

Using Xcode to develop for 3rd party hardware

Amanda Walker
MacTech 2014

Moore's Law is making embedded devices affordable

- More than just Arduino and Raspberry Pi
- Lots of inexpensive boards based on AVR, ARM, etc.
- Internet of Things gaining momentum
- Maker movement fueling tinkering (and hardware sales)

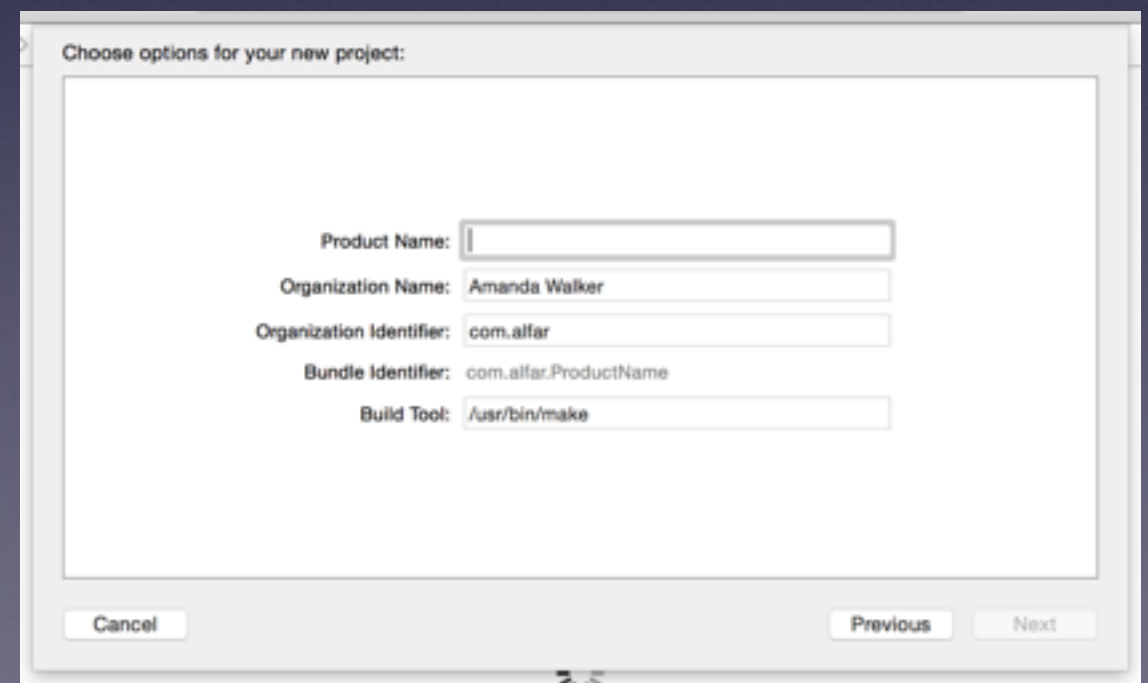
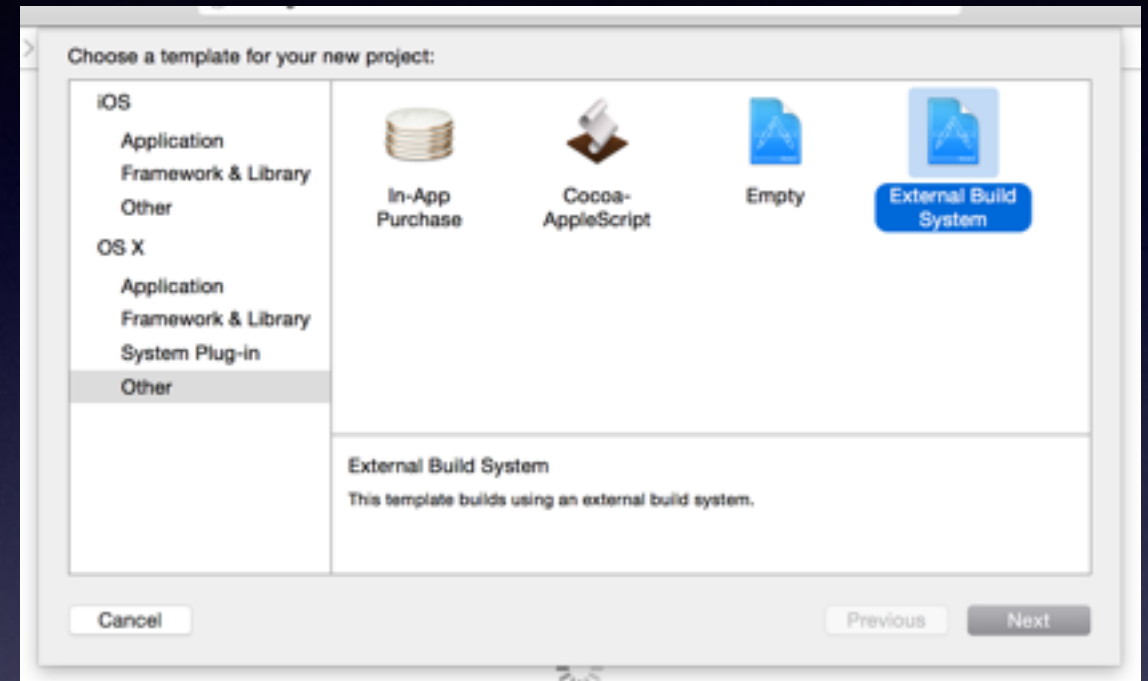


