THE INTERNET OF THINGS

Chapter 1

THE INTERNET OF THINGS: AN OVERVIEW

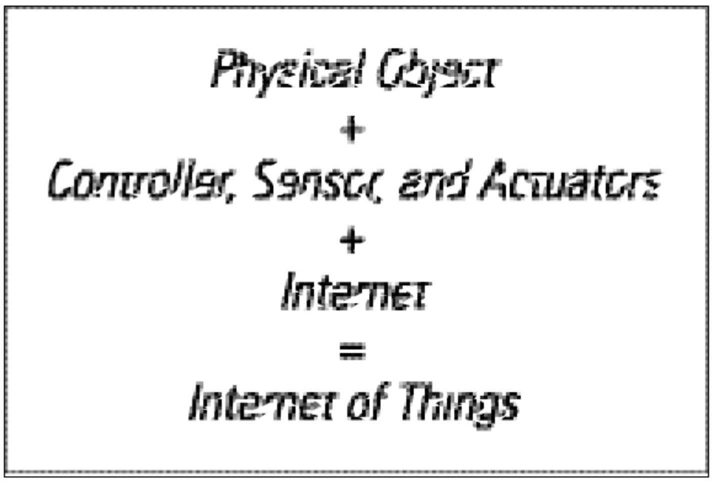
- **THE FLAVOUR OF THE INTERNET OF THINGS:**
- (Examples of IOT)
- 1) The alarm rings. As you open your eyes , you see that it's five minutes later than your usual wake-up time.
- The clock has checked the train times online, and your train must be delayed, so it lets you sleep in a little longer.
- 2) In your kitchen, a blinking light reminds you it's time to take your tablets.
- 3) If you forget, the medicine bottle cap goes online and emails your doctor to let her know.
- 4) On your way out of the house, you catch a glow in the corner of your eye.
- Your umbrella handle is lit up, which means that it has checked the BBC weather reports and predicts rain.
- 5) As you pass the bus stop on the way to the station, you notice the large LCD display flash that the number 23 is due.
- When the bus company first installed those displays, they ran on the expected timetable information only, but now that every bus has GPS tracking its location, they simply connect to the bus company's online service and always give the updated information.
- 6) An ornament with a dial notices the change and starts to turn so that the text on it points to the word "Travelling".
- Your family will also see later that you've arrived at "Work" safely.
- 7) The wrist band's large display also makes it easy to glance down and see how fast you are running and how many calories you've burned.
- All the data is automatically uploaded to your sports tracking site, which also integrates with your online supermarket shopping account to make it easy to compare with how many calories you've eaten.

THE "INTERNET" OF "THINGS":

- All the cases we saw used the *Internet* to send, receive, or communicate information.
- And in each case, the gadget that was connected to the Internet wasn't a computer, tablet, or mobile phone but an object, a *Thing*.
- **These Things are designed for a purpose:**
- the umbrella has a retractable canopy and a handle to hold it.
- A bus display has to be readable to public transport users, including the elderly and partially sighted and be able to survive poor weather conditions.
- The sports bracelet is easy to wear while running, has a display that is large enough and bright enough to read even when you are moving, and will survive heat, cold, sweat, and rain.
- Unlike a calm light in the umbrella stand, gives piece of information to process subconsciously when you pass it on the way out of your home, an app requires you to perform several actions. (you have to take the phone out of your pocket

or bag, unlock it, navigate to the right website , you have to type the URL and read the data from a small screen.)

- Rather than having greater capabilities, the smart umbrella simply moves the same intelligence into your environment so that you don’t have to change your routine.
- So the idea of the Internet of Things suggests that rather than having a small number of very powerful computing devices in your life (laptop, tablet, phone)
- you might have a large number of devices which are perhaps less powerful (umbrella, bracelet, mirror, fridge, shoes).
- The definition of ubicomp, however, would also include the air fresheners which release scent when they detect movement in the room as part of its domain.
- That is to say, such a device is an intelligently programmed computer processor, driven by sensors in the real world, and driving output in the real world, all embedded into an everyday object.
- These factors make this ubicomp, and it is only differentiated from the “Internet of Things” by the fact that these days most of the really interesting things done with computing also involve an Internet connection.
- But what does it mean to “connect an object to the Internet”?
- Clearly, sticking an Ethernet socket into a chair or a 3G modem into a sewing machine doesn’t suddenly inspire the object with mysterious properties.
- Rather, there has to be some flow of information which connects the defining characteristics of the Thing with the world of data and processing represented by the Internet.
- The Thing is present, physically in the real world, in your home, your work, your car, or worn around your body.
- This means that it can receive inputs from your world and transform those into data which is sent onto the Internet for collection and processing.
- So your chair might collect information about how often you sit on it and for how long.
- The presence of the Thing also means that it can produce outputs into your world with what we call “actuators”.
- Some of these outputs could be triggered by data that has been collected and processed on the Internet.
- So your chair might vibrate to tell you that you have received email.
- We could summarize these components in the following simple equation:
- Note that in all the cases we’ve looked at, the form of the object follows the function of the Thing:
- your chair is designed to sit on, the sewing machine to sew at, and so on.
- The fact of also being connected to the Internet and having general-purpose computing capabilities doesn’t necessarily have an impact on the form of the object at all.



