www.acuityeducare.com

Acuity Educare

Internet Of Things
SEM : V

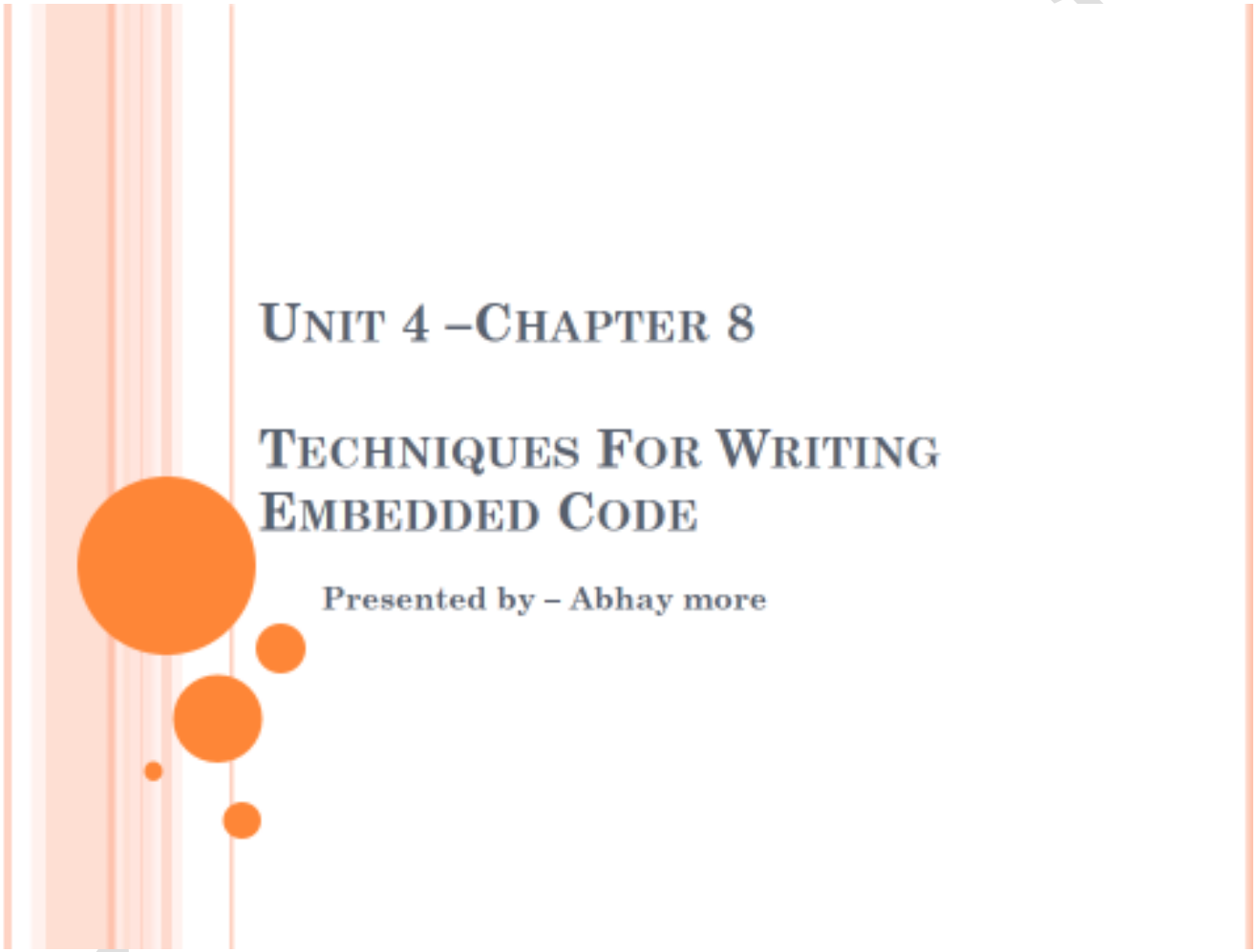
SEM V: UNIT 4

ACUITY EDUCARRE PVT. LTD.
Caring for Education...always!

607A, 6th floor, Ecstasy business park, city of joy, JSD road, mulund (W) | 8591065589/022-25600622

YouTube
Abhay More

Telegram
abhay_more



UNIT 4 –CHAPTER 8

TECHNIQUES FOR WRITING EMBEDDED CODE

Presented by – Abhay more

INTRODUCTION

- FOR THE MOST part, writing code for an embedded platform is no different to writing code for a desktop or server system.
- However, there are a few differences, and it is worth bearing them in mind as you write.
- In this chapter we explore what some of these issues are and outline ways that you can avoid or work around the problems.
- As you saw in Chapter 5, "Prototyping Embedded Devices", one of the big differences between embedded systems and "normal" computing platforms is the lack of resources available.

INTRODUCTION

- Whereas on laptops or servers you have gigabytes of memory and hundreds of gigabytes of storage, on a microcontroller the resources are typically measured in kilobytes.
- For example, your web browser will think nothing of slurping 330KB of HTML, CSS, JavaScript, and images into memory, and then copying it around to parse it and rework it into a better data structure for displaying just to show you the (relatively simple) home page of Google.
- That's 150 times the total memory available on an Arduino Uno, just for the download—before you start any processing of it.

INTRODUCTION

- Aside from resource constraints, connected devices are, by their nature, likely to be something that people turn on and then "forget about"—not literally, of course, because, one hopes, they provide a valuable service or brighten up people's lives.
- However, the owner of the device doesn't expect to have to regularly restart or maintain it. Consequently, your system should expect to run for months or years at a time without any user intervention.

INTRODUCTION

- The same goes for any configuration or tuning of the system.
- Although it's just about acceptable for server software to require an administrator to keep an eye on things and run through some maintenance procedure from time to time, this is not true for devices such as laptops and PCs.
- The aim with ubiquitous computing devices should be to take that idea of automated or *self-maintenance further still.*

Memory Management

- When you don't have a lot of memory to play with, you need to be careful as to how you use it.
- This is especially the case when you have no way to indicate that message to the user.
- The computer user presented with one too many "low memory" warning dialog boxes will try rebooting, and so will the system administrator who spots the server thrashing its disk as it pages memory out to the hard drive to increase the amount of virtual memory.

Memory Management

- On the other hand, an embedded platform with no screen or other indicators will usually continue blindly until it runs out of memory completely—at which point it usually "indicates" this situation to the user by mysteriously ceasing to function.
- Even while you are developing software for a constrained device, trying to debug these issues can be difficult. Something that worked perfectly a minute ago now stops inexplicably.

MEMORY MANAGEMENT- TYPES OF MEMORY

- Before we get into the specifics of how to make the most of the resources you have available, it's worth explaining the different types of memory you might encounter.

ROM:

- Read-only memory refers to memory where the information stored in the chips is hard-coded at the chips' creation and can only be read afterwards.

MEMORY MANAGEMENT- TYPES OF MEMORY

- This memory type is the least flexible and is generally used to store only the executable program code and any data which is fixed and never changes.
- Originally, ROM was used because it was the cheapest way of creating memory, but these days it has no cost advantage over Flash chips, so their greater flexibility means that pure ROM chips are all but extinct.

MEMORY MANAGEMENT- TYPES OF MEMORY

Flash:

- Flash is a semi-permanent type of memory which provides all the advantages of ROM—namely, that it can store information without requiring any power, and so its contents can survive the circuit being unplugged—without the disadvantage of being unchangeable forever more.
- The contents of flash memory can be rewritten a maximum number of times, but in practice it is rare that you'll hit the limits.

MEMORY MANAGEMENT- TYPES OF MEMORY

- Reading from flash memory isn't much different in speed as from ROM or RAM.
- Writing, however, takes a few processor cycles, which means it's best suited to storing information that you want to hold on to, such as the program executable itself or important data that has been gathered.

RAM:

- Random-access memory trades persistence for speed of access.

MEMORY MANAGEMENT- TYPES OF MEMORY

- It requires power to retain its contents, but the speed of update is comparable with the time taken to read from it.
- As a result it is used as the working memory for the system—the place where things are stored while being processed.
- If you know that the contents of a variable won't ever change, it is better to define that variable as a constant instead. In the C and C++ programming languages, you do this by using the **const** keyword.