Conventional Embedded Development

- Windows-based IDEs
- A few vendors with Linux IDEs ported from Windows
- Eclipse Java-based IDE with plugins
- For the hobbyist/“maker” market, Java-based IDEs based on Wiring (example: Arduino)

The documented way

- External Build System template provided with Xcode
- Allows any external build command (make, ant, homegrown scripts, etc.)
- Lets you use Xcode as a text editor, but not much more.
- A step up from running Emacs or vi in a terminal window, but only barely.

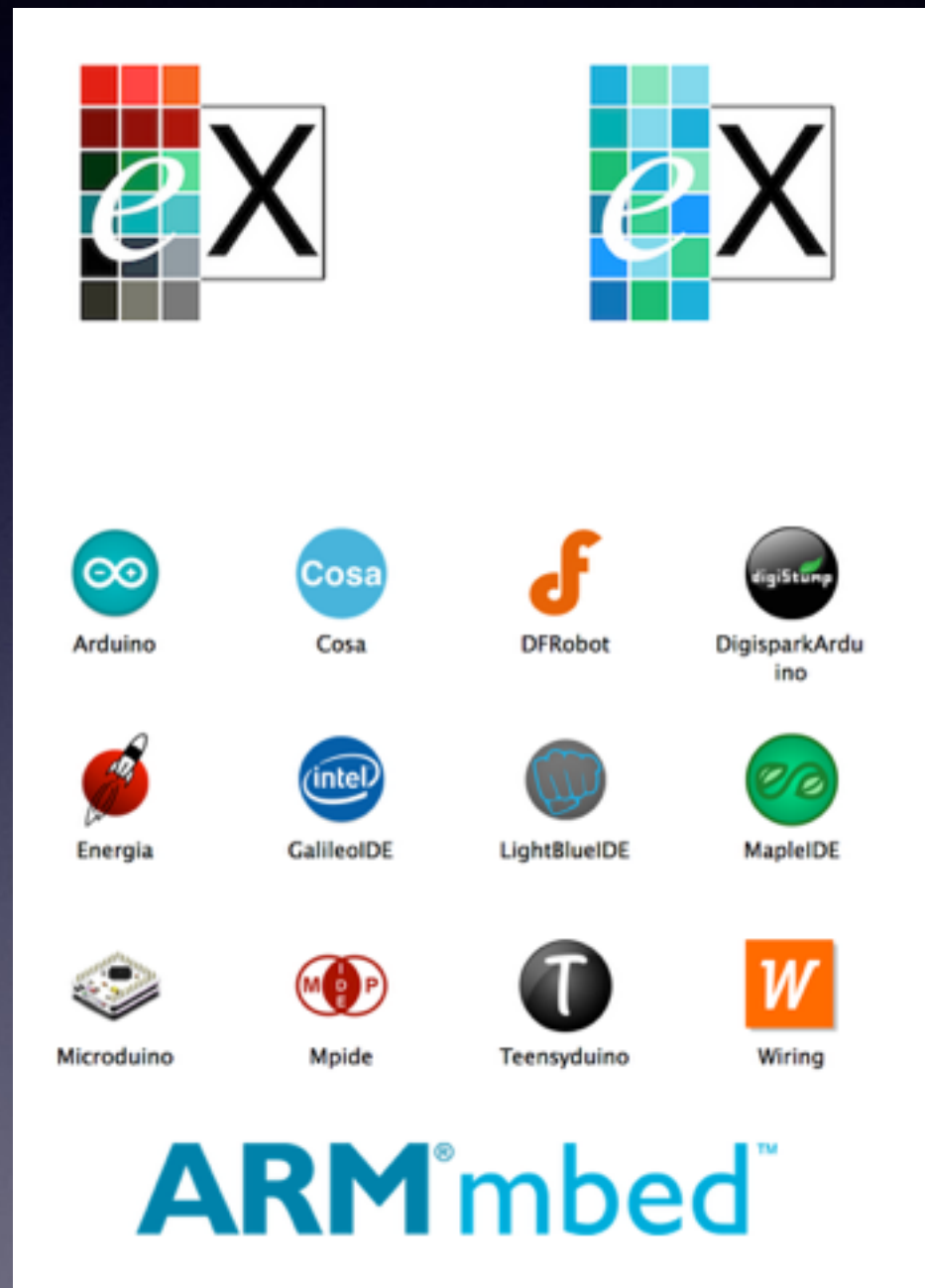


The documented way

- Compile from GNU source
- Detailed instructions by Ethernut
- Scripts, installers, etc.:
<https://github.com/esden/summon-arm-toolchain>
<http://www.obdev.at/products/crosspack/>
- Can take time and repeated attempts to get everything set up and working.