An equation for the Internet of Things.

THE TECHNOLOGY OF THE INTERNET OF THINGS:

- It is worth taking a little time to look at the Internet of Things through a lens of the history of technology to more clearly understand how and where it fits.
- Technology's great drivers have initially been fundamental needs, such as food and water, warmth, safety, and health.
- Hunting and foraging, building and medicine grow out of these needs.
- Then, because resources for these things are not always distributed where and when one might like, technological advances progress with enabling and controlling the movement of people.
- Trade develops as a movement of goods from a place where they are plentiful and cheap to one where they are rare and valuable.
- **Storage** is a form of movement in time—for example, from harvest time, when food is plentiful and cheap, to the following winter, when it is highly valued.
- Information becomes key, too—hence, the development of **language to communicate** technology to others.
- Travellers might pass on messages as well as goods and services, and an oral tradition allows this information to pass through time as well as space.
- From writing, via the telegraph, radio, and television, to digital information, more and more technology has been about enabling the movement of information or doing interesting things with that information.
- As technology has progressed, new categories of objects have been created:
- in the electronic age, they have included telephones, radios, televisions, computers, and smartphones.
- As with most new technology, these devices tended to start out very expensive and gradually come down in price.
- Demand drives down prices, and research leads to optimization and miniaturisation.
- Ultimately, it becomes not just possible but also feasible to include functionality that would previously have required its own dedicated device *inside* another one.
- mere computing power isn't a sufficient precondition for the Internet of Things.

- Rather, we are looking at computing power linked on the one hand to electronic sensors and actuators which interact with the real world and on the other to the Internet.
- It turns out that the rapid sharing and processing of *information* with services or other consumers is a huge differentiator.
- Internet connectivity is also cheaper and more convenient than it used to be.
- Whereas in the past, we were tied to expensive and slow dial-up connections, nowadays we have broadband subscriptions, providing always-on connectivity to the Net.
- Wired Ethernet provides a fairly plug-and-play networking experience, but most home routers today also offer WiFi, which removes the need for running cables everywhere.
- For situations in which a fixed network connection isn't readily available, mobile phone connectivity is widespread.
- Another factor at play is the maturity of online platforms.
- Whereas early web apps were designed to be used only from a web browser, programming using an Application Programming Interface (API), which allows other programs, rather than just users, to interact with and use the services on offer.
- As the online services mature, so too do the tools used to build and scale them.
- Web services frameworks such as Python or Ruby on allow easy prototyping of the online component.
- Similarly, cloud services such as Amazon Web Services mean that such solutions can scale easily with use as they become more popular.

- **ENCHANTED OBJECTS:**
- Technologist, David Rose has talked about Enchanted Objects and has categorised various objects drawn from fairy tales and fantasy literature in ways that apply as much to technological objects.
- **For *Protection*,**
- Ex: In story:
- The **magical swords** and helmets protected the main characters of fairy tales from their enemies,
- In reality:
- the development of science and technology throughout history been driven by the need for **military** superiority, for the purpose of **security**.
- ***Health*** has been a driver for many quests to find an ingredient for a health potion and for research into various branches of medicine, pharmacology and surgery, physiotherapy, and diet.
- Ex: In story:
- Snow White's wicked stepmother asking "Mirror mirror on the wall, who's the fairest of them all?"
- In reality:
- to the friends settling an argument of fact by looking up articles from Wikipedia on their smartphones.
- ***Human Connection**,* even when one's loved ones are far away.

