THE INTERNET OF THINGS
Chapter 1
THE INTERNET OF THINGS: AN OVERVIEW

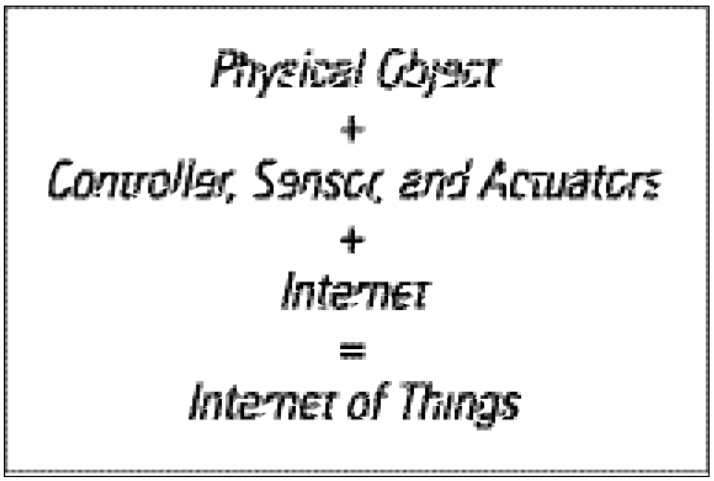
- **THE FLAVOUR OF THE INTERNET OF THINGS:**
- (Examples of IOT)
- 1) The alarm rings. As you open your eyes , you see that it's five minutes later than your usual wake-up time.
- The clock has checked the train times online, and your train must be delayed, so it lets you sleep in a little longer.
- 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.

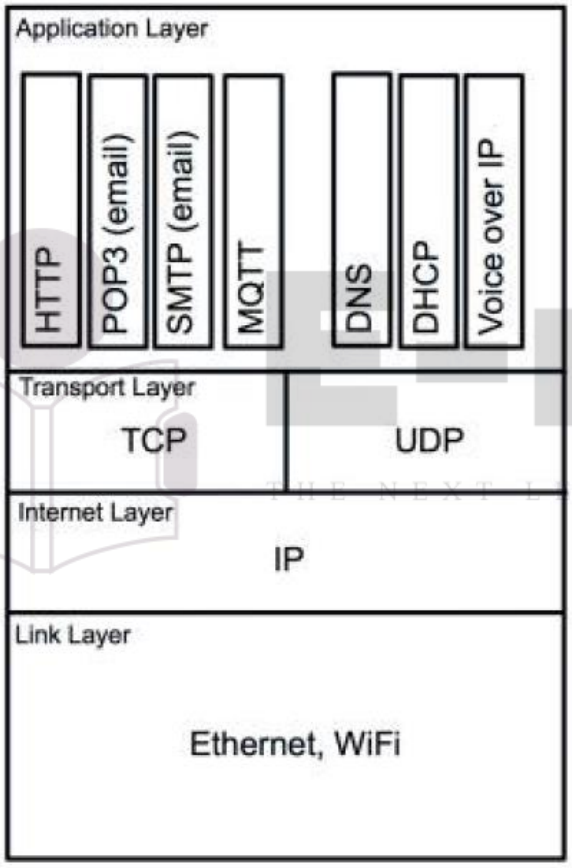
THE IP PROTOCOL SUITE (TCP/IP)

- 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.

Chapter 4
THINKING ABOUT PROTOTYPING

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

SKETCHING

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

FAMILIARITY

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

COSTS VERSUS EASE OF PROTOTYPING

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

writing and reading values to and from the GPIO pins.

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

Case Study: Bubblino

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

Prototypes And Production

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

Changing Embedded Platform

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

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

Physical Prototypes And Mass Personalisation

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

Climbing Into The Cloud

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

Open Source Versus Closed Source

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

Why Closed?

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

Why Open?

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

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

Being a Good Citizen:

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

Open Source as a Competitive Advantage

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

Open Source as a Strategic Weapon:

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

Mixing Open And Closed Source

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

Closed Source For Mass Market Projects

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

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

Tapping Into The Community

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

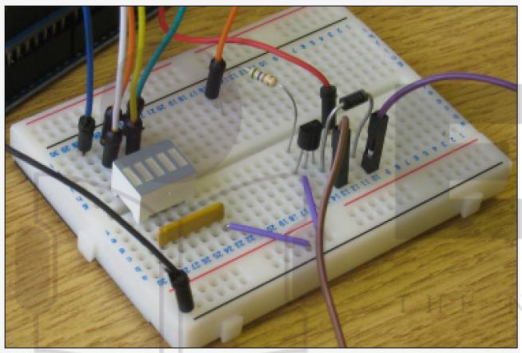
IOT
UNIT 2
Chapter 5
PROTOTYPING EMBEDDED DEVICES

- **ELECTRONICS:**
- Most of the prototyping can be done on what are called *solderless breadboards.*
- They enable you to build components together into a circuit with just a push-fit connection.
- When it comes to thinking about the electronics, it's useful to split them into two main categories:
- ▪ **Sensors:** Sensors are the ways of getting information *into* your device, finding out things about your surroundings.
- ▪ **Actuators:** Actuators are the *outputs* for the device—the motors, lights, and so on, which let your device do something to the outside world.
- Within both categories, the electronic components can talk to the computer in a number of ways.
- The simplest is through digital I/O, which has only two states: a button can either be pressed or not; or an LED can be on or off.
- These states are usually connected via general-purpose input/output (GPIO) pins and map a digital 0 in the processor to 0 volts in the circuit and the digital 1 to a set voltage.
- Usually the voltage that the processor is using to run (commonly 5V or 3.3V)
- If you want a more nuanced connection than just on/off, you need an analogue signal.
- If you want to run a motor at a speed other than off or full-speed, you need to feed it with a voltage somewhere between 0V and its maximum rating.
- Because computers are purely digital devices, you need a way to translate between the analogue voltages in the real world and the digital of the computer.
- An analogue-to-digital converter (ADC) lets you measure varying voltages.
- Microcontrollers often have a number of these converters built in.
- They will convert the voltage level between 0V and a predefined maximum into a number, depending on the accuracy of the ADC.
- The flipside of an ADC is a DAC, or digital-to-analogue converter.
- DACs let you generate varying voltages from a digital value.
- **SENSORS**
- Pushbuttons and switches, which are probably the simplest sensors, allow some user input.
- Sensing the environment is another easy option.
- Light-dependent resistors (LDRs) allow measurement of ambient light levels, temperature sensors allow you to know how warm it is, and sensors to measure humidity or moisture levels are easy to build.
- Microphones obviously let you monitor sounds and audio.
- Distance-sensing modules, which work by bouncing either an infrared or ultrasonic signal off objects, are readily available.
- **ACTUATORS**
- One of the simplest and yet most useful actuators is light, because it is easy to create electronically and gives an obvious output.
- Light-emitting diodes (LEDs) typically come in red and green but also white and other colours.

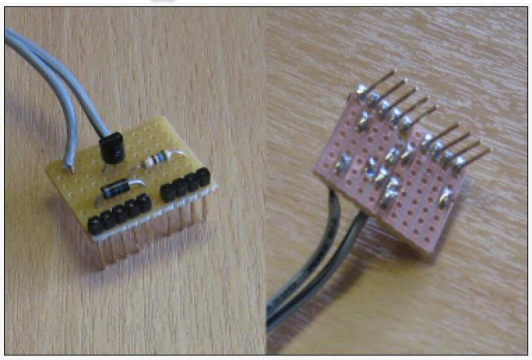
- More complicated visual outputs also are available, such as LCD screens to display text or even simple graphics.
- You can wire up outputs to speakers to create more complicated synthesised sounds.
- More complicated again are motors.
- Stepper motors can be moved in *steps*, as the name implies.
- Usually, a fixed number of steps perform a full rotation.
- DC motors simply move at a given speed when told to.
- Both types of motor can be one-directional or move in both directions.

SCALING UP THE ELECTRONICS

- From the perspective of the electronics, the starting point for prototyping is usually a "breadboard".
- it's common to solder the components onto some protoboard, which may be sufficient to make the circuit more permanent.
- Moving beyond the protoboard (stripboard) option tends to involve learning how to layout a PCB.
- It involves learning how to use a new piece of software and understanding some new terminology.



The breadboard.



The stripboard.

The PCB.

EMBEDDED COMPUTING BASICS

- **MICROCONTROLLERS:**
- These systems combine the processor, RAM, and storage onto a single chip, which means they are much more specialised, smaller than their PC equivalents, and also easier to build into a custom design.
- Microcontrollers are very limited in their capabilities.
- Usually, they offer RAM capabilities measured in kilobytes and storage in the tens of kilobytes.
- The modern chips are much smaller, require less power, and run about five times faster than their 1980s counterparts.
- The microcontroller market consists of many manufacturers each with a range of chips for different applications.
- (Atmel, Microchip, NXP, Texas Instruments, to name a few).
- The devices using them are focused on performing one task.
- **SYSTEM-ON-CHIPS:**
- In between the low-end microcontroller and a full-blown PC sits the SoC.
- Like the microcontroller, these SoCs combine a processor and a number of peripherals onto a single chip but usually have more capabilities.
- The processors usually range from a few hundred megahertz to the gigahertz.
- RAM measured in megabytes rather than kilobytes.
- Storage for SoC modules tends not to be included on the chip, with SD cards being a popular solution.
- The greater capabilities of SoC mean that they need some sort of operating system to marshal their resources.
- A wide selection of embedded operating systems, both closed and open source, is available and from both specialised embedded providers and the big OS players, such as Microsoft and Linux.

CHOOSING YOUR PLATFORM:

- The platform you choose depends on the particular blend of price, performance, and capabilities that suit what you're trying to achieve.
- And just because you settle on one solution, that doesn't mean somebody else wouldn't have chosen a completely different set of options to solve the same problem.
- Start by choosing a platform to prototype in.
- some of the factors that you need to weigh when deciding how to build your device:
-
- **1) Processor Speed**
- The processor speed, or clock speed, of your processor tells you how fast it can process the individual instructions in the machine code for the program it's running.
- Naturally, a faster processor speed means that it can execute instructions more quickly.
- You might also make a comparison based on millions of instructions per second (MIPS).
- Some processors may lack hardware support for floating-point calculations, so if the code involves a lot of complicated mathematics, slower processor with hardware floating-point support could be faster than a slightly higher performance processor without it.
- Microcontrollers tend to be clocked at speeds in the tens of MHz, whereas SoCs run at hundreds of MHz or possibly low GHz.

- If your project doesn’t require heavyweight processing then some sort of microcontroller will be fast enough.
- If your device will be crunching lots of data(ex: processing video in real time) then you’ll be looking at a SoC platform.
-
- **2) RAM**
- RAM provides the working memory for the system.
- If you have more RAM, you may be able to do more things or have more flexibility over your choice of coding algorithm.
- It is difficult to give exact guidelines to the amount of RAM you will need, as it will vary from project to project.
- However, microcontrollers with less than 1KB of RAM are unlikely to be of interest, and if you want to run standard encryption protocols, you will need at least 4KB, and preferably more.
- For SoC boards, particularly if you plan to run Linux as the operating system, we recommend at least 256MB.
-
- **3) Networking**
- How your device connects to the rest of the world is a key consideration for Internet of Things products.
- Wired Ethernet is often the simplest for the user—generally plug and play—and cheapest, but it requires a physical cable.
- Wireless solutions obviously avoid that requirement but introduce a more complicated configuration.
- WiFi is the most widely deployed to provide an existing infrastructure for connections, but it can be more expensive and less optimized for power consumption than some of its competitors.
- Other short-range wireless can offer better power-consumption profiles or costs than WiFi but usually with the trade-off of lower bandwidth
- There is, of course, the existing Bluetooth standard as another possible choice.
- For remote or outdoor deployment the mobile phone networks can be used.
-
- **4)USB**
- If your device can rely on a more powerful computer being nearby, tethering to it via USB can be an easy way to provide both power and networking.
- You can buy some of the microcontrollers in versions which include support for USB.
- Instead of the microcontroller presenting itself as a device, some can also act as the USB “host”.
-
- **5)Power Consumption**
- Faster processors are often more power hungry than slower ones.
- For devices which might be portable or rely on an unconventional power supply (batteries, solar power) depending on where they are installed, power consumption may be an issue.
- However, processors may have a minimal power-consumption sleep mode.
- This mode may allow you to use a faster processor to quickly perform operations and then return to low-power sleep.
-

- **6)Interfacing with Sensors and Other Circuitry**
- In addition to talking to the Internet, your device needs to interact with something else—either
- sensors to gather data about its environment;
- Or motors, LEDs, screens, and so on, to provide output.
- You could connect to the circuitry through some sort of peripheral bus—SPI and I2C being common ones—
- or through ADC or DAC modules to read or write varying voltages;
- or through generic GPIO pins, which provide digital on/off inputs or outputs.
-
- **7) Physical Size and Form Factor**
- The continual improvement in manufacturing techniques for silicon chips means that we've long passed the point where the limiting factor in the size of a chip is the amount of space required for all the transistors and other components that make up the circuitry on the silicon.
- Nowadays, the size is governed by the number of connections it needs to make to the surrounding components on the PCB.
- The limit to the size that each connection can be reduced to is then governed by the capabilities and tolerances of your manufacturing process.
- Some surface-mount designs are big enough for home-etched PCBs and can be hand-soldered.
- Others require professionally produced PCBs and accurate pick-and-place machines to locate them correctly.
- Due to these trade-offs in size versus manufacturing complexity, many chip designs are available in a number of different form factors, known as *packages*.
- This lets the circuit designer choose the form that best suits his particular application.

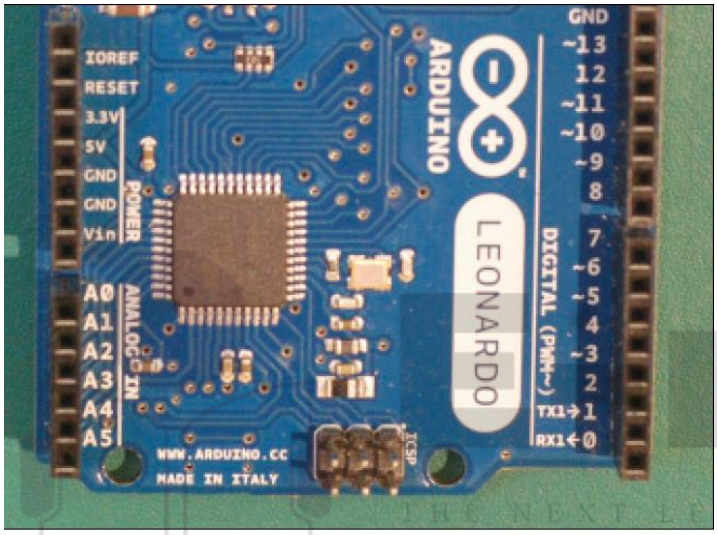
ARDUINO

- The Arduino team's focus on simplicity rather than raw performance for the code has made the Arduino the board of choice in almost every beginner's physical computing project.
- The "standard" Arduino board has gone through a number of iterations:
- Arduino NG, Diecimila, Duemilanove, and Uno.
- The Uno features an ATmega328 microcontroller and a USB socket for connection to a computer.
- It has 32KB of storage and 2KB of RAM
- The Uno also provides 14 GPIO pins.
- Arduino Mega 2560 provides 256KB of Flash storage, 8KB of RAM, three more serial ports, a massive 54 GPIO pins.
- the more recent Arduino Due has a 32-bit ARM core microcontroller.
- Its specs are similar to the Mega's, although it ups the RAM to 96KB.
-
- **DEVELOPING ON THE ARDUINO**
- Using a single USB cable, you can not only power the board but also push your code onto it, and (if needed) communicate with it
- For example, for debugging or to use the computer to store data retrieved by the sensors connected to the Arduino.
-

- **Integrated Development Environment**
- You usually develop against the Arduino using the integrated development environment (IDE) that the team supply at http://arduino.cc.
- Most Arduino projects consist of a single file of code, so you can think of the IDE mostly as a simple file editor.
- The controls that you use the most are those to check the code (by compiling it) or to push code to the board.
-
- **Pushing Code**
- Connecting to the board should be relatively straightforward via a USB cable.
- Sometimes you might have issues with the drivers or with permissions on the USB port.
- You need to choose the correct serial port and the board type (look carefully at the labelling on your board and its CPU to determine which option to select).
- When your setup is correct, the process of pushing code is generally simple:
- first, the code is checked and compiled, with any compilation errors reported to you.
- If the code compiles successfully, it gets transferred to the Arduino and stored in its flash memory.
- At this point, the Arduino reboots and starts running the new code.
-
- **Operating System**
- The Arduino doesn't, by default, run an OS as such, only the bootloader, which simplifies the code-pushing process described previously.
- When you switch on the board, it simply runs the code that you have compiled until the board is switched off again (or the code crashes).
- It is, however, possible to upload an OS to the Arduino, usually a lightweight real-time operating system (RTOS) such as FreeRTOS/DuinOS.
- The main advantage of one of these operating systems is their built-in support for multitasking.
- It is even possible to compile code without using the IDE but by using the toolset for the Arduino's chip—for example the avr-gcc toolset.
- The avr-gcc toolset (www.nongnu.org/avr-libc/) is the collection of programs that let you compile code to run on the AVR chips used by the rest of the Arduino boards and flash the resultant executable to the chip.
- It is used by the Arduino IDE behind the scenes but can be used directly, as well.
-
- **Language**
- The language usually used for Arduino is a slightly modified version of C++.
- It includes some libraries used to read and write data from the I/O pins provided on the Arduino and to do some basic handling for "interrupts".
- The code needs to provide only two routines:
- ▪ setup():
- This routine is run once when the board first boots.
- You could use it to set the modes of I/O pins to input or output or to prepare a data structure which will be used throughout the program.
- ▪ loop():
- This routine is run repeatedly in a tight loop while the Arduino is switched on.
- Typically, you might check some input, do some calculation on it, and perhaps do some output in response.

- Ex: blinking a single LED:
- // Pin 13 has an LED connected on most Arduino boards. give it a name:
- int led = 13;
- // the setup routine runs once when you press reset:
- void setup()
- {
- // initialize the digital pin as an output.
- pinMode(led, OUTPUT);
- }
- // the loop routine runs over and over again forever:
- void loop()
- {
- digitalWrite(led, HIGH); // turn the LED on
- delay(1000); // wait for a second
- digitalWrite(led, LOW); // turn the LED off
- delay(1000); // wait for a second
- }
-
- **Debugging**
- Because C++ is a compiled language, a fair number of errors, such as bad syntax or failure to declare variables, are caught at compilation time.
- Because the Arduino isn't generally connected to a screen, it is hard for it to tell you when something goes wrong.
- Even if the code compiled successfully, certain errors still happen.
- An error could be raised that can't be handled, such as a division by zero, or trying to access the tenth element of a 9-element list.
- If Bubblino stops blowing bubbles, how can we distinguish between the following cases?
- ▪ Nobody has mentioned us on Twitter.
- ▪ The Twitter search API has stopped working.
- ▪ Bubblino can't connect to the Internet.
- ▪ Bubblino has crashed due to a programming error.
- ▪ Bubblino is working, but the motor of the bubble machine has failed.
- ▪ Bubblino is powered off.
- The first commercially available version of the WhereDial has a bank of half a dozen LEDs specifically for consumer-level debugging.
- In the case of an error, the pattern of lights showing may help customers fix their problem.
- Runtime programming errors may be tricky to trap because although the C++ language has exception handling, the avr-gcc compiler doesn't support it.
- So the Arduino platform doesn't let you use the usual try... catch... logic.
- Effectively, this means that you need to check your data before using it: if a number might conceivably be zero, check that before trying to divide by it.
- Test that your indexes are within bounds.
- In the absence of a screen, the Arduino allows you to write Information over the USB cable using Serial.write().
- The Arduino IDE provides a serial monitor which echoes the data that the Arduino has sent over the USB cable.

