Chapter 4
THINKING ABOUT PROTOTYPING

- With the Internet of Things, we are always looking at **building three things in parallel: the physical Thing; the electronics** to make the Thing smart; and the **Internet service** that we'll connect to.
- The **prototype** is optimized **for ease and speed of development** and also the **ability to change and modify it.**
- Many Internet of Things projects **start with a prototyping microcontroller, connected by wires to components on a prototyping board**, such as a "breadboard", and housed in some kind of container.
- At the end of this stage, you'll **have an object that works.**
- **It's a *demonstrable* product** that you can use to convince yourself, your business partners, and your investors.
- Finally, the **process of manufacture will iron out issues of scaling up** and polish.
- You might **substitute prototyping microcontrollers and wires with smaller chips on a printed circuit board (PCB).**

SKETCHING

- Jot down some ideas or **draw out some design ideas with pen and paper.**
- That is an important **first step in exploring your idea** and one we'd like to extend beyond the strict definition to **also include sketching in hardware and software**.
- What we mean by that is the **process of exploring the problem space: iterating through different approaches and ideas to work out what works and what doesn't.**

FAMILIARITY

- If you **can already program in Python**, for example, maybe **picking a platform such as Raspberry Pi, which lets you write the code in a language you already know, would be better than having to learn Arduino from scratch**.

COSTS VERSUS EASE OF PROTOTYPING

- It is also worth **considering the relationship between the costs** (of prototyping and mass producing) **of a platform against the development effort that the platform demands**.
- It is beneficial if you can **choose a prototyping platform in a performance/ capabilities bracket similar to a final production solution**.
- That way, **you will be less likely to encounter any surprises over the cost**.
- **For the first prototype, the cost is probably not the most important issue:** the smartphone or computer options are particularly convenient if you already have one available, at which point they are effectively zero-cost.
- **if your device has physical interactions , you will find that a PC is not optimized for this kind of work.**
- **An electronics prototyping board**, unsurprisingly, *is* **better suited to this kind of work.**
- An important factor to be aware of is that the **hardware and programming choices you make will depend on your skill set**.
- **For many beginners to hardware development, the Arduino toolkit is a surprisingly good choice.**
- The **input/output choices are basic** and **require an ability to follow wiring diagrams and, ideally, a basic knowledge of electronics**.
- Yet the interaction **from a programming point of view** is essentially simple—

writing and reading values to and from the GPIO pins.

- And the language is C++.
- (A **general-purpose input/output** (**GPIO**) is an uncommitted digital signal pin on electronic circuit board whose behavior—including whether it acts an input or output—is controllable by the user at run time.)
- The **IDE pushes the compiled code onto the device where it just runs, automatically, until you unplug it.**

Case Study: Bubblino

- Its original purpose was precisely to demonstrate "how to use Arduino to do Internet of Things stuff".
- So the original **hardware connected an Arduino to the motor** for an off-the- shelf bubble machine.
- The original prototype **had a Bluetooth-enabled Arduino, which was meant to connect to Nokia phone,** which was **programmed with Python**.
- The **phone did** the hard **work of connecting to the Internet** and simply **sent the Arduino a number, being the number of recent tweets**.
- **Bubblino responded by blowing bubbles for that many seconds**.
- The **current devices are based on an Arduino Ethernet**.
- This means that the **Twitter search and XML processing are done on the device**, so **it can run completely independently of any computer, as long as it has an Ethernet connection**.
- **In a final** twist, the concept of Bubblino has been **released as an iPhone app, "Bubblino and Friends"**, which simply **searches Twitter for defined keywords and plays an animation and tune.**
- A kit such as an Arduino easily connects to a computer via USB, and you can speak to it via the serial port in a standard way in any programming language.

Prototypes And Production

- *The* biggest obstacle to getting a project started—**scaling up to building more than one device,** perhaps many thousands of them—**brings a whole new set of challenges and questions**.

Changing Embedded Platform

- When you **scale up**, you may well **have to think about moving to a different platform, for cost or size reasons**.
- If the first prototype you built on a PC, iPhone
- If you've used a **constrained platform** in prototyping, you may find that you have to **make choices and limitations in your code**.
- Dynamic memory allocation on the 2K that the Arduino provides may not be especially efficient.

- In practice, **you will often find that you don't need to change platforms**.
- Instead, you might look at, for example, **replacing an Arduino prototyping microcontroller with an AVR chip** (the same chip that powers the Arduino) and just those components that you actually need, connected on a custom PCB.

Physical Prototypes And Mass Personalisation

- Chances are that the **production techniques** that you use for the physical side of your device **won't translate directly to mass production**.
- **Digital fabrication** tools can allow each item to be slightly different, **letting you personalise each device in some way.**
- (*mass personalisation*, as the approach is called, means you can **offer something unique**).

Climbing Into The Cloud

- The server software is the easiest component to take from prototype into production.
- Scaling up in the early days will involve buying a more powerful server.
- If you are **running on a cloud computing platform**, such as Amazon Web Services, **you can even have the service dynamically expand and contract, as demand dictates**.