- the postal service, telephones, and social networking help keep us in touch with our family and friends.
- Ex: In story:
- for *Effortless Mobility* invented **flying carpets**, and even teleportation.
- In reality:
- Through technology, we have invented cars and railways, bicycles, and **aeroplanes.**
- The need for ***Creative Expression***
- Ex: in stories by the enchanted paintbrushes and magic flutes
- In reality:
- from charcoal to paint to computer graphics, or from drums to violins and electronic synthesisers.
- So, **technology** has always been associated with **magic,** and so this will be true almost by default for the Internet of Things.
- A **key element** of many enchanted objects is that above and beyond their practical enchantment they are given a name and a personality—implying an **intelligence greater than strictly necessary to carry out the task for which they are designed.**
- so our connected devices, or Things, have processing and communicating capabilities well beyond the needs of the average lamp or umbrella.
- **WHO IS MAKING THE INTERNET OF THINGS?**
- There are many crossover points between all the disciplines listed.
- Artists may collaborate with designers on installations or with traditional craftspeople on printmaking.
- Designers and engineers work closely to make industrial products, and hobbyist "hackers" (in the sense of tinkerers (unskilled person).
- In the Internet of Things:
- A hacker might tinker at the prototype for a Thing;
- A software developer might write the online component;
- A designer might turn the ugly prototype into a thing of beauty, possibly invoking the skills of a craftsperson
- And an engineer might be required to solve difficult technical challenges, especially in scaling up to production.

Chapter 2
DESIGN PRINCIPLES FOR CONNECTED DEVICES

- **CALM AND AMBIENT TECHNOLOGY:**
- ubicomp is often also referred to as *ambient computing*.
- the term "**ambient**" is **not** something to which we **actively pay attention** and in some cases as something which we seek to remove (e.g., ambient noise in a sound recording).
- the term ***calm** technology*—systems which **don't compete for attention** yet are ready to provide utility or useful information when we decide to give them some attention.

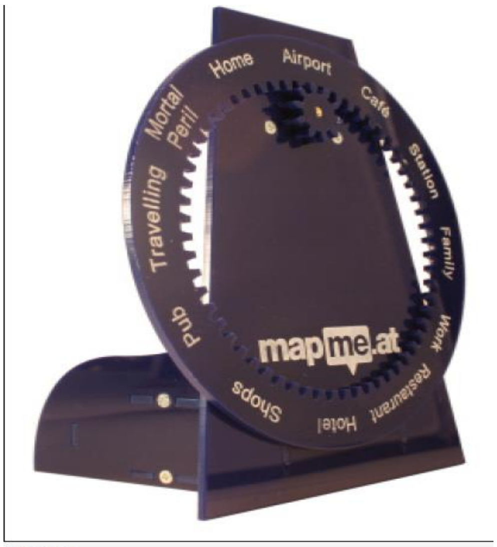
- Proliferation of computing devices into the world comes with all manner of new challenges.
- Issues include configuration, **how to provide power to all these items**, **how they talk to each other, and how they communicate with us.**
- The **networking challenges.**
- **Configuration and user interaction**, however, obviously **involve people** and **so** are **difficult** problems **to solve with just technical solutions.**
- This is where **good design can aid in adoption and usability.**
- Designing a connected device in isolation is likely to lead you to design decisions which aren't ideal when that object or service is placed into the real world.
- In addition to thinking of a device in the physical context one step larger— **"Always design a thing by considering it in its next larger context"** —
- a chair in a room, a room in a house, a house in an environment, an environment in a city plan"—we should do the same for the services.
- For **connected devices** which are just sensing their world, (or **acting as *inputs***), as long as their activity **doesn't require them to query the people** around them, there **shouldn't be any issues**.
- They will **collect information** and **deposit it into some repository online** for processing or analysis.
- When the **devices start interacting with people**, **things get more complicated.**
- Already we're seeing the **number of notifications, pop-ups, and indicator noises on our computers and mobile phones proliferate.**
- **When we scale up this number to include hundreds of new services and applications** and then spread that across the rest of the objects in our world, **it will become an attention-seeking cacophony (an unpleasant mixture of loud sounds).**
- **an antidote** to such a problem is to **design ubicomp** computing systems **to seek to blend into their surroundings;** in so doing, we could keep them in our peripheral perception until the right time to take centre stage:
- ***Calm technology engages both the center and the periphery of our attention, and in fact moves back and forth between the two.***
- A great example of this approach is ***Live Wire***, one of the first Internet of Things devices.
- Live Wire (also sometimes called **Dangling String**) is a simple device: **an electric motor connected to an eight-foot long piece of plastic string.**
- The **power** for the motor is **provided by the data transmissions on the Ethernet network** to which it is connected, so **it twitches whenever a packet of information is sent across the network**.
- Under normal, **light network load,** the string **twitches** (sudden jerk) **occasionally.**
- If the network is **overloaded**, the **string whirls madly**, **accompanied by a distinctive noise** from the motor's activity.
- Conversely, if **no network activity** is occurring, an unusual **stillness** comes over the string.