- This could include any textual information, such as logging information, comments, and details about the data that the Arduino is receiving and processing (to double-check that your calculations are doing the right thing).
-
- **SOME NOTES ON THE HARDWARE**
- The Arduino exposes a number of GPIO pins and is usually supplied with "headers" (plastic strips that sit on the pin holes, that provide a convenient solderless connection for wires, especially with a "jumper" connection).
- The headers are optimised for prototyping and for being able to change the purpose of the Arduino easily.
- Each pin is clearly labelled on the controller board.
- In general, you have power outputs such as 5 volts or 3.3 volts.
- one or more electric ground connections (GND), numbered digital pins, and numbered analogue pins prefixed with an *A*.



- You can power the Arduino using a USB connection from your computer.
- The Arduino also has a socket for an external power supply.
- Either way should be capable of powering the microcontroller and the usual electronics that you might attach to it.
- The Arduino Ethernet has an on-board Ethernet chip and trades the USB socket for an Ethernet one, making it easier to hook up to the Internet.
- This is obviously a strong contender for a useful board for Internet of Things projects.
- The LilyPad has an entirely different specialism, as it has a flattened form (shaped, as the name suggests, like a flower with the I/O capabilities exposed on its "petals") and is designed to make it easy to wire up with conductive thread, and so a boon for wearable technology projects.
- Most of the boards share the same layout of the assorted GPIO, ADC, and power pins, and you are able to piggyback an additional circuit board on top of the Arduino which can contain all manner of componentry to give the Arduino extra capabilities.
- In the Arduino world, these add-on boards are called *shields*, perhaps because they cover the actual board as if protecting it.
- Some shields provide networking capabilities—Ethernet, WiFi, or Zigbee wireless, for example.
- Motor shields make it simple to connect motors and servos;
- there are shields to hook up mobile phone LCD screens;
- others to provide capacitive sensing;
- others to play MP3 files or WAV files from an SD card;

- (http://shieldlist.org/, is dedicated to comparing and documenting them)
-
- **OPENNESS:**
- The Arduino project is completely open hardware and an open hardware.
- The only part of the project protected is the Arduino trademark, so they can control the quality of any boards calling themselves an Arduino.

RASPBERRY PI

- The Raspberry Pi, unlike the Arduino, wasn't designed for physical computing at all, but rather, for education.
- Uses Broadcom BCM2835 system-on-chip, powerful graphics processing unit (GPU), capable of high-definition video and fast graphics rendering.
- the Raspberry Pi is effectively a computer that can run a real, modern operating system, communicate with a keyboard and mouse, talk to the Internet, and drive a TV/monitor with high-resolution graphics.

The following table compares the specs of the latest, most powerful Arduino model, the Due, with the top-end Raspberry Pi Model B:

	Arduino Due	Raspberry Pi Model B
CPU Speed	84 MHz	700 MHz ARM11
GPU	None	Broadcom Dual-Core VideoCore IV Media Co-Processor
RAM	96KB	512MB
Storage	512KB	SD card (4GB +)
OS	Bootloader	Various Linux distributions, other operating systems available
Connections	54 GPIO pins 12 PWM outputs 4 UARTs SPI bus I²C bus USB 16U2 + native host 12 analogue inputs (ADC) 2 analogue outputs (DAC)	8 GPIO pins 1 PWM output 1 UART SPI bus with two chip selects I²C bus 2 USB host sockets Ethernet HDMI out Component video and audio out

- **DEVELOPING ON THE RASPBERRY PI**
-
- **Operating System**
- Although many operating systems can run on the Pi, we recommend using a popular Linux distribution, such as
- ▪ **Raspbian:**
- Released by the Raspbian Pi Foundation, It is based on Debian.
- This is the default "official" distribution and is certainly a good choice for general work with a Pi.

- ▪ **Occidentalis:**
- This is Adafruit's customised Raspbian.
- Unlike Raspbian, the distribution assumes that you will use it "headless"—not connected to keyboard and monitor—so you can connect to it remotely by default.
- The main advantages are that
- ▪ The sshd (SSH protocol daemon) is enabled by default, so you can connect to the console remotely.
- ▪ The device registers itself using zero-onfiguration networking (zeroconf) with the name raspberrypi.local, so you don't need to know or guess which IP address it picks up from the network in order to make a connection.
- The following command, from a Linux or Mac command line, lets you log in to the Pi just as you would log in to a remote server:
- $ ssh root@raspberrypi.local
- From Windows, you can use an SSH client such as PuTTY
-
- **Programming Language**
- One choice to be made is which programming language and environment you want to use.
- Python is a good language for educational programming (and indeed the name "Pi" comes initially from Python).

- **"blinking lights" example:**
- import RPi.GPIO as GPIO
- from time import sleep
- GPIO.setmode(GPIO.BOARD)
- # set the numbering scheme to be the
- # same as on the board
- GPIO.setup(8, GPIO.OUT)
- # set the GPIO pin 8 to output mode
- led = False
- GPIO.output(8, led)
- # initiate the LED to off
- while 1:

```
        GPIO.output(8, led)
        led = not led
        # toggle the LED status on/off for the next iteration
        sleep(10)
        # sleep for one second
```

Contrast Python with C++

- **Python, as with most high-level languages, compiles to relatively large (in terms of memory usage) and slow code, compared to C++.**
- There is no issue because the Pi has more than enough memory.
- The speed of execution may or may not be a problem: Python is likely to be "fast enough" for most tasks, and certainly for anything that involves talking to the Internet.
- **Python handles memory management automatically.**
- Automatic memory management generally results in fewer bugs.
- However, this automatic work has to be scheduled in and takes some time to complete.
- Depending on the strategy for garbage collection, this may result in pauses in operation which might affect timing of subsequent events.

- **Linux itself arguably has some issues for "real-time" use.**
- With many processes that may run simultaneously, precise timings may vary due to
- how much CPU priority is given to the Python runtime at any given moment.
- **An Arduino runs only the one set of instructions, in a tight loop, until it is turned off or crashes.**
- The Pi constantly runs a number of processes.
- If one of these processes misbehaves, or two of them clash over resources (memory, CPU, access to a file or to a network port), they may cause problems that are entirely unrelated to your code.
-
- **Debugging**
- While Python's compiler also catches a number of syntax errors and attempts to use undeclared variables, it is also a relatively permissive language (compared to C++) which performs a greater number of calculations at runtime.
- This means that additional classes of programming errors won't cause failure at compilation but will crash the program when it's running, Python code on Linux gives you the advantages of both the language and the OS.
- You could step through the code using Python's integrated debugger, attach to the process using the Linux strace command, view logs, see how much memory is being used, and so on.
- As long as the device itself hasn't crashed, you may be able to ssh into the Raspberry Pi and do some of this debugging while your program has failed.
- Because the Pi is a general-purpose computer, without the strict memory limitations of the Arduino, you can simply use try... catch... logic so that you can trap errors in your Python code and determine what to do with them.
-
- **SOME NOTES ON THE HARDWARE**
- The Raspberry Pi has 8 GPIO pins, which are exposed along with power and other interfaces in a 2-by-13 block of male header pins.
- The pins in the Raspberry Pi aren't individually labelled.
- The block of pins provides both 5V and 3.3V outputs.
- The GPIO pins themselves are only 3.3V tolerant.
- The Pi doesn't have any over-voltage protection, so you are at risk of breaking the board if you supply a 5V input!
- The Raspberry Pi doesn't have any analogue inputs (ADC), which means that options to connect it to electronic sensors are limited.
- To get readings from light-sensitive photocells, temperature sensors, potentiometers, and so on, you need to connect it to an external ADC via the SPI bus.
-
- **OPENNESS**
- Many of the components are indeed highly open: the customised Linux distributions such as "Raspbian" (based on Debian), the ARM VideoCore drivers, and so on.
- The core Broadcom chip itself is a proprietary piece of hardware.

IOT
Chapter 6
PROTOTYPING THE PHYSICAL DESIGN

PREPARATION

- To best prepare for design work involves **developing an interest in design and having examples of design ideas you like**.
- All you need to do is **develop an interest in the world around you and start paying attention to the wealth of objects and experiences that you encounter.**
- This first step **will help you work out what you like and what you don't like.**
- Over time, **you'll start to work out which qualities you particularly appreciate.**
- As you go, **collect details about the items that inspire you.**
- **Take photos, jot down things in your notebook, or sketch them out.**
- Paste them into an old-school scrapbook.
- Over time you'll **build up an archive of good design that will help you spark your creativity when designing things of your own.**
- The other side to a (very) basic design education is **an understanding and appreciation of the tools available** and the ***materiality* of the processes involved in making things in the physical.(rather than the digital)**
- **Items crafted in** the **digital** world can focus exclusively on appearances and **take on any form they like**; in the **real world,** those annoying laws of **physics** and the **limitations of the tools for fabrication** come into play.
- As with most things, **the more experience you gain with different materials and tools, the better your understanding will be when you come to design something.**
- **Start playing around with the tools you've got to hand.**
- Get involved in local (or not so local) hackdays(event in which computer programmers and others involved in software development, including graphic designers, interface designers, project managers, and others, often including subject-matter-experts, collaborate intensively on software projects).
- **Make things to give as gifts** (or just to see what happens).

SKETCH, ITERATE, AND EXPLORE

- In the early stages of a design, you can never do too much iterating through ideas and trying out different approaches to solving the problem.
- However, **your first idea is unlikely to be the best, so you should be optimising for speed of iteration rather than quality of prototype.**
- You could **iterate through designs** with a 3D printer, but doing so **with a pen and paper is much quicker.**
- **Use whatever tools make most sense to help with the idea generation and exploration.**
- You might **use** a mood board—**a whiteboard where you jot down thoughts and sketches over a few days**—or a notebook that you doodle sketches in while sitting on a park bench watching the world pass by.
- **The sketches aren't for an art gallery; they're to help you work through your thinking.**
- They only need to **capture and convey your ideas**.
- **The more sketching you do, the better they will get.**

- **Don’t be afraid to take your sketching into three dimensions.**
- **Try out different sizes** and see how the changes in dimensions affect the feel or the look of the design.
- **Then give it or show it to some of the people who might use the finished item to find out how they interact with it.**
- The **key lesson** is to **use these techniques to experiment with different possibilities** and learn which features of which designs are best.
- Consider, for example, the **evolution of the design for the Good Night Lamp.**
- The **original design** was a **more traditional lamp shape**, but in a design workshop, the team batted around a range of ideas with the help of a purely functional prototype.
- They realised that a design echoing the **shape of a house better conveyed the core concept of connecting loved ones in their homes.**

The evolving design for the Good Night Lamp (from left to right): original design; functional mockup; redesign mockup; redesign functional prototypes (orange and wood/acrylic); revised size mockup; revised size functional prototype.

NONDIGITAL METHODS

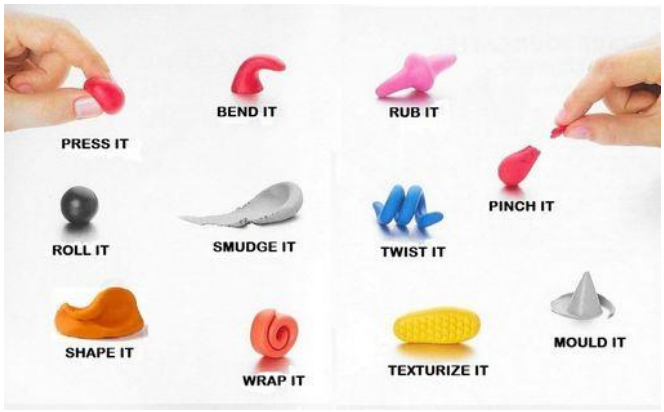
- Many of **traditional craft techniques** are just as valid for use when prototyping the physical form of your device.
- One of the **key advantages** that these techniques have over the newer digital fabrication methods is their **immediacy.**
- **Three-dimensional printing times are often measured in hours**, and although laser cutting is much faster, performing a cut still takes minutes.
- And all this is without including the **time taken to revise the design on the computer first.**
- Let’s look at some of the more common options here:
- ▪ **Modelling clay:**
- The most well-known brands are Play-Doh and Plasticine, but you can find a wealth of different versions with slightly different qualities.
- Some, like **Play-Doh**, have a **tendency to dry out and crack if left exposed to the air.**
- **Modelling clay is best used for short-term explorations of form, rather than longer-term functional prototypes.**

- ▪ **Epoxy putty:**

- **It is adhesive to most materials, tough and durable, and self hardening.**
- Product as the brand Milliput; it is similar to modelling clay although usually available in fewer colours.
- **It comes in two parts, one of which is a hardener.**
- **You mix equal parts together to activate the epoxy.**
- **You then mould it to the desired shape, and in about an hour, it sets solid.**
- If you like, you can then sand it or **paint it for a better finish**, so this product works well for more durable items.

- ▪ **Sugru:**

-



- **Sugru is a mouldable silicone rubber.**
- Like epoxy putty, **it can be worked for only a short time before it sets** (about 30 minutes, and then about a day to fully cure); but unlike epoxy, once cured, it remains flexible.
- **It is also good at sticking to most other substances** and **gives a soft-touch grippy surface**, which makes it a great addition to the designer's (and hacker's) toolkit.

- ▪ **Toy construction sets:**

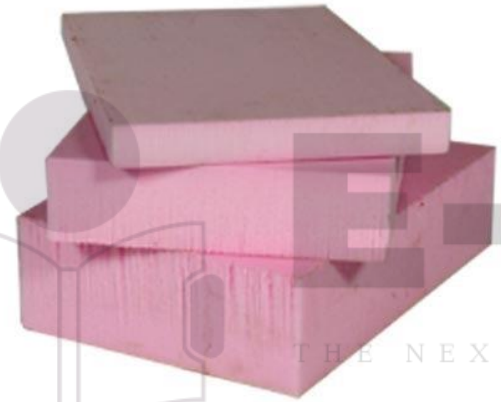
- The other interesting feature of these sets is the **availability of gears, hinges, and other pieces to let you add some movement to your model.**
- You can purchase systems to control LEGO sets from a computer.
- **Many hackers combine an Arduino for sensing and control with LEGO for form.**

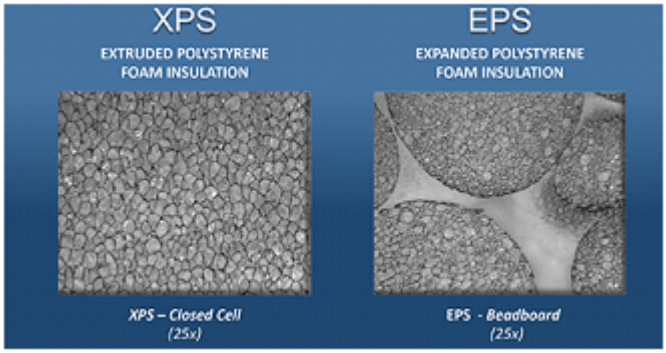
- ▪ **Cardboard:**
- Cardboard **is cheap and easy to shape with a craft knife or scissors**, and available in all manner of colours and thicknesses.
- It provides a reasonable amount of structural integrity and **works well for sketching out shapes** that you'll later cut out of thin plywood or sheets of acrylic in a laser cutter.

- ▪ **Foamcore or foamboard:**

- This sheet material is **made up of a layer of foam sandwiched by two sheets of card.**
- It's readily available at art supplies shops and comes in 3mm or 5mm thicknesses in a range of sizes.
- **Like cardboard, it is easily cut with a craft knife.**

- ▪ **Extruded polystyrene:**





- **Extruded polystyrene** boards are used as an insulation.
- This product is similar to the *expanded* polystyrene that is used for packaging but is **a much denser foam that is better suited to modelling purposes.**
- It is often referred to as "blue foam", although it's the density rather than the colour which is important.
- **Light yet durable, it can be easily worked: you can cut it with a craft knife, saw it, sand it, or, for the greatest ease in shaping it, buy a hot-wire cutter.**
- Sheets of extruded polystyrene are much **thicker than foamboard**, usually between 25mm and 165mm.
- As a result, it is great for mocking up solid three-dimensional shapes.

- If you need something thicker than the sheet itself, you can easily glue a few layers together.
- **The dust from sanding it and the fumes given off when cutting it with a hot-wire cutter aren't too nice, so make sure you wear a dust mask and keep the area ventilated when working with it.**

LASER CUTTING

- **Three-dimensional printers can produce more complicated parts**, but the simpler design process **greater range of materials which can be cut, and faster speed make the laser cutter a versatile piece of kit.**
- **Laser cutters range from desktop models to industrial units which can take a full 8' by 4' sheet in one pass.**
- Most commonly, though, **they are floorstanding and about the same size as a large photocopier.**
- Most of the laser cutter is given over to the **bed**; this **is a flat area that holds the material to be cut.**
- The bed **contains a two-axis mechanism with mirrors and a lens to direct the laser beam to the correct location** and focus it onto the material being cut.
- It is similar to a flatbed plotter but one that burns things rather than drawing on them.
- **The computer controls the two-axis positioning mechanism and the power of the laser beam.**
- At a sufficiently **low power**, this feature **enables you to etch additional detail into the surface of the piece.**
- You can also **etch things at different power levels to achieve different depths of etching.**
-
- CHOOSING A LASER CUTTER
- When choosing a laser cutter, you should **consider two main features:**
- ▪ **The size of the bed:**

- This is the place where the sheet of material sits while it's being cut, so **a larger bed can cut larger items.**
- A larger bed allows you to buy material in bigger sheets (which is more cost effective), and **if you move to small-scale production, it would let you cut multiple units in one pass.**
- ▪ **The power of the laser:**
- **More powerful lasers can cut through thicker material.**
- Depending on what you're trying to create, you can cut all sorts of different materials in a laser cutter.
- **You are able to get laser cutters which can cut metal**, they tend to be the more powerful and industrial units.
- **The lower-powered models don't cut through the metal**; and worse, as the shiny surface of many metals does an excellent job of reflecting the laser beam, you run a real risk of damaging the machine.
- If you don't have a laser cutter of your own, there is a good chance that your local makerspace or hackspace will have one that you could use.