MEMORY MANAGEMENT- TYPES OF MEMORY

- This keyword lets the compiler know that the variable doesn't need to live in RAM because it will never be written to—only read from.
- Using constants this way can be a big saving if you have any large lookup tables or other big data structures. It's much better to get this storage out of RAM and into flash.
- The Arduino platform, for example, provides an additional macro to let you specify that certain strings should be stored in flash memory rather than RAM. Wrapping the string in F(...) tells the system that this is a "flash" string rather than a "normal" one:

```
Serial.println("This string will be stored in RAM");
Serial.println(F("This one will be in flash"));
```

Memory Management- Making The Most Of Your RAM

- Now that you've moved everything that you can out of RAM and into flash, all that remains is to work out ways to make better use of the free memory you have.
- When you have only a few kilobytes or tens of kilobytes of RAM available, it is easier to fill up that memory, causing the device to misbehave or crash.
- Yet you may want to use as much of the memory as possible to provide more features.

Memory Management- Making The Most Of Your RAM

- This consideration is important, and it's easier to make the best trade-off between maximising RAM usage and reliability if your memory usage is *deterministic—that is, if you know the maximum amount of* memory that will be used.
- The way to achieve this result is to not allocate any memory dynamically, that is, while the program is running.
- To people coming from programming on larger systems, this concept is exotic; after all, when you're downloading some information from the Internet, for example, how could you possibly know beforehand exactly how large it is going to be?

MEMORY MANAGEMENT- MAKING THE MOST OF YOUR RAM

- What happens if the web page you're retrieving has gained an extra paragraph or two since you wrote the code?
- Rather than download the entire page into memory at once, you download it in chunks—filling the buffer each time and then working through that chunk of data before moving on to the next one.
- An upside of this approach is that you are able to process pages which are much larger than you could otherwise process; that is, you can handle datasets which are bigger than the entire available memory for the system!

MEMORY MANAGEMENT- MAKING THE MOST OF YOUR RAM

- The code used for Bubblino, which runs on an Arduino board with only 2KB of RAM, can easily process the standard response XML from Twitter's search API, which is typically 10–15KB per download.
- Because Bubblino's code condenses all that text into a single number—the count of new tweets.
- All it needs to track is whether or not each tweet is newer and the timestamp for the newest tweet it has seen (so it can pick up where it left off next time around).

Memory Management- Making The Most Of Your RAM

- The downside of this sort of single-pass parsing, however, is that you have no way to go back through the stream of data. After you discard the chunk you were working on, it's gone.
- In case you come to know whether the data is required or not only by processing the complete page. In these cases you could choose a system with more memory available in the first place.
- Alternatively, you might cache the download to an SD card or other area of flash memory.
- In systems with only a few kilobytes of RAM, we recommend exclusively using the stack to store variables.

Memory Management- Making The Most Of Your RAM

- But strictly speaking, it is still possible to run out of memory when just using the stack. This situation is called *stack overflow*. *Stack Overflow* happens when your program has used more than during execution (for example, in a deeply nested recursive routine).
- One way to reduce the chance of this situation occurring is to keep the number of global variables to a minimum.
- Global variables are attractive, particularly to newcomers to coding because they're available everywhere.

Memory Management- Making The Most Of Your RAM

- You don't have to worry about how to pass them from function to function.
- *However, because they are always allocated (as you* can see from the example in the preceding sidebar), any global variables take up valuable RAM at all times.
- In comparison, local variables exist only during the function in which they're declared and so take up space only when they're needed.
- The other way to keep down your stack is to avoid using recursive algorithms. They are elegant and can make your code easier to understand, the stack grows with each recursion.

Performance And Battery Life

- When it comes to writing code, performance and battery life tend to go hand in hand—what is good for one is usually good for the other.
- Whether either or both of these are things that you need to optimize depends on your application.
- A device which is tethered to one place and powered by an AC adaptor plugged into the wall isn't as reliant on energy conservation.
- However, consuming less energy is something to which all devices should aspire.

Performance And Battery Life

- Similarly, if you're building something which doesn't have to react instantly—maybe an ambient notifier for a weather forecast, which doesn't have any ill effect if it updates a few seconds later—or if it doesn't have an interactive user interface which needs to respond promptly to the user's actions, maximising performance might not be of much concern.
- For items which run from a battery or which are powered by a solar cell, and those which need to react instantaneously when the user pushes a button, it makes sense to pay some attention to performance or power consumption.

Performance And Battery Life

- A lot of the biggest power-consumption gains come from the hardware design.
- In particular, if your device can turn off modules of the system when they're not in use or put the entire processor into a low-power sleep mode when the code is finished or waiting for something to happen, you have already made a quick win.
- That said, it is still important to optimize the software, too!
- After all, the quicker the main code finishes running, the sooner the hardware can go to sleep.

Performance And Battery Life

- One of the easiest ways to make your code more efficient is to move to an event-driven model rather than polling for changes.
- The reason for this is to allow your device to sit in a low power state for longer and leap into action when required.
- On the hardware side, look to use processor that invoke your processing code only when the relevant sensor conditions are met.
- If your code needs to pause for a given amount of time to allow some effect to occur before continuing, use calls which allow the processor to sleep rather than wait in a busy-loop.

Performance And Battery Life

- The service API that you're talking to might have options which reduce how much information it sends you.
- When you're downloading tweets from a certain account on Twitter, for example, you can ask for only tweets after a specified ID.
- The first time you call the API, you have to deal with all the tweets it sends, but on subsequent calls, you can ask just for tweets since the ID of the most recent one that you already processed.

Performance And Battery Life

- Following are a few habits that help make your code generally more efficient:
  - When you are writing if/else constructs to choose between two possible paths of execution, try to place the more likely code into the first branch—the *if rather than the else part—as follows:*

```
if something is true
    The more likely to happen code goes here
else
    The less likely path of execution should go here
```

  - Declaring data as constant where it is known never to change can help the compiler to place it into flash memory or ROM.
  - Avoid copying memory around. Moving big chunks of data from place to place in memory can be a real performance killer.

Libraries

- These days, when developing software for server or desktop machines, you are accustomed to having a huge array of possible libraries and frameworks available to make your life easier.
- Need to parse a chunk of RSS XML? No problem. Just pull in the RSS parsing library for your language of choice.
- Want to send an email? Don't worry; there's a module for that, too. And so on and so on.
- In the embedded world, tasks are often a little trickier.

LIBRARIES

- It's getting better with the rise of the system-on-chip offerings and their use of embedded Linux, where most of the server packages can be incorporated in the same way as you would on "normal" Linux.
- On the other hand, microcontrollers are still too resource-constrained to just pull in mainstream-operating system libraries and code.
- You may use a library but if it does lots of memory allocations or extensive processing, you probably are better off starting from scratch or finding one that's already written with microcontroller limitations in mind.

LIBRARIES

- We don't have space here to cover all the possible libraries that are available, and we're by no means aware of everything that's available. However, here are a few which might be of interest:
  - **lwIP:** lwIP, or LightWeight IP, is a full TCP/IP stack which runs in low-resource conditions. It requires only tens of kilobytes of RAM and around 40KB of ROM/flash. The official Arduino WiFi shield uses a version of this library.
  - **uIP:** uIP, or micro IP, is a TCP/IP stack targeted at the smallest possible systems. It can even run on systems with only a couple of kilobytes of RAM.

LIBRARIES

- **uClibc:** uClibc is a version of the standard GNU C library (glibc) targeted at embedded Linux systems. It requires far fewer resources than glibc.
- **Atomthreads:** Atomthreads is a lightweight real-time scheduler for embedded systems. The scheduler switches between the tasks quickly enough that it looks that way, just like the multitasking on your PC.

DEBUGGING

- One of the most frustrating parts of writing software is knowing your code has a bug, but it's not at all obvious where that bug is.
- In embedded systems, this situation can be doubly frustrating because there tend to be fewer ways to inspect what is going on so that you can track down the issue.
- Building devices for the Internet of Things complicates matters further.
- Modern desktop IDEs have excellent support for digging into what is going on while your code is running.

DEBUGGING

- You can set breakpoints which stop execution when a predefined set of conditions is met, at which point you can poke around in memory to see what it contains, evaluate expressions to see whether your assumptions are correct, and then step through the code line by line to watch what happens.
- You can even modify the contents of memory or variables on the fly to influence the rest of the code execution and in the more advanced systems rewrite the code while the program is stopped.

