Prototype to production: Arduino for the professional

Jacob Beningo - May 12, 2016

Despite its popularity among hobbyists and electronics enthusiasts, the Arduino has become infamous among professional embedded systems developers. I must admit that for the longest time I also viewed the Arduino as so simple it was nearly useless for professional developers. But I have changed my mind.

I've found that on a number of occasions over the last few years, rapid prototyping using Arduinos and Arduino shields has proven invaluable in moving a project forward. Despite, or perhaps because of, its abstracted simplicity, the Arduino has been key in turning an abstract idea into a defined product. For that reason, let's take a closer at the Arduino and how professional developers can benefit from it.

The Arduino hardware platform

One of the most powerful aspects of the Arduino for professional developers is the hardware ecosystem that supports it. Every Arduino board and derivative has a standard hardware interface that allows custom designed electronics to be stacked on top of the processor board to flesh out the prototype of an embedded system. The custom electronic boards, known as shields as probably most developers are aware, can literally have any type of electronics onboard such as motor drivers, sensors, actuators, LEDs or whatever the application needs may be. The popularity of Arduino among hobbyists has greatly benefited embedded system professionals because the result has been a wide variety of Arduino shields for nearly every application imaginable available off the shelf.

One of my personal favorite shields, seen in Figure 1, is the Sparkfun weather shield. This shield provides a collection of analog and digital sensors that are perfect for teaching embedded systems courses. But if you have a different requirement, a quick search on nearly any electronic vendors' website will reveal dozens of commercially available and stocked Arduino shields of all kinds. Arduino shields are typically inexpensive, costing less than $50 depending on the collection of sensors and electronics onboard.