- Both extremes of activity therefore alert the nearby human
- The mention of the distinctive sound from the motor when the Live Wire is under heavy load brings up another interesting point.
- **Moving the means of conveying information away from screens and into the real world often adds a new dimension to the notification.**

MAGIC AS METAPHOR:

- In addition to the technology becoming capable of a particular action, **we** often **need *society***, to be **ready to accept it**.
- There are many examples when the **main difference between a failed technology and** a wildly **successful one is** that the **successful one arrived a few years later, when people were more receptive to what was offered.**
- Technology blogger Venkatesh Rao came up with a good **term** to help **explain how new technology becomes adopted.**
- He posits (suggest something as a basic fact) that **we don't see the present, the world that we live in now, as something that is changing.**
- **If we step back for a second, we do *know* that it has changed**.
- Rao called this concept **the *manufactured normalcy*** *(situation in which everythong is normal)* ***field***.
- **For a technology to be adopted, it has to make its way inside the manufactured normalcy field.**
- As a result, the **successful user-experience designer** is the one **who presents users with an experience which doesn't stretch the boundaries of their particular normalcy field too far, even if the underlying technology being employed is a huge leap ahead of the norm.**
- For example, the **mobile phone** was first introduced as a **phone that wasn't tethered to a particular location.**
- Now broadly the **same technology** is used to **provide a portable Internet terminal**, which can play movies, carry your entire music collection, and (every now and then) make phone calls.
- **The way that portable Internet terminals made it into our manufactured normalcy field was through the phone metaphor.**
- **Introducing technology to people in terms of something they already understand is a tried and tested effect:** computers started off as glorified typewriters; graphical user interfaces as desktops....
- Arthur C. Clarke has claimed that **"any sufficiently advanced technology is indistinguishable from magic,"** and given that the Internet of Things commonly bestows semi-hidden capabilities onto everyday objects, maybe the enchanted objects of magic and fairy tale are a good metaphor to help people grasp the possibilities.
- **Some Internet of Things projects draw their inspiration directly from magic.**
- For example, John McKerrell's **WhereDial** takes its lead from the **clock in *Harry Potter*** which tracked the location of the members of the Weasley family.

- The WhereDial, by comparison, has to rely on mere technology for its capabilities;
- however, **with the GPS chipsets in smartphones and location check-in services like FourSquare,** it isn't much of a leap to also own **an ornament which updates to show when you are at work, or travelling, or at a restaurant.**



The WhereDial.

- The **ambient orb** is a "single-pixel display" that can **show the status of a metric of its user's choosing—the price of a stock, the weather forecast etc.**
- Ambient Devices then took the idea one step further and built an **enchanted umbrella**.
- **It can read the weather forecast, and the handle glows gently if rain is expected,** alerting you to the fact that you may need to pick it up as you head out of the house.
- Everyday sort of magic that makes tasks a bit easier and lives a little more fun.
- **Using our understanding of magic and fairy tales to help make sense of these strange new gadgets.**
- **PRIVACY:**
- With more sensors and devices watching us and reporting data to the Internet, **the privacy of third parties who cross our sensors' paths is an important consideration.**
- Designers of an Internet of Things service will need to balance these concerns carefully.
- **KEEPING SECRETS:**
- An example from an early **instrumented car park** in a Westfield shopping mall in Australia.
- **Each parking bay is overlooked by a small sensor from Park Assist, which uses a cheap camera to tell whether the space is occupied.**
- The sensors are all networked and presumably can **provide analytics to the owner of the car park as to its usage.**
- A light on the sensor can help guide drivers to a free space.
- **The shopping mall provided a smartphone app for visitors to download so that they could find out more information about the facilities.**

- **One of the features of the app was a Find My Car option.**
- Choosing that, **you were prompted to enter the first few characters of your licence plate, and the app would then return four small photos of potential matches—**from optical character recognition software processing the sensor data on the mall's server.
- **security professional** Troy Hunt **was able to watch what information the app was requesting from the server and found that it was a simple unencrypted web request.**
- The initial request **URL had a number of parameters, including the search string**, but also including information such as the number of results to return.
- That **request returned** a chunk of data, which included the URLs for the four images to download, but **also included some additional pieces of information.**
- It was **easier for** the developer of the **web service to just return all the available data than to restrict it to just what was needed** in this case.
- The **extra data included**, for example, **the IP addresses of each of the sensor units**, but more importantly, **it also included the full licence plate for each vehicle and the length of time it had been parked in the space.**
- **By altering the search parameters, Troy found that he could request many more than the four matches, and it was also possible to omit the licence plate search string.**
- That meant he **could download a full list of licence plates from all 2550 parking spaces in a single web request, whenever he liked.**
- Once alerted to the problem, Westfield and Park Assist were quick to disable the feature and then work with Troy to build a better solution.
-
- **Important points:**
- ***Don't share more than you need to provide the service.***
- **"The best way to keep a secret is to never have it".**
- **If you can avoid gathering and/or storing the data in the first place, you need not worry about disclosing it accidentally.**
- In this day and age, **it is standard practice to never store passwords as cleartext**.
- You could also **consider applying** the standard mechanisms for **password encryption**, such as the **one-way hash**, to other pieces of data.
- One-way hashing is a cryptographic technique used to **condense an arbitrarily sized chunk of data into a fixed-sized piece, called the hash.**
- It's called one-way hashing because **there isn't an easy way, given the resultant hash, to work out what the original data was.**
- Hashing algorithms are so designed such that even a small difference in the input data leads to a huge difference in the output hash.