SOFTWARE

- The file formats or software which you need to use to provide your design vary across machines and providers.
- Although some laser-cutting software will let you define an engraving pattern with a bitmap, typically you **use some type of vector graphics format.**
- Vector formats **capture the drawing as a series of lines and curves, which translate much better into instructions for moving the laser cutter than the grid-like representation of a bitmap.**
- **With a bitmap**, as you might have seen if you've ever tried blowing up one small part of a digital photo, **the details become jagged as you zoom in closely**, whereas the **vector format knows** that **it's still a single line and can redraw it with more detail.**
- **CorelDRAW is a common choice for driving the laser cutters themselves, and you can use it to generate the designs too.**
- Other popular options are **Adobe Illustrator**, as many designers already have a copy installed and are familiar with driving it.
- The ***best* choice** is the **one you're most comfortable working with**, **or** failing that, **either the one your laser cutter uses or the one you can afford.**
- **When creating your design, you use the stroke (or outline) of the shapes and lines rather than the filled area to define where the laser will cut and etch.**
- The ***kerf*, the width of the cut made by the laser**, is about 0.2mm but isn't something you need to include in the design.
- **A thinner stroke width is better,** as it will stop the laser cutter from misinterpreting it as two cuts when you need only one.
- **Different types of operation—cut versus etch or even different levels of etching—can usually be included in the same design file just by marking them in different colours.**
- **Whoever is doing your cutting may have a set convention of colour scheme for different settings, so you should make sure that you follow this convention if that is the case.**

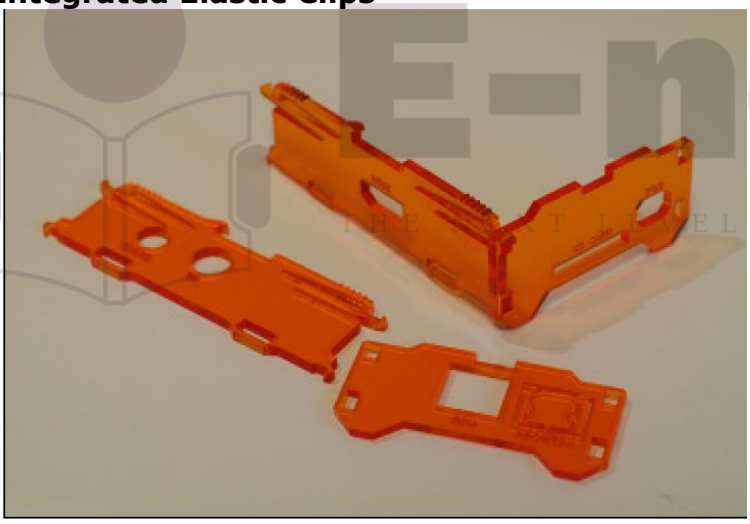
HINGES AND JOINTS

Lattice (or Living) Hinges

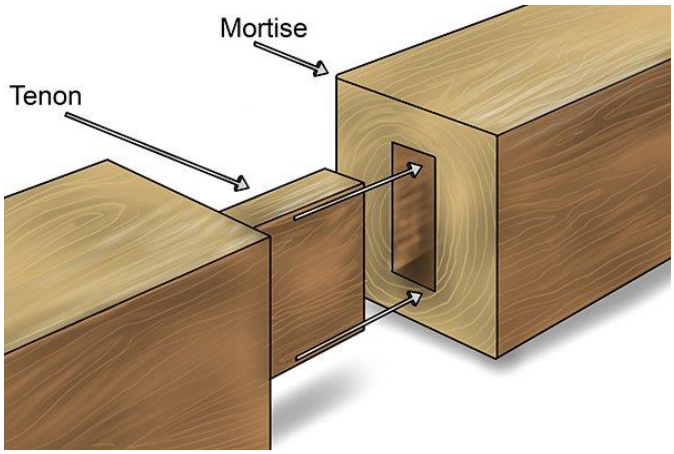
- To introduce some curves into your design.
- A series of closely laid-out cuts, perpendicular to the direction of the curve, allows the material to be bent after it has been cut.
- Varying the number of cuts and their separation affects the resulting flexibility of the hinge.

- The lattice (or living) hinge.

- **Integrated Elastic Clips**



- Integrated elastic clips.



- This jointing technique is used in situations similar to a through mortise-and-tenon joint, when joining two sheets of material at 90 degrees.
- The tenon is replaced with two hooks for holding the mortise sheet tight to the tenon sheet without any need for glue or additional fixings.
- To provide the required flexibility in the tenon to fit it through the mortise during assembly, additional, deeper cuts are made into the tenon side.

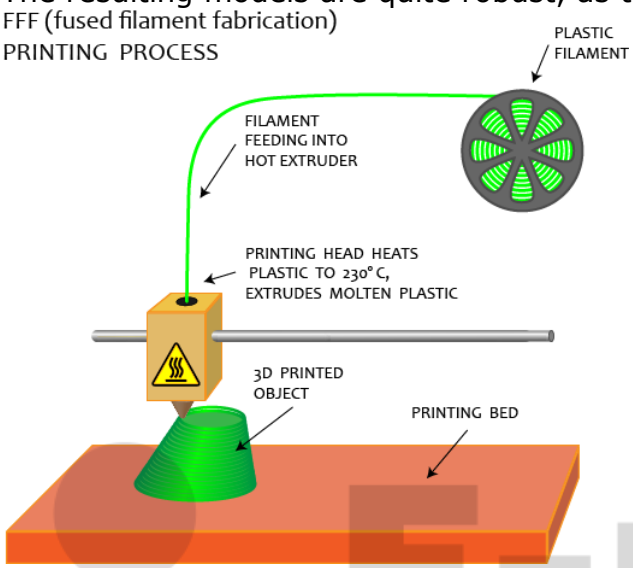
- **Bolted Tenon (or T-Slot) Joints**
- It is a modified version of the standard mortise-and-tenon joint which adds a T- or crossshaped slot to the tenon sheet, with the crossbar of the T or cross being just big enough to hold a nut.
- You can then thread a bolt through a hole in the mortise sheet, down the slot and through the nut.
- 

3D PRINTING

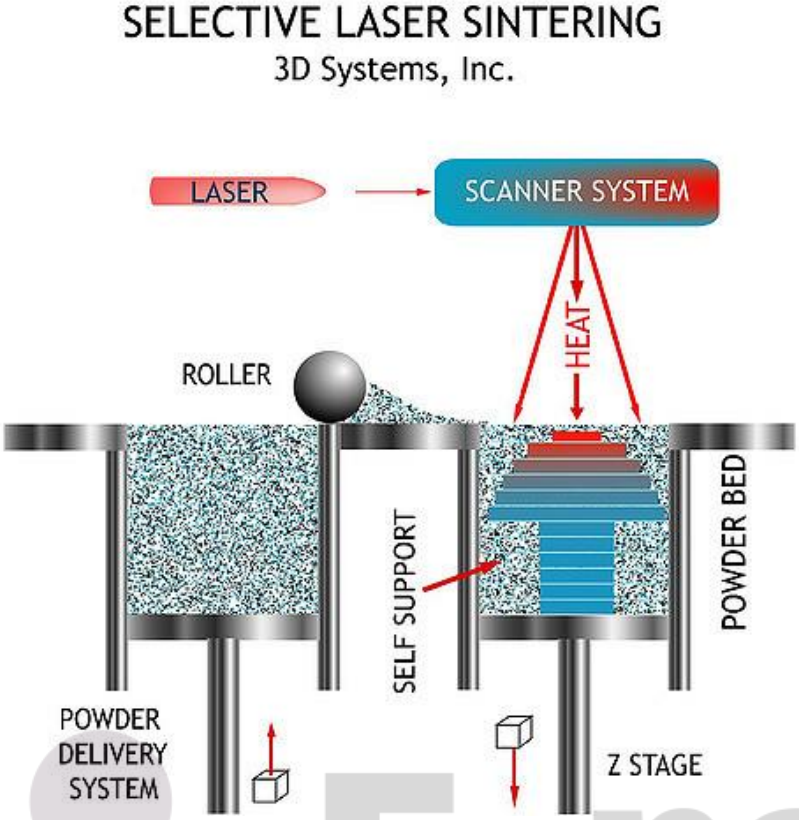
- The term *additive manufacturing* is used because all the various processes which can be used to produce the output start with nothing and *add* material to build up the resulting model.
- This is in contrast to *subtractive manufacturing* techniques such as laser cutting , where you start with more material and cut away the parts you don't need.
- We have three-dimensional computer model as the input.
- The software slices the computer model into many layers, each a fraction of a millimetre thick, and the physical version is built up layer by layer.
- It can produce items which wouldn't be possible with traditional techniques.
- For example, because you can print interlocking rings without any joins, you are able to use the metal 3D printers to print entire sheets of chain-mail which come out of the printer already connected together.
- Another common trick with 3D printing is to print pieces which include moving parts: it is possible to print all the parts at the same time and print them ready-assembled.
- Other processes, such as the extruded plastic techniques, require you to print a second material, which takes the supporting role.
- When the print is finished, this support material is either broken off or washed away.
- (The support material is specifically chosen to dissolve in water or another solution which doesn't affect the main printing material.)

TYPES OF 3D PRINTING

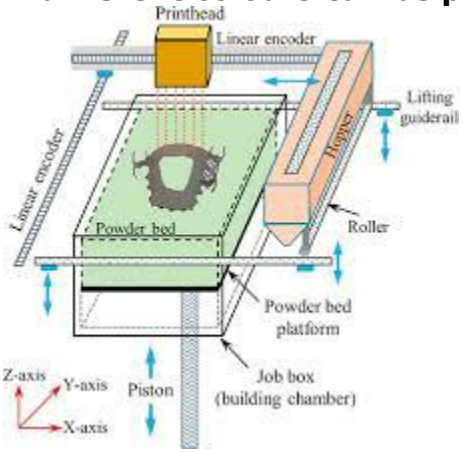
- ▪ **Fused filament fabrication (FFF):** Also known as fused deposition modeling (FDM).
- It works by extruding (**Plastics extrusion** is a high-volume manufacturing process in which **raw plastic is melted and formed into a continuous profile.**) a fine filament of material (usually plastic) from a heated nozzle.
- The nozzle can be moved horizontally and vertically by the controlling computer, as can the flow of filament through the nozzle.
- The resulting models are quite robust, as they’re made from standard plastic.



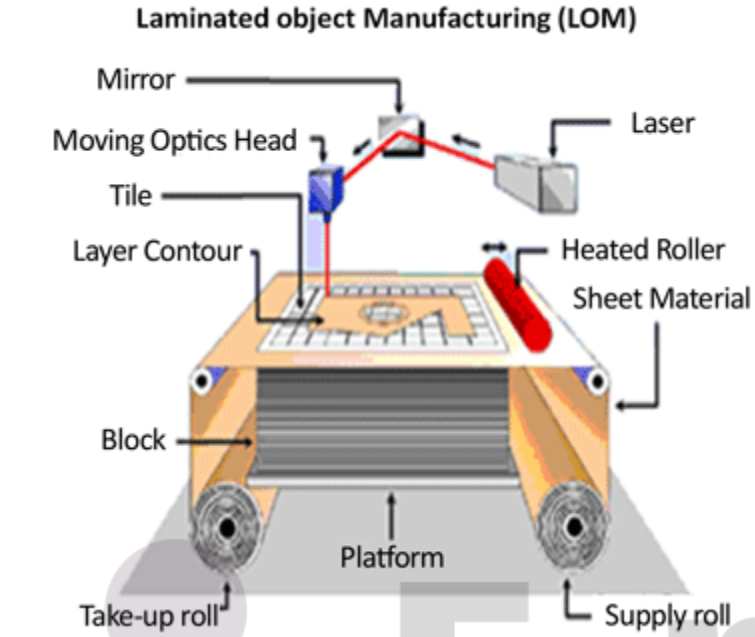
- **Laser sintering:**
- This process is sometimes called selective laser sintering (SLS), electron beam melting (EBM), or direct metal laser sintering (DMLS).
- It is used in more industrial machines but **can print any material which comes in powdered form and which can be melted by a laser.**
- It **provides a finer finish than FDM**, but the models are just as robust, and they’re even stronger when the printing medium is metal.
- This technique is **used to print aluminium** or titanium, although it can just as easily print nylon.



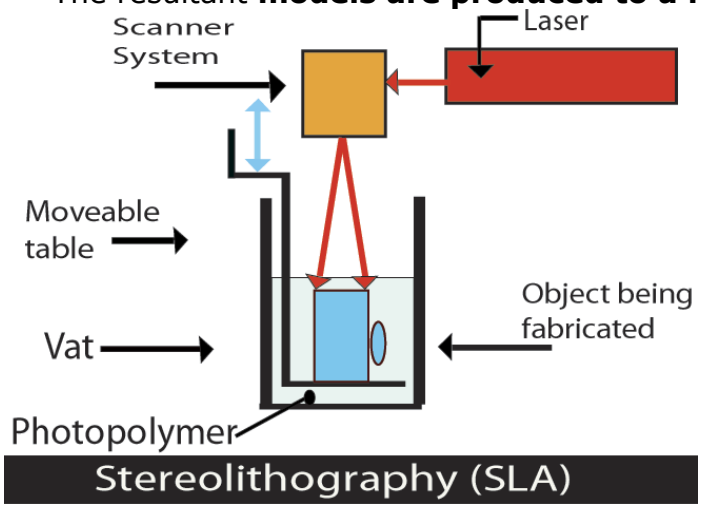
- **Powder bed:**
- Like laser sintering, the powder-bed printers **start with a raw material in a powder form, but rather than fusing it together with a laser, the binder is more like a glue which is dispensed by a print head similar to one in an inkjet printer.**
- After the printing process, the **models are** quite brittle(**delicate and easily broken**) and so **need postprocessing where they are sprayed with a hardening solution.**
- The great advantage of these printers is that **when the binder is being applied, it can be mixed with some pigment**; therefore, full-colour prints in **different colours can be produced in one pass**.



- ▪ **Laminated object manufacturing (LOM):**
- This is another method which can produce full-colour prints.
- LOM **uses traditional paper printing** as part of the process.
- Because it **builds up the model by laminating many individual sheets of paper together, it can print whatever colours are required onto each layer** before cutting them to shape and gluing them into place.



- **Stereolithography and digital light processing:**
- Both approaches **build their models from a vat (tank) of liquid polymer resin (adhesive)** which is **cured** (refers to the toughening **or hardening** of a polymer material by cross-linking of polymer chains, brought about by electron beams, heat, or **chemical** additives.) **by exposure to ultraviolet light.**
- **Stereolithography uses a UV laser to trace the pattern for each layer,** whereas **digital light processing uses a DLP projector to cure an entire layer at a time.**
- The resultant **models are produced to a fine resolution**.



SOFTWARE

- If you are already familiar with one 3D design program, see whether it can export files in the correct format for the machine you'll use to print.
- If you are using a printing service, it will advise on which program it prefers you to use or what formats it accepts.
- Or failing that, choose one to suit your budget.
- Working out how to design items in three dimensions through a two dimensional display isn't trivial, so it's more important than usual to work through the tutorials for the software you choose.
- Tinkercad (http://tinkercad.com) and Autodesk's 123D Design Online (http://www.123dapp.com/design) are two options which just run in your web browser.
- So they let you start designing without having to install any additional software.
- Autodesk also has a range of 123D apps available to download and install.
- You can find a desktop version of 123D Design and also of 123D Catch, a clever application which takes an array of photos of an object and automatically converts them into a 3D model.
- SolidWorks (http://www.solidworks.com) and Rhino (http://www.rhino3d.com) are the industry-standard commercial offerings.
- In the open source camp, the main contenders are OpenSCAD (http://www.openscad.org) and FreeCAD (http://free-cad.sourceforge.net).
- When you have your design ready, you need a further piece of software to convert it into a set of instructions which will be fed to the printer.
- This is usually known as the *slicing algorithm* because its most important function is to carve the model into a series of layers and work out how to instruct the printer to build up each layer.
- Skeinforge was the first slicing software used by the open source printers, but it has been largely overtaken by the newer and more user-friendly Slic3r.
- Both will let you change parameters to fine-tune your 3D prints, specifying options like the temperature to which the plastic should be heated, how densely to fill the solid objects, the speed at which the extruder head should move, etc

CNC MILLING

- Computer Numerically Controlled (CNC) milling is similar to 3D printing but is a *subtractive* manufacturing process rather than *additive*.
- A computer controls the movement of the milling head,much like it does the extruder in an FDM 3D printer.
- However, rather than building up the desired model layer by layer from nothing, it starts with a block of material larger than the finished piece.
- It cuts away the parts which aren't needed—much like a sculptor chips away at a block of stone to reveal the statue, except that milling uses a rotating cutting bit (similar to an electric drill)
- Because cutting away material is easier, CNC mills can work with a much greater range of materials than 3D printers can.
- CNC mills can also be used for more specialised (but useful when prototyping electronic devices) tasks, such as creating custom printed circuit boards.
- the CNC mills away lines from the metal surface on the board, leaving the conductive paths.

- An advantage of milling over etching the board is that you can have the mill drill any holes for components or mounting at the same time, saving you from having to do it manually afterwards with your drill press.
- A wide range of CNC mills is available, depending on the features you need and your budget.
- Sizes range from small mills which will fit onto your desktop through to much larger machines with a bed size measured in metres.
- The challenges of accurately moving the carriage around increase with their size.
- Beyond size and accuracy, the other main attribute that varies among CNC mills is the number of axes of movement they have:
- ▪ **2.5 axis:** This type has three axes of movement—X, Y, and Z—it can move only any two at one time.
- ▪ **3 axis:** Like the 2.5-axis machine, this machine has a bed which can move in the X and Y axes, and a milling head that can move in the Z.
- However, it can move all three at the same time
- ▪ **4 axis:** This machine adds a rotary axis to the 3-axis mill to allow the piece being milled to be rotated around an extra axis, usually the X (this is known as the *A axis*).
- ▪ **5 axis:** This machine adds a second rotary axis—normally around the Y—which is known as the *B axis*.
- ▪ **6 axis:** A third rotary axis—known as the *C axis* if it rotates around Z—completes the range of movement in this machine.
- As with 3D printing, the software you use for CNC milling is split into two types:
- ▪ **CAD** (Computer-Aided Design) software lets you design the model.
- ▪ **CAM** (Computer-Aided Manufacture) software turns that into a suitable toolpath—a list of co-ordinates for the CNC machine to follow which will result in the model being revealed from the block of material.

REPURPOSING/RECYCLING

- As with the other elements of building your connected device, a complete continuum exists from buying-in the item or design through to doing-it yourself.
- So, just as you wouldn't think about making your own nuts and bolts from some iron ore, sometimes you should consider reusing more complex mechanisms or components.
- One reason to reuse mechanisms or components would be to piggyback onto someone else's economies of scale.
- If sections or entire subassemblies that you need are available in an existing product, buying those items can often be cheaper than making them in-house.
- If the final design requires processes with massive up-front costs or the skills of a designer that you don't have the funds to hire right now, maybe a product already exists that is near enough to work as a proxy.
- That lets you get on with taking the project forwards, ending up at a point.
- And, of course, it doesn't have to be a finished item that you reuse.
- The website Thingiverse (http://www.thingiverse.com) is a repository of all manner of designs, most of which are targeted at 3D printing or laser cutting.
- All available under creative commons licenses which allow you to use the design as is or often allow you extend it to better suit your own needs.

Case Study: The Ackers Bell

- The Ackers Bell is an Internet-connected bell, which was commissioned by the big-data startup ScraperWiki (http://scraperwiki.com).
- It is connected to the company's online billing system and rings the bell whenever a new payment hits its bank account, giving the sales team further incentive to make more sales and everyone else a chance to celebrate every success.
- Casting a bell from scratch was always going to be a stretch, so the first step was to investigate what bells could be sourced elsewhere and reused.
- An initial discussion with a campanologist friend quickly found some very nice tuned bells, but sadly they were outside the available budget.
- The customer on the choice led to their settling on a traditional brass ship's bell.
- the next step was to work out how to mount it.
- They just needed to devise a way to assemble the two parts: electronics and bell.
- The following photo shows some of the initial sketches made in Adrian's notebook,

Early design ideas for the Ackers Bell.