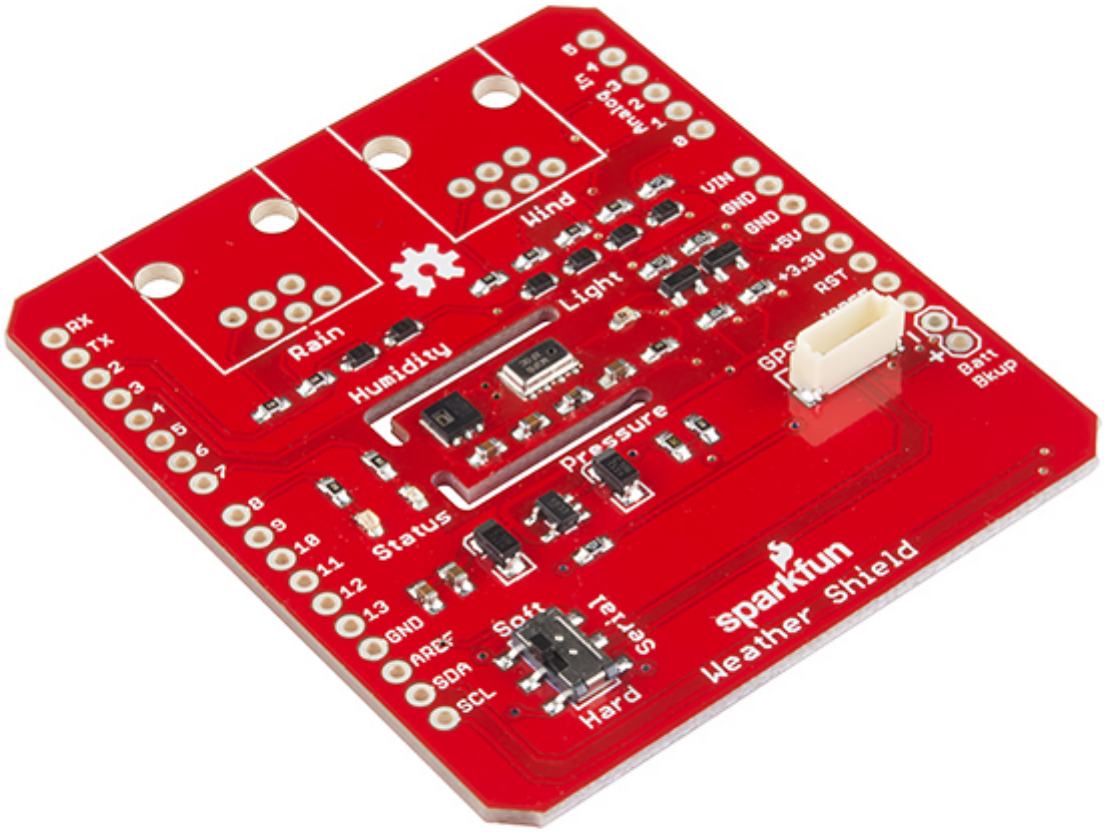

Figure 1 – Sparkfun Weather Shield

Professional developers can also leverage the Arduino hardware platform to interface with commercial devices of interest. Using available shields for CAN, SPI, RS-485, Ethernet, and other equipment interfaces it's possible to perform rapid prototyping activities for proof-of-concepts or one-off customer demos. For a few hundred dollars, a developer can easily assemble a complete hardware representation of the proposed embedded system and write some "dirty code" to make it functional.

The Arduino hardware interface has changed slightly over the years with the latest revision being based on the UNO R3 pinout. The standard interface consists of six analog inputs, fourteen digital input, outputs, a dedicated I2C channel and then miscellaneous power rails and references. An example of the UNO R3 pinout can be found in Figure 2.