DEBUGGING

- The debugging environment for embedded systems is usually more primitive.
- If your embedded platform is running a more fully featured operating system such as Linux, you are better placed than if you're developing on a tiny microcontroller.
- Systems such as embedded Linux usually have support for remote debugging with utilities such as gdb, the GNU debugger.
- You have access to a range of capabilities similar to desktop debugging— the ability to set breakpoints, single-step through code, and inspect variables and memory contents.

DEBUGGING

- Another way to get access to desktop-grade debugging tools is to emulate your target platform on the desktop.
- Because you are then running the code on your desktop machine, you have access to the same capabilities as you would with a desktop application.
- The downside of this approach is that you aren't running it on the exact hardware that it will operate on.
- Although it is a useful way to flush out initial bugs, there's likely to be the odd problem that you don't catch this way and that reveals itself only when you run it on the final hardware.

DEBUGGING

- If you need on-the-hardware debugging and your platform doesn't allow you to use **gdb**, JTAG access might give you the capabilities you need.
- JTAG is named after the industry group which came up with the standard: the Joint Test Action Group.
- Initially, it was devised to provide a means for circuit boards to be tested after they had been populated, and this is still an important use.
- However, since its inception, JTAG has been extended to provide more advanced debugging features.

DEBUGGING

- If you don't have access to any of these tools, you have to fall back on some of the simpler, yet tried-and-tested techniques.
- The most obvious, and most common, poor-man's debugging technique is to write strings out to a logging system.
- This approach is something that almost all software does, and it enables you to include whatever information you deem useful.
- That could be the value of certain variables at key points in the code.

DEBUGGING

- Or if you suspect the system is running out of RAM, writing out the amount of free space at startup and then at various points throughout the code can help you work out whether that is the case.
- If all else fails, the debugging tool of (often not-so-) last resort is a variation on what was probably your first step in building hardware: flashing an LED.
- As long as you have one GPIO pin free, you should be able to connect an LED and have your code turn it on at a given point.

Organising RAM: Stack versus Heap

When the system first boots up, it has all RAM available to store things in, but how does it decide what goes where and how to find it later? Two general concepts for arranging memory are used: the stack and the heap. Each has its advantages and disadvantages, and computers (including most embedded systems) tend to make use of both.

The stack is organised just as the name implies—like a stack of papers. New items which are added to the stack go on the top, and items can be removed only in strict reverse order, so the first thing to be removed is the last item that was placed onto the stack.

This arrangement makes it easy for the processor to keep track of where things are and how much space is being used because it has to track only the top of the stack. The downside to this approach is that if you're finished with a particular variable, you can release the memory used for it only when you can remove it from the stack, and you can do that only when everything added since it was allocated is removed from the stack, too.

Consequently, the stack is really only useful for

- Items that aren't going to survive for long periods of time
- Items that remain in constant use, from the beginning to the end of the program

Global variables, which are always available, are allocated first on the stack. After that, whenever the path of execution enters a function, variables declared within it are added. The parameters to the function get pushed onto the stack immediately, while the other variables are pushed as they are encountered. Because all the variables within a function are available only to code inside it, when you reach the end of that function, all those parameters and variables are ready to be discarded. So the stack gets unwound back to the same size it was just before control passed to the function.

The space on the stack is used up as the algorithm dives deeper into the nest of functions and released as execution winds back. For example, consider this pseudocode:

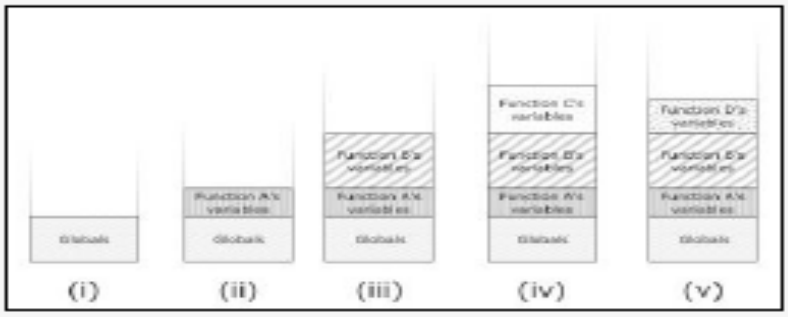
```
// global variables
function A {
    variable A1
    variable A2
    call B()
}
function B {
    variable B1
    variable B2
    variable B3
    call C()
    call D()
}
```

```
function C {
    variable C1
    // do some processing
}
function D {
    variable D1
    variable D2
    // do some other processing
}

// Main execution starts here
// Just call function A to do something...
call A()
...
```

Then, stack usage proceeds as follows:

1. Before function A is called, the stack looks like state (I).
2. As execution moves into function A, its variables are added to the stack (II).
3. Function A then calls function B, resulting in its variables being added to the stack (III).
4. Inside function B, first function C is called, resulting in its variables being added to the stack (IV).
5. When execution returns from function C, its variables are removed from the stack, taking you back to stack (III).
6. Then function D is called, so its variables are pushed onto the stack instead (V).
7. Then execution returns to function B, with D's variables removed (III).
8. And back to A, removing B's variables (II).
9. And, finally, you leave function A, dropping back to just the global variables being defined (I).



Stack usage at points during the example program execution.

continued

continued

As you can see, the maximum memory usage on the stack depends very much on the execution path that code can take through your code.

The heap, in comparison, enables you to allocate chunks of memory whenever you like and keep them around for as long as you like.

The heap is a bit like the seating area of a train where you fill up the seats strictly from the front and have to keep everyone who is travelling as a group in consecutive seats. To begin, all the seats are empty. As groups of people arrive, you direct them to the next available block of seats.

If, for example, a group of six people gets off at one stop, you are left with a block of six empty seats in the middle of the train. If the next group to get on is a group of three, those people can take up half of those empty seats, which leaves you with an empty block of three.

When a group of four people gets on at the following stop, they can't fit into the three empty seats because they have to sit together, so they sit towards the rear of the train where all the empty seats are.

This sort of behaviour continues happily, with people coming and going and filling up and vacating seats. But two possible problems exist. First, you might simply have more people to fit on the train than there are seats (this is the same problem as running out of memory). The second problem is more subtle: though you theoretically have enough free seats for the next group of passengers, those free seats are spread across the train and aren't available in a continuous block of free seats. This last situation is known as memory fragmentation.

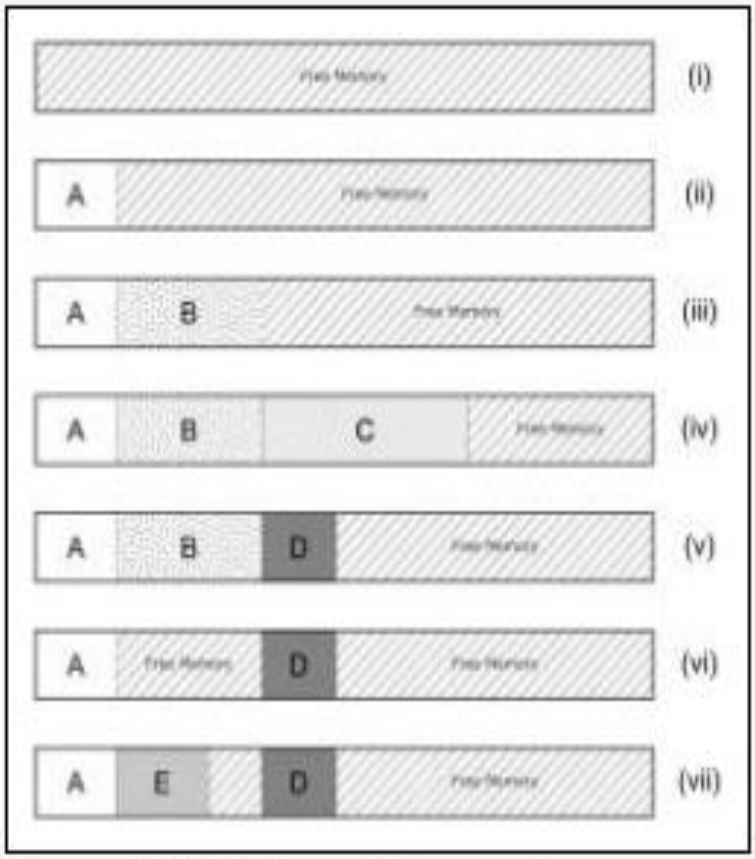
Like we did with the stack, some pseudocode will help demonstrate normal usage of the heap:

```
create object A (size 20 bytes)
create object B (size 35 bytes)
create object C (size 50 bytes)
// do some work that needs object C
delete object C
create object D (size 18 bytes)
// do more work with objects B and D
delete object B
create object E (size 22 bytes)
```

