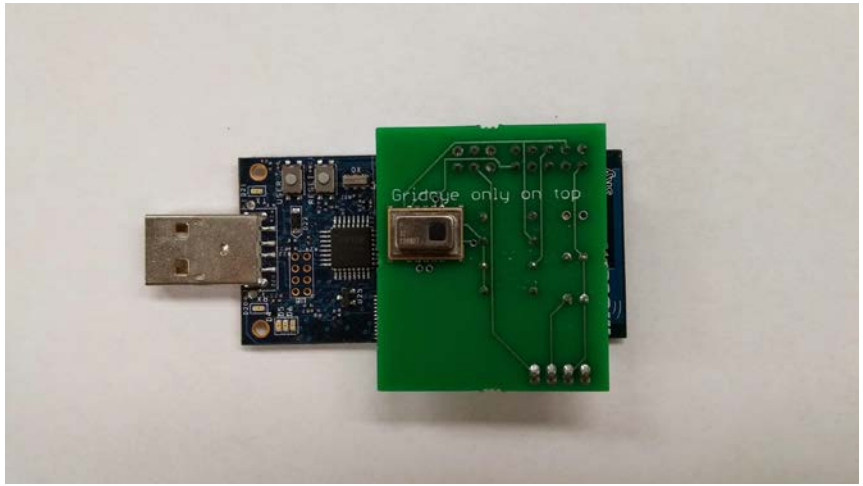
CSE160 – Project 0

Originally done by Ashish Yadav
ayadav6@ucmerced.edu

Modified by Alberto Cerpa
acerpa@ucmerced.edu

Wireless Sensor Networks

A wireless sensor network (WSN) is a wireless network consisting of spatially distributed autonomous devices using sensors to monitor physical or environmental conditions.



Potential Applications

- Environmental monitoring of air, water, and soil
- Structural monitoring for buildings and bridges
- Industrial machine monitoring
- Process monitoring
- Asset tracking

TinyOS

- Open Source operating system designed for low-power wireless devices
- Used in Academia
- Components/Interfaces
- Event Driven
- Mostly Synchronous
- Open Source

Why TinyOS?

Traditional OSes are not suitable for networked sensors

Huge !

Multi-threaded architecture =>large memory

I/O model

Kernel and user space separation

Typically no energy constraints

Ample available resources

Sensor Hardware Constraints

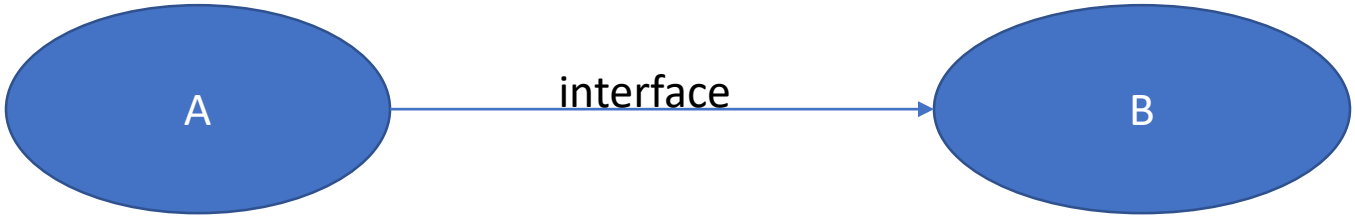
- Lower Power
- Limited memory
- Slow CPU
- Size (Small)
- Limited hardware parallelisms
- Communication using radio
  - Low-bandwidth
  - Short range

Tiny OS properties

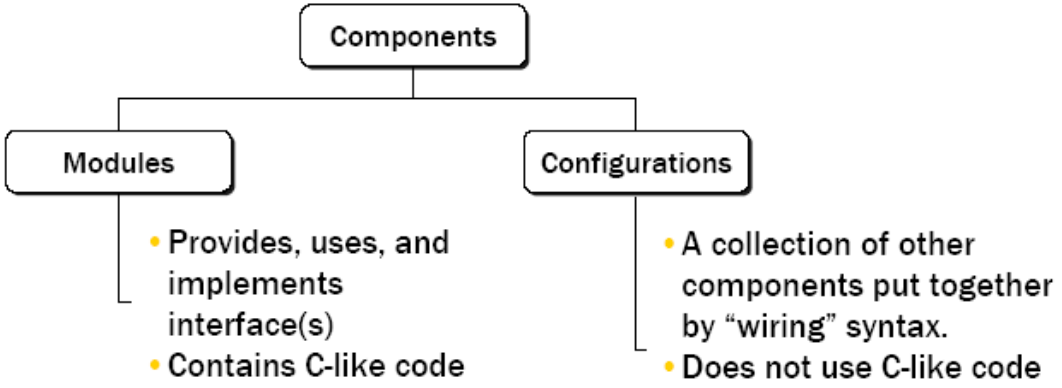
- Small memory footprint
- Efficient in power and computation
- Communication is fundamental
- Real-time
- Support diverse application design
- Concurrency: uses event-driven architecture
- Communication
  - Uses event/command model
  - FIFO and non pre-emptive scheduling
- Modularity
  - Application composed of *components*
  - OS + Application compiles into single executable

