Demo Abstract: YETI - An Eclipse Plug-in for TinyOS 2.1

Nicolas Burri, Roland Flury, Silvan Nellen, Benjamin Sigg, Philipp Sommer, Roger Wattenhofer
Computer Engineering and Networks Laboratory
ETH Zurich, Switzerland
{burri, rflury, snellen, besigg, sommer, wattenhofer}@tik.ee.ethz.ch

Abstract

We present YETI[1], an Eclipse plug-in providing support for TinyOS development. YETI provides features well-known from development environments for other languages such as syntax highlighting, code completion and error detection. Furthermore, it includes an additional set of tools which are designed to ease the TinyOS development process for both newcomers and experienced developers. The plug-in seamlessly integrates with the existing TinyOS toolchain and provides debugging support on the target platform using a JTAG hardware adapter.

Categories and Subject Descriptors

D.2.3 [**Software Engineering**]: Coding Tools and Techniques—*Program editors*

General Terms

Design, Experimentation, Languages

Keywords

Sensor Networks, TinyOS, Debugging, Development

1 Introduction

Development of applications for wireless sensor networks has a steep learning curve. Newcomers have to familiarize with hardware platforms and software tools that are different from what they might already know. *TinyOS* is a component-based operating system targeting wireless sensor networks. Due to its flexibility it is widely used in academia and industry. TinyOS modules are written in nesC [4] which is an extension of the C programming language. Although the TinyOS homepage provides a well-written tutorial explaining the basic concepts of the operating system and the toolchain, it is still a cumbersome task to get the first self-written application running on a sensor node. Existing development environments for TinyOS [6, 7] support only TinyOS 1.x or have their main focus on the graphical wiring of components [3, 5]. We present YETI, an Eclipse-based integrated development environment (IDE) for TinyOS. The current YETI version 2.0 is the successor of the YETI TinyOS 1.x plug-in [2] and extends it with support for TinyOS 2.1. Various improvements and new features, such as JTAG debugging support from the graphical user interface, make it even more powerful.

[1]**Y**ETI is an **E**clipse-based **T**inyOS **I**DE

SenSys'09, November 4–6, 2009, Berkeley, CA, USA.
ACM 978-1-60558-748-6

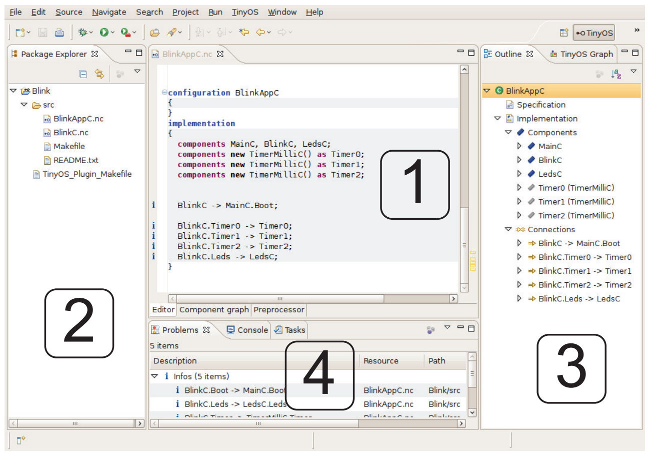
2 Architecture

The goal of the YETI project is to provide an efficient, convenient and easy-to-use development tool for both experienced users and TinyOS newcomers. YETI integrates as a plug-in into the Eclipse platform. Therefore, we can benefit from existing features available within the Eclipse framework such as the powerful editor support, the persistency system and a convenient update mechanism. Since YETI itself does not include a copy of TinyOS, the user has to specify the location of a TinyOS source tree before first use. Due to this decoupling of the TinyOS installation from the IDE, it is very easy to switch between different versions of TinyOS without having to change the corresponding environment variables.

The YETI parser reads nesC and header files of the current project and builds an abstract syntax tree (AST). An internal model of the TinyOS application is then derived from the AST. YETI uses this internal model for error checking, code completion and hyperlinks. Whenever a source file in the workspace is modified by the user, the AST and the internal model are updated accordingly.

3 Features

Once installed within the Eclipse framework, YETI provides a custom *TinyOS Perspective* which hosts helpful features for all development stages of wireless sensor network applications, see Figure 1. The *TinyOS Editor* provides syntax highlighting for nesC code as well as code completion. Source code containing syntax errors is underlined with red color and a user-friendly error message is displayed in the *Problem View*. If possible, the editor offers a suggestion how to fix the problem, e.g., by adding missing methods of an interface which the module has to implement. When holding the `ctrl`-key and clicking on a component, interface or method, the corresponding source file is opened in the editor, sparing the user from manually searching for the appropriate file in the TinyOS source tree. The *TinyOS Project Wizard* helps the user to create a new TinyOS project from scratch or to import one of the various sample applications shipped with TinyOS into the workspace. The *Launch Configuration* dialog can be used to define the target platform and include directives for the current project, thus saving the user from



```
configuration BlinkAppC
{
}
implementation
{
  components MainC, BlinkC, LedsC;
  components new TimerMilliC() as Timer0;
  components new TimerMilliC() as Timer1;
  components new TimerMilliC() as Timer2;


  BlinkC -> MainC.Boot;

  BlinkC.Timer0 -> Timer0;
  BlinkC.Timer1 -> Timer1;
  BlinkC.Timer2 -> Timer2;
  BlinkC.Leds -> LedsC;
}
```

