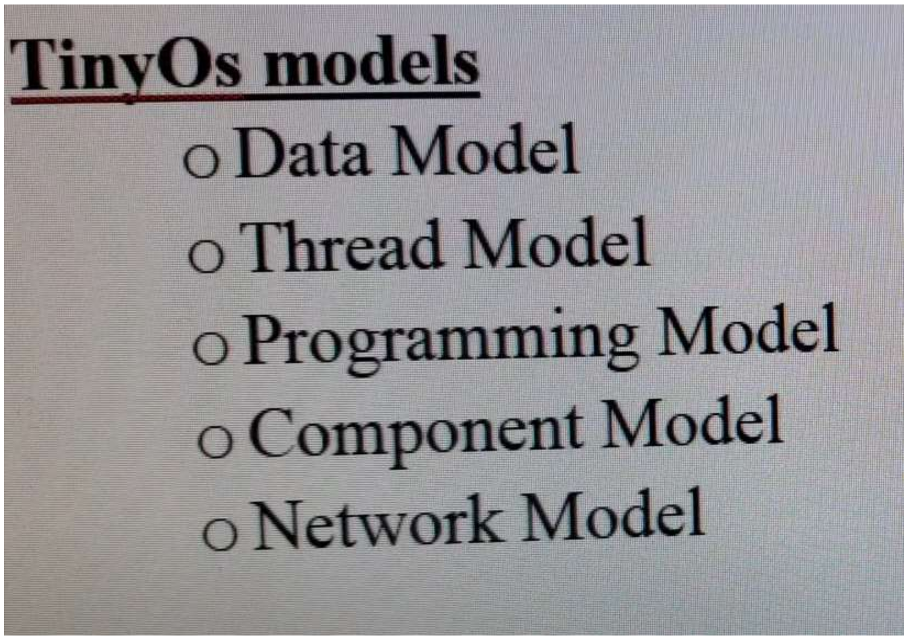


Features of Tiny0S

- Completely non-blocking
- Programs are built out of software components.
- Tasks are non-preemptive and run in FIFO order.
- TinyOS code is statically linked.


TinyOS as a Solution

- Component based architecture allows frequent changes while still keeping the size of code minimum.
- Event based execution model means no user/kernel boundary and hence supports high concurrency.
- It is power efficient as it makes the sensors sleep as soon as possible.
- Has small footprint as it uses a non-preemptable FIFO task scheduling


TinyOs models

- Data Model
- Thread Model
- Programming Model
- Component Model
- Network Model

TinyOS is an embedded, component-based operating system and platform for low-power wireless devices**.**
TinyOS, a flexible, application-specific operating system for sensor networks, which form a core component of ambient intelligence systems. Sensor networks consist of (potentially) thousands of tiny, low-power nodes, each of which executes concurrent, reactive programs that must operate with severe memory and power constraints.
The sensor network challenges of limited resources, event-centric concurrent applications, and low-power operation drive the design of TinyOS.

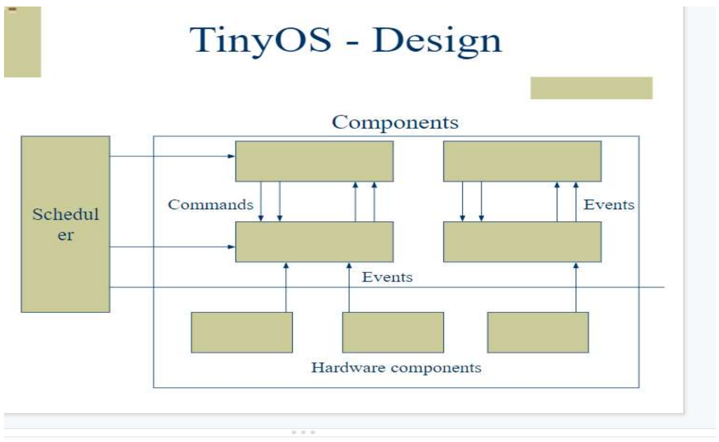
TinyOS combines flexible, fine-grain components with an execution model that supports complex yet safe concurrent operations. TinyOS meets these challenges well and has become the platform of choice for sensor network research; it is in use by over a hundred groups worldwide, and supports a broad range of applications and research topics.

It provide a qualitative and quantitative evaluation of the system, showing that it supports complex, concurrent programs with very low memory requirements (many applications fit within 16KB of memory, and the core OS is 400 bytes) and efficient, low-power operation.

TinyOS has a component-based programming model, codified by the NesC language , a dialect of C. TinyOS is not an OS in the traditional sense; it is a programming framework for embedded systems and set of components that enable building an application-specific OS into each application. A typical application is about 15K in size, of which the base OS is about 400 bytes; the largest application, a database-like query system, is about 64 K bytes.

Tiny OS computational concepts:-

A microthreaded OS that draws on previous work done for lightweight thread support and efficient network interfaces.
Two level scheduling structure.
Long running *tasks* that can be interrupted by hardware *events.*
Small, tightly integrated design that allows crossover of software components into hardware.



Events:
Event handlers deal with hardware events (interrupts) directly or indirectly.
Deposit information into a frame.
Post tasks.
Signal higher level events.
Call lower level commands.
Commands:
Non-blocking requests to lower level components.
Deposit request parameters into a component's frame, and post a task for later execution.
Can also invoke lower level commands, but cannot block.
To avoid cycles, commands cannot signal events.
Return status to the caller.
Tasks :
Perform the primary computation work.
Atomic with respect to other tasks, and run to completion, but can be preempted by events.

Allow the OS to allocate a single stack assigned to the currently executing task.
Call lower level commands. Signal higher level events.
Schedule other tasks within a component.
Simulate concurrency using events.

NesC model :
nesC (pronounced "NES-see") is a component-based, event-driven programming language used to build applications for the TinyOS platform. TinyOS is an operating environment designed to run on embedded devices used in distributed wireless sensor networks.
● A “holistic” approach to networked embedded systems.
● Supports and reflects TinyOS's design.
● Extends a subset of C.
● A static language.
– All resources known at compile-time.
– Call-graph fully known at compile-time.

Design of NesC model

● Components.
● Bidirectional interfaces.
● Simple expressive concurrency model.
● Whole-program analysis.

Nes C Component

● Challenge: platform diversity, flexible SW/HW boundary, applications deeply tied to hardware.
● Encourages modular design.
● Restrict access to private data.
● Allow underlying implementations to be replacedeasily.
● Can abstract HW using thin wrappers.
● Allow specialization of applications to hardware.
● Modules implement application code.
● Modules have private state. – Sharing of data among components is discouraged.
● Convention:
– Module names end with 'M', e.g. BlinkM.
– Module variables start with 'm_', e.g. m_timers.