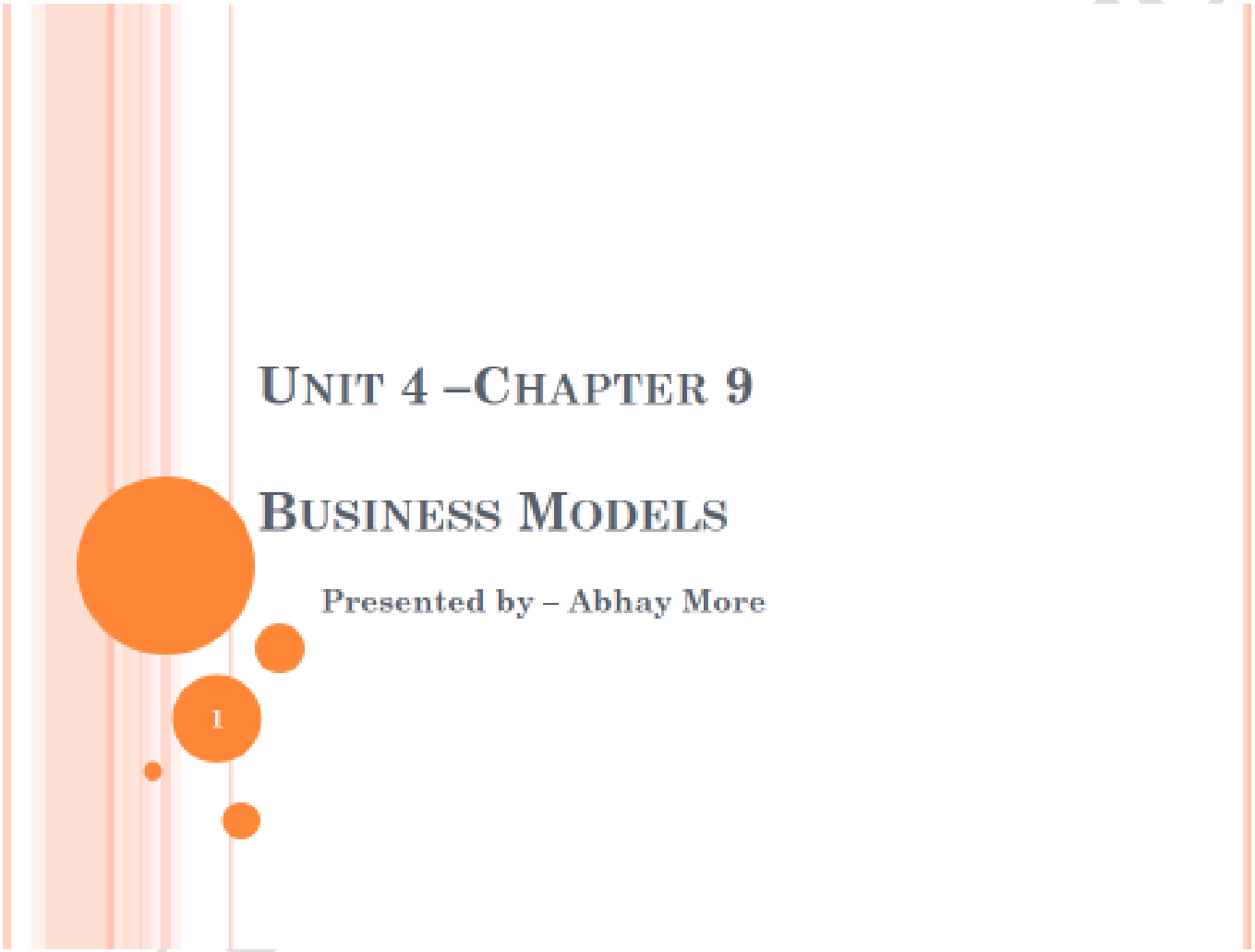
Then, as execution flows through the code, the heap will evolve as follows:

1. At the start of execution, the heap will be empty (I).
2. Object A is added to the heap (II), taking up 20 bytes of space.
3. Object B is added to the heap (III), consuming a further 35 bytes straight after the space for object A.
4. Object C is added to the heap (IV), adding 50 bytes to the heap right after object B.
5. Object C is no longer needed and is deleted, releasing the space it consumed on the heap and taking us back to heap (III).

6. Object D is created and takes up 18 bytes of the space just vacated by object C (v).
7. Now object B is finished with and deleted. As other code might be relying on the position of object D, we can't move it, so there's now a free space between objects A and D (vi).
8. Object E is created. It requires 22 bytes of space, which means it will fit in the hole left by object B (vii).



Heap usage at points during the example program execution.



INTRODUCTION

- IF YOU ARE primarily a maker or a programmer, & not an entrepreneur, you may have only a dim idea of what a "business model" is.
- In casual discussion, this expression seems to refer almost exclusively to how the business makes money.
- For example, when one is talking about Twitter or Pinterest or the latest social media sensation, a common put-down is "Have they got a business model yet?
- Sure, they're big now, but do they have any idea how they're going to make money out of it?"

3

INTRODUCTION

- But there is more to a business model than just money. We could define it as a "hypothesis about what customers want, how they want it, and how an enterprise can organize to best meet those needs, get paid for doing so, and make a profit".
- This definition brings a number of factors:
  - A group of people (customers)
  - The needs of those customers
  - A thing that your business can do to meet those needs
  - Organisational practices that help to achieve this goal and to be able to carry on doing so, sustainably
  - A success criterion, such as making a profit

5

INTRODUCTION

- All these aspects are relevant as much to hobbyist or not-for-profit projects as they are to commercial enterprises, though for the last point profit might be substituted for "improving the world" or "having fun" as criteria for success.
- We start with an overview of business models over time, to get a flavour for the topic, and then look at a commonly used way to evolve a model.
- We then look at how existing Internet of Things companies have modelled themselves and think about where they may end up.
- Finally we take a practical look at starting a company, from initial funding, and discuss the advantages of a "lean startup" approach.

7

A SHORT HISTORY OF BUSINESS MODELS

- From the earliest times, and for the great majority of human existence, we have gathered in tribes, with common property and shared resources.
- This is an almost universal pattern amongst hunter-gatherers, as it means that every member of the tribe can find food and shelter even if they have not been lucky foraging or hunting that day.
- We could describe this form of collectivism as a basic *gift economy*.

11

A Short History Of Business Models

- *Gift economies develop where those with the* appropriate skills can provide their products or services—hunting, pottery, livestock, grain, childcare—and expect repayment of this obligation not immediately but with a gift of comparable worth later.
- This is not a written debt but a social obligation, which the recipient will repay in due course, perhaps when hunting is good, when she happens upon the raw materials for her craft, or even much later in the year at harvest time.

13

A Short History Of Business Models

- Development of systems such as barter and money developed only at the edges, between different tribes.
- We could argue that the first of what we could recognise as modern business models developed at these borders and resulted from the technology required to move products and obligations through space and time.

15

A Short History Of Business Models – Space And Time

- While neighbouring tribes might have discovered variants in the local area's resources—animal, vegetable or mineral—it is when trade develops with others from far-off lands that it becomes really interesting.
- A merchant might sell silks made in his village to a region where these cloths are rare and in demand in exchange for aromatic spices which will be highly prized back home.

17

A Short History Of Business Models – Space And Time

- But long-distance trade brings with it a whole set of problems: while nomadic hunter-gatherers were adept at finding food and making a home on the move, merchants have to carry larger quantities of goods for sale and want to maximise the time travelling rather than doing the myriad tasks required for subsistence and shelter.
- Their goods and food carried will have to last far longer, so they will need to be protected and preserved.
- Above all, they need to have a reliable means of transport for themselves and their merchandise.