WHOSE DATA IS IT ANYWAY?

- With the number of sensors being deployed, it isn't always clear whose data is being gathered.

- Consider the case of **a camera deployed in an advertising hoarding which can check to see whether people are looking at the different adverts.**
- Does the data belong to the company that installed the camera or to the members of the public who are looking at the adverts?
- Adam Greenfield, a leading practitioner of urban computing, makes a convincing argument that **in a public space this data is being generated by the public, so they should at least have equal rights to be aware of, and also have access to, that data.**
- **On private property**, you can more easily claim that the members of the **public don't have such a right**, but perhaps **the property owner might assert rights** to the data rather than whoever installed the camera.

WEB THINKING FOR CONNECTED DEVICES:

- When you are thinking of the networked aspect of Internet of Things objects, it might help to draw on **experiences and design guidelines from existing network deployments.**
- You should aim to get into the mindset of the web and **create devices which are *of* the web rather than those which just exist *on* the web.**
- **SMALL PIECES, LOOSELY JOINED:**
- Even if you are building all the components of your service, **it makes sense not to couple them too tightly together.**
- The **Internet** flourished not because it is neatly controlled from a central location, but because it isn't; it **is a collection of services and machines following the maxim of *small pieces, loosely joined*.**
- **each piece should be designed to do one thing well and not rely too much on tight integration with the separate components it uses.**
- **make the components more generalised** and able to serve other systems which require a similar function.
- That will **help you**, and others, **to reuse and repurpose the components to build new capabilities**
- **Where possible, use existing standards and protocols rather than inventing your own.**

FIRST-CLASS CITIZENS ON THE INTERNET:

- What do we mean by that?
- Where possible, you should use the same protocols and conventions that the rest of the Internet uses.
- a good rule of thumb for the past 20 years or more has been to **expect the IP protocol to penetrate everywhere.**
- We see no reason for it not to continue into the Internet of Things.
- **In the few cases where the existing protocols don't work,** such as in extremely low-powered sensors, a better solution is to **create new open standards which address the issue.**
- **When mobile phones were first being connected to the Internet**, it was deemed too difficult for them to talk to web servers directly, and a whole suite of **new protocols, Wireless Application Protocol (WAP), were developed.**

GRACEFUL DEGRADATION:

- The **endpoints have** a massively **disparate and diverse range of capabilities.**
- As a result, **building services which can be used by all of them is a nearly impossible task**.
- However, a number of **design patterns have evolved to mitigate the problem:**
- **1)** If you need to come up with a format for some data being transferred between devices, **include a way to differentiate between successive versions of the formats**—ideally in such **a way that older devices can still mostly read newer formats.**
- This is known as ***backwards compatibility***.
- The **HTML format** does this by stating that **any client should ignore any tags** (the text inside the <>) **that it doesn't understand**, so newer versions can add new tags without breaking older parsers.
- The **HTTP protocol** uses a slightly different technique in which **each end specifies the version of the protocol that it supports, and the other end takes care not to use any of the newer features for that particular session**.
- The other common technique is to use something called ***graceful degradation*****.**
- This technique involves aiming to **provide a fully featured experience if the client is capable of it but then falling back—potentially in a number of levels—to a less feature-rich experience on less capable clients.**
- Such as in Gmail, the **coder wants to use advanced JavaScript features in modern browsers.**
- Well-written **apps check that the features are available before using them, but if those features aren't available, the apps might limit themselves to a version using simpler** (and more common) **JavaScript code.**
- And **if JavaScript isn't available at all, they fall back to basic HTML forms.**
- This experience is not as nice as the full one but better than no experience at all!

AFFORDANCES:

- Donald Norman defines *affordances* as follows:
- *Affordances provide strong clues to the operations of things.*
- *Knobs are for turning.*
- *Balls are for throwing or bouncing.*
- *When affordances are taken advantage of,* ***the user knows what to do just by looking:***
- ***no picture, label, or instruction is required.***
- *Complex things may require explanation, but simple things should not.*

- ***When simple things need pictures, labels, or instructions, the design has failed.***
- What are the affordances of digitally enhanced objects?
- How do we convey to the user of an object that it can communicate with the cloud?
- An important start is to keep the existing affordances of the object being enhanced.
- **Users who don't realise that a device has any extra capabilities should still be able to use it as if it hasn't**.
- Similar rules apply when designing physical interfaces.
- Don't overload familiar connectors with unfamiliar behaviours.