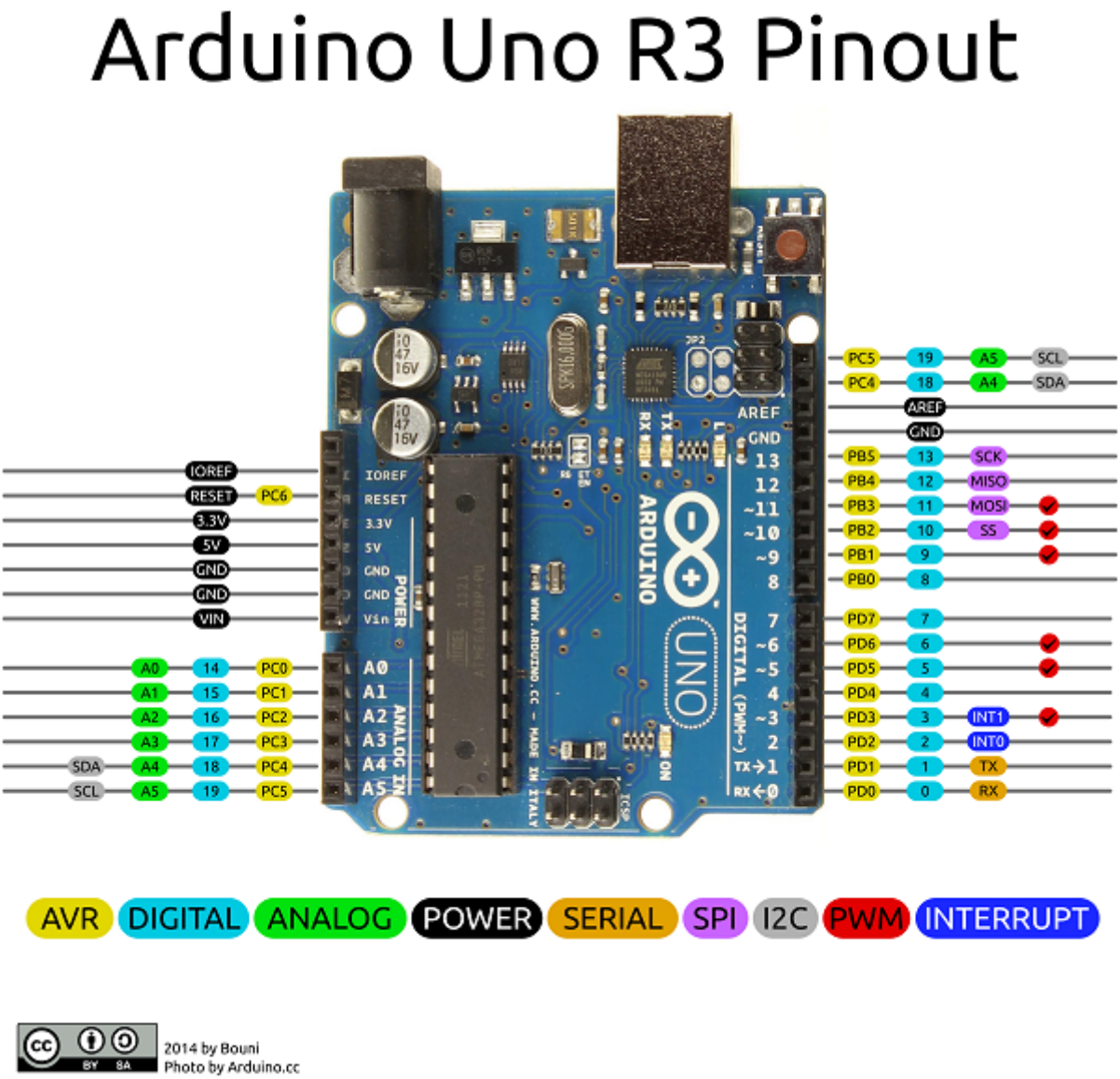

Figure 2 – Arduino Uno R3 Standard Pinout

(Source https://github.com/Bouni/Arduino-Pinout)

The Arduino shield interface is designed for low cost, low pin count microcontrollers, which can potentially be an issue for professional embedded systems developers needing more. Microcontroller companies have tried to resolve this issue by creating development boards for their more powerful processors while following the footprint for an Arduino shield. They then expanded the headers for additional functionality. By expanding their headers in the same way, developers can build their own custom shields for these enhanced development boards that utilize the extra functionality. Yet they can still also purchase off-the-shelf Arduino shields that remain compatible with the development board. The NXP FRDM-26Z and FRDM-64F are prime examples of how microcontroller companies are utilizing the Arduino shield interface and then expanding on those capabilities (There are plenty of other examples; just pick your favorite and check out its own website).

The Arduino software platform

The Arduino is more than hardware; it's a complete hardware and software prototyping system. Its software development environment and libraries leave much to be desired from a professional developer's point of view, but it is still useful to get a basic understanding of how Arduino handles

software development.

First, a developer examining the Arduino website -- arduino.cc -- will discover that there is some really strange language going on when it is talking about software. Arduino has invented a concept for the general public known as sketching, which to a professional developer is "writing code". A sketch is really nothing more than a software project but the terminology sketch comes from the fact that Arduino was originally developed as a rapid prototyping tool for individuals who knew little to nothing about software or electronics, artists for example.

Next, a would-be Arduino developer will discover that the Arduino programming language is used to program Arduino devices. Never heard of the Arduino programming language? That is because the Arduino programming language is actually nothing more than C/C++. The "Arduino language" as they refer to it is actually just a collection of libraries that provide a consistent set of APIs for controlling microcontroller peripherals.

As a professional developer, the Arduino libraries can provide a fast track for rapid prototyping. For example, API calls for controlling a digital input/output pin are *digitalWrite()* and *digitalRead()*. There are loads of different library functions for internal microcontroller peripherals and external device control such as EEPROMs and motor controllers. Developers can choose to use these libraries or instead write their own. Many of the library calls tends to be inefficient and not optimized for speed or size, though, so any development effort needs to pay careful attention to the real-time response of the built-in libraries.

The Arduino software is open source and can be used for any purpose, but developers and managers need to keep in their mind that the software was developed for prototyping purposes. The code is not written to be fault tolerant, secure, or be used in any production-intent environment. A developer will still need to go through the whole production process to take a product to market. But Arduino can at least be used to prove early on that the system could work, rather than spend months only to fail.

Conclusions

Professional developers can leverage the Arduino ecosystem to rapidly prototype and prove out an embedded system concept. Existing Arduino libraries can be used for quick and dirty development but many developers will find the software development environment wanting and will likely choose to use their own development tools and environments. Despite the professional deficiencies in the software platform, though, the use of the Arduino shields and hardware environments offer a great opportunity to help accelerate development through the use of readily available shields. Just don’t forget that Arduino is meant for rapid prototyping rather than developing production-intent systems.

Jacob Beningo is principal consultant at Beningo Engineering, an embedded software consulting company. Jacob has experience developing, reviewing and critiquing drivers, frameworks and application code for companies requiring robust and scalable firmware. Jacob is actively involved in improving the general understanding of embedded software development through workshops, webinars and blogging. Feel free to contact him at jacob@beningo.com, at his website www.beningo.com, and sign-up for his monthly Embedded Bytes Newsletter here.

Also see:

- Prototype to production - A hands-on series
- Arduino: embedded engineering for all