ARDUINO CDT IDE

Bringing Eclipse CDT to Hobbyists

Doug Schaefer
QNX/BlackBerry, CDT Project Co-lead
What is Arduino?

- “Arduino is an open-source electronics platform”
  - http://arduino.cc
- Hardware design is open and forked by a few companies
  - Adafruit, Sparkfun
- Originally based on AVR 8-bit microcontrollers
  - Starting to see ARM 32-bit Cortex-M3 based designs
- Supported by the GNU toolchain for builds
  - No debugger support other than complex JTAG setups
- Inexpensive and accessible, lots of accessories
  - $35 down to <$10 available at most on-line electronic shops
- A great community with lots of tutorials on the web
  - Portable, wearable, awesome!
Why not Raspberry Pi, BeagleBone?

• Power!

• Arduinos run at about 60 mA.
  • Run forever on a good battery
  • Very portable, growing line of wearable projects

• Raspberry Pi 2 is rated 800 mA
  • Many USB ports have a 500 mA fuse, i.e. pop!
  • Great for home electronic projects but need to be plugged in
  • But certainly another great area we need to look at for CDT

• Microcontrollers teach true systems programming
  • No OS, only your code runs
  • But great new libraries that make it easy for beginners
    • And experts!
Arduino Programming

• Based on Wiring which is based on Processing
  • Wiring is a simplified version of C++
  • Teach non-programmers now to program
• A fairly extensive library to abstract hardware details
  • Use pin number instead of GPIO register addresses
  • Simple serial port API for debug messages back to console
  • Additional APIs provided by peripheral vendors
    • E.g. Adafruit’s Neopixel library
    • Interesting collection of FFT libraries for processing analog inputs
• Many algorithms depend on speed of processor
  • Build environment must know what board you are building for
Why C++ for Arduino

• Wiring is a simplified C++
• But it’s not that much different from real C++
  • Auto includes system headers, i.e. Arduino.h
  • Auto forward declares all functions
• Why not teach Arduino programmers about these things?
  • Use the same language and environment the pros use
  • Start down the path to become a pro embedded software engineer
Why CDT for Arduino

• Arduino has an IDE
  • Based on the Processing IDE
• Simple, fairly easy to use
  • Data driven UI to select the Board you are working with
  • Easy to access the Serial Monitor to see output
  • Single buttons to build and launch
• But lacks many of the advanced features CDT has
  • Content assist and source navigation
  • Integration with Source Control
• And using Eclipse lets you do all your projects in one IDE
  • E.g., Arduino connected to a Raspberry Pi to a web app, etc.
• Makes a great exemplary extension for CDT!
Introducing CDT for Arduino

- New Project wizard and template
- Build tools integration
- Native serial port support added to CDT
- Remote (org.eclipse.remote) connection type
- Launch Bar support
- Terminal for serial monitor
- p2 touchpoint action to download and install toolchain
New Project Wizard

• The current CDT new project wizard is horribly confusing
  • So much so, few adopters use it, they usually create their own
• And we create our own for Arduino
  • It’s really simple – just the standard pages
  • Maybe at some point we will add configuration options like selecting libraries
• Creates project, sets nature, adds build settings
• Project files create from templates
  • Using freemarker which is being added to cdt.core, maybe Orbit
  • “Powerful template language”
    • Conditionals, looping through lists, macros
  • Generates initial source code and Makefile
Build Integration

• Set up to use the AVR GCC compiler
  • Shipped with the official Arduino IDE

• CDT Managed Build Definition extension
  • Defines compiler and adds Board Type setting
  • Not using managed build, just calling standard make
  • But board setting sets environment vars for the build

• Language settings provider
  • Defines built-in macros and include path
    • Sets up the correct arguments to the compiler to do that
  • Build output parser extends the base GCC output parser
    • Looks for avr-g++ on the command line to trigger
Serial Port

- Serial port is all you have when you boot a microcontroller
  - Simple, ancient, readily available
  - And with Serial to USB converters, don’t even need a port on your host machine
- But serial port in Java isn’t standardized, need a JNI lib
  - RXTX, but it’s dead and was LGPL anyway
  - jSSC is alive but it’s LGPL
  - Serial port programming if you keep it simple is simple
- Adding SerialPort class to the CDT native plug-in
  - InputStream and OutputStream to read and write
  - SerialPort class managed the connection properties
  - And lists the available serial ports
Arduino Remote Connection Type

- Uses the new Remote v2 API
- Implements the IRemoteCommandShell service
  - To use the Remote connection type for the TM Terminal

- Also provides a target to select in the Launch Bar
  - Launch Delegate determines serial port address of active connection
  - It pauses the serial port to allow external tools to write to it
  - Calls on avrdude for AVR based boards to download the image
  - Then resumes the serial port so Terminal can see the output again
Launch Bar Support

- Select target board for launch
- Run only
- Auto generate launch configuration
  - Looks for Arduino project nature on new projects
Terminal

- Terminal View to interact with serial port
- Reuse Remote’s Terminal Connection
  - Using IRemoteCommandShell service
Installing the Toolchain

• IDE’s include build and launch tools
• Tools are GPL
• Want to ship Arduino CDT as part of CDT
• Proposed Solution
  • p2 touchpoint action that downloads and unpacks the toolchain
  • Downloads directly from the Arduino download site
  • Attached to a plug-in that shows all applicable licenses
• Other solution’s possible if this isn’t approved
  • Host elsewhere but put entry in Eclipse Marketplace
• Release with Mars in June
Questions?

• Thank you!