A Graphical Development and Simulation Environment for TinyOS-based Wireless Sensor Networks

http://ptolemy.eecs.berkeley.edu/viptos

Elaine Cheong Edward A. Lee Yang Zhao Christopher Brooks

Summary

Viptos (Visual Ptolemy and TinyOS) is an integrated graphical development and simulation environment for TinyOS-based wireless sensor networks.

Graphical Development. Viptos allows developers to create block and arrow diagrams to construct TinyOS programs from any standard library of nesC/TinyOS components.

Code Generation. The tool automatically transforms the diagram into a nesC program that can be compiled and downloaded from within the graphical environment onto any TinyOS-supported target hardware.

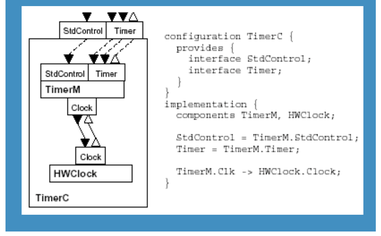
Simulation. Viptos provides interrupt-level simulation of actual TinyOS programs, with packet-level simulation of the network, while allowing the developer to use other models of computation available in Ptolemy II for modeling various parts of the system.

Background

Viptos is built upon TOSSIM and the Ptolemy II VisualSense environment for network-level modeling and simulation of wireless sensor networks.

TOSSIM is a text-based interrupt-level simulator for TinyOS programs. It runs actual TinyOS code but provides software replacements for the simulated hardware, and models network interaction at the bit or packet level.

TinyOS/nesC Architecture

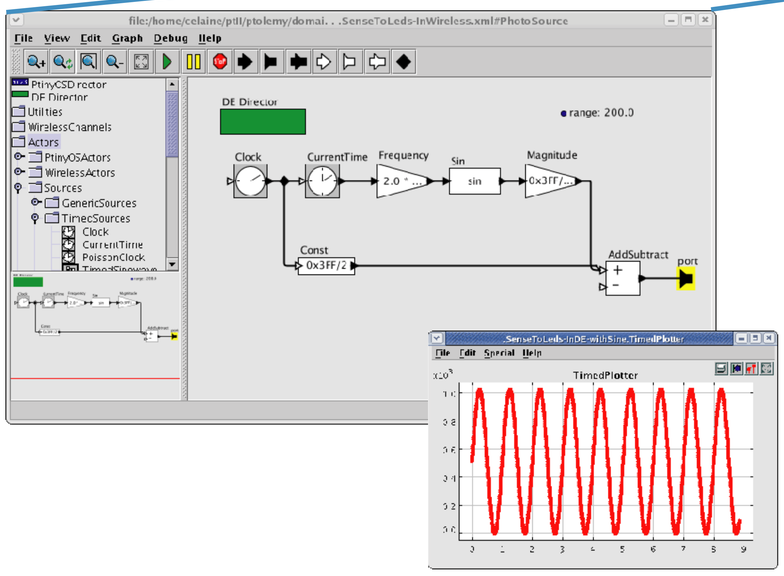


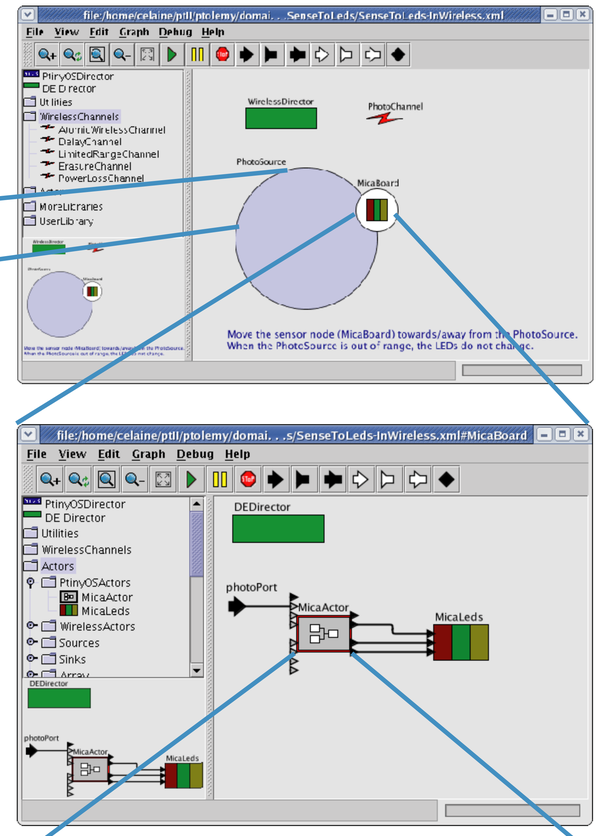
Ptolemy II is a graphical software system for modeling, simulation, and design of concurrent, real-time, embedded systems. Ptolemy II focuses on assembly of concurrent components with well-defined models of computation that govern the interaction between components.

Viptos Harvest Tools

The Viptos utilities, *nc2moml* and *ncapp2moml*, harvest existing TinyOS components and applications and convert them into a format that can be displayed as block (and arrow) diagrams and simulated.

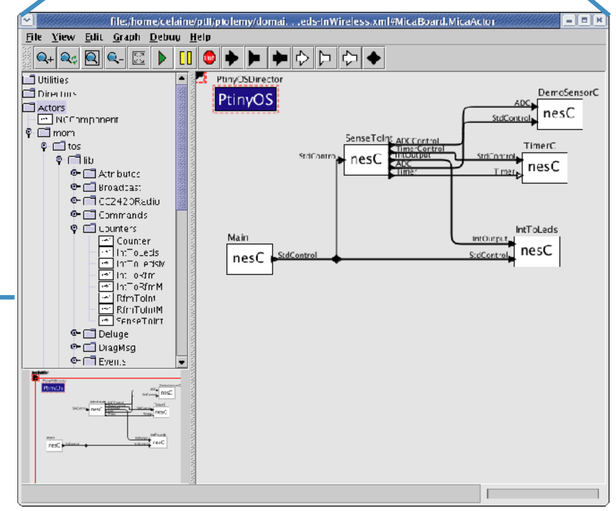
Example: Display Incoming Light Level on LEDs





nesC Code Generation for Target Hardware

```
configuration _SenseToLeds_InWireless_MicaBoard_MicaActor3198 {
}
implementation {
components Main, TimerC, IntToLeds, SenseToInt, DemoSensorC;
SenseToInt.TimerControl -> TimerC.StdControl;
SenseToInt.Timer -> TimerC.Timer[unique("Timer")];
SenseToInt.IntOutput -> IntToLeds.IntOutput;
Main.StdControl -> IntToLeds.StdControl;
Main.StdControl -> SenseToInt.StdControl;
SenseToInt.ADC -> DemoSensorC.ADC;
SenseToInt.ADCControl -> DemoSensorC.StdControl;
}
```



Chess University of California, Berkeley
Center for Hybrid & Embedded Software Systems