Why not both?



- Xcode template by Rei Vilo
- Uses Wiring-based IDE app bundles **as SDKs**
- Adds an “index” target that enables code completion, class browsing, etc.
- Free and donationware versions
- embedxcode.weebly.com

DEMOS

Limitations

- External build systems rely on Makefiles, which can get pretty complex
- There's no supported way to define an external debug server or device platform
- Many pieces of Xcode are open source, but not how to build a real Xcode SDK

In an ideal world

- Use Xcode's toolchain
- Use external build target to post-process the executable into a binary image or hex file and load it onto the device
- Get Ildb talking to an external debugserver like the one in [OpenOCD](#)

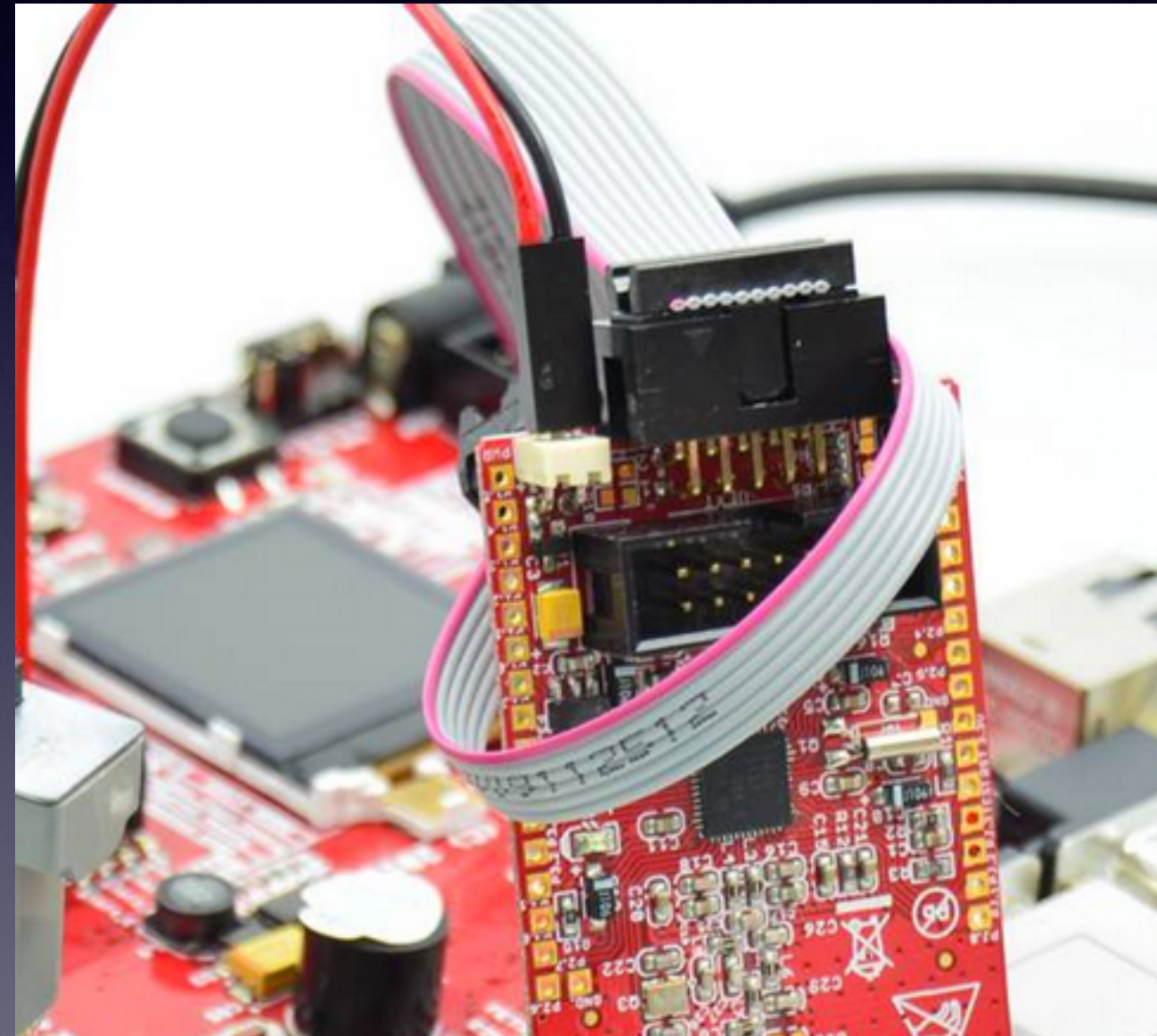


Photo credit:
OpenOCD web site (www.openocd.org)