19

A Short History Of Business Models – Space And Time

- Technological advancements such as waterway navigation and portage of boats over land opened up new possibilities, as did the domestication of animals such as the camel, which unlocked trade routes through the Western Arabian deserts.
- We touched briefly on the preservation of food, whether through salting or smoking, or simply better storage technology such as grain silos.
- As well as facilitating transportation through space, preservation is also a way of transporting goods through *time*.

21

A Short History Of Business Models – Space And Time

- A farmer or trader who can afford to not eat or sell all his produce during the glut of harvest can fetch a better price months later at a higher price.
- So, a merchant trader's business is transporting goods through space and time, and his suppliers, the producers, benefit from that by being able to sell a bulk of their produce in one go, after which they can continue with their daily life and work.
- Money, then, abstracted trade further, setting an easy-to-calculate exchange rate between a fixed currency and the product being exchanged.

23

A Short History Of Business Models – Space And Time

- In the original gift economies, producers could pay their obligations only periodically or intermittently, according to the rhythms of hunting, farming, or craft.
- With money, this obligation was abstracted and could be paid back at arbitrary times.
- In this sense, money is another technology which allows travel through time.
- The versatility and ease of calculation which this development brought with it made it easier to develop new business models, such as investing in other merchants' trade expeditions in return for a given share or the development of interest on loans.

25

A Short History Of Business Models – From Craft To Mass Production

- When Gutenberg demonstrated his printing press circa 1450, books changed from being priceless treasures, hand-crafted by monks and artisans, to a commodity that could be produced.
- Soon every bourgeois family could afford their own books, at least a copy of the Gutenberg Bible, the first mass-produced book.
- It is no exaggeration to suggest that the invention laid the foundations for an information culture which is currently exemplified by the Internet and the World Wide Web.

27

A Short History Of Business Models – From Craft To Mass Production

- Now, painstakingly copying ancient texts onto vellum and stamping them onto paper with the latest innovation appear to end with the same result, a book.
- That the latter was some thousand times faster was not simply a quantitative change but a *qualitative one: information is no longer so rare, valuable,* and fragile that it must be preserved by gatekeepers (the ruling classes and the church) but can be so widely spread that everyone can have access to it.

29

A Short History Of Business Models – From Craft To Mass Production

- Trade routes would play their part here too, as the printing press spread to the New World and India via the sea routes that would be discovered by the end of the century.
- The cost of printing would become ever smaller as the technology spread, leading to new business models with the rise of newspapers and pamphlets.
- By the time Dickens was writing his novels in the mid-nineteenth century, he could publish them a chapter at a time, by monthly or weekly subscription.

31

A Short History Of Business Models – From Craft To Mass Production

- In 1884, the British company Lever Brothers launched Sunlight Soap, the first household soap to be sold not by weight, to be cut in the shop by the grocer, but packaged in bars and branded with a logo.
- This was an innovation in mass consumerism, whereby the brand established a link of trust direct with the consumer, relegating the middleman, the grocer, to becoming just a way to deliver the product to the consumer.
- Mass production, perfected by Ford Motor Company, was another major change in business model, driven not by how Henry Ford sold his cars but by how he made them.

33

A Short History Of Business Models – From Craft To Mass Production

- Ford moved away from the "craft production" of cars sold by commission to highly custom requirements and made by skilled craftsmen.
- Rather, his workers specialised on a single task, and he insisted on standard gauges for parts so that the cars could be assembled and fitted together, ending up identical.
- This approach made it simple to maintain and repair a Ford car, so the average person could afford to buy one without employing a mechanic to keep it working.
- The fact that mass production also drove down the costs to produce these cars also helped keep them affordable.

35

A Short History Of Business Models – From Craft To Mass Production

- In other areas, the ethic of mass production resulted in new business models such as supermarkets, which pioneered both "self-service shopping" and the sale of a whole range of products under one roof.
- The first recognizable supermarkets appeared in the 1930s, evolved into the hypermarkets of the 1960s, and now the concept of self-service shopping has evolved to the automated tills where every shopper can be his own checkout assistant.

37

A Short History Of Business Models – From Craft To Mass Production

- Fast-food franchising began in the 1930s and exploded with McDonald's and Burger King in the 1950s.
- Standardized menus, pre-prepared ingredients, and standard practices for each franchisee to follow meant that you could now eat exactly the same meal in any of a chain restaurant's stores in your country.

39

A Short History Of Business Models – The Long Tail Of The Internet

- As we have seen, huge changes in business practice are usually facilitated by, or brought about as a consequence of, technological change.
- One of the greatest technological paradigm shifts in the twentieth century was the Internet.
- From Tim Berners-Lee's first demonstration of the World Wide Web in 1990, it took only five years for eBay and Amazon to open up shop and emerge another five years later as not only survivors but victors of the dot-com bubble.
- Both companies changed the way we buy and sell things. Chris Anderson of *Wired magazine coined and popularized the phrase "long* tail" to explain the mechanism behind the shift.

41

A Short History Of Business Models – The Long Tail Of The Internet

- A physical bricks & mortar shop has to pay rent and maintain inventory, all of which takes valuable space in the shop; therefore, it concentrates on providing what will sell to the customers who frequent it: the most popular goods, the "hits", or the Short Head.
- In comparison, an Internet storefront exposes only bits, which are effectively free.
- Of course, Amazon has to maintain warehouses and stock, but these can be much more efficiently managed than a public-facing shop.

43

A Short History Of Business Models – The Long Tail Of The Internet

- A physical bricks & mortar shop has to pay rent and maintain inventory, all of which takes valuable space in the shop; therefore, it concentrates on providing what will sell to the customers who frequent it: the most popular goods, the "hits", or the Short Head.
- In comparison, an Internet storefront exposes only bits, which are effectively free.
- Of course, Amazon has to maintain warehouses and stock, but these can be much more efficiently managed than a public-facing shop.

43

A Short History Of Business Models – The Long Tail Of The Internet

- Therefore, it can ship vastly greater numbers of products, some of which may be less popular but still sell in huge quantities when all the sales are totalled across all the products.
- Whereas a specialist shop in Liverpool; Springfield, Oregon; or Florence, Italy, may or may not find enough customers to make its niche sustainable, depending on the town's size and cultural diversity, on the Internet all niches can find a market.
- Long tail Internet giants help this process by *aggregating* products from smaller providers, as with Amazon Marketplace or eBay's sellers.

45

A Short History Of Business Models – The Long Tail Of The Internet

- This approach helps thousands of small third-party traders exist, but also makes money for the aggregator, who don't have to handle the inventory or delivery at all, having outsourced it to the long tail.
- E-books and print-on-demand are also changing the face of publishing with a far wider variety of available material and a knock-on change in the business models of writers and publishers that is still playing out today.

47

A Short History Of Business Models – The Long Tail Of The Internet

- Newer business models have been created and already disrupted, as when Google overturned the world of search engines, which hadn't even existed a decade previously.
- Yet although Google's stated goal is "to organize the world's information and make it universally accessible and useful", it makes money primarily through exploiting the long tail of advertising, making it easy for small producers to advertise effectively alongside giant corporations.

49

A Short History Of Business Models – Learning From Histroy

- We've seen some highlights of business models over the sweep of human history, but what have we learnt that we could apply to an Internet of Things project that we want to turn into a viable and profitable business?
- First, we've seen that some models are ancient, such as Make Thing Then Sell It.
- The way you make it or the way you sell it may change, but the basic principle has held for millennia.

51

A Short History Of Business Models – Learning From Histroy

- Second, we've seen how new technologies have inspired new business models.
- We haven't yet exhausted all the new types of business facilitated by the Internet and the World Wide Web.... If our belief that the Internet of Things will represent a similar sea change in technology is true, it will be accompanied by new business models we can barely conceive of today.
- Third, although there are recurring patterns and common models, there are countless variations.