Software

- TinyOS (Platform)
- Coding language
  - NesC (Network Embedded System C)
    - Basic unit of nesC code is a component
    - Components connect via interfaces
      - Connections called “wiring”



NesC


Components
Modules
Configurations
• Provides, uses, and implements interface(s)
• Contains C-like code
• A collection of other components put together by “wiring” syntax.
• Does not use C-like code

TOSSIM

- Simulates entire TinyOS application
- Replaces components with simulation implementation
- Core code = tos/lob/tossim
- 2 interfaces :
  - C
  - Python

Project Outlines

Project #0 - Getting Started

Project #1 - Flooding and Neighbor Discovery – 5 %

Project #2 - Routing Table – 17.5 %

Project #3 - Reliable Transport - 25 %

Project #4 - Application Layer – 7.5 %

Words of Advice

- This class is tough, so allocate enough time to it
- Finish early
- Partial credit is better than no credit
- If a project is not fully functional by submission, be sure to catch up quickly.
- Use good programming practices
  - No magic numbers
  - Descriptive names
  - Comments

Suggestions and Expectations

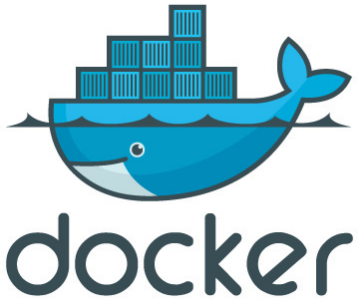
This semester, we will be implementing various protocols on top of a TinyOS network emulator and simulator called "TOSSIM".

- Get comfortable with programming
- Start using Linux
- C / Python
- Docker -- (Makes development a breeze) – not related to the course

Project #0 - Virtualization

- Docker
  - Lightweight
  - Reproducible
  - Higher learning curve
- Vagrant
  - Lightweight
  - Reproducible
- Virtualbox
  - Backend to handle the VM
- Available on Mac/Windows/Linux

+



Project #0 - Native

- Only tested on Ubuntu 14.04 LTS and 16.04 LTS
- Install the toolchain for compiling nescand supporting tools for simulation.

Native Installation

Here is a quick way of getting tinyos installed on the lab machines in SE1-100. This has has not been tested on machines outside of the lab and will likely not work on other machines.

Open a terminal and enter the following command to download a bundle of the necessary binaries.

1. wget http://andes.ucmerced.edu/~abeltran2/cse160/fall2016/lab_tinyos.tar.gz
2. tar xvfz lab_tinyos.tar.gz
3. cd bundle
4. ./install.sh
5. export TINYOS_ROOT_DIR=$HOME/bundle/tinyos-main/

You will need to exit your terminal for the path to be added correctly and start your terminal again.

Installing TinyOS on your PC

- **Docker** – catcourses.ucmerced.edu under files>Projects>TinyOS installation>Docker-Installation-Instructions.pdf
- **Virtual Machines/Native -** catcourses.ucmerced.edu under files>Projects>TinyOS installation>Native-Installation-Instructions.pdf

Comparison

Virtualization

+ Consistent
+ Persistent
+ Can use IDE using a shared folder
- Bad on slow/old computers
- Requires to have a separate terminal for building.

Native

+ Fast
+ Easier to build in an IDE
- Lab machines do not have persistent software

Resources

- The official TinyOS Documentation:
  - http://tinyos.stanford.edu/tinyos-wiki/index.php/TinyOS_Documentation_Wiki
- Free Book:
  - http://csl.stanford.edu/~pal/pubs/tos-programming-web.pdf
- nesC: http://nescc.sourceforge.net/