Chapter 3
INTERNET PRINCIPLES

- **INTERNET COMMUNICATIONS:AN OVERVIEW**
- **IP (Internet Protocol)**
- **Data is sent** from one machine to another **in a packet**, with a destination address and a source address **in a standardised format (a "protocol").**
- Most of the time, the packets of data **have to go through a number of intermediary machines, called *routers***, to reach their destination.
- The underlying networks aren't always the same.
- a postcard was placed in an envelope before getting passed onwards.
- This happens with Internet packets, too.
- So, **an *IP packet*** is a block of data along with the same kind of information you would write on a physical envelope: **the name and address of the server**, and so on.
- There is **no guarantee**, and you can send only what will **fit in a single packet**.

TCP

- What if you wanted **to send longer messages** than fit on a postcard?
- Or wanted to **make sure your messages got through**?
- TCP is **built on top of the basic IP protocol** and adds **sequence numbers, acknowledgements, and retransmissions.**
- This means that a message sent with TCP can be arbitrarily long and give the sender some assurance that it actually arrived at the destination intact.

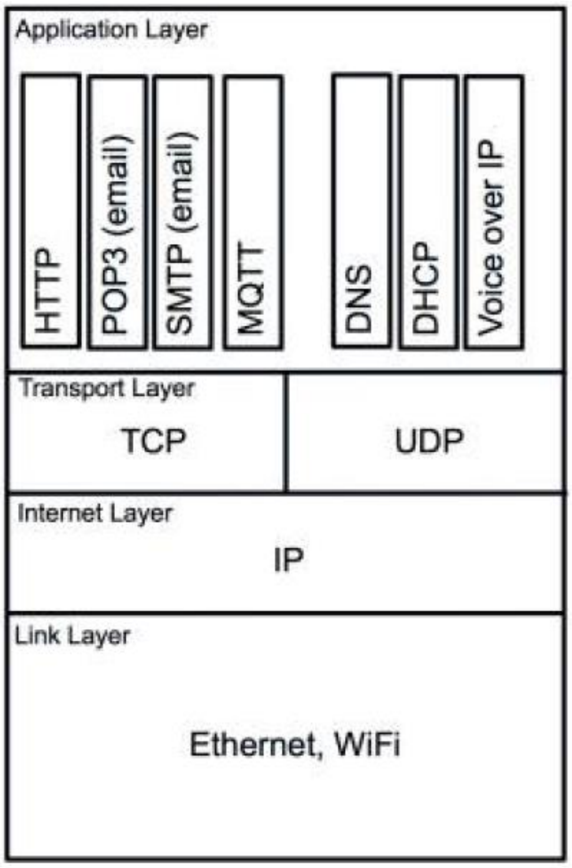
THE IP PROTOCOL SUITE (TCP/IP)

- whole suite or stack of protocols layered on top of each other, each layer building on the capabilities of the one below.
- ▪ The low-level protocols at the ***link layer*** manage the **transfer of bits of information** across a network link.
- The ***Internet layer uses IP address****.*
- Then TCP, which lives in the ***transport layer,*** sits on top of IP and extends it with more sophisticated **control of the messages passed.**

- Finally, the ***application layer*** contains the protocols that deal with fetching web pages, **sending emails,** and Internet telephony.

UDP

- It is protocol in the transport layer.
- **In UDP each message may or may not arrive.**
- **No handshake or retransmission occurs, nor is there any delay to wait for messages in sequence.**
- These limitations **make TCP preferable for** many of the tasks that **Internet of Things devices** will be used for.
- The lack of overhead, however, makes UDP useful for applications such as streaming data, which can cope with minor errors but doesn't like delays.
- Voice over IP (VoIP)—computer-based telephony, such as Skype—is an example of this.



The Internet Protocol suite.