53

A Short History Of Business Models – Learning From Histroy

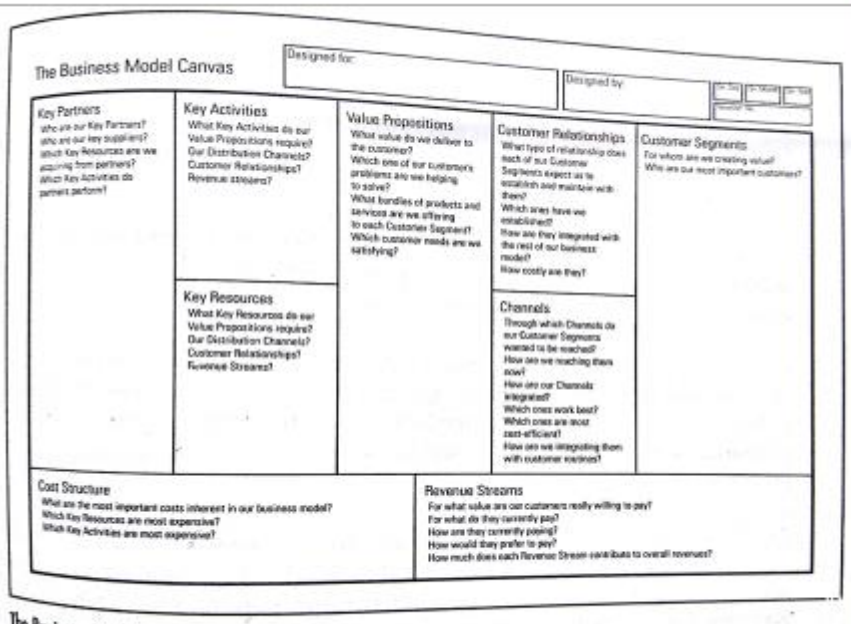
- Subtle changes to a single factor, such as the manufacturing process or the way you pay for a product or resource, can have a knock-on effect on your whole business.
- Finally, new business models have the power to change the world, like the way branded soap ushered in mass consumerism and mass production changed the notion of work itself. If the Internet of Things does change the world, it may well be through the business models it permits.

55

The Business Model Canvas

- One of the most popular templates for working on a business model is the Business Model Canvas by Alexander Osterwalder and his startup, the Business Model Foundry. The canvas is a Creative Commons–licensed single-page planner.
- At first sight, it looks as though each box is simply an element in a form and the whole thing could be replaced by a nine-point checklist.
- However, the boxes are designed to be a good size for sticky notes, emphasizing that you can play with the ideas you have and move them around.
- Also the layout gives a meaning and context to each item.

59



The Business Model Canvas.

THE BUSINESS MODEL CANVAS

- At the bottom right, we have *Revenue Streams, which is more or less the* question of "how are you going to make money?"
- The central box, *Value Propositions, is, in plainer terms, what you will be* producing—that is, your Internet of Things product, service, or platform.
- The *Customer Segments are the people you plan to deliver the product to.* That might be other makers and geeks, the general public, families, businesses.
- The *Customer Relationships might involve a lasting communication between* the company and its most passionate customers via social media.

63

THE BUSINESS MODEL CANVAS

- *Channels are ways of reaching the customer segments. From advertising and* distributing your product, to delivery and after-sales, the channels you choose have to be relevant to your customers.
- On the left side, we have the things without which we have no product to sell. The *Key Activities are the things that need to be done. The Thing needs* to be manufactured; the code needs to be written.
- *Key Resources include the raw materials that you need to create the product* but also the people who will help build it.

65

THE BUSINESS MODEL CANVAS

- Of course, few companies can afford the investment in time and money to do all the Key Activities themselves or even marshal all the Key Resources. (Henry Ford tried hard, but even he didn't manage.)
- You will need *Key Partners, businesses that are better placed to supply specific skills or* resources, because that is their business model, and they are geared up to do it more cheaply or better than you could do yourself.
- The *Cost Structure requires you to put a price on the resources and activities* you just defined. Which of them are most expensive?

67

Who Is The Business Model For?

- Primarily, the reason to model your business is to have some kind of educated hypothesis about whether it might deliver what you want from it.
- Even if you don't use a semi-formal method like the canvas we just discussed, anyone who starts up any business will have thought, at least briefly, about whether she can afford to do it, what the business is, and whether she'll get paid.
- As a programmer or a maker, you might believe it counterintuitive to think of a piece of paper with nine boxes in it as a "tool".

71

Who Is The Business Model For?

- But when you have a well-tested separation of factors to consider, the small amount of structure the canvas provides should help you think about the business and give you ways to brainstorm different ideas:
  - What if we target the product at students instead of businesses?
  - What if we outsource our design to an agency?
  - What if we sell at low volume/high value instead?
- Many great product ideas turn out to be impractical, ahead of their time, or unprofitable.

73

WHO IS THE BUSINESS MODEL FOR?

- Being able to analyze how the related concepts mesh will help you challenge your product idea and either make it stronger or know when to abandon it.
- The model is also useful if you want to get other people involved. This could be an employee or a business partner…or an investor.
- In each of these cases, the other parties will want to know that the business has potential, has been thought out, and is likely to survive and perhaps even go places. With a new business startup, you have no track record of success to point to.

75

WHO IS THE BUSINESS MODEL FOR?

- Perhaps to a lesser extent, your *customers will also be considering whether to* invest their time and money in your product. They will ask themselves certain questions about it. Let us look at some of these likely questions, from the wider field of Internet products in general.
  - **Why should I waste time trying out Yet Another Social Network? I think I'll wait and see whether all my friends join it first. This first** question is about your "Value Proposition" (that is, the product) and a reasonable concern if you are trying to get into a market that already has good or popular solutions.

77

WHO IS THE BUSINESS MODEL FOR?

- **Your online document collaboration looks great, but is it worth my moving my whole business to it? If you stop trading or change the platform, we may have to redo all the work again. Such customers** may well be interested in the details of your business model to calculate whether the risk they've identified is worth their commitment.
- **This free service is fantastic, but why don't you let me pay for it, so I can get consistency, receive support, and avoid adverts? Lastly, many** customers are aware of alternative charging models that they would prefer. Not all customers vote for the free option.

79

WHO IS THE BUSINESS MODEL FOR?

- It has been stated about "free" products: "If you're not paying for something, you're not the customer; you're the product being sold".
- Several assumptions often made about this:
  - Not paying means not complaining.
  - You're either the product or the customer.
  - Companies you pay treat you better.
  - So startups should all charge their users.

81

MODELS

- We have looked at the Business Model Canvas as a tool for generating and analysing models.
- As we saw from our history, the models have many common variants.
- It is a good idea to have a look at some of the models that Internet of Things companies have used or might use and consider some of the parameters these models relate to on the canvas.

85

MODELS – MAKE THING, SELL THING

- The simplest category of models, "make a Thing and sell it," is, of course, valid for the Internet of Things.
- Adrian sells custom-built Bubblino, and the startup Good Night Lamp is preparing to ramp up production of its eponymous lamps as an off-the-shelf product.

87

MODELS – SUBSCRIPTIONS

- A Thing would be a dumb object if it weren't for the important Internet component which allows the device to remain up to date with useful and current content.
- But, of course, this ongoing service implies costs to the provider—development, maintenance of servers, hosting costs, and in some cases even connection costs.
- A subscription model might be appropriate, allowing you to recoup these costs and possibly make ongoing profit by charging fees for your service.

89

MODELS – SUBSCRIPTIONS

- Many products could legitimately use this method, but perhaps the more complex, content-driven services would find it more convincing.
- Paying Bubblino a monthly fee to blow bubbles might seem steep, but the BERG Cloud, which delivers nicely formatted news and entertainment to its Little Printer, might have seemed an ideal product for this model.
- As it stands, content consumers do *not pay for either BERG* Cloud or for any content subscriptions.

91

MODELS – SUBSCRIPTIONS

- In the future, content publishers may pay for certain premium services. Perhaps this example shows that there is not yet a market for paid subscriptions to Internet of Things products.
- This may mean that there is a market to be built. People happily pay subscriptions to music services, corporate groupware, and of course, mobile phones, so perhaps Internet of Things products in these spaces will find subscription more appealing to their consumers.

93

MODELS – SUBSCRIPTIONS

- The so-called freemium model (a portmanteau of "free" and "premium") has always been a way to encourage paying customers while not alienating free ones.
- In this model, a smaller or larger part of your product is free, while the users are also encouraged to pay a premium to get additional features or remove limits.