- They thought wood made the best choice of material to complement the brass of the bell but also wanted a darker wood.
- In addition, the potential joint design would require wood a fair bit thicker than the 3mm ply, so they finally chose an 8mm thick oak from the range of hardwoods at the local timber merchant.

- Unfortunately, some test cuts—or more accurately, test burns—showed that the oak was too hard for either of the readily available laser cutters to manage.
- Further experimentation was required.
- For the choice of wood, they tried a variety of different veneers.
- In the end, a dark beeswax (used in polishes) on the birch ply (**Birch** wood **is** a light wood with a very fine grain.) gave a good colour with the added benefit of some protection and treatment for the wood.
- student from Liverpool University who was spending some time at the company on an internship. He came up with the idea of housing the bell inside a lattice framework of stock 3mm ply, shaped to echo the lines of the bell itself.
- That design meant that although the final form was a fairly complex three-dimensional design, it could be constructed from flat sheets of wood cut on the laser cutter.
- In addition to providing a support to hang the bell, the laser-cut design included a platform to hold the Arduino board and a mounting point for the solenoid— cleverly hidden inside the bell to keep the external profile clean.
- Eventually, they fashioned a shim which adjusted the angle at which the solenoid struck the bell, to make it perpendicular to the bell's surface at the point of impact.

The Ackers Bell, ready for delivery to the customer.

IOT
Chapter 7
PROTOTYPING ONLINE COMPONENTS

Introduction:

- Each component has a critical part to play.
- The physical, designed object ties into the design principles of context.
- The controller and associated electronics allow it to sense and act on the real world.
- The Internet adds a dimension of communication.
- The network allows the device to inform you or others about events or to gather data and let you act on it in real time.
- It lets you aggregate information from disparate locations and types of sensors.
- Similarly, it extends your reach, so you can control or activate things from afar

Q1)What is an API? Explain the concept of Mashing and scrapping APIs.

GETTING STARTED WITH AN API

- The most important part of a web service, with regards to an Internet of Things device, is the Application Programming Interface, or API.
- An API is a way of accessing a service that is targeted at machines rather than people.
- If you think about your experience of accessing an Internet service, you might follow a number of steps.
- **For example, to look at a friend's photo on Flickr, you might do the following:**
- **1.** Launch Chrome, Safari, or Internet Explorer.
- **2.** Search for the Flickr website in Google and click on the link.
- **3.** Type in your username and password and click "Login".
- **4.** Look at the page and click on the "Contacts" link.
- **5.** Click on a few more links to page through the list of contacts till you see the one you want.
- **6.** Scroll down the page, looking for the photo you want, and then click on it.
- Although these actions are simple for a human, they involve a lot of looking, thinking, typing, and clicking.
- A computer can't look and think in the same way.
- The tricky and lengthy process of following a sequence of actions and responding to each page is likely to fail the moment that Flickr slightly changes its user interface.
- For example, if Flickr rewords "Login" to "Sign in", or "Contacts" to "Friends", a human being would very likely not even notice, but a typical computer program would completely fail.
- Instead, a computer can very happily call defined commands such as login or get picture #142857.

- **MASHING UP APIS**
- (Meaning: something created by combining elements from two or more sources.)
- Perhaps the data you want is already available on the Internet but in a form that doesn't work for you?
- The idea of "mashing up" multiple APIs to get a result has taken off and can be used to powerful effect.
- For example:
- ▪ Using a mapping API to plot properties to rent or buy.
- ▪ Showing Twitter trends on a global map or in a timeline or a charting API.

- ▪ Fetching Flickr images that are related to the top headlines retrieved from newspaper's API.

SCRAPING

- (meaning: the action of using a computer program to copy data from a website.)
- **Screen scraping** is the process of collecting **screen** display data from one application and translating it so that another application can display it.
- In many cases, companies or institutions have access to fantastic data but don't want to or don't have the resources or knowledge to make them available as an API.
- This is normally done to capture data from a legacy application in order to display it using a more modern user interface.
-
- **Here are a few examples:**
- Adrian has scraped the Ship AIS system to get data about ships on the river Mersey, and this information is then tweeted by the @merseyshipping account.
- The Public Whip website (www.publicwhip.org.uk/) is made possible by using a scraper to read the Hansard transcripts of UK government sessions (released as Word documents).
- With the resultant data, it can produce both human- and machine-readable feeds of how our elected representatives vote.
- The ScraperWiki site (https://scraperwiki.com) has an excellent platform for writing scrapers, in a number of dynamic programming languages, which collate data into database tables.
- It provides infrastructure for scripts that you could run on your own computer or server but allows you to outsource the boring, repetitive parts to ScraperWiki.
- **LEGALITIES**
- Screen-scraping may break the terms and conditions of a website.
- For example, Google doesn't allow you to screen-scrape.
- Alternative sources of information often are available.
- For example, you could use OpenStreetMap instead of Google Maps.

Q2)How to write new API? Explain with example of Timer.

WRITING A NEW API

- You plan to assemble the data from free or licensed material you have and process it.
- Or perhaps your Internet-connected device can populate this data.
- The process of building your own API is explained with an example project, Clockodillo.
- This is an Internet of Things device that Hakim built to help him use the Pomodoro time management technique
- The technique uses a timer to break down work into intervals, traditionally 25 minutes in length, separated by short breaks.
- These intervals are named ***pomodoros***, the plural in English of the Italian word *pomodoro* (tomato)
- (the tomato-shaped kitchen timer).
- 

- Clockodillo explores how the Internet of Things might help with that:
- connecting the kitchen-timer to the Internet to make the tracking easier while keeping the simplicity of the physical twist-to-set timer for starting the clock and showing progress as it ticks down.
- Clockodillo is an Internet-connected task timer
- The user can set a dial to a number of minutes, and the timer ticks down until completed.
- It also sends messages to an API server to let it know that a task has been started, completed, or cancelled.
- **A number of API interactions deal precisely with those features of the physical device:**
- ▪ Start a new timer
- ▪ Change the duration of an existing timer
- ▪ Mark a timer completed
- ▪ Cancel a timer
- Some interactions with a timer data structure are too complicated to be displayed on a device consisting mostly of a dial—for example, anything that might require a display or a keyboard!
- Those could be done through an app on your computer or phone instead.
- ▪ View and edit the timer's name/description And, naturally, the user may want to be able to see historical data:
- ▪ Previous timers, in a list
- • Their name/description
- • Their total time and whether they were cancelled
- Assume that each device will send some identifying token, such as a MAC address.
- The user will somehow identify himself with the server, after which all the preceding actions will relate just to a given user ID.

Q3) Explain the concept of security with example of Timer.

SECURITY

- How important security is depends a lot on how sensitive the information being passed is and whether it's in anyone's interest to compromise it.
- For Clockodillo, perhaps a boss might want to double-check that employees are using the timer.
- Or a competitor might want to check the descriptions of tasks to spy what your company is working on.
- Or a competitor might want to disrupt and discredit the service by entering fake data.
- If the service deals with health or financial information, it may be an even more attractive target.
- Location information is also sensitive; burglars might find it convenient to know when you are out of the house.

Task	Inputs	Outputs
1. Create a new timed task	User, Timer duration	Timer ID
2. Change duration of timed task	User, Timer ID, New duration	OK
3. Mark timer complete	User, Timer ID	OK
4. Cancel timer	User, Timer ID	OK
5. Describe the timed task	User, Timer ID, Description	OK
6. Get list of timers	User	List of Timer IDs
7. Get information about a timer	User, Timer ID	Description, Create time, Status

-
- The request has to pass details to identify the user, which is the problem of identity; that is, the application needs to know for which user to create the timer so that the user can retrieve information about it later.
- But the application should also authenticate that request.
- A password is "good enough" authentication for something that isn't hypersensitive.
- You have to consider the risks in sending the identification or authentication data over the Internet
- If the username and password are in "clear text", they can be read by anyone who is packet sniffing.

- **The two main cases here are as follows:**
- ▪ **Someone who is targeting a specific user and has access to that person's wired or (unencrypted) wireless network.**
- This attacker could read the details and use them (to create fake timers or get information about the user).
- ▪ **Someone who has access to one of the intermediate nodes.**
- This person won't be targeting a specific device but may be looking to see what unencrypted data passes by, to see what will be a tempting target.
- if a software password is compromised, a website can easily provide a way of changing that password.
- But while a computer has a monitor and keyboard to make that task easy, an Internet-connected device may not.
- So you would need a way to configure the device to change its password—for example, a web control panel hosted on the server or on the device itself.
- This solution is trickier (and does require the machine to have local storage to write the new password to).
- One obvious solution to the problem of sending cleartext passwords would be to encrypt the whole request, including the authentication details.
- For a web API, you can simply do this by targeting https:// instead of http://.
- Instead of username/password on HTTPs allowing a MAC address over HTTP for requests 1-4 that will be sent by the timer.
- Here's a revised table;

Task	Auth	Inputs	Outputs
1. Create a new timed task	MAC or User/Pass	Timer duration	Timer ID
2. Change duration of timed task	MAC or User/Pass	Timer ID, New timer duration	OK
3. Mark timer complete	MAC or User/Pass	Timer ID	OK
4. Cancel timer	MAC or User/Pass	Timer ID	OK

Q4)List and explain the standards to consider for implementing the API.
IMPLEMENTING THE API

- An API defines the messages that are sent from client to server and from server to client.
- You can send data in whatever format you want, but it is almost always better to use an existing standard because convenient libraries will exist for both client and server to produce and understand the required messages.
- Here are a few of the most common standards that you should consider:
- ▪ **Representational State Transfer (REST):**
- Access a set of web URLs using HTTP methods such as GET and POST.
- The result is often XML or JSON but can often depend on the HTTP content-type negotiation mechanisms.
- ▪ **JSON-RPC:**
- JavaScript Object Notation (JSON), is a way of formatting data so that it can be easily exchanged between different systems.
- Remote Procedure Call (RPC) is a term to describe ways of calling programming code which isn't on the same computer as the code you are writing.
- Access a single web URL like http://timer. roomofthings.com/api/, passing a JSON string such as
- {'method':'update', 'params':
- [{'timer-id':1234,'description':'Writing API chapter for book'}],
- 'id':12
- }.
- The return value would also be in JSON like {'result':'OK', 'error':null, 'id':12}.
- ▪ **XML-RPC:** This standard is just like JSON-RPC but uses XML instead of JSON.
- **Simple Object Access Protocol (SOAP):** This standard uses XML for transport like XML-RPC but provides additional layers of functionality, which may be useful for very complicated systems.

- In REST, you attach each resource to a URL and act on that.
- You can use different "methods" depending on whether you want to GET a resource, POST it onto the server in the first place, PUT an update to it, or DELETE it from the server.
- So the REST API will finally look like this:

Resource URL	Method	Auth	Parameters	Outputs
1. /timers	POST	MAC or Cookie	Timer duration	Timer ID
2. /timers/:id/ duration	PUT	MAC or Cookie	Timer duration	OK
3. /timers/:id/ complete	PUT	MAC or Cookie		OK
4. /timers/:id	DELETE	MAC or Cookie		OK
5. /timers/:id/ description	PUT	Cookie	Description	OK
6. /timers	GET	Cookie		List of Timer IDs
7. /timers/:id	GET	Cookie		Info about Timer
8. /user/device	PUT	Cookie	MAC address	OK
9. /user/device	GET	Cookie		MAC address
10. /login	POST	User/Pass	User/Pass	Cookie + OK
11. /user	POST		User/Pass	Cookie + OK

-

Q6) Explain parameters you should consider when deciding on a platform for your web back end with the help of example (PERL).

Following are some of the parameters you should consider when deciding on a platform for your web back end:

- ▪ What do you already know (if you are planning to develop the code yourself)?
- ▪ What is the local/Internet recruiting market like (if you are planning to outsource)?
- ▪ Is the language thriving? Is it actively developed? Does it have a healthy community (or commercial support)? Is there a rich ecosystem of libraries available?
- Ex: Back-end Code in Perl
- Each handler declares the HTTP verb (GET, POST, PUT, DELETE) and the route that it handles.
- Parameters can be passed within the route, marked by a colon (:id, for example), or as part of the HTTP request.
-
- #1 Create a new timed task
- post "/timers.:format" => sub {
- my $user = require_user;
- # 'require_user' wants a session cookie # OR a valid MAC address.
- my $duration = param 'duration'
- or return status_bad_request('No duration passed');
- my $timer = schema->resultset('Timer')->create({
- user_id => $user->id,
- duration => $duration,
- status => 'O', # open
- });

- return status_created({
- status=>'ok',
- id => $timer->id,
- });
- };
-
- **USING CURL TO TEST**
- While you're developing the API, and afterwards, to test it and show it off, you need to have a way to interact with it.
- You could create the client to interface with it at the same time
- # the -F flag makes curl POST the request
- $ curl http://localhost:3000/user.json \
- -F user=hakim -F pass=secret
- {
- "status" : "ok",
- "id" : 2
- }
- curl simply makes an HTTP request and prints out the result to a terminal.

Q7) Explain the following factors with respect to API.

- **API Rate Limiting.**
- **Interaction via HTML**
- **Designing a Web Application for Humans**

GOING FURTHER

- **API Rate Limiting**
- If the service becomes popular, managing the number of connections to the site becomes critical.
- Setting a maximum number of calls per day or per hour or per minute might be useful.
- You could do this by setting a counter for each period that you want to limit.
- Then the authentication process could simply increment these counters and fail if the count is above a defined threshold.
- The counters could be reset to 0 in a scheduled cron job.
- software application can easily warn users that their usage limit has been exceeded and they should try later.

- **Interaction via HTML**
- The API currently serialises the output only in JSON, XML, and Text formats.
- You might also want to connect from a web browser.
- Note that not every Internet of Things product needs a browser application.
- Perhaps the API is all you need, with maybe a static home page containing some documentation about how to call the API.
- In the case of Clockodillo, however, we do want to have a set of web pages to interact with: Users should be able to look at their timers, assign descriptions, and so on.
- Drawbacks:
- Although web browsers do speak HTTP, they don't commonly communicate in all the methods that we've discussed.
- Web browsers don't commonly support PUT and DELETE.
- They support only GET and POST.
- As a solution we can "tunnel" the requests through a POST.

- There is a convention in Perl to use a field called x-tunneled-method, which you could implement like this:
- <form method="POST" action="/timer.html?x-tunneled-method=DELETE">
- <input type="hidden" name="id" value="1234">
- <input type="submit" value="Cancel this timer!">
- </form>

- **Designing a Web Application for Humans**

- For example, the above figure shows a static login page.
- This page is entirely for the convenience of a human.
- All the labels like "Your email address" and the help text like "Remember your password is case sensitive" are purely there to guide the user.
- The logo is there as a branding and visual look and feel for the site.

Q8)What are REAL-TIME REACTIONS? Explain two options such as "polling" and "Comet"

- To establish an HTTP request requires several round-trips to the server.
- There is the TCP "three-step handshake" consisting of a SYN (synchronise) request from the client, a SYN-ACK from the server to "acknowledge" the request, and finally an ACK from the client.
- Although this process can be near instantaneous, it could also take a noticeable amount of time.
- If you want to perform an action the instant that something happens on your board, you may have to factor in the connection time.
- If the server has to perform an action immediately, that "immediately" could be nearly a minute later, depending on the connection time.
- For example, with the task timer example, you might want to register the exact start time from when the user released the dial, but you would actually register that time plus the time of connection.
- We look at two options here: polling and the so-called "Comet" technologies.

POLLING

- If you want the device or another client to respond immediately, how do you do that?
- You don't know when the event you want to respond to will happen, so you can't make the request to coincide with the data becoming available.
- Consider these two cases:

- ▪ The WhereDial should start to turn to “Work” the moment that the user has checked into his office.
- ▪ The moment that the task timer starts, the client on the user’s computer should respond, offering the opportunity to type a description of the task.
- The traditional way of handling this situation using HTTP API requests was to make requests at regular intervals. This is called *polling*.
- You might make a call every minute to check whether new data is available for you.
- However, this means that you can’t start to respond until the poll returns.
- So this might mean a delay of (in this example) one minute plus the time to establish the HTTP connection.
- You could make this quicker, polling every 10 seconds, for example.
- But this would put load on the following:
- ▪ **The server:** If the device takes off, and there are thousands of devices, each of them polling regularly, you will have to scale up to that load.
- ▪ **The client:** This is especially important if, as per the earlier Arduino example, the microcontroller blocks during each connect!

- **COMET**
- *Comet* is an umbrella name for a set of technologies developed to get around the inefficiencies of polling.
- **Long Polling (Unidirectional)**
- starts off with the client making a polling request as usual.
- However, unlike a normal poll request, in which the server immediately responds with an answer, even if that answer is “nothing to report”, the long poll waits until there is something to say.
- This means that the server must regularly send a keep-alive to the client to prevent the Internet of Things device or web page from concluding that the server has simply timed out.
- Long polling would be ideal for the case of WhereDial: the dial requests to know when the next change of a user’s location will be.
- As soon as WhereDial receives the request, it moves the dial and issues a new long poll request.
- However, it isn’t ideal for the task timer, with which you may want to send messages from the timer quickly, as well as receive them from the server.

- **Multipart XMLHttpRequest (MXHR) (Unidirectional)**
- Many browsers support a multipart/x-mixed-replace content type, which allows the server to send subsequent versions of a document via XHR.
- It is perfectly possible to simply long poll and create a new request on breaking the old one, but this means that you might miss a message while you’re establishing the connection.
- In the example of WhereDial, this is unlikely; you’re unlikely to change location first to Home and then to Work in quick succession.

- **HTML5 WebSockets (Bidirectional)**
- Traditionally, the API used to talk directly to the TCP layer is known as the *sockets API*.
- When the web community was looking to provide similar capabilities at the HTTP layer, they called the solution *WebSockets*.
- WebSockets have the benefit of being bidirectional.

- You can consider them like a full Unix socket handle that the client can write requests to and read responses from.
- This might well be the ideal technology for the task timer.
- After a socket is established, the timer can simply send information down it about tasks being started, modified, or cancelled, and can read information about changes made in software, too.

Q9) Explain how to implement Comet? And how scaling works with Comet.

- **Implementations**
- Let's look at support for these techniques on the three main platforms that you may need to consider for an Internet of Things application: the browser web app (if applicable), the microcontroller itself, and the server application.
- On the browser side, it is often possible to abstract the actual transport using a library which chooses which method to connect to the server.
- For example, it might use WebSockets if available; otherwise, it will fall back to MXHR or long polling.
- Many web servers have abstractions to support Comet techniques.
- Web::Hippie::Pipe provides a unified bidirectional abstraction for Perl web servers.
- There are also libraries for the microcontroller; however, they tend to support only one scheme.
- For example, several dedicated WebSockets libraries are available for Arduino.

- **Scaling**
- An important consideration is that all these Comet techniques require the client to have a long-term connection with the server.
- For a single client, this is trivial.
- But if there are many clients, the server has to maintain a connection with each of them.
- If you run a server with multiple threads or processes, you effectively have an instance of the server for each client.
- As each thread or process will consume system resources, such as memory, this doesn't scale to many clients.
- Instead, you might want to use an asynchronous web server, which looks at each client connection in turn and services it when there is new input or output.
- If the server can service each client quickly, this approach can scale up to tens of thousands of clients easily.