- **IP ADDRESSES:**
- In the world of **low-level computer networking**, however, **numbers are much easier to deal with.** So, **IP addresses are numbers**.
- In Internet Protocol version 4 (**IPv4**), 2^32 addresses are possible.
- **usually written as four 8-bit numbers separated by dots** (from 0.0.0.0 to 255.255.255.255).
- This "dotted quad" is still exactly equivalent to the 32-bit number.
- Every machine on the Internet has at least one IP address.
- **Your home or office network might have only one publicly visible IP address.**

- **DNS**
- Although computers can easily handle **32-bit numbers**, even formatted as dotted quads they are **easy for most humans to forget**.
- The Domain Name System (DNS) helps our brains navigate the Internet.
- Domain names such as the following:
- google.com
- bbc.co.uk
- **Each domain name has a top-level domain (TLD),** like .com or .uk, which further subdivides into .co.uk and .gov.uk, and so on.
- **This top-level domain knows where to find more information about the domains within it**; for example, .com knows where to find google.com and wiley.com.
- **The domains then have information about where to direct calls to individual machines or services.**
- For example, the DNS records for .google.com know where to point you for the following:
- www.google.com
- mail.google.com
- calendar.google.com
- **But DNS can also point to other services on the Internet**—for example:
- pop3.google.com — For receiving email from Gmail
- smtp.google.com — For sending email to Gmail
- **STATIC IP ADDRESS ASSIGNMENT**
- How do you get assigned an IP address?
- If you have bought a server-hosting package from **an Internet service provider (ISP)**, you might typically be **given a single IP address**.
- But **the company itself has been given a block of addresses to assign.**
- Historically, **these were ranges of different sizes, typically separated into "classes" of 8 bits, 16 bits, or 24 bits:**
- Class A — From 0.x.x.x
- Class B — From 128.0.x.x
- Class C — From 192.0.0.x
- The class C ranges had a mere 8 bits (256 addresses) assigned to them, while the class A ranges had many more addresses and would therefore be given only to the very largest of Internet organisations.
- With the explosion of the number of devices connecting to the Internet we can use **Classless Inter-Domain Routing (CIDR)**, which allows you to specify exactly how many bits of the address are fixed.
- the **class A** addresses we mentioned above would be equivalent to 0.0.0.0**/8**, while a class C might be 208.215.179.0**/24**.
- In many cases, the **system administrator** simply **assigns server numbers in order**.
- He makes a note of the addresses and updates DNS records and so on to point to these addresses.
- We call this kind of address ***static*** **because once assigned it won't change again without human intervention.**

- **DYNAMIC IP ADDRESS ASSIGNMENT**
- Thankfully, **we don’t typically have to choose an IP address for every device we connect to a network.**
- Instead, when you connect a laptop, a printer, **it can request an IP address from the network itself using the Dynamic Host Configuration Protocol (DHCP).**
- When the device tries to connect, instead of checking its internal configuration for its address, **it sends a message to the router asking for an address.**
- **The router assigns it an address.**
- This is not a static IP address which belongs to the device indefinitely; rather, **it is a temporary “lease” which is selected *dynamically* according to which addresses are currently available.**
- If the router is rebooted, **the lease expires**, or the device is switched off, **some other device may end up with that IP address.**
- Using a **static address** may be fine for development (if you are **the only person connected to it with that address**), but **for working in groups** or **preparing a device to be distributed** to other people on arbitrary networks, you almost certainly want **a dynamic IP address.**

IPv6

- If your mobile phone, watch, MP3 player, telehealth or sports-monitoring **devices are all connected to the Internet, then you personally are carrying half a dozen IP addresses already.**
- At home you would start with all your electronic devices being connected.
- But beyond that, you might also have **sensors at every door and window for security.**
- More sensitive sound sensors to detect the presence of mice or beetles.
- Other **sensors to check temperature, moisture, and airflow levels for efficiency.**
- It is hard to predict what **order of number of Internet connected devices** a household **might have in the near future. Tens? Hundreds? Thousands?**
- **Enter IPv6, which uses 128-bit addresses,** usually displayed to users as **eight groups of four hexadecimal digits**—for example, 001:0db8:85a3:0042 :0000:8a2e:0370:7334.
- The address space (2^128).
- You can find many ways to work around the lack of public IP addresses using subnets.
- **IPv6 and Powering Devices**
- We know that **we can regularly charge and maintain a small handful of devices**.
- The **requirements for large numbers of devices, however, are very different**.
- **The devices should be low power and very reliable, while still being capable of connecting to the Internet.**
- Perhaps to accomplish this, these **devices will team together in a mesh network.**