95

MODELS – CUSTOMISATION

- We touched on the improvements to mass production whereby the process of buying a car can be tweaked to the buyer's requirements.
- For an Internet of Things device, at the intersection between solid thing and software, there are options for customisation that we believe may lead to new business models.
- For a mass-produced item, any customisation must be strictly bounded to a defined menu: a selection of different colours for the paintwork, options for fittings such as tyres, and for features like the onboard computer control and display.

97

MODELS – CUSTOMISATION

- The world of software is, by contrast, pathologically malleable, if we let it be.
- Early websites explored the new medium of HTML to its garish extremes, with <blink> tags and animated .gif images.
- Yet today's equivalent of home pages, offered by incumbents such as Facebook, Twitter, and Pinterest, offer small degrees of customisation within strictly defined boundaries: a selection of (tasteful) colour schemes and a choice of image to use as your avatar.

99

MODELS – CUSTOMISATION

- Many Internet of Things products have some possibility of customisation: Every Bubblino has a name (given to it by Adrian), but the user can also change which phrases he listens to on Twitter.
- BERG's Little Printer offers a selection of content to be printed but also an option of which smiley face it will print for display while waiting for a new delivery.
- The new manufacturing techniques, such as laser cutting and 3D printing, should allow great possibilities for customising even the physical devices.

101

MODELS – CUSTOMISATION

- MakieLab make dolls that can be designed online.
- Built to your specification, they are therefore unique and entirely *yours in a* way that a mass-produced doll couldn't be.

103

MODELS – BE A KEY RESOURCE

- Not every Internet of Things business will be selling a product to the mass market.
- Some will sell components or expertise to other companies—that is, component manufacturing or consultancy services.
- Effectively, in this kind of business, you are positioning yourself as a "key resource" or a "partner" in somebody else's business model.
- These business models are perfectly valid. Small companies such as Adafruit and Oomlout sell electronic components to hobbyist makers.

105

MODELS – BE A KEY RESOURCE

- These fairly straightforward supplier/consultant relationships make the point that enterprises will need to solve problems, just as consumers will.
- Environmental data consultancy amee provides means for not only consumers but also businesses and government bodies to improve their environmental impact by getting hard data about their carbon footprint—not just their direct energy usage but also the energy used to dispose of their waste.

107

MODELS – PROVIDE INFRASTRUCTURE: SENSOR NETWORKS

- Sensor data is a fascinating topic in the Internet of Things: Although there are official data sources, often very accurately calibrated and expensive to create, they may be hard to access and of course can exist only where a government body or company has chosen to apply its large but finite resources.
- The long tail of third-party data sensor enthusiasts can supplement or sometimes outclass the official streams of information.
- What is needed is a platform to aggregate that data, and one of the companies competing to fulfil that role is Xively.

109

MODELS – PROVIDE INFRASTRUCTURE: SENSOR NETWORKS

- They allow any consumer to upload a real-time feed of sensor data—for example, radiation levels in Japan after the Fukushima Daiichi nuclear disaster—and for the data from many feeds to be mapped, graphed, and compared.
- Xively have, since the beginning, intended to provide a free, public infrastructure for open source data.
- While their provision of an Internet of Things "middleware" may be hard to monetize at time of this writing, becoming a major player in the infrastructure for the growing Internet of Things market is a huge potential prize.
- Xively sold in 2011 to LogMeIn for $15 million.

111

MODELS – PROVIDE INFRASTRUCTURE: SENSOR NETWORKS

- For this kind of project to succeed, Australian technologist Andrew Fisher proposes that it must
  - Gain trust
  - Become dispersible
  - Be highly visible
  - Be entirely open
  - Be upgradeable

113

MODELS – TAKE A PERCENTAGE

- In the example of sensor networks, if the value of the data gathered exceeds the cost of the physical sensor device, you might be able to provide that physical product for free.
- In fact, energy companies quite often do this with their smart meters.
- You could also link devices to advertising to reduce the price. Although this practice is controversial, many US consumers have chosen an ad-supported version of the Kindle e-Book reader to save the initial outlay.

115

Models – Take A Percentage

- As we suggested earlier, even without charging the *end user of your Internet* of Things device, there will be many options to make a profit from somewhere (ad revenues, payment for data services from companies or state organisations, commission for data bandwidth incurred, etc.)
- Within the burgeoning field of the Internet of Things, exactly what the "product being sold" consists of is a field that remains to be explored.

117

Funding An Internet Of Things Start Up

- As important as future costs and revenues are to a well-planned business model, there will most likely be a period when you have only costs and no income.
- The problem of how to get initial funding is a critical one, and looking at several options to deal with it is worthwhile.
- If you have enough personal money to concentrate on your new Internet of Things startup full time without taking on extra work, you can, of course, fund your business yourself.

121

Funding An Internet Of Things Start Up

- Apart from the risk of throwing money into a personal project that has no realistic chance of success (which this chapter's aim is to avoid!), this would be a very fortunate situation to be in.
- And luckier still if you have the surplus money to bankroll costs for materials and staff.
- If the initial stages don't require a huge investment of money, your time will be the main limiting factor.
- If you can't afford to work full time on your new project, perhaps you can spare a day in the weekend or several evenings after work.

123

Funding An Internet Of Things Start Up

- You might be able to arrange to work part time on your day job; even an extra afternoon or day might be enough to get things moving.
- Many people try to combine a startup with a consultancy business, planning to take short but lucrative contracts which support the following period of frantic startup work.
- Paul Graham advises some caution on this approach, as the easy money from consulting may be too much of a crutch and remove one of the primary motors for a startup, the fear of failure.

125

Funding An Internet Of Things Start Up

- Making sure that you *don't need to spend huge amounts on the startup is* key.
- You probably don't need an office in the early stages, and perhaps you don't need expensive Aeron chairs.
- You can work from your kitchen table, a café, or out of a co-working space.
- Everything we've discussed in the chapters on prototyping is designed to get a Minimum Viable Product out to show to people and start gathering interest.

127

Funding An Internet Of Things Start Up – Hobby Projects And Open Source

- If your project is also your hobby, you may have no extra costs than what you would spend anyway on your free-time activity.
- One way to make a project grow faster might be to release all the details as open source and try to foster a community around it.
- This approach can be hard work and can benefit from a natural talent, experience, or luck in attracting and maintaining good collaborators.
- After you have open-sourced a project, you can't close-source it again.

129

Funding An Internet Of Things Start Up – Hobby Projects And Open Source

- Indeed, your idea, code, and schematics could be used by others in their own commercial offering.
- Careful consideration of the license used may be critical here: A more restrictive license such as the GPL requires those who build on your work to share their source code also under the same terms.
- Hence, using the GPL may help restrict commercial exploitation to only those groups that are happy for *you to, in turn, reap the benefits of their work.*

131

Funding An Internet Of Things Start Up – Hobby Projects And Open Source

- Also, when thinking about open source, remember that as the project initiator and owner, you would be the best placed in forming a company around the project and are more likely to reap benefits from the relationship with the community:
  - Many pairs of eyes and hands testing, reporting problems, fixing them, and building new features
  - Many passionate users with real use cases and opinions about the product—better than any focus group
  - The goodwill of that community, with its ready-made network of personal recommendations and social-media marketing

133

Funding An Internet Of Things Start Up – Venture Captial

- Of course, getting funding for a project from an external investor presents its own work and risks.
- The process of applying for funding takes time.
- Startups often concentrate their fundraising activities into *rounds, periods in which they dedicate much of their effort into raising a* target amount of money, often for a defined step in their business plan.
- Before any official funding round comes the informal idea of the *friends, family, and fools (FFF) round.*

135

Funding An Internet Of Things Start Up – Venture Captial

- *This stage may be the one in which you've* contributed your life savings, and persuaded your aunt, your best friend, and a local small business to pitch in the rest, on the basis of your reputation.
- A common next step would be an *angel round. The so-called angels are* usually individual investors, often entrepreneurs themselves, who are willing to fund some early-stage startups which a more formal investor (venture capitalists) might not yet touch.
- The reason might be that these angels have a technical or business background in your product or simply that, as individual investors, they may have more scope to go with their own intuition about your worth.