Q10)What OTHER PROTOCOLS are available to replace HTTP.

- **MQ TELEMETRY TRANSPORT**
- MQTT (http://mqtt.org) is a lightweight messaging protocol, designed specifically for scenarios where network bandwidth is limited.
- It was developed initially by IBM but has since been published as an open standard, and a number of implementations, both open and closed source, are available, together with libraries for many different languages.
- Rather than the client/server model of HTTP, MQTT uses a publish/ subscribe mechanism for exchanging messages via a *message broker*.
- Rather than send messages to a pre-defined set of recipients, senders publish messages to a specific *topic* on the message broker.
- Recipients subscribe to whichever topics interest them, and whenever a new message is published on that topic, the message broker delivers it to all interested recipients.

- This makes it much easier to do one-to-many messaging, and also breaks the tight coupling between the client and server that exists in HTTP.

- **EXTENSIBLE MESSAGING AND PRESENCE PROTOCOL**
- Another messaging solution is the Extensible Messaging and Presence Protocol, or XMPP (http://xmpp.org).
- This is both a blessing and a curse: it is well understood and widely deployed, but because it wasn't designed explicitly for use in embedded applications, it uses XML to format the messages.
- This choice of XML makes the messaging relatively verbose ("using more words than necessary".) which could preclude (prevent from happening; make impossible) it as an option for RAM-constrained microcontrollers.

- **CONSTRAINED APPLICATION PROTOCOL**
- The Constrained Application Protocol (CoAP) is designed to solve the same classes of problems as HTTP but, for networks without TCP.
- There are proposals for running CoAP over UDP, SMS mobile phone messaging.
- CoAP draws many of its design features from HTTP and has a defined mechanism to proxies to allow mapping from one protocol to the other.

IOT
Chapter 8
TECHNIQUES FOR WRITING EMBEDDED CODE

Q1) Explain MEMORY MANAGEMENT issues in Embedded code?

- 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.
- 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 that 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.
- **The different types of memory you might encounter are:**
- **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.
- 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.
- **Flash**
- Flash is a semi-permanent type of memory which provides all the advantages of ROM, 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.
- 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.
- 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.

- Systems tend to have a lot more persistent storage than they do RAM, so it makes sense to keep as much in flash memory as is possible.
- You can also provide hints to the compiler to help it place as much as possible of the running program into flash.
- 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 (which are commonly used in embedded systems), you do this by using the const keyword.
- 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.
- 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"));

Q2)How to make THE MOST OF YOUR RAM ?

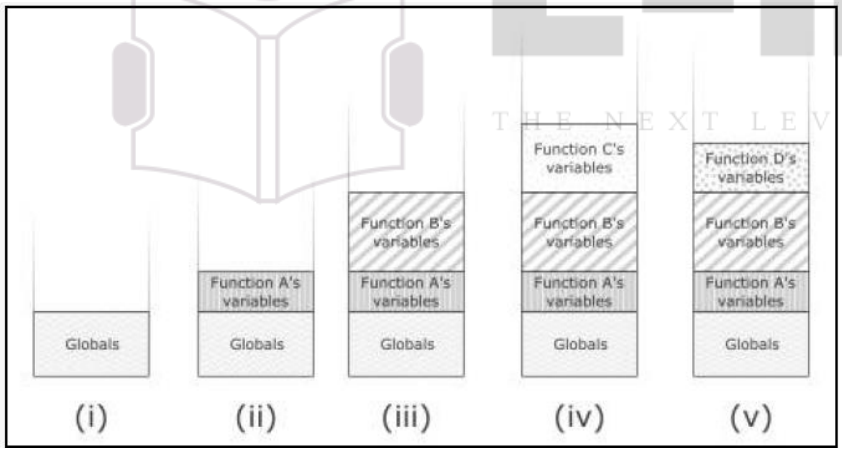
- Now that you've moved everything that you can out of RAM and into flash, all that remains is to work out ways to make better use of the free memory you have.
- When you have only a few kilobytes or tens of kilobytes of RAM available, it is easier to fill up that memory, causing the device to misbehave or crash.
- Yet you may want to use as much of the memory as possible to provide more features.
- There is a 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.
- For example when you're downloading some information from the Internet how could you possibly know beforehand exactly how large it is going to be?
- The standard mechanism on desktop or server systems would be to allocate just enough memory at the time you're downloading things, *when* you know how much you'll need.
- In a deterministic model, you need to take a different tack.
- Rather than allocate space for the entire page, you set aside space to store the important information that you're going to extract and also a buffer of memory that you can use as a working area while you download and process the page.
- 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!
- 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.
- If the format of the data you're consuming means that you know if something is needed only by the time you reach a later part of the file, you have to set aside space to save the potentially useful segment when you encounter it.
- Then when you reach the decision point, you still have it available to use and can discard it then if it's not required.
- You might cache the download to an SD card or other area of flash memory, where it could be processed in multiple passes without needing to read it all into RAM at once.

Q3) Explain Organising RAM: Stack versus Heap.

- Two general concepts for arranging memory are used: the stack and the heap.
- 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.
- The processor 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.
- 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.

- 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()
```

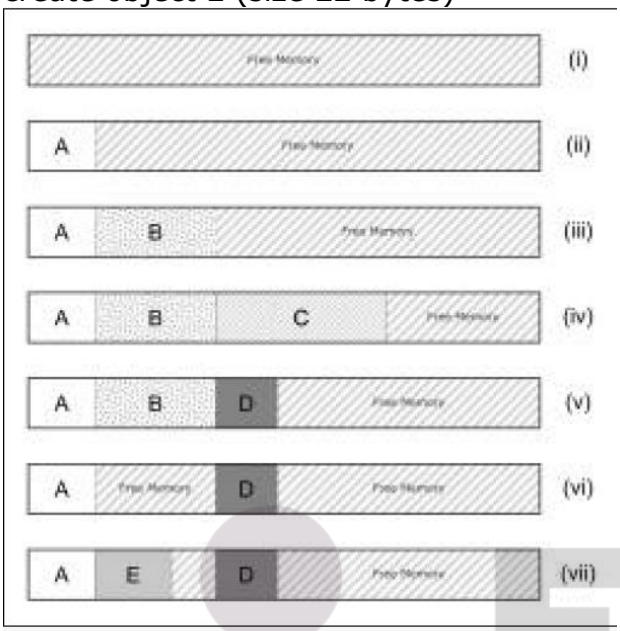


HEAP

- 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.
- 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.

- Following 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)
```



- *Deterministic memory usage suggests* avoiding placing things on the heap at all costs.
- That is, in systems with only a few kilobytes of RAM, we recommend exclusively using the stack to store variables.
- But it is still possible to run out of memory when just using the stack.
- This situation is called *stack overflow*.
- On some processors this situation occurs because the stack is a fixed block partitioned off in memory:
- if your program has used more of the stack at some point during its execution , it could outgrow this fixed space.
- One way to reduce the chance of this situation occurring is to keep the number of global variables to a minimum.
- 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.
- If you can move more of your variables inside the functions where they're actually used, you can free up more space for use during other parts of the execution path.
- The other way to keep down your stack usage, or at least within well-defined bounds, is to avoid using recursive algorithms.
- The stack grows with each recursion.
- If you know how many times a given function will recurse for the expected inputs, this may not be a problem.
- But if you cannot be certain that the algorithm won't recurse so many times that it blows your stack, it may be better, in an embedded system, to rework your algorithm as an iterative one.
- An iterative algorithm has a known stack footprint, whereas a recursive one adds to the stack with each level of recursion.

Q4) Explain the concept of 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.
- 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.
- Similarly, if you're building something which doesn't have to react instantly 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.
- 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.
- 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, instead of having to regularly do busywork to check whether things have changed and it has real work to do.
- On the hardware side, look to use processor features such as comparators or hardware interrupts to wake up the processor and 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.
- If you can reduce the amount of data that you're processing, that helps too.
- 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.
- But if but the existing API doesn't provide that possibility, you always have the option of writing a service of your own to sit between the device and the proper API (known as a *shim service*).
- The shim service has all the processing power and storage available to a web server and so can do most of the heavy lifting.
- After this, it just sends the minimum amount of data across to be processed in your embedded system.
- For instance, Bubblino talks directly to Twitter and downloads a full set of search results as XML each time it checks for new messages.
- All this data is processed and finally reduced down to a single number—how many new tweets it finds.
- In theory, an optimised intermediary service could perform the searches and just transmit a single number to the Bubblino device.
- The downside to that is you would need to write another service and maintain infrastructure to support it.

Q5) list and explain habits that help make your code generally more efficient (to increase performance).

- **Following are a few habits that help make your code generally more efficient:**
- **1]W**hen 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
- This allows the pre-fetch and lookahead pipelining of instructions to work in the more common cases, rather than requiring the lookahead instructions to be discarded and the pipeline refilled.
- 2] Declaring data as constant where it is known never to change can help the compiler to place it into flash memory or ROM.
- In some scenarios it is quicker to insert a value into the code as a plain number rather than to load that value from a variable's location in memory somewhere, and if the compiler knows what the variable's value is always going to be, it can make that substitution.
- 3] Avoid copying memory around.
- Moving big chunks of data from place to place in memory can be a real performance killer.
- In an ideal world, your code would look only at the data it needed to and read it only once.
- This is particularly an issue in protocol work, where data is initially read into a packet buffer for the Ethernet layer, say, and then passed up to the IP layer, then the TCP layer, followed by the HTTP code, and finally the application layer.
- A naive approach would copy the data at each step, as you unpack the relevant protocol headers.
- This approach could result in all the application data being copied five times as it travelled up the stack.
- A better approach would be to have a pointer or a reference to the initial buffer passed from layer to layer instead.
- In addition to the length of the data, you may need to store an offset into the buffer to show where the relevant data starts, but that's a small increment in complexity to greatly reduce how much data is copied around.
- 4] When you do need to copy data around, the system's memory copying and moving routines (such as memcpy and memmove) usually do a more efficient job than you could, so use them.
- Where possible, use processor instructions that copy more than one byte in a single operation, thus considerably speeding up the process.

Q6)Why we need LIBRARIES? List few libraries available.

- 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.
- In the embedded world, tasks are often a little trickier. 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.
- The trickiest part is likely to be working out how to recompile a library for your target processor if a prebuilt version isn't readily available for your system
- On the other hand, microcontrollers are still too resource-constrained to just pull in mainstream-operating system libraries and code.
- You might be able to use the code as a starting point for writing your own version, but if it does lots of memory allocations or extensive processing its better finding one that's already written with microcontroller limitations in mind.
- Here are a few libraries that are available:
- ▪ **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.
- It does this by not using any buffers to store incoming packets or outgoing packets which haven't been acknowledged.

- ▪ **uClibc:** uClibc is a version of the standard GNU C library (glibc) targeted at embedded Linux systems.
- It requires far fewer resources than glibc.
- Changing code to use it normally just involves recompiling the source code.
- ▪ **Atomthreads:** Atomthreads is a lightweight real-time scheduler for embedded systems.
- You can use it when your code gets complicated enough that you need to have more than one thing happening at the same time.
- ▪ **BusyBox:** Although not really a library, BusyBox is a collection of a host of useful UNIX utilities into a single, small executable and a common and useful package to provide a simple shell environment and commands on your system.

Q7) Explain various methods of DEBUGGING software and hardware.

- 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 by introducing both custom electronic circuits (which could be misbehaving or incorrectly designed) and communication with servers across a network.
- Modern desktop integrated development environments (IDEs) have excellent support for digging into what is going on while your code is running.
- 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.
- The debugging environment for embedded systems is usually more primitive.
- Systems such as embedded Linux usually have support for remote debugging with utilities such as gdb, the GNU debugger.
- This utility allows you to attach the debugger from your desktop system to the embedded board, usually over a serial connection but sometimes also over an Ethernet or similar network link.
- Once it is attached, you then 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.
- 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 in the wild.
- 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.
- JTAG has been extended to provide more advanced debugging features such as features available when connected to some software on a separate PC called an *in-circuit emulator* (ICE).
- These allow you to use the additional computer to set breakpoints, single-step through the code running on the target processor, and often access registers and RAM too.
- Some systems even allow you to trigger the debugger from complex hardware events, which gives you even better control and access than debuggers such as gdb.
- 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.
- 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.
- And if your code appears to hang during execution, something as simple as some "reached line X" debug output.

Q8) what is Logging? Explain issues with logging.

- If you have access to a writable file system, for example, on an SD card, you can write the output to a file there, but it is more common to write the information to a serial port.
- This approach enables you to attach a serial monitor, such as HyperTerminal on Windows, to the other end of the connection and see what is being written in real time.
- Issues with this are:
- 1] the amount of space needed for any logging information: all the strings and code to do the logging have to fit into the embedded system along with your code.
- 2] As the serial communication runs at a strict tempo, typically a small buffer is used to store the outgoing data while it is being transmitted.
- If your code hangs or crashes very soon after your logging output, there's a chance that it won't be written out over the serial connection before the system halts.
- Because most Internet of Things devices have a persistent connection to the network, you could add a debugging service to the network interface.
- This way, you can create a simple service which lets you connect using something as basic as telnet and find out more about what is happening in real time.
- Actually, even without a dedicated debug interface on the network, you can use the fact that your device has network connectivity to help figure out what has gone wrong.
- TCP/IP stacks generally offer basic capabilities such as responding to ICMP ping requests, even if the higher-level code isn't doing what you expect.
- If you know the IP address of the device and it responds when you query it with the ping network utility, you can infer that at least some part of the system is still functioning.
- If you can connect a computer somewhere on the network path between the device and the service it communicates with, running a *packet sniffer* enables you to see what is happening at the network level.
- At a slightly higher level, you can also use the logging and software on the server to gather information about the device's activity.
- If the transport between the device and the server is a standard protocol, such as the HTTP communication with a web server, there is a good likelihood that any requests were recorded in the server log file.
- If all else fails, the debugging tool can be: 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.

IOT Ch-9
BUSINESS MODELS

- **Introduction:**
- Definition: It is “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 together 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

Q1) Explain the HISTORY OF BUSINESS MODELS.

- Gift economies develop where those with the appropriate skills can provide their products or services and expect repayment of this obligation not immediately but with a gift of comparable worth later.
- Modern business models are resulted from the technology required to move products and obligations through space and time.

- **SPACE AND TIME**
- While neighbouring tribes might have discovered variants in the local area’s resources 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.
- But long-distance trade brings with it a whole set of problems:
- merchants have to carry larger quantities of goods for sale
- Their goods and food carried will have to last far longer, so they will need to be protected and preserved.
- they need to have a reliable means of transport for themselves and their merchandise
- Preservation is also a way of transporting goods through *time*.
- A farmer or trader who can afford to not sell all his produce during the harvest can fetch a better price months later at a higher price.
- Money, then, abstracted trade further, setting an easy-to-calculate exchange rate between a fixed currency (a certain size disc of gold or weight of grain).
- Ease of calculation which this development brought with it made it easier to develop new business models, such as the development of interest on loans.

- **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.
- The invention laid the foundations for an information culture.
- Information is no longer so rare and valuable that it must be preserved by gatekeepers but can be so widely spread that everyone can have access to it.
- 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.
- 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.

- 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.

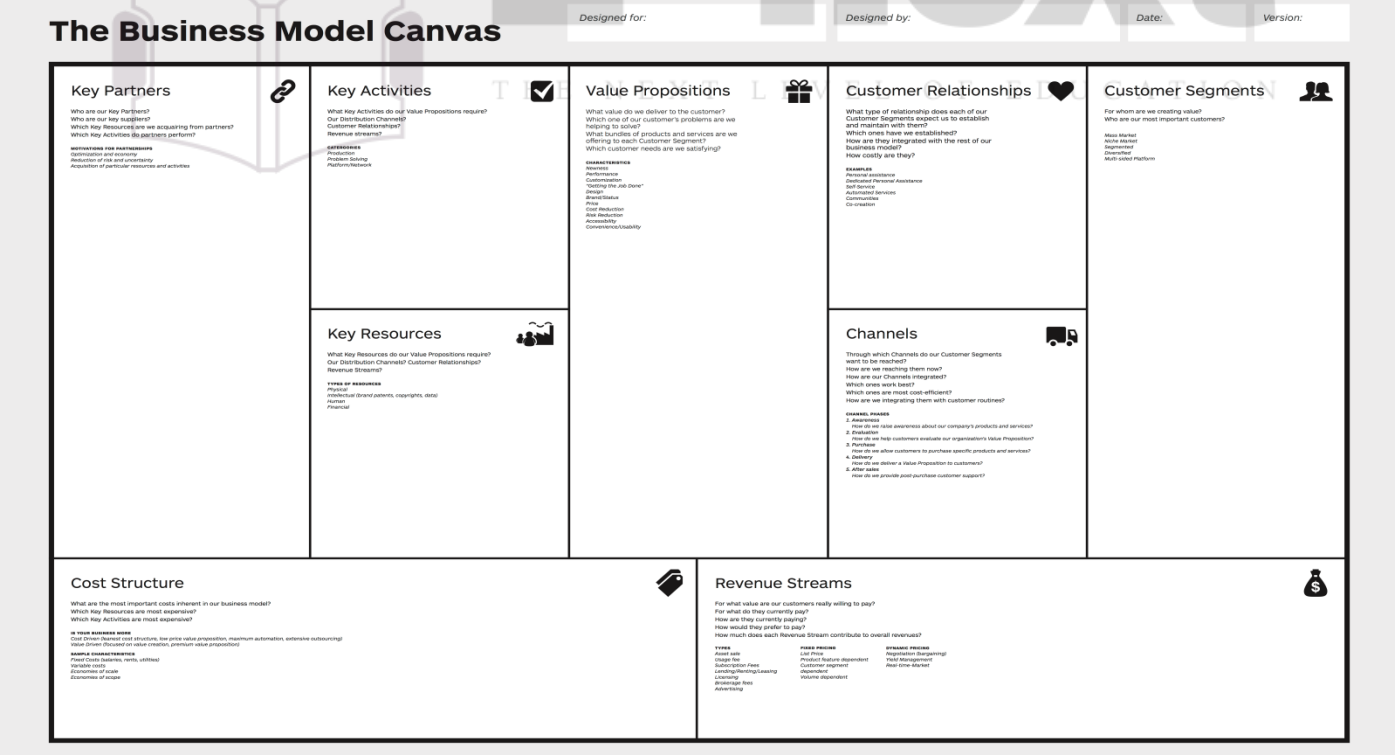
- **THE LONG TAIL OF THE INTERNET**
- From Tim Berners-Lee's first demonstration of the World Wide Web in 1990, it took only five years for eBay and Amazon to open up shop and emerge another five years later as not only survivors but victors of the dot-com bubble. Both companies changed the way we buy and sell things.
- A physical bricks 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.
- 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.
- Long tail Internet giants help this process by *aggregating* products from smaller providers, as with Amazon Marketplace or eBay's sellers.
- This 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.
- 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.

- **LEARNING FROM HISTORY**
- 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
- Second, we've seen how new technologies have inspired new business models.
- Third, although there are recurring patterns and common models, there are countless variations. 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 mass production changed the notion of work itself.

Q2) What is the use of THE BUSINESS MODEL CANVAS? Explain each template box.

- 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.
- 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.
- Let's look at the model, starting with the most obvious elements and then drilling down into the grittier details that we might neglect without this kind of template.
- At the bottom right, we have *Revenue Streams*, which is more or less the question of "how are you going to make money?"
- Although its position suggests that it is indeed one of the important desired *outputs* of the business.