Open Source Versus Closed Source

- We're looking at two issues:
- Your assertion, as the creator, of **your Intellectual Property rights**
- Your **users' rights to freely tinker with your creation**

Why Closed?

- Asserting Intellectual Property rights is often the default approach, especially for larger companies.
- If you **declared copyright on some source code or a design**, someone who wants to market the same project cannot do so by simply reading your instructions and following them.
- You might also be able to **protect distinctive elements of the visual design with trademarks and of the software and hardware with patents**.
- **Note that starting a project as closed source doesn't prevent you from later releasing it as open source.**

Why Open?

- In the open source model, you release the sources that you use to create the project to the whole world.
- Why would you give away something that you care about, that you're working hard to accomplish?
- There are several **reasons to give away your work:**
- You may **gain positive comments** from people who liked it.
- It acts as a public showcase of your work, which may **affect your reputation and lead to new opportunities**.
- People who used your work may **suggest or implement features or fix bugs**.
- By generating early interest in your project, you may **get support** and **mindshare of a quality** that it would be hard to pay for.
- A **few words of encouragement** from someone who liked your design and your blog post about it may be invaluable to get you moving **when you have a tricky moment on it**.
- A **bug fix from someone** who tried using your code in a way you had never thought of may **save you hours of** unpleasant **debugging later**.

- **Disadvantages of Open Source**
- deciding to release as open source may take more resources.
- If you're **designing for other people**, you have to **make something of a high standard**, but for yourself, you often might be tempted to cut corners.
- Then having to **go back and fix everything so that you can release it in a form that doesn't make you ashamed** will take time and resources.
- **After** you **release** something as open source, you may still have a perceived **duty to maintain and support it, or at least to answer questions about it via email, forums, and chatrooms.**
- Although you **may not have *paying* customers**, your users are a community that you may want to maintain.

Being a Good Citizen:

- If you say you have an open platform, **releasing only a few libraries, months afterwards, with no documentation or documentation of poor quality could be considered rude.**
- Also, your open source work should make some attempt to play with other open platforms.

Open Source as a Competitive Advantage

- First, *using* open source work is often a **no-risk** way of **getting software** that has been **tested, improved, and debugged by many eyes**.
- Second, using open source aggressively gives your product the **chance to gain mindshare**.
- (ex: Arduino : one could easily argue that it isn't the most powerful platform ever and will surely be improved.)
- If an open source project is good enough and gets word out quickly and appealingly, it can much more easily **gain the goodwill and enthusiasm to become a platform.**

Open Source as a Strategic Weapon:

- In economics, **the concept of *complements* defines products and services that are bought in conjunction with your product**—for example, DVDs and DVD players.
- If the **price of one of those goods goes down, then demand for both goods is likely to rise**.
- Companies can therefore **use improvements in open source versions of complementary products to increase demand for their products**.
- If you **manufacture microcontrollers**, for example, then **improving the open source software frameworks that run on the microcontrollers can help you sell more chips.**
- While open sourcing your core business would be risky indeed, trying to standardise things that you use but which are core to *your competitor*'s business may, in fact, **help to undermine that competitor.**
- So **Google releasing Android as open source could undermine Apple's iOS platform**.
- **With the Internet of Things** because **several components in different spaces interact to form the final product: the physical design, the electronic components, the microcontroller, the exchange with the Internet, and the back-end APIs and applications**.\

Mixing Open And Closed Source

- We've discussed open sourcing many of your libraries and keeping your core business closed.
- It's also true that **not all our work is open source.**
- We have undertaken some **for commercial clients who wanted to retain IP**.
- **Some of the work was simply not polished enough** to be worth the extra effort to make into a viable open release.
- Adrian's project Bubblino has a mix of licences:
  1. Arduino code is open source.
  2. Server code is closed source.

Closed Source For Mass Market Projects

- **A project might be not just successful but *huge***, that is, a mass market commodity.
- The costs and effort required in moving to mass scale show how, for a physical device, the importance of supply chain can affect other considerations.

- Consider Nest, an intelligent thermostat: the area of smart energy metering and control is one in which many people are experimenting.
- The moment that an international power company chooses to roll out power monitors to all its customers, such a project would become instantaneously mass market.

Tapping Into The Community

- While thinking about which platform you want to build for, having a community to tap into may be vital or at least useful.
- **If you have a problem with a component or a library, or a question about how to do something you could simply do a Google search on the words** "arduino servo potentiometer" and find a YouTube video, a blog post, or some code.
- If you are doing something more obscure or **need more detailed technical assistance, finding someone who has already done exactly that thing may be difficult.**
- **When you are an inexperienced maker, using a platform in which other people can mentor you is invaluable.**
- **Local meetings are also a great way to discuss your own project and learn about others.**
- While to discuss your project is in some way being "open" about it, **you are at all times in control of how much you say and whom you say it to**.
- In general, **face-to-face meetings** at a hackspace may well be a friendlier and more supportive way to dip your toes into the idea of a "community" of Internet of Things makers.