Programming the Arduino with the Eclipse IDE

Erica Kane for NovaLabs

http://www.thekanes.org/

What we will learn

- What Eclipse can do, and when to use it
- Installation of the Arduino software, Eclipse, and Jantje's plug-in
- Creating a new Arduino project and importing existing Arduino projects
- Working with libraries
- Common pitfalls and traps
- How to start fresh if needed

What is Eclipse?

- Full featured integrated development environment (IDE), open source and free
- Includes auto-completion, code navigation, bookmarks, auto task list *demo*

Arduino IDE vs. Eclipse

Arduino IDE

- Works out of the box
- Well supported
- Compiles existing Arduino libraries easily
- Files can be kept in their original directories
- Extremely hard to navigate large filesets
- Missing code completion and other IDE features

Eclipse

- Very powerful IDE with professional features
- Requires plug-in or extensive tuning to work with Arduino
- Compilation has some differences
- Recommended (by me) to use symbolic links for files
- Make and compilation can be customized

Installation procedure

- Install Arduino IDE, confirm operation
- Install Eclipse Juno C/C++, 32-bit version
- Plug in Arduino board
- Install Jantje's Arduino Eclipse plug-in

Arduino software installation

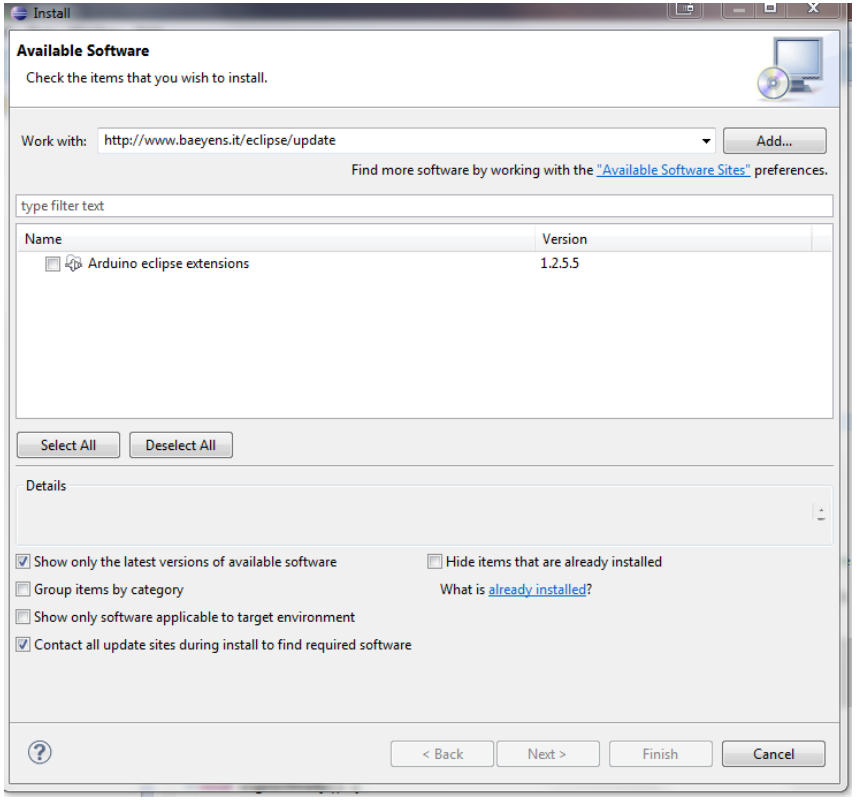
- Download at http://arduino.cc/en/Main/Software
- Choose the standard, non-beta version; this will work with version 1 of the plug-in *In theory, the beta version with work with version 2 of the plug-in but I was unable to compile the Arduino libraries (may be possible if C compilation is forced)*
- Put in directory with no spaces, i.e. C:\dev\arduino-1.0.5
- Can have beta and non-beta Arduino installations installed side by side
- Confirm you can upload to board!

Eclipse installation

- Download from http://www.eclipse.org/ Many versions available, be sure to pick Juno C/C++, 32-bit version
- Download for Windows is a simple zip file. Hang on to it as you may wish to reinstall or have multiple installations
- Put in writable directory with no spaces, i.e., C:\dev\eclipse NOT Program Files
- Accept default workspace directory
- Note, to compile non-Arduino projects you will need to use/download a C/C++ toolchain (see minGW), optional for Arduino