- 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.
- The *Customer Relationships* might involve a lasting communication between the company and its most passionate customers via social media.
- Maintaining a “community” of your customers may be beneficial, but which relationships will you prioritise to keep communicating with your most valuable customer segments?
- *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.
- Few companies can afford the investment in time and money to do all the Key Activities themselves or even marshal all the Key Resources.
- 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.
- Perhaps you will get an agency to do your web design and use a global logistics firm to do your deliveries.
- The *Cost Structure* requires you to put a price on the resources and activities you just defined.
- Helps you determine whether you will be more cost driven (sell cheaply, and in great volume via automation and efficiency) or more value driven (sell a premium product at higher margins, but in smaller quantities).



Q3) 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.
- the canvas 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?
- 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.
- Your *customers* will also be considering whether to invest their time and money in your product.
- They will ask themselves certain questions about it.
- some of these likely questions are:
- ▪ **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.
- ▪ **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.**
- ▪ **Your online document collaboration looks great, but is it worth 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 and might prefer a different one.
- Not all customers vote for the free option.
- 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 are:
- ▪ 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.
- Powazek suggests that the actual lesson to be learned is that:
- *Your business plan cannot be secret anymore. People are too smart for that, too tired of getting burned, too wary of losing their contributions when a startup dies, and too annoyed by sudden changes to the terms. Communicate your business plan from the start and you'll avoid a thousand problems down the road.*

Q4) Explain some of the models that Internet of Things companies have used or might use.

MAKE THING, SELL THING

- Adrian sells custom-built Bubblini, and the startup Good Night Lamp is preparing to ramp up production of its eponymous lamps as an off-the-shelf product.
- Many small-scale projects take the option of selling the product in "kit" form, with some assembly required.
- Because kits are assumed to be for specialists and hobbyists rather than the general public, the administrative burden may be lower.
- However, making a decision to limit your target market may well limit the potential revenue also.

SUBSCRIPTIONS

- An 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.
- In the future, content publishers may pay for certain premium services.
- 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.
- In so-called freemium model (“free” and “premium”) 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.
- This model could be combined with our first two models:
- Buying the physical device gives free lifetime access to the associated Internet service, but additional paid services are also available.

CUSTOMISATION

- For a mass-produced item, any customisation must be strictly bounded to a defined menu: a selection of different colours for the paintwork.
- The websites Facebook, Twitter etc offer small degrees of customisation within strictly defined boundaries: a selection of (tasteful) colour schemes and background of your choice.
- 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.
- The new manufacturing techniques, such as laser cutting and 3D printing, should allow great possibilities for customising even the physical devices.
- MakieLab (http://makie.me) make dolls that can be designed online.

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.
- Small companies such as Adafruit sell electronic components to hobbyist makers.
- On the consultancy side, work will be available either simply providing your skills for hire or indeed in providing vision and expertise for strategic planning to a company that wants to engage with the Internet of Things.
- Environmental data consultancy amee (www.amee.com) 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.

PROVIDE INFRASTRUCTURE: SENSOR NETWORKS

- Sensor data is a fascinating topic in the Internet of Things: There are official data sources which are expensive to create and hard to access and they 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.
- They allow any consumer to upload a real-time feed of sensor data.
- Xively was intended to provide a free, public infrastructure for open source data while also providing enhanced commercial offerings with enhanced capacity and privacy options and formal service level agreements (SLAs).

- Sensor data is information, which can be shared freely or might simply be sold. Many energy suppliers are rolling out “smart meters”, which promise greater efficiency and therefore cheaper bills but also aggregate huge quantities of information.
- As regards the business model, you need to consider the legality of such collection and whether it fits with your company values.

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.
- You could also link devices to advertising to reduce the price.
- 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.)
- Perhaps future versions of Bubblino could also be triggered for occasional promoted tweets.
- Perhaps your Internet-enabled fridge will make tutting noises when you fill it and suggest other (promoted) options for your next shop.

Q5) List and explain the possible ways to get funds for AN INTERNET OF THINGS STARTUP?

- 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.
- If you have enough personal money to concentrate on your new Internet of Things startup full time, you can fund your business yourself.
- For others there are still ways to kick off a project.
- 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.
- Many people try to combine a startup with a consultancy business, planning to take short contracts which support the following period of frantic startup work.
- 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.

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.
- Yes, you can probably fork the project and continue to work on it in secret, but the existing project may carry on if your collaborators are enthusiastic enough about it.
- 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.
- 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 get 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

VENTURE CAPITAL

- Getting funding for a project from an external investor presents its own work and risks.
- 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.
- 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.
- Although it's important to consider the possible impact on your personal relationships, this round of funding may be the most straightforward to get hold of.
- 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.
- 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.
- 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.
- Angels typically disburse sums that are significant for early-stage startups—in the region of tens or possibly hundreds of thousands of pounds.
- 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.
- 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.
- Typically, VC funding will be larger chunks of money, from half a million pounds up.
- In the early stages, would be an *accelerator*, which might be run by a venture capital firm.
- In this case, part or all of the money that could be awarded to your company is paid in kind, in the form of free office space, consultancy, and specific training and mentoring in areas that the investor believes will make you a success.

- 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".
- 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.

GOVERNMENT FUNDING

- Governments typically want to promote industry and technological development in their country, and they may provide funds to help achieve particular aims.
- Although governments can and do set up their own venture capital funds or collaborate with existing funds in various ways, they generally manage the majority of their funds differently.
- For one thing, they also want to fund *existing* companies to do new research and innovation.
- The money provided still has "strings attached", but they are likely to be handled differently:
- ▪ **Outputs:** Deliverables 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 simply be a test that you are managing the money well or may be related to the goals that the body itself wishes
- to promote.
- 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 may be required to *match funds*; that is, if you were awarded L10,000, you would also have to raise L10,000 yourself
- ▪ **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.
- Governments will, however, try to split their pot of money to fund the outcomes that they are interested in as policy.
- Quite reasonably, this may tend to favour grants for research over grants to help get to market.
- After all, after the product is proven, the company should be able to afford to fund it by itself or get VC funding.
- It is perfectly normal for companies to work through multiple sources of funding.

CROWDFUNDING

- 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.
- Over millennia many civic and religious monuments and constructions have been funded at least partly by the public.
- However, such projects have been mostly sponsored and given focus by some influential person or body.
- 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".
- Appealing text, slick videos, and great design may make the difference between yours and a competing project.
- Because even some successfully funded projects may fail, older and wiser crowdfunders may be more likely to fund projects in which they see some attention to the business model or a track record for successful completion by the project team.
- Our funders are real people and will have all the variety of concerns 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 know the customers interest in your product.
- Assuming that the aggregators (Kickstarter, Indiegogo, and the rest) are doing their job well, they will reach a good segment of the potential customers for your Internet of Things 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.

Q6) what are LEAN STARTUPS?

- 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)
- 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.
- These simple pages allow you to propose many projects and focus only on the ones that have most feedback.
- 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.
- 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.
- 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.
- But adding those enhancements to an incomplete prototype would not result in a working business model.
- 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.
- You can tweak in response to the feedback you get from iterating your product in the real world.
- 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.
- ▪ **Technology pivot:**
- Accomplish the same goals as before, but change the implementation details.
- This pivot would be a business decision, made to improve manufacturing costs, speed, or quality.

IOT
ch-10
MOVING TO MANUFACTURE

1) What are the different possibilities that we should consider before producing a Product. Also Explain Designing KITS in detail.

- Depending on your motives and desires for the product, and the amount of time and money that you have available to devote to it, a full spectrum of possibilities is open to you.
- If your ambition is just to enable others to build a version of it for themselves, you only need to document the build process and share it, along with any source code and design files you used, on the Internet—either on your own blog or website, or on a site like Instructables.
- When you decide that you want to get your invention into the hands of many more people way), you can generally split your choices into three categories:
- 1 ▪ A kit that your customers can assemble themselves
- 2 ▪ A ready-made electronics board which users can use as a sub-assembly in their own projects
- 3 ▪ A fully fledged product, complete with its housing, instructions, and even packaging, just waiting to be put on the shelf at your local department store.

- **DESIGNING KITS**
- Many of those people would jump at the chance to buy a kit in which someone else has done the hard (to them) part of selecting and sourcing the right components and putting them together with a step-by-step guide.
- Most kits tend to provide only the electronics and software for a particular application rather than any physical housing.
- Kits tend to piggyback on existing microcontrollers and often take the form of a standard format plug-on board—for example, a *shield* in the Arduino.
- Given that you've built your prototype, you've already done most of the work needed to create a kit.
- You've worked out which components you need, where to source them, and how to wire them up to create a functioning circuit.
- And you've written the necessary software to interface to and control the electronics.
- The parts you might not have done include designing a PCB, documenting the build process, and working out the costs.
- You can make the PCBs a few different ways after they are designed.
- Documenting the build process isn't too tricky.
- When you have everything together for one of your kits, run through assembling one of them yourself.
- As you go, take photos or video of each step and write down the order of each step.
- Working out what price to charge is harder.
- Some rules of thumb can help, though.
- The most obvious is to understand what your costs are.
- You should create a spreadsheet (or at least a list) of all the items that make up your product, along with their cost to you to buy.
- You should list every single electronic component, connector, cable, PCB, case, and so on, in addition to the packing box you'll ship it in and the time taken to put things together.

- This list is called the *bill of materials* (BOM), and it forms the starting point for all your costings and prices.
- A good starting point for working out your price (how much you will charge consumers) is to take the total cost of the BOM and multiply it by 4 or 5.
- That calculation gives you a margin to cover that item's portion of the fixed costs and also some profit.
- It also provides enough margin for resellers to cover *their* fixed costs and make some profit.
- The final step from kit to consumer product is to manufacture and assemble the housings, linkages, and whatever else is part of the finished device.

Q2) How to design PRINTED CIRCUIT BOARDS? Explain software choices for it.

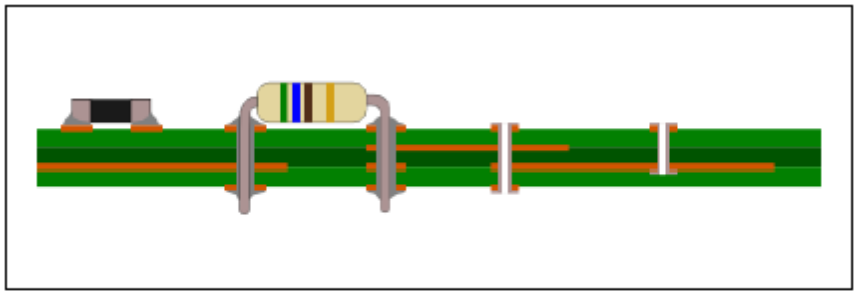
- Soldering things up is a good step towards making your prototype more robust.
- After you've done this, you should have something that will survive being given to an end user.
- You will soon get fed up with soldering each item by hand.
- Moving beyond stripboard, designing and etching your own custom PCBs gives you more options on how to lay out the circuit and makes it easier to solder up as the only holes in the board will be those for components.
- Homemade boards will still lack that fully professional finish.
- That's because they won't have the green solder mask or silkscreen layers that a PCB manufacturer will give you.
- Moving to professionally manufactured boards further simplifies the assembly process because the solder mask will make the soldering a bit easier, and, more importantly, the silkscreen provides outlines of where each component should be placed.

- SOFTWARE CHOICES
- You have many different choices when looking for some software to help you design your PCB.
- If you are working with a contract electronics design house, the staff may well use something like Altium Designer.
- But you're more likely to use one of the following lower-end (and cheaper) options.
- Fritzing (http://fritzing.org) is a free, open source design package aimed particularly at beginners in PCB design.
- It deliberately starts with a design screen resembling a breadboard and lets you map out your circuit by copying whatever you have prototyped in real life.
- It then converts that design to a schematic circuit diagram and lets you arrange components and route the traces on the PCB view.
- When you are happy with your design, you can export it for manufacture as a PCB.
- Fritzing even offers a fabrication service to make it simple to make your design a reality.
- KiCad (www.kicad-pcb.org) is another open source offering but with a more traditional workflow.
- It has a more comprehensive library of predefined parts and can be used to design boards with up to 16 layers of copper, compared to the double-sided boards that Fritzing produces.
- In addition to letting you see what the finished board will look like, it also enables you to export the 3D model. Doing so enables you to import it into your CAD software when designing the enclosure to ensure that the relevant clearances are allowed for all the components.

- Probably the most popular PCB layout software for the hobbyist and semi-professional market is EAGLE from CadSoft.
- The reason for its popularity most likely comes down to its long having a free version for non-commercial use, allowing beginners to get started.
- Although in addition to its noncommercial licence, the free version is also restricted to two layers and a maximum board size of 100mm × 80mm.
- A more recent rival to EAGLE in the commercial pro-sumer market is DesignSpark PCB.
- Unlike EAGLE, DesignSpark PCB does not restrict the size of boards you can design (up to 1m2) or the number of layers (it supports up to 14 layers).
- However, it is the only program described here which isn't available for Linux or Mac.

Q3)What makes up a PCB?

- The PCB is made up of a number of layers of fibreglass and copper, sandwiched together into the board.
- The fibreglass provides the main basis for the board but also acts as an insulator between the different layers of copper, and the copper forms the "wires" to connect the components in the circuit together.
- Given that you won't want to connect all the components to each other at the same time, which would happen if you had a solid plate of copper across the whole board, sections of the copper are etched away—usually chemically,
- But it is possible to use a CNC mill for simple boards.
- These remaining copper routes are called *tracks* or *traces* and make the required connections between the components.
- The points on the tracks where they join the leg of a component are known as *pads*.
- Single-sided boards have only one layer of copper, usually on the bottom of the board; because they're often for home-made circuits with through-hole components, the components go on the top with their legs poking through the board and soldered on the underside.
- Double-sided boards have two layers of copper: one on the top and one on the bottom.
- More complicated circuits may need even more layers to allow room to route all the traces round to where they are needed.
- Additional layers require a more complex manufacturing procedure in which alternating layers of copper and fibreglass are built up, a bit like you would make a sandwich.
- When the holes drilled through the board are *plated*—a process in which the walls of the holes are coated in
- A thin layer of copper—any layers with copper at that point are connected together.
- When you need to connect traces on two layers together at a point where there isn't a hole for the leg of a component, you use a *via*.
- It is smaller, hole through the board purely to connect different layers of copper once plated.
- You also can use blind vias, which don't pierce all the way through the board , when you don't want to connect every layer together.



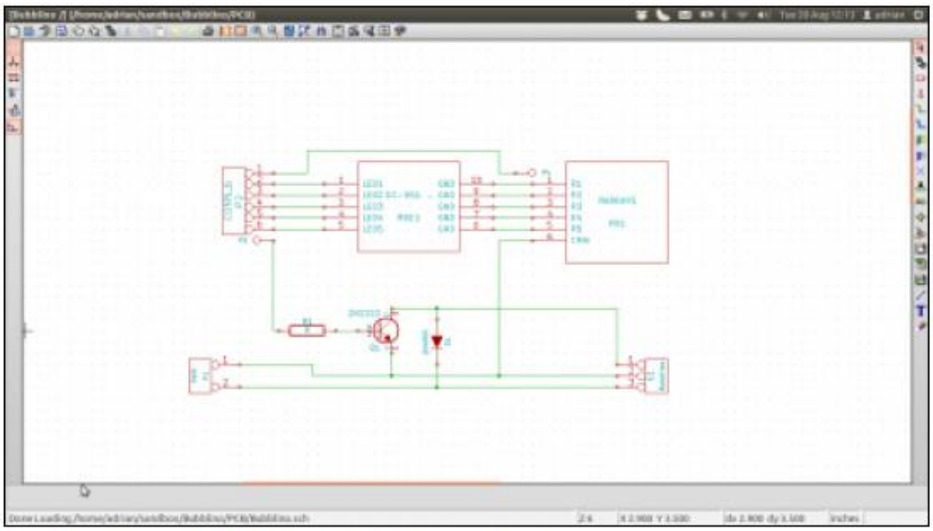
PCB features (left to right): surface-mount component; through-hole component; via; blind via.

- In places where you have many connections to a common point, rather than run lots of tracks across the circuit, you can more easily devote most of a layer of the board to that connection and just leave it as a filled area of copper.
- This is known as a *plane* rather than a trace and is frequently used to provide a route to ground.

- The surfaces of professionally manufactured PCBs undergo processes to apply two other finishes which make them easier to use.
- First, all the parts of the board and bare copper which aren't the places where component legs need to be soldered are covered in solder mask.
- Solder mask is most commonly green.
- The mask provides a solder-resistant area, encouraging the solder to flow away from those areas and to adhere instead to the places where it is needed to connect the components to the tracks.
- This reduces the likelihood of a solder joint accidentally bridging across two points in the circuit where it shouldn't.
- Then, on top of the solder mask is the silkscreen.
- This is a surface finish of paint applied, as the name suggests, via silkscreen printing.
- It is used to mark out where the components go and label the positions for easy identification of parts.
- It generally also includes details such as the company or person who designed the board, a name or label to describe what it is for, and the date of manufacture or revision number of the design.

Q4) Explain THE DESIGN PROCESS for PCB.

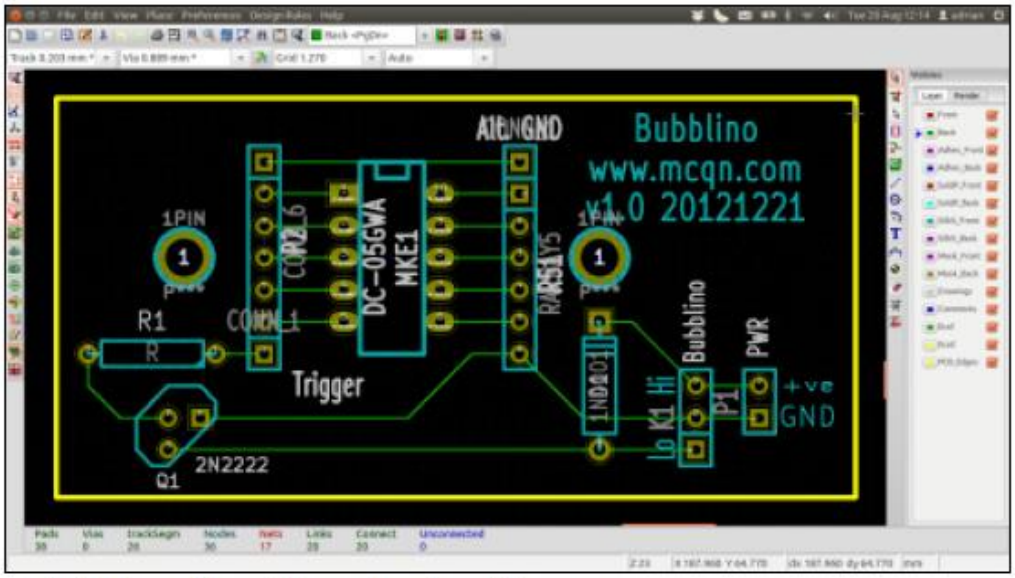
- Regardless of the exact program you decide to use to lay out your PCB, your task in creating the design is split between two main views: the schematic and the board.
- **The Schematic**
- You usually start the design in the schematic view. It lets you lay out the components logically and make the necessary connections without having to worry about exactly where they'll sit in physical space or whether any of the tracks cross.
- The schematic provides a conceptual view of how the circuit is laid out.
- Components are represented by their standardised symbols, and you usually group them into areas of common functionality instead, which won't necessarily match the groupings they'll have in the physical layout.
- Your software package includes most of the common items you need in its library: common resistors, diodes, capacitors, transistors, integrated circuits (ICs), and more.



