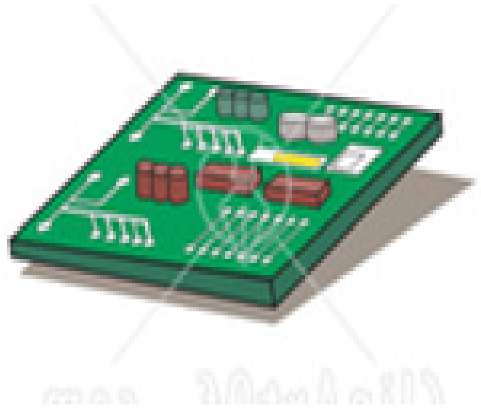
# TinyOS

Presented by
Bretny Khamphavong & Nancy White

## TinyOS: What is it?

- Free and open source component based operating system and platform targeting wireless sensor networks (or WSNs)
- Embedded operating system written in the nesC programming language as a set of cooperating tasks and processes

## nesC Basics



- Dialect of C optimized for the limitations of sensor networks
- Separation: construction and composition; programs are built out of components, which are assembled to form whole programs

## nesC Basics Con’t

- Specification: interfaces intended to represent the functionality that the component provides to its user, interfaces represent the functionality the component needs to perform its job
- Interfaces are bidirectional and are statically linked to each other via their interfaces

## Hardware Platform Constraints

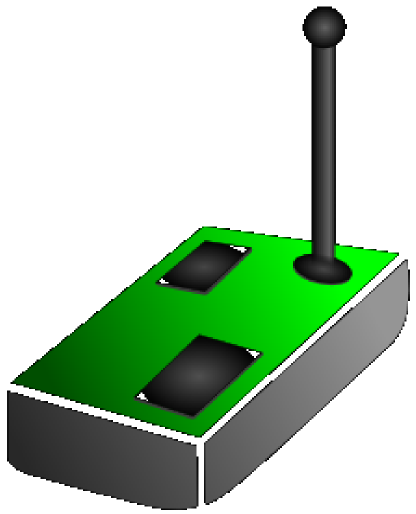
- Available memory is typically very low
- Written in such a way to allow for maximum concurrency with only one stack
- Little processing power for the greatest efficiency

# TinyOS is Non-Blocking

- Processing can occur while waiting for the input/output (I/O) completion of transmitted data
- Example:  Command line utility asking for a user's input
  - Normal execution occurs while waiting
  - Information appropriately processed when received

## Transmitting Information

- Works from one wireless sensor to another
- All I/O operations lasting longer than a few hundred microseconds are made asynchronous via callback
- Creates key feature of non-blocking I/O from a single stack



# Callbacks

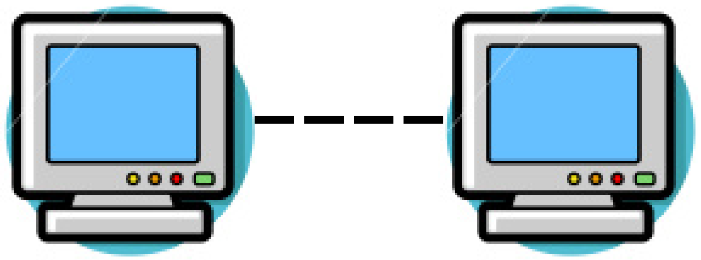
- Way to reference an executable piece of code by another executable piece of code
- Applications written for TinyOS need to be able to provide pieces of code that can be executed when the transmission of data or I/O operation is complete
- TinyOS uses these “events” extensively, and they are linked into the application to increase performance

## Enabling Performance Abilities

- Programmers must understand new programming concepts:
  - Instead procedural code, complex tasks are linked together through a series of events
  - Tasks can be scheduled at a later time from a FIFO queue
  - Sufficient for high I/O applications, but may be problematic for high CPU applications

# Deferred Procedure Call

- Takes higher procedure tasks and executes them before the less important tasks.
- The operating system can also post tasks that do not need to be run immediately
- The tasks are done in a first in first out or FIFO order
- The compiler converts the code into binary

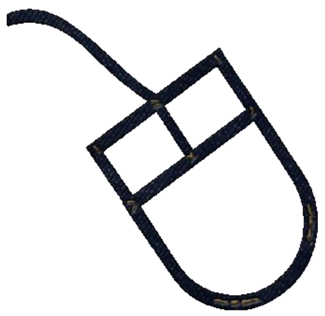


## GNU Toolchain

- GNU Toolchain: blanket term for a collection of programming tools produced by the GNU Project
- GNU Project: was developed to create a complete Unix-like operating system of free software

## GNU Toolchain Con’t

- These tools form a toolchain or suite of tools used in a serial manner to develop applications and operating systems
- GNU toolchain was very important in developing software for embedded systems



## In Conclusion…

- TinyOS is a light weight operating system which emphasizes low resource usage and high concurrency for I/O operations
- Concept of event handlers are common in some programming areas, but challenges and exposes developers to a different level of thinking

# References

- TinyOS Wikipedia: http://en.wikipedia.org/wiki/TinyOS
- TinyOS Documentation Wiki: http://docs.tinyos.net/
- Callback (computer science) Wikipedia: http://en.wikipedia.org/wiki/Callback_(computer_science)
- Asynchronous I/O Wikipedia: http://en.wikipedia.org/wiki/Asynchronous_I/O
- GNU Project: http://www.gnu.org/
- GNU toolchain: http://en.wikipedia.org/wiki/GNU_toolchain