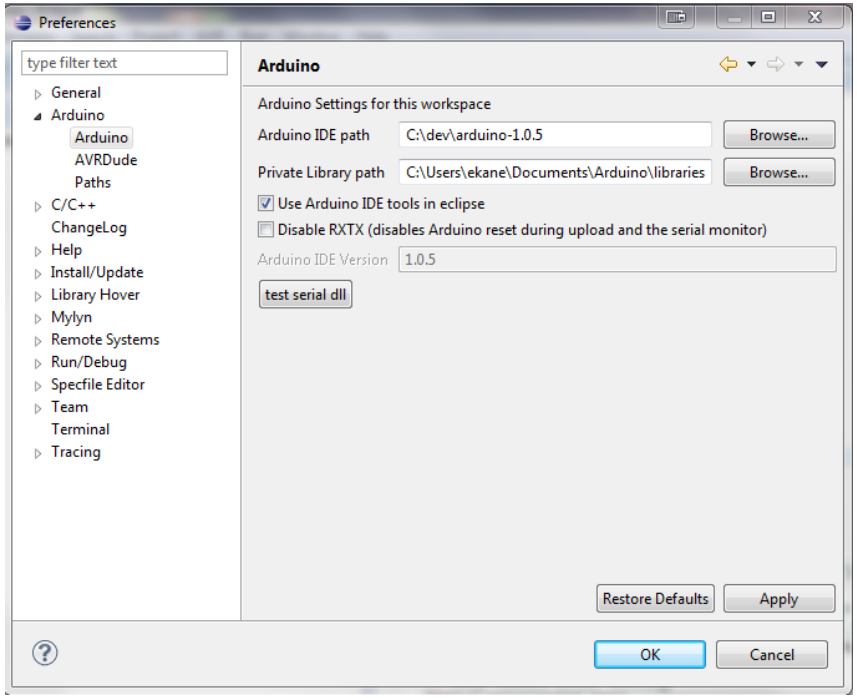
Arduino Eclipse plug-in installation

- Home page is http://www.baeyens.it/eclipse/
- Plug in your Arduino board!
- In Eclipse, Help->Install New Software
- Uncheck “Group by category”
- http://www.baeyens.it/eclipse/update in Work with, then press enter
- Select All, then Next



Arduino Eclipse plug-in configuration

- Windows -> Preferences -> Arduino -> Arduino
- Enter Arduino home and library path (Arduino IDE has library path in its Preferences if you are uncertain)
- Then C/C++->File Types
- Add *.ino and *.pde as C++ source file



Creating a new Arduino project

- For the very first one, be sure your board is plugged in! Needed to create static library in the workspace
- File -> New -> Project, Arduino -> New Arduino sketch
- Choose/confirm board (you can change later)
- Special .h file created
- Rename main .cpp file to .ino
- Compile and upload
- Serial monitor: Window -> Show view -> Other -> Arduino -> Show serial monitor; manually connect to board

Working with libraries

- To add a library, use File -> Import -> Arduino -> Import Arduino libraries
- Note this will *copy* libraries into the workspace; OK for system ones but if you wish to edit them, symbolic links recommended
- **BE CAREFUL** removing libraries, this can corrupt your workspace! From the FAQ:
  - Right click the library. Select Resource configurations->exclude from build. Select all checkboxes (normally you only have release). Select OK.
    Now the library you want to delete will be the last entry in your project and is grayed out. You can safely delete it now.
- You may also need to look at Project -> Properties -> C/C++ General -> Paths and Symbols after a library deletion

A word on symbolic links

- I recommend keeping your source files in a separate directory for version control, then symbolic linking them to the workspace
- Safer in case of workspace corruption – you can delete at will – also cleaner
- Built into Linux and Windows 7+ (yes), the Link Shell Extension makes Windows usage easier: http://schinagl.priv.at/nt/hardlinkshellext/hardlinkshellext.html

Import an existing project

- Create a new Arduino project with the same name
- Rename main .cpp file to .ino
- Import any libraries needed
- Erase generated .cpp file, link in existing one(s). *KEEP* the generated .h file, merge if necessary
- If you plan to edit the libraries, delete and link in existing ones (not usually needed for system libraries) *demo*

Common pitfalls and traps

- Must use the plug-in to add libraries
- Delete libraries as instructed earlier
- Calls to Serial and other libraries may show up as red semantic “errors” but compilation still works
  - Right-click project -> Index. Use the various options here and rebuild project until errors vanish
- See the plug-in FAQ also at: http://www.baeyens.it/eclipse/Arduino%20eclipse%20plugin%20FAQ.html
- Wire library needs tweaks to compile (see FAQ)
- We are using the AVR C++ compiler, not C. For Megas you may wish to upgrade, see http://www.thekanes.org/2013/03/14/more-robot-software-improvements/
- And if all else fails….

Starting from scratch

- Delete workspace (make sure your files are linked from elsewhere or copy them first!)
- Delete Eclipse directory
- Delete any Eclipse or .eclipse directories in the Documents or Users area
- Then unzip from your installer and start again…

Resources

- Eclipse: http://www.eclipse.org/
- Plug-in: http://www.baeyens.it/eclipse/
- My tweaks: http://www.thekanes.org/tag/arduino/