The schematic view of the Bubblino PCB, as designed in KiCad.

- You should also make sure you choose the correct *package* for the component; this is the physical format for it. Many parts are available in a range of different formats, depending on the target application.
- Making the right choices when initially choosing the component will save you having to rework that down the line
- When you come to the board view.
- When you are happy with the schematic design of your board, you can proceed to laying out the physical board.
- **The Board**
- There are no hard-and-fast rules on how to arrange the board.
- It makes sense to keep together groups of components that constitute an area of functionality, such as the power supply, for a more logical layout.
- The design has certain fixed aspects—things such as connectors which all need to be along one side for access once in a case, for example, or arrangements of headers to mate with an Arduino board.
- Start by placing the components which have the most connections first and then fitting the remainder around those constraints.
- As you position the components, you should try to reduce how tangled they are (rotating or moving them around for the best combination).
- The thin, straight-line connections from point-to-point are known as *air wires*.
- They show where the connections need to be made but won't appear in the finished PCB design until you turn them into real tracks on the PCB.
- Each one needs to be turned into a track on the PCB and routed round the board so that it makes the connection without crossing any of the other tracks or running too close to the pads of other components.
- By finding enough room for the tracks and stopping them from crossing, you can make the different layers of the PCB come into play.
- Placing tracks on different layers of the board means that they won't make electrical contact when they cross, and for more complex designs you might run one track on one layer for part of its path and switch it to another layer with a via to continue the rest of its route.
- Your PCB software has an auto-route function, which you can use to route all the tracks for you.
- But in practice, you will find it best to at least lay out the more important tracks first by hand.
- After all the tracks have been routed, your PCB design is almost finished.
- You should add any labels to the silkscreen layer to explain things.

- Also include a version number so that you can differentiate any future revisions, and also add maybe your name or the company name and a website for people to find out more.



The finished view of the Bubblino PCB board, ready for export.

- Then give your PCB a final once-over.
- Run the design rules check to catch any missed connections or problems where tracks are too close to each other or too thin for your PCB manufacturer to reliably manufacture.
- The design rules effectively contain the manufacturing tolerances for your PCB manufacturer.
- They define things like the minimum track width or the minimum distance between pads.
- Then print out a copy on paper; this way, you can compare the components to their location on the PCB in real life and spot any errors in their footprints.
- After you fix any issues thrown up by those last checks, you're ready to make the PCBs.

Q5)List and explain possible ways of Manufacturing PCB.

- If you want only a couple of boards, or you would like to test a couple of boards (a very wise move) before ordering a few hundred or a few thousand, you may decide to make them in-house.
- **ETCHING BOARDS**
- The most common PCB-making technique for home use is to etch the board.
- Some readily available kits provide all you need.
- The first step is to get the PCB design onto the board to be etched.
- This process generally involves printing out the design from your PCB design software onto a stencil.
- If you're using photo-resist board, it will be onto a stencil which masks off the relevant areas when you expose it to UV light.
- Your stencil then needs to be transferred to the board.
- For photo-resist board, you will expose it under a bright lamp for a few minutes.
- With the board suitably prepared, you can immerse it into the etching solution, where its acidic make-up eats away the exposed copper, leaving the tracks behind.
- After all the unnecessary copper has been etched away, and you've removed the board from the etching bath and cleaned off any remaining etchant, your board is almost ready for use.

- The last step is to drill the holes for any mounting points or through-hole components.
- **MILLING BOARDS**
- In addition to using a CNC mill to drill the holes in your PCB, you can also use it to route out the copper from around the tracks themselves.
- To do this, you need to export the copper layers from your PCB software as Gerber files.
- These were first defined by Gerber Systems Corp., hence the name, and are now the industry standard format used to describe PCBs in manufacture.
- To translate your Gerber file into the G-code that your mill needs requires another piece of software.
- Some CNC mills come with that software already provided, or you can use a third-party program such as Line Grinder.
- The mill effectively cuts a path round the perimeter of each track to isolate it from the rest of the copper.
- As a result, PCBs which have been milled look a bit different from those which are etched because any large areas of copper that aren't connected to anything are left on the board.
- **THIRD-PARTY MANUFACTURING**
- If your design has more than two layers, if you want a more professional finish, or if you just don't want to go to the trouble of making the PCBs yourself, many companies can manufacture the boards for you.
- The price for getting the boards made varies based on the complexity and the size of the design but also varies quite a bit from company to company.
- If you need the boards quickly, a local firm is best.
- If you have more time you can give it outside country such as china, it might reduce cost.
- Either way, the Gerber files are what you need to provide to the manufacturer.
- Make sure you export all the relevant layers from your design, meaning each of the copper layers you're using, plus the solder mask, silkscreen and drill files.

Q6)Explain the concept of ASSEMBLY and testing with respect to PCBs.

- **ASSEMBLY**
- After your PCBs have been manufactured, you still need to get the components soldered onto them.
- If you're selling them as kits, the customers will solder things up, so you just need to pack everything into bags and let them get on with it.
- Otherwise, you have to take responsibility for making that happen.
- For small runs, you can solder them by hand.
- For through-hole boards, break out your soldering iron.
- For assembling surface-mount boards, you need one more item from your PCB design Gerber collection: the solder paste layer.
- You use it to generate a stencil that allows you to apply the solder.
- You can laser-cut one from a thin sheet of Mylar plastic or have one made for you out of thin steel.
- When you have all the components on the board, you need to melt the solder to fix everything in place.
- You can do this with a soldering iron, but doing it by hand is easier if you use a hot-air rework station.
- It uses hot air to provide the necessary heat.
- You can solder all the connections at once if you use a reflow oven.

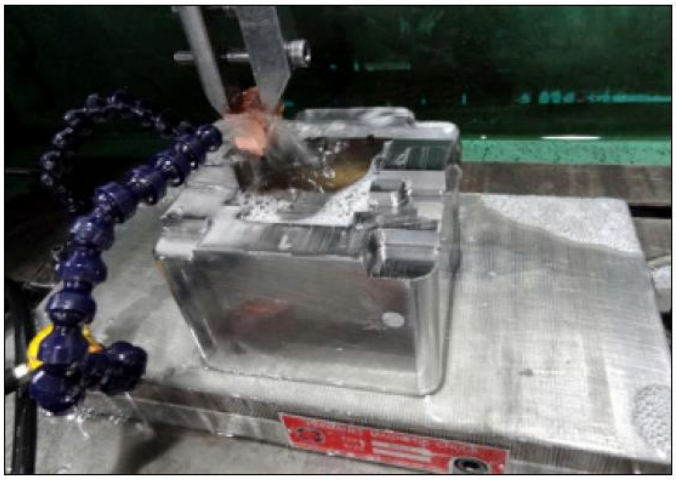
- As the name suggests, this oven heats up the PCB and components evenly until the solder melts.
- Professional reflow ovens allow you to set different temperature profiles, allowing you to match the specification in the manufacturers' datasheets for more exacting components.
- After you outgrow hand assembly, you will need some help from robots.
- In this case, you will need robots that can pick up components using a tiny vacuum nozzle, rotate and place them in the right location on the PCB, and then repeat that process at a rate of tens of thousands of components per hour. These robots are known as *pick-and-place assembly machines*.
- It is worth to offload some of the work to an *assembly house*, also known as a *contract manufacturer*.
- Contract manufacturers are firms geared up to helping people produce finished, populated PCBs.
- Often they offer a range of services from PCB design, through dealing with PCB manufacturers, to soldering up the components and even testing the completed boards
- Using an assembly house saves you from buying the expensive machinery yourself.
- If you do decide to use a contract manufacturer, having a conversation about components is worthwhile.
- For common parts, if you don't have specific requirements for tolerances, and the like, you might be able to specify that the assembler should use the parts it already holds in stock.
- That might mean you don't have to buy an entire reel just to get a few components.
- Even for the parts the contract manufacturer doesn't already carry and that you need to buy in, it may make sense to work with the manufacturer to place the order.
- If the manufacturer has dealt with the supplier before, its reputation might let you negotiate a better deal.

- **TESTING**
- Now your boards are all ready and assembled, but how do you know that they all work as they're meant to?
- This is where testing comes in.
- A better approach is to build a specific test rig to exercise the different parts of the circuit and measure the voltages at set points on the board.
- These measurements can then be compared against known-good values and a decision made automatically as to whether or not the device under test (DUT) has passed.
- Because you don't want to spend time making individual connections for each test, the normal practice for the test rig is to use the mounting holes for the PCB for alignment and then have it held by some clips against a number of carefully prepositioned, spring-loaded pins.
- These pins are known as *pogo pins*, and the spring means they can make a good connection to the board without any extra work, such as soldering, when the board is placed into the test rig.
- The test program can then run through its tests and measure voltages at different pogo pins at the relevant time in the test to see how the board being tested performs.

- If the DUT includes a microcontroller chip, the test rig could flash it with a test program to help with the testing, and even at the end of the test, assuming it has passed, flash it with the final firmware image ready for deployment.

Q7) How to go for MASS-PRODUCING THE CASE AND OTHER FIXTURES

- A good rule of thumb for keeping down the costs of production is to minimise the amount of time a person has to work on each item.
- Machines tend to be cheaper than people, and the smaller the proportion of labour is in your costs, the more you'll be able to afford to pay a decent wage to the people who *are* involved in assembling your devices.
- Production rates are also important.
- Though they're fairly labour free, 3D printers and laser cutters aren't the fastest of production techniques.
- Waiting a couple of hours for a print is fine if you just want one, but a production run of a thousand is either going to take a very long time or require a *lot* of 3D printers.
- We look at what must be the most common method of mass production: injection moulding of plastic.
- As the name suggests, the process involves injecting molten plastic into a mould to form the desired shape.
- After the plastic has cooled sufficiently, the mould is separated and the part is popped out by a number of ejection pins and falls into a collection bin.
- The whole cycle is automated and takes much less time than a 3D print, which means that thousands of parts can be easily churned out at a low cost per part.
- The expensive part of injection moulding is producing the mould in the first place; this is known as *tooling up*.
- Often for a super-smooth surface, the moulds are finished with a process called *electrical discharge machining* (EDM), which uses high-voltage sparks to vaporise the surface of the metal and gives a highly polished result.
- The moulds are machined out of steel or aluminium and must be carefully designed and polished so that the moulding process works well and the parts come out with the desired surface finish.
- The mould also needs to include space for the ejection pins to remove the part after it's made and a route for the plastic to flow into the mould.
- Because the parts need to be extracted from the mould after they're formed, very sharp corners and completely vertical faces are best avoided.
- A slight angle, called the *draft*, from vertical allows for clean parting of the part and its mould, and consistent wall thicknesses avoid warping of the part as it cools.
- The simplest moulds are called *straight-pull* and consist of the mould split into two halves.
- If your design needs to include vertical faces or complex overhangs, more complicated moulds which bring in additional pieces from the side are possible but add to the tooling-up cost.
- One way to reduce the tooling-up costs and also increase the production rate is to mould more than one part at a time.
- In a process known as *multishot moulding*, you can even share parts of different colours on the same mould.
- With carefully measured volumes for each part, one of the colours of plastic is injected first to fill the parts which need to be that colour.
- Then the other colour is injected to fill the remainder of the mould.



One of the moulds for BERG's Little Printer being finished on an EDM machine.

Q8)Explain Manufacturing Process with case study of : BERG's little printer

- The Little Printer, made by London design firm BERG, is a delightful, tiny Internet connected Printer.
- In 2006, Matt Webb, one of the founders of BERG, shared some thinking-out-loud about how the printer connected to his computer could be shared with close friends and family regardless of where they are in the world (http://berglondon.
- com/blog/2006/10/06/my-printer-my-social-letterbox/).
- He called this a social letterbox, with the idea being that people he was close to socially would be able to share printed things with him even when they weren't close to him physically.
- In the comments to his initial blog post, there's a remark from a certain Adrian McEwen, suggesting that a better approach than repurposing the printer connected to your computer would be to give the printer its own network connection and intelligence.
- This initial contemplation led to a number of people, including Matt, some of his friends, and the wider maker community, experimenting with repurposed receipt printers hooked up to Arduino boards and connected to the Internet.
- They experimented with printing out the weather forecast, their appointments for the day, tweets from friends, and much more.
- From those early investigations, they worked through what made sense when printed out and what did not.
- These experiments continued over a few years, operating almost as a background project for BERG as they engaged in a combination of client work and self-directed products.
- To bring Little Printer into existence the physical form required careful design of the printer mechanism and tooling for both injection moulding and metal pressing to provide the housing.
- The team ran into issues with the way that the injection-moulded plastic cooled around the holes for the button and status light on the top of the case: it introduced a ripple on the surface.
- Solving this problem resulted in a two-stage process for that part of the case. It is now injection-moulded without the holes, which gives the correct finish, and then drilled to provide the holes.
- On the electronics and software side, they wanted to minimise the amount of setup required and to be able to control the look and feel of the whole user experience, from ordering the printer through to using it in the home.

- This led them to choose a ZigBee-based wireless option which didn't require any username or password to pair with other endpoints.
- However, while home routers support WiFi, they don't have ZigBee support built in.
- This dictated a two-box solution: in addition to your Little Printer, you receive a BERG Cloud Bridge, a box with both ZigBee wireless and Ethernet which you plug into an Ethernet port on your network to talk to the Internet.
- That allows the printer to communicate with the BERG Cloud server software, which provides for both the delivery of "publications" to the printers and the user-facing website.
- Through the user interface of their website, BERG allows users to register their Little Printer and to choose which publications they want to subscribe to and when they should be delivered.
- Details of all this work broke cover in November 2011.
- By August 2012, things were nearing completion, and they had started taking pre-orders.
- A combination of the switching frequency of their switch-mode power supply and the PCB design for the Bridge units conspired to make it fail the electromagnetic compatibility emissions test.
- Solving that problem caused the schedule to slip by more than a month, and then a smaller issue in assembly with the magnetic catches on the case delayed things by another week or so.
- At the end of November 2012, these playful Little Printers started winging their way out to customers, bringing more Internet of Things goodness into numerous homes.

Q9) Discuss CERTIFICATION issues with IOT products.

- One of the less obvious sides of creating an Internet of Things product is the issue of certification.
- These regulations are there for good reason.
- They make the products you use day in, day out, safer for you to use; make sure that they work properly with complementary products from other suppliers; and ensure that one product doesn't emit lots of unwanted electromagnetic radiation and interfere with the correct operation of other devices nearby.
- The regulations that your device needs to pass vary depending on its exact functionality, target market (consumer, industrial, and so on), and the countries in which you expect to sell it.
- The best approach is to work with a local testing facility.
- They not only are able to perform the tests for you but also are able to advise on which sets of regulations your device falls under and how they vary from country to country.
- Such a testing facility subjects your device to lots of tests.
- Testers check over the materials specifications to ensure you're not using paint containing lead; zap it with an 8KV static shock of electricity to see how it copes; subject it to probing with a hot wire—heated to 500 degrees Celsius—to check that it doesn't go up in flames; and much more.
- Of particular interest is the electromagnetic compatibility, or EMC, testing.
- This tests both how susceptible your device is to interference from other electronic devices, power surges on the main's electricity supply, and so on, and how much electromagnetic interference your product itself emits.
- Electromagnetic interference is the "electrical noise" generated by the changing electrical currents in circuitry.

- When generated intentionally, it can be very useful: radio and television broadcasts use the phenomenon to transmit a signal across great distances, as do mobile phone networks and any other radio communication systems such as WiFi and ZigBee.
- The problem arises when a circuit emits a sufficiently strong signal *unintentionally* which disrupts the desired radio frequencies.
- In addition to the test report, you need to gather together PCB layouts, assembly certificates, the certificates for any precertified modules that you have used, and datasheets for critical components.
- This information is all held in a safe place by the manufacturer (that is, you) in case the authorities need to inspect it.
- The location of the technical file is mentioned on the declaration of conformity, which is where you publicly declare to which directives in the regulations your device conforms.
- In Europe, you must also register for the Waste Electrical and Electronic Equipment Directive (WEEE Directive).
- It doesn't cover any of the technical aspects of products but is aimed instead at reducing the amount of electronic waste that goes to landfill.
- Each country in the EU has set up a scheme for producers and retailers of electronic and electrical products to encourage more recycling of said items and to contribute towards the cost of doing so.

Q10) Which things affect the COST of PCB?

- You have many things to consider as you move to higher volume manufacturing.
- Unfortunately, lots of them involve sizeable up-front costs.
- In fact, the further you get into the process, the less you will need your hardcore coding or critical design skills, and the more time you'll spend balancing cash flow and fund-raising.
- If you don't have much experience with managing delivery of inter-related tasks and deadlines, it is worth seeking out a trusted advisor or partner to help.
- *Someone* on the team needs to keep an eye on how things are progressing; which things must be done before other tasks can proceed; and, peering further into the project timeline, spot (and deal with) problems before they become a roadblock for the rest of the team.
- Many of the online PCB services include a quoting tool, so even before the design is finished, you can get a feel for the likely price.
- You should be able to make a reasonable guess on the size of PCB you need and the number of layers it's likely to require; those are the main factors that the quoting tools use to work out the cost, plus extras if you want something out of the ordinary such as a different coloured solder mask.
- For assembling the PCBs, you should budget something around €150–€800 for setup costs—getting the solder paste stencil made, programming the pick-and-place machine, and so on—and then between 3 and 14 pence for placing each component.
- This doesn't include the cost of the components themselves.
- Lawrence Archard, a hardware engineer with much more experience of going to manufacturing suggests that after you know the price for a few hundred of a component, each time you increase quantity by a factor of 10, the item cost will drop to three-quarters of the price.
- Pricing the plastics or other physical components depends far too much on the design for us to be able to give meaningful numbers here, but getting a hardened

steel tool made, which will be good for churning out 100,000 parts before it needs replacing, could easily run to €10,000.
- And to get through certification, you are likely to need a similar amount again.
- Simpler devices being certified in fewer territories might get through certification for around €2,000; more complicated designs, particularly those involving uncertified RF modules, could see costs 10 times that amount (€20,000) to get all the certifications in place.
- Naturally, as the project progresses, you will get more accurate quotes (and eventually completely accurate invoices…) as you talk to suppliers and get ready to place orders with them.
- This will let you update your BOM and cost spreadsheets and have the new information ripple through your plans.

Q11) How to SCALE UP SOFTWARE? Also explain various factors that require polish.

- Software is intangible and malleable.
- There are no parts to order, no Bill of Materials to pay for.
- The bits of information that make up the programs which run in the device or on the Internet are invisible.
- The software you wrote during project development and what ends up in production will be indistinguishable to the naked eye.
- Software has to be polished before it can be exposed to the real world.