5 items

Description	Resource	Path
Infos (5 items)		
BlinkC.Boot -> MainC.Boot	BlinkAppC.nc	Blink/src
BlinkC.Leds -> LedsC.Leds	BlinkAppC.nc	Blink/src

Figure 1. Screen shot of the YETI TinyOS IDE. (1) NesC editor window showing the current TinyOS component. (2) Project Explorer listing files of the current project. (3) Outline showing the component structure. (4) Panel for the problem view and console output.

editing the Makefile by hand. Available target platforms are automatically identified and listed together with further valid parameters such as possible extension boards. Compiling applications and flashing the resulting binary image to the sensor node is accomplished using a single click. Furthermore, the *TinyOS Search* view lets the user quickly browse through interfaces, modules and configurations found in the search path of the current target platform. The wiring of an application's components can be displayed in the *TinyOS Graph* view.

4 Debugging with JTAG

Debugging applications on sensor nodes is hard, especially for newcomers not familiar with the programming of embedded systems. A common approach to debug code on sensor nodes is to use the LEDs to provide some information about the current state of the program. Sending debug text to the serial port of the node using *printf* or transmitting a radio message containing debug information is more powerful, but requires more changes in the program code and alters the program flow. The *JTAG* standard [1] defines a hardware-based solution to test integrated circuits. Using a JTAG hardware adapter, the host software on the PC can communicate with the microcontroller to read or write data registers. Currently, YETI supports debugging for sensor node platforms based on the Atmel AVR and TI MSP430 microcontrollers. YETI uses the platform-specific toolchains (`avr-gdb`/`avarice` for the AVR architecture and `msp430-gdb`/`msp430-gdbproxy` for the MSP430) and Eclipse's C Development Tools (CDT) to control the debugging process as shown in Figure 2. During the build process, the component model of nesC is mapped to a single C file by rewriting the names of variables and methods. For example the `Boot.booted()` event in the `BlinkC` component is mapped to `BlinkC__Boot__booted()` in C by the nesC compiler. YETI uses its internal model of the application to perform the mapping between the nesC and C code in

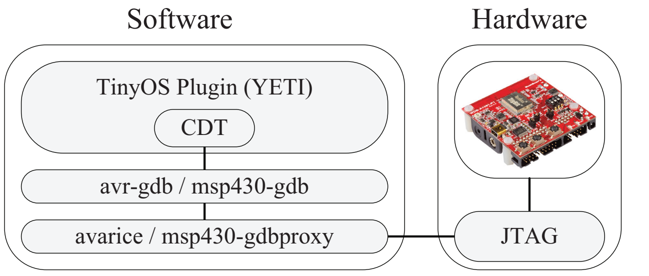


Figure 2. Integration of JTAG debugging support for TinyOS into YETI. Platform-specific tools (`avr-gdb`/`avarice` or `msp430-gdb`/`msp430-gdbproxy`) are used to interact with the sensor node hardware.

a way transparent to the user. That is, all user-interaction is based on the nesC program and not the cryptic C code. The debugging process can be configured using a user-friendly launch configuration dialog. Breakpoints can be set directly in the nesC editor component. Furthermore, the current state of registers and variables can be inspected and modified. To the best of our knowledge, YETI2 is the only development environment supporting the debugging of TinyOS code from the graphical user interface using JTAG-enabled hardware.

5 Demonstration Setup

Our demonstration setup will consist of two laptops which have the YETI plug-in installed. TinyOS code running on a sensor node can be debugged using YETI and a JTAG adapter connected to the USB port of the laptop. Debugging using JTAG will be possible on the AVR-based Meshnetics Meshbean development board and on the MSP430-based Shockfish TinyNode. Visitors are encouraged to try out the different features of YETI.

6 Download

YETI is available for Windows and Linux from the project website at `http://tos-ide.ethz.ch/`.

7 References

[1] IEEE Standard Test Access Port and Boundary-Scan Architecture. *IEEE Std 1149.1-2001*, pages 1–200, 2001.

[2] N. Burri, R. Schuler, and R. Wattenhofer. YETI: A TinyOS Plug-in for Eclipse. In *Proceedings of the ACM Workshop on Real-World Wireless Sensor Networks (REALWSN)*, pages 1–5, Uppsala, Sweden, 2006.

[3] E. Cheong, E. A. Lee, and Y. Zhao. Viptos: A Graphical Development and Simulation Environment for TinyOS-based Wireless Sensor Networks. In *Proceedings of SenSys '05*, page 302, San Diego, California, USA, 2005.

[4] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. The nesC Language: A Holistic Approach to Networked Embedded Systems. In *Proceedings of PLDI '03*, pages 1–11, San Diego, California, USA, 2003.

[5] W. P. McCartney and N. Sridhar. TOSDev: A Rapid Development Environment for TinyOS. In *Proceedings of SenSys '06*, pages 387–388, Boulder, Colorado, USA, 2006.

[6] J. Sallai, G. Balogh, and S. Dora. TinyDT. http://tinydt.sourceforge.net/.

[7] R. Tynan. TinyOS IDE. http://tinyoside.ucd.ie.