137

Funding An Internet Of Things Start Up – Venture Captial

- Angels typically disburse sums that are significant for early-stage startups—in the region of tens or possibly hundreds of thousands of pounds.
- However, the personal interest and experience that angels can bring to your company means that their advice, contacts, and other help may well be as useful as any money they provide.
- Though angels take on a lot of risk in investing so early, before companies have proved themselves, they tend to invest in a number of companies to spread the risk.

139

Funding An Internet Of Things Start Up – Venture Captial

- They usually want equity in your company, a percentage of the value of the company, that will pay back their investment if and when you do well.
- These angels might also demand a place on your board of directors, to oversee their investment, but also out of interest in helping the company to succeed.

141

Funding An Internet Of Things Start Up – Venture Captial

- The *venture capital (VC) round is similar, but instead of your courting* individual investors, the investor is a larger group with significant funds, whose sole purpose is to discover and fund new companies with a view to making significant profit.
- VCs may be interested if angels have already funded you and will certainly be interested if other VC companies are already looking at funding you.
- VCs will certainly want equity, probably a significant amount of it, and a position on your board of directors.
- Typically, VC funding will be larger chunks of money, from half a million pounds up.

143

Funding An Internet Of Things Start Up – Venture Captial

- Even though funding may sound like "free money", we've already seen that getting investment comes with conditions: equity and some measure of control via your board of directors.
- In addition, you need to be aware that by accepting investment through venture capital, you are committing yourself to an *exit*.
- *An exit strategy is a "method by which a* venture capitalist or business owner intends to get out of an investment that he or she has made".

145

Funding An Internet Of Things Start Up – Venture Captial

- Because your investors will want a return, your long-term goal can't just be to make your company successful but to do it in such a way as to pay back the investment. Typically, you have only two exits:
  - **You get bought by a bigger company: In this case, the buyer buys out** the investors; that is, the buyer pays the investors the value of their percentage equity of their perceived valuation of the worth of the company.
  - **You do an IPO (initial public offering)—that is, float on the stock market: This involves new shares being issued and sold to the stock** market. Although this option "dilutes" the value of the shares already issued, the existing holders are able to then sell their shares on the market too, to get back their investment, or to retain the shares if they believe that the shares will grow in value.

147

Funding An Internet Of Things Start Up – Government Funding

- Governments typically want to promote industry and technological development in their country, and they may provide funds to help achieve particular aims. Attempting to cover the variety of funding schemes across the whole world would be a project in itself, so we make some mostly general notes based in many cases on the current situation in the UK.
- The money provided still has "strings attached", but they are likely to be handled differently:
  - **Outputs:** Deliverables (aka outputs) are the metrics that an awarding body may use to tell if you are doing the kind of thing that the body wants to fund. This metric may be related to the goals that the body itself wishes to promote.

149

Funding An Internet Of Things Start Up – Government Funding

- You might be required to write regular reports or pass certain defined milestones on schedule. If your funding is given in stages, the later payments may be conditional on successful delivery of previous outputs. You should be very clear on what needs to be done and how onerous the task is. Even a large sum of money is worth less if you are going to spend significant amounts of time on secondary activities. You may be required to *match funds; that is, if you were awarded* £10,000, you would also have to raise £10,000 yourself. Certain UK bodies such as the Technology Strategy Board (TSB) currently operate in this way. It is important to understand the way this matching works. For example, you may be required to make payments with your matched funds first and then reclaim the amount spent in arrears. You therefore need to understand the process, as it may require additional fundraising, cashflow management.

151

Funding An Internet Of Things Start Up – Government Funding

- **Spending constraints:** Some funding may require you to spend a proportion of the money on, for example, business consultancy or web development, perhaps with the fund facilitator's company or associates.

153

Funding An Internet Of Things Start Up – Crowdfunding

- We've already looked at the long tail as a business model; we can think of crowdfunding as the long tail of funding projects.
- Getting many people to contribute to a project isn't exactly a new phenomenon.
- Walter Gervaise built the first stone bridge over the River Exe in 1238 through public subscription by approaching friends and other wealthy citizens. Earlier, after 27 BC, Augustus Caesar sponsored a public subscription for a statue of his physician Antonius Musa.
- Over millennia many civic and religious monuments and constructions have been funded at least partly by the public.

155

Funding An Internet Of Things Start Up – Crowdfunding

- As of 2013, the main options for crowdfunding are Kickstarter (www.kickstarter.com) and Indiegogo (www.indiegogo.com).
- Historically, Kickstarter was available to use only for funding projects based in the US, whereas Indiegogo set itself up to be "the world's funding platform".
- Now Kickstarter is available also for UK projects. If you are based in a country not covered by this organization, you might consider Indiegogo.
- Kickstarter currently has somewhat better traction than Indiegogo.

157

Funding An Internet Of Things Start Up – Crowdfunding

- More people have heard of it, and this may make it more likely that people will fund you.
- Indiegogo is open to all types of projects, including community and charitable ones, whereas Kickstarter is only for those creative projects that end up with a product, be it artistic or technological.
- With the greater restrictions set by Kickstarter, it is not surprising that there is an application process required to create a project.
- Not all projects are approved.
- Indiegogo plays much less of a gatekeeper role, allowing you to start promoting your project immediately without an approval process.

159

Funding An Internet Of Things Start Up – Crowdfunding

- Your funders are real people and will have all the variety of concerns and foibles that any group of real people have.
- This interaction with a large and diverse group is a key part of the interest of this method of funding: It is far more than just the money.
- Crowdsourcing allows you to do this before even investing your time and money in the product!
- If there is no interest, perhaps the product is not a winner as currently specified and advertised.
- If the project goes viral, as happens occasionally, and gains far more than the targeted amount, you know you have a potential hit on your hands.

161

LEAN STARTUPS

- We've looked at the advantages of running a startup on a low budget.
- The mentality needed to do this includes spending time and money only when it's really necessary—staying hungry and *lean*.
- *The concept of a "lean* startup," pioneered by Silicon Valley entrepreneur Eric Ries, springs from this idea (*The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses, Crown Business, 2011).*

165

LEAN STARTUPS

- The option in the preceding section of crowdfunding a project presented an even more appealing step on this route: running the project only *if there is a* demonstrable niche market for it.
- Many lean proponents suggest setting up a landing page for a project with a simple form to register interest.
- This is quick and simple to do, especially as numerous startups do exactly this (unbounce.com, landerapp.com, and others).

167

LEAN STARTUPS

- These simple pages allow you to propose many projects and focus only on the ones that have most feedback.
- However, if you've already done some prototyping work and have a good feeling about a single idea, taking things a step further and creating a project on a crowdfunding site may be even more appropriate.
- Doing so represents more work than creating a simple form, but you will learn far more from it!
- In many ways, this "laziness"—doing the minimum now and putting off the hard work till later—is also the reason that we have split the *prototype from* the final product.

169

LEAN STARTUPS

- There is a time to market your project, a time to ensure that the idea works, and a time to build a sellable product.
- If you are thinking "lean", you should be applying this idea at all stages.
- For example, at the first stages of production and marketing, you should be working towards the "Minimum Viable Product".
- This is still a sellable product rather than a prototype, but with all extraneous features removed, it may *feel like a* prototype of your final vision for the product.

171

LEAN STARTUPS

- All the initial efforts are towards making this product because it can be sold.
- If you have time and money afterward to add additional enhancements to the product, service, packaging, and so on, this will add more value.
- The essence, then, of lean is to be able to iterate, performing the tasks that are required to get things moving at this stage, without investing time upfront to make everything perfect.
- The fact that your business model is a *hypothesis and not set in stone can encourage you to tweak it in response to* the feedback you get from iterating your product in the real world.

173

LEAN STARTUPS

- Such tweaks are known as *pivots and usually work by changing one part of your* model—think one of the boxes on the Business Model Canvas.
- For instance:
  - **Zoom-in pivot:** Focus on what was only a part of the value proposition, and turn that into the whole Minimum Viable Product.
  - **Customer segment pivot:** Realise that the people who will actually buy your product aren't the ones you were originally targeting. While you can continue to make exactly the same product, you have been marketing it to the wrong people.

175

LEAN STARTUPS

- **Technology pivot:** Accomplish the same goals as before, but change the implementation details. While prototyping will almost certainly involve many changes in technology while you establish the best way to make the product from an engineering perspective, this pivot would be a business decision, made to improve manufacturing costs, speed, or quality.

177