- **The various factors that require this polish: correctness, maintainability, security, performance, and community.**

- **DEPLOYMENT**
- Copying software from a development machine to where it will be run from in production is typically known as *deployment*, or sometimes, by analogy to physical products, *shipping*.
- In the case of the firmware running on the microcontroller, this will (hopefully) be done once only, during the manufacture of the device.
- For a simple device like the Arduino which usually only runs a single program, the process will be identical to that in development.
- For a device like a BeagleBone or Raspberry Pi which runs an entire operating system, you will want to "package" the various program code, libraries, and other files you have used in a way that you can quickly and reliably make a new build—that is, install all of it onto a new board with a single command.
- The software for an online service will tend to run in a single location.
- However, it will generally be visible to the whole of the Internet, which means that it will potentially be subject to greater load and to a greater range of accidental or malicious inputs that might cause it to stop working.
- It is necessary, to update the online software components more regularly than those on the device.
- Having a good deployment process will allow you to do this smoothly and safely.
- Ideally, with one trigger (such as running a script, or pushing code to a release branch in your code repository), a series of actions will take place which update the software on the server.

- **CORRECTNESS AND MAINTAINABILITY**
- So, you've sold a thousand Internet-connected coffee machines. Congratulations!

- Now it's time to cross your fingers and hope that you don't get a thousand complaints that it doesn't actually work. Perhaps it makes cappuccinos when the customer asked it for a latte.
- Maybe it tweets "Coffee's on!" when it isn't, or vice versa.
- Clearly, as a publicly available product, your software has to do what you claimed it would, and do it efficiently and safely.
- Testing your code before it is deployed is an important step in helping to avoid such a situation, and your chosen language and development framework will have standard and well-understood testing environments to help ease your task.
- The embedded code in the device, however, is particularly important to test, as that is the hardest to update once the product has been sent out to the users.
- It may be possible, given that the devices will be connected to the Internet anyway, for the code to be updated *over-the-air.*
- If your device can update its own code, the channels for delivering and authenticating any updates must be rock solid, lest they be compromised and, as mobile technologist Brian Proffitt warns, "The Internet of Things might try to kill you".

- **SECURITY**
- These concerns lead us back to the important issue of security.
- Following are some of the more important guidelines:
- 1.Make sure that your servers are kept up-to-date with the latest security patches, are hardened with the appropriate firewalls, and detect and mitigate against password hacking attempts and rootkit attacks.
- 2. User passwords should never be stored in plain text.
- If your database were ever compromised, an attacker could easily log in as any user.
- 3. Never simply trust user input.
- Check that anything that is entered into a web application fits the type of data you expect, and refuse or clean anything which doesn't. Although you may think input from your connected devices would be okay (because you wrote the code), it is possible that it has been compromised or an attacker may be "spoofing" it.
- In particular, be wary of passing user input to your database without checking it (otherwise, you risk an SQL injection attack were it to include SQL commands)
- 4. Be aware of cross-site request forgery (CSRF) attacks from other malicious or compromised websites.
- For example, if one of your users browses a bad site which uses JavaScript to open http://some.example.com/heating?switch=off on your site, and the user is already logged in, he may come home to a cold house.

- **PERFORMANCE**
- The first thing many people think of when considering scaling up software is whether it will be fast enough and handle a large number of users.
- If your web service is running on a modern framework, it should be easy to scale up by deploying the code onto a more powerful machine, or by running multiple servers, with a front-end server or proxy managing the load.
- You should concentrate on running your web application with appropriate standard infrastructure which is good enough for most problems.
- If you're lucky enough to have so much traffic that you need to scale, you should always optimise using this algorithm:
    - Identify that you actually have a problem.
    - Measure and profile the tasks which are slow and identify the problem.

- Fix that problem. The web community does a good job in sharing best practice in how to address such problems, so you often find that others have trodden the path before and documented their tried-and-tested solutions.

- **USER COMMUNITY**
- Whether you are launching a mass-market commercial product or an open-source community effort, your project will be successful only if people actually use it.
- At a minimum, you need a support email or a bug-reporting tool (ideally a public one, so that users can easily find related issues), but you will probably have a presence on various social media forums (Twitter, Facebook, and the like).
- As well as responding to queries, however, you should think about the user experience before launching: does your website have documentation, introductory videos, and tutorials?
- Blogging regularly about product development and related topics helps build a readership of users, both current and potential.
- Finally, some form of forum, mailing list, or chat room lets users support each other, which reduces your workload, but more importantly helps to build up a community and expertise around your product.

IOT
chapter -11
ETHICS

Q1) summarize what the Internet of Things is?

According to technologist and policymaker Vannevar Bush:

- *[Machines with interchangeable parts can now be constructed with great economy of effort.*
- *In spite of much complexity, they perform reliably.*
- *Such as typewriter, or the movie camera, or the automobile.*
- *Electrical contacts have ceased to stick when thoroughly understood.*
- *Note the automatic telephone exchange, which has hundreds of thousands of such contacts, and yet is reliable.*
- *The world has arrived at an age of cheap complex devices of great reliability; and something is bound to come of it.]*

- The availabilities of technology brings certain abilities within the reach of not just the powerful but the ordinary citizen.
- Earlier examples were concerned with publication, transport, and communication.
- The advances in the Internet of Things are also primarily related to communication, but now allow the publication and transmission of vast streams of data, from the social to the environmental without needing the permission or expertise of a technological or political elite.
- The "form follows function" applies primarily to the physical usage of the "Thing", its affordances, sensors, and actuators, and only minimally to its digital communications.
- This leads to objects that can look innocuous (harmless) but have arbitrary and potentially unexpected capabilities.
- Connecting the Internet to the real world allows both your physical actions to be made public and, in the opposite direction, for events on the Internet to affect your environment.
- Applying this bidirectional communication to Things can lead to features that interact with the concept of privacy. When you switch on your Good Night Lamp, the "little lamp" at your mother's bedside will also turn on, letting her know you are home.
- When you leave the office, the WhereDial at home turns to let your partner know you're travelling.
- We have repeatedly noted that the Internet of Things is made up of
- Physical object + controllers, sensors, and actuators + Internet service
- Each of these aspects has a part to play in the ethical issues specific to the Internet of Things.

Q2) Explain the issues on PRIVACY. Also explain how internet affects PRIVACY.

- The Internet, as a massive open publishing platform, has been a disruptive force as regards the concept of privacy.
- Everything you write might be visible to anyone online.
- There is a *value* in making such data public: the story told on the Internet becomes your persona and defines you in respect of your friends, family, peers, and potential employers.
- A common argument is "if you've got nothing to hide, then you've got nothing to fear."

- You *change* and your persona changes.
- Yet your past mistakes (drunken photos, political statements) may be used against you in the future.
- As the Internet of Things is about *Things*, which are rooted in different contexts than computers, it makes uploading data more ubiquitous.
- Even innocuous photos can leak data. With GPS coordinates (produced by many cameras and most smartphones) embedded into the picture's EXIF metadata, an analysis of your Flickr/Twitpic/Instagram feed can easily let an attacker infer where your house, your work, or even your children's school is.
- Similar issues exist with sports-tracking data, whether produced by an actual Thing, such as Nike+ or a GPS watch.
- This data is incredibly useful to keep track of your progress, and sharing your running maps, speed and heartbeat.
- It may be trivial for an attacker to infer where your house is (probably near where you start and finish your run) and get information about the times of day that you are likely to be out of the house.
- To the extent that you allow your location to be shared with people you've *chosen* to share it with, there is no infringement of privacy.
- We saw previously that many "things" have little in their external form that suggests they are connected to the Internet.
- It is very important to note that even aggregate data can "leak" information.
- If you can see data collected for a street, for example, then comparing a week when a household is away on holiday with a normal week when they are at home might tell you about their usage.
- Some very interesting questions can be raised about this: should companies be prevented from trading data with each other?
- Should there be legal limits to what data can be kept or what analyses performed on it?
- Or do we have to think the unthinkable and admit that privacy is no longer possible in the face of massive data combined with data mining?
- As sensors such as CCTV cameras, temperature meters and Bluetooth trackers are installed in public and private spaces, from parks to shops, data about you is collected all the time.
- The term "data subject" has been coined for this purpose. Although you may not own the data collected, you are the subject of it and should have some kind of rights regarding it: transparency over what is collected and what will be done with it, the access to retrieve the data at the same granularity that it was stored, and so on.

Q3)Who CONTROLs data collected by devices (sensors)?

- Some of the privacy concerns manifest only if the "data subject" is not the one in control of the data.
- For example if the drunken photo was posted by *someone else*, without your permission.
- This is a form of cyberbullying, which is increasingly prevalent in schools and elsewhere.
- If you are gifted a WhereDial or a Good Night Lamp, is there an *expectation* that you use it, even if you don't really want to?
- While the technology itself doesn't cause any controlling behaviour, it could easily be applied by a spouse/parent/employer in ways that manifest themselves as abusive, interfering, or restrictive.

- Already, companies and organisations are looking at mashing up data sources and apps and may start to offer financial incentives to use Internet of Things devices: for example, reductions in health insurance if you use an Internet-connected heart monitor, have regular GPS traces on a run-tracking service, or regularly check in to a gym.
- As with questions about privacy, there are almost always good reasons for giving up some control.
- From a state perspective, there may be reasons for collective action, and information required to defend against threats, such as that of terrorism.
- The threat of one's country becoming a police state is not merely a technological matter: institutions such as democracy, the right to protest, free press, and international reputation should balance this.
- Now that surveillance equipment is cheap, and the processing power required to *analyse* the mountain of data produced by this equipment gets ever more accessible.
- As the Canadian open government activist David Eaves has eloquently discussed, it is not only authoritarian states such as Iran and China which are intent on controlling their Internet but also democratic ones.
- The US, UK, Canada, France, and others have already enacted various laws to give the state and its favoured corporations greater control over its citizens' use of the Internet.
- Of course, it may not be "the State" that profits from the control but corporations.
- Companies have the expertise and the technology to interact with the Internet. This is particularly true of the Internet of Things, which has largely been driven by monitoring and logistics concerns within large businesses.

Q4)Explain the terms "DISRUPTING CONTROL" and "CROWDSOURCING".

- **DISRUPTING CONTROL**
- The other major possibility that Eaves suggests is that "The Internet Destroys the State".
- When we refer to a technology as "disruptive", we mean that it affects the balance of power.
- If one of the fears about the Internet of Things is that it will transfer power away from the citizens, the subjects of technology, to the elite, then perhaps it can also be used to redress that balance.
- One extreme example of this would be how surveillance and fears of the Big Brother state (CCTV cameras, remote-controlled helicopter-drones) might be mitigated by "sousveillance".
- Here, activists might have compromised public cameras, or perhaps installed additional spy cameras, routed through self-healing networks in people's homes, hidden in public spaces, flying in the airspace, or even crawling the sewers.

- **CROWDSOURCING**
- One fascinating feature of modern Internet life is "crowdsourcing", from knowledge to funding projects to work.
- In the Internet of Things world, this concept has manifested itself in sensor networks such as Xively.
- Founder Usman Haque has said that their original intent wasn't simply "making data public" but also letting "the public making data".
- Governments and companies simply do not and cannot have a monopoly on all recording of data: there are infinite combinations of data sources.

- Choosing which data to record is a creative and engaged act, as well as, perhaps, a political one.
- Former Xively evangelist Ed Borden has led a "call to arms" for a citizen-led air-quality sensor network.
- As he points out, "The air quality data collected by the government is likely sampled from far, far away and then applied to you on a regional level, almost completely useless from the standpoint of trying to understand or change the local dynamics of pollution that affect you".
- Crowdsourcing this data is an entirely innocent scientific activity yet is profoundly radical, too.
- Javaun Moradi, product manager for NPR Digital, clarifies, "These networks aren't trying to replace scientific and government detection equipment, they're trying to both fill a data gap and advance conversation".

Q5) List and explain five critical requirements observed in Fisher's original definition for a sensor commons project.

- Fisher's original definition observed five critical requirements for a sensor commons project.
- It must:
- ▪ **Gain trust:**
- Trust is largely about the way that an activist project handles itself beyond the seemingly neutral measurements; understanding local issues, being sensitive about the ways that the sensor network itself affects the environment, engaging the public with accessible and readable information about the project, and dealing with the local authorities to get access to the systems the project wants to measure.
- ▪ **Become dispersible:**
- Becoming dispersible means spreading the sensors throughout the community.
- Getting mass adoption will be easier if the proposed sensors are inexpensive and if the community already trusts the project.
- If the sensors are complicated to set up or require massive lengths of cabling, they will get much less take-up!
- The Xively air-quality project led to the creation of the "air-quality egg", a simple, inexpensive sensor with precisely these features.
- ▪ **Be highly visible:**
- Being visible involves explaining why the project's sensors are occupying a public space.
- Being honest and visible about the sensor will help to engender trust in the project and also advertise and explain the project further.
- Advertising not just the sensors but the *data* (both online and in real life) and the ways that data has helped shape behaviour will also generate a positive feedback loop.
- ▪ **Be entirely open:**
- Being open is perhaps what distinguishes the sensor commons from a government project the most.
- Government data sets are often entirely closed, but the data that *is* released from them will be given a lot of attention because of the rigour and precision that their sensor projects will have.
- A community sensor network may have uncalibrated devices—that is, the readings for a device may be consistently out from the "correct" value and may have additional noise at the extremes of the scale.

- The openness makes up for this because all the facts about the devices and the possible errors are admitted upfront and can be improved by anyone in the community.
- ▪ **Be upgradable:**
- Finally, the project should be designed to be upgradable, to enable the network to remain useful as the needs change or hardware gets to the end of its working life.
- This requirement interplays with the dispersibility and openness of the project, and the up-front thought to managing the project long term will feed back into the trust in the project.

Q6)Which things affect more to the ENVIRONMENT and how to measure environmental cost ?

- The classic environmental concerns about the production and running of the *Thing* itself.
- **PHYSICAL THING**
- Creating the object has a carbon cost, which may come from the raw materials used, the processes used to shape them into the shell, the packing materials, and the energy required to ship them from the manufacturing plant to the customer. It's easier than ever to add up the cost of these emissions: for example, using the ameeConnect API (www.amee.com/ pages/api), you can find emissions data and carbon costs for the life-cycle use of different plastics you might use for 3D printing or injection moulding.
- You may need to consider other environmental factors, such as emissions produced during normal operation or during disposal of the object.
- For example, thermal printer paper may contain Bisphenol-A, which has health and environmental concerns.
- **ELECTRONICS**
- The electronics contained in a Thing have their own environmental cost.
- Buying PCBs locally or from a foreign manufacturer affects the carbon cost of shipping the completed units.
- If your product needs to conform to RoHS legislation, then every single component that could be extracted from it must be RoHS compliant.
- More worryingly, many electronic components rely on "rare earth minerals" (REMs) which have been extracted in China or from other locations worldwide.
- The mining process must be managed properly; otherwise, slurries formed of mildly radioactive waste minerals will be left behind long after the mines cease production. Refining them involves the use of toxic acids.
- Shipping the raw material from mine to refinery to manufacturer has its own carbon cost too.
- **INTERNET SERVICE**
- As Nicholas Negroponte (founder of MIT's Media Lab) preaches, "Move bits, not atoms".
- In the digital world, moving data rather than physical objects is faster, is safer, and has a lower environmental cost.
- Of course, "data" doesn't exist in the abstract.
- The stone tablets, and libraries of paper books that have historically been used to store analogue data always had their own environmental cost.
- Now, running the Internet has a cost: the electricity to run the routers and the DNS lookups, plus establishing the infrastructure—laying cabling across the sea, setting up microwave or satellite links, and so on.
- As well as the cost of transferring the data across the Internet, running your own web server uses power.

Q7)Explain use of THE INTERNET OF THINGS AS PART OF THE SOLUTION.

- Gavin Starks, former CEO of amee, has spoken convincingly of instrumenting the world precisely to *save* it.
- While Starks's lectures are timely and necessary, as a good hacker, he prefers to *do* something about the problem: try to solve it through technology, information, and awareness.
- We already discussed distributed sensor networks as a social and political act: the potential for global environmental action is also massive.
- If community-led sensor networks can help supplement government and international science measurements, then we should be doing everything we can to help.
- Instrumenting production lines, home energy usage, transport costs, building energy efficiency, and all other sources of efficiency might seem extreme, but it may be a vital, imperative task.
- Other technologies which aren't principally linked with the Internet of Things will also be important.
- If 67 percent of the world's water usage is in agriculture, then are there ways to reduce that quantity through technology?
- Instrumenting the supply chains, measuring to be certain that new methods really *are* more efficient, and reducing inefficiencies by automation could well use Internet of Things solutions to help measure and implement the solutions.
- The Internet of Things could become a core part of the solution to our potentially massive environmental problems.
- Projects such as a carbon score for every company in the UK will help change attitudes, perhaps simply by gamifying the process of improving one's emissions, but also by having an objective measure that could, in future, be as important to a company's success as its credit score.
- In the face of these suggestions—collective sensor networks and massive business process engineering not for profit but for environmental benefits—you might wonder whether these calls to action amount to critiques of capitalism.
- As resources become ever scarcer, a greater percentage of income might be spent on covering rental of all goods—cars, food, possibly even housing.
- The Internet of Things will also, if we let it, become a platform for whatever people want it to be.

Q8) What is CAUTIOUS OPTIMISM?

- Technology *did* change the world that they knew, for the worse, in many senses.
- But without the changes that disrupted and spoilt one world, we wouldn't have arrived at a world, our world, where magical objects can speak to us, to each other, and to vastly powerful machine intelligences over the Internet.
- As a massively interdisciplinary field, practitioners of Internet of Things may have an opportunity (or perhaps responsibility) to contribute to providing moral leadership in many of the upcoming ethical challenges.
- we should remember an important lesson on humility from Laura James's keynote at the OpenIoT assembly:
- *[Don't assume you know it all.*
- *The [I]nternet of [T]hings is interdisciplinary and it stretches most of the individual disciplines too.*
- *You will need help from others.*
- *Be ready to partner with other organisations, collaborate with people from very different backgrounds to you.]*

- When designing the Internet of Things, or perhaps when designing *anything*, you have to remember two contrasting points:
- ▪ **Everyone is not you.**
- Though you might not personally care about privacy or flood levels caused by global warming, they may be critical concerns for other people in different situations.
- ▪ **You are not special.**
- If something matters to you, then perhaps it matters to other people too.

Q9) Explain THE OPEN INTERNET OF THINGS DEFINITION.

- The Open IoT Assembly 2012 culminated in the drafting of the "Open Internet of Things Definition".
- An emergent document, created after two days of open discussion, it seeks to define and codify the points of interest around the technology of the Internet of Things and to underscore its potential to "deliver value, meaning, insight, and fun".
- A particularly interesting consensus in the definition was that, even though the Data Licensor (usually the person who has set up the sensor or paid for that data) should quite reasonably own the data from that sensor, some rights should also apply to individuals whose data is recorded (the Data Subjects).
- They *must* be granted licence to any data that regards them and *should* be allowed to license the anonymised aggregate data for their own purposes.
- We can summarize the main goals of the definition as follows:
- ▪ **Accessibility of data:**
- As a stated goal, all open data feeds should have an API which is free to use, both monetarily and unrestricted by proprietary technologies with no alternative open source implementation.
- ▪ **Preservation of privacy:**
- The Data Subjects should know what data will be collected about them and be able to decide to consent or not to that data collection.
- **Transparency of process:**
- Data Subjects should be made aware of their rights—for example, the fact that the data has a licence—and that they are able to grant or withdraw consent.
- In addition, where the data is collected from a *public* space, the public should get a right to participate in decision making and governance of that data.

- The importance placed by these principles on data is unsurprising: the Internet of Things brings the gathering and collation of data into the everyday world and has real consequences on individual privacy and power.