- **MAC ADDRESSES**
- Every network-connected device also has a MAC address, which **is like the final address on a physical envelope in our analogy.**
- It is **used to differentiate different machines on the same physical network** so that they can exchange packets.
- This **relates to the lowest-level "link layer"** of the TCP/IP stack.
- Though MAC addresses are globally unique, they don't typically get used outside of one Ethernet network (for example, beyond your home router).
- So, when an IP message is routed, it hops from node to node, and when it finally reaches a node which knows where the *physical* machine is, that **node passes the message to the device associated with that MAC address.**
- **MAC stands for *Media Access Control*.**
- It is a **48-bit number**, usually **written as six groups of hexadecimal digits, separated by colons**—for example:
- 01:23:45:67:89:ab
- **Most devices**, such as your laptop, come with the MAC address **burned into their Ethernet chips**.
- **Some chips**, such as the Arduino Ethernet's WizNet, **don't have a hard-coded MAC address.**
- This is **for production reasons: if the chips are mass produced, they are, of course, *identical***.
- So they can't, physically, contain a distinctive address.
- **The address could be stored in the chip's firmware,** but this would then require **every chip to be built with custom code compiled in the firmware.**
- **Alternatively, one could provide a simple data chip which stores just the MAC address** and have the WizNet chip read that.
- (*WizNet is a Korean manufacturer which specialises in networking chips for embedded devices.*)

TCP AND UDP PORTS

- when you send a **TCP/IP message** over the Internet, **you have to send it to the right port**.
- TCP ports are **referred to by numbers (from 0 to 65535).**
- **AN EXAMPLE: HTTP PORTS:**
- If your **browser requests an HTTP page**, it usually sends that request **to port 80**.
- The **web server is "listening" to that port** and therefore replies to it.
- **If you send** an HTTP message **to a different port**, one of several things will happen:
- ▪ **Nothing is listening to that port**, and the **machine replies with an "RST" packet** (a control sequence resetting the TCP/IP connection) to complain about this.
- ▪ Nothing is listening to that port, but the **firewall lets the request simply hang instead of replying.**

- The client has decided that trying to send a message to that port is a bad idea and refuses to do it. (list of "restricted ports".)
- The message arrives at a port that is expecting something other than an HTTP message.
- The **server** reads the client's response, **decides that it is garbage, and then terminates the connection.**
- **Ports 0–1023 are "well-known ports"**, and only a system process or an administrator can connect to them.
- **Ports 1024–49151 are "registered"**, so that common applications can have a usual port number.
- You see custom port numbers **if a machine has more than one web server**; for example, in development **you might have another server, bound to port 8080:**
- http://www.example.com:8080
- The **secure (encrypted) HTTPS** usually **runs on port 443.** So these two URLs are equivalent:
- https://www.example.com
- https://www.example.com:443
- OTHER COMMON PORTS
- 22 SSH (Secure Shell)
- 23 Telnet
- 25 SMTP (outbound email)
- 110 POP3 (inbound email)
- 220 IMAP (inbound email)

APPLICATION LAYER PROTOCOLS

- This is the **layer you are most likely to interact with** while prototyping an Internet of Things project.
- A ***protocol* is a set of rules for communication between computers.**
- It includes rules about **how to initiate the conversation** and **what format the messages should be in.**
- It determines **what inputs are understood** and **what output is transmitted.**
- It also specifies **how the messages are sent and authenticated** and **how to handle errors caused by transmission**.
- **HTTP**
- The **client requests a resource by sending a command to a URL, with some headers**.
- Ex: try to get a simple document at http://book.roomofthings.com/hello.txt.
- The basic structure of the request would look like this:
- GET /hello.txt HTTP/1.1
- Host: book.roomofthings.com
- We specified the **GET method** because we're **simply getting the page.**
- We then tell the server **which resource we want** (/hello.txt) and **what version of the protocol we're using**.
- we **write the headers**, which **give additional information** about the request.

- The **Host header is the only required header in HTTP 1.1.**
- It is used to let a web server that serves multiple virtual hosts **point the request to the right place**.
- Accept: text/html,application/xhtml+xml, application/ xml;
- Accept-Charset: UTF-8
- Accept-Encoding: gzip
- Accept-Language :en-US
- The **Accept- headers** tell the server what kind of content the client is willing to receive and are part of **"Content negotiation"**.
- Finally, the server sends back its response.
- The server replies, giving us a 200 status code (which it summarizes as "OK"; that is, the request was successful).

HTTPS: ENCRYPTED HTTP

- **If someone eavesdropped your connection** (easy to do with tools such as Wireshark if you have access to the network at either end), **that person can easily read the conversation**.
- The HTTPS protocol is actually just **a mix-up of plain old HTTP over the Secure Socket Layer (SSL) protocol.**
- An HTTPS server **listens to a different port (usually 443)** and on connection sets up a secure, encrypted connection with the client.
- When that's established, both sides just speak HTTP to each other as before!
- *Diffie–Hellman (D-H) key exchange is a way for two people to exchange cryptographic keys in public.*
- *without an eavesdropper being able to decode their subsequent conversation.*
- *This is done by each side performing mathematical calculations which are simple to do but not to undo.*
- *Neither side ever sends their own secret key unencrypted, but only the result of multiplying it with a shared piece of information.*

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:
- ▪ Arduino code is open source.
- ▪ 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.