T.Y.B.SC (I.T)

SEMESTER - V (CBCS)

INTERNET OF THINGS

SUBJECT CODE: USIT502

Dr. Suhas Pednekar
Vice Chancellor
University of Mumbai, Mumbai

Prof. Ravindra D. Kulkarni
Pro Vice-Chancellor,
University of Mumbai

Prof. Prakash Mahanwar
Director,
IDOL, University of Mumbai

Programme Co-ordinator : **Shri Mandar Bhanushe**
Head, Faculty of Science and Technology IDOL
Univeristy of Mumbai – 400098

Course Co-ordinator : **Gouri S.Sawant**
Assistant Professor B.Sc.I.T, IDOL
University of Mumbai- 400098

Editor : **Dr Triveni Kaul**
Assistant Professor
Yashwantrao Chavan College of Arts, Science and Commerce Navi Mumbai.

Course Writers : **Ms Reena Kothari**
Assistant Professor,
Vidyalankar School of Information Technology,
Vidyalankar Campus, Wadala (E), Mumbai.

: **Mr Sabir Moin Moinuddin Shaikh**
Assistant Professor,
M V Mandali's Colleges of Commerce and Science, Veera Desai Road, Andheri West.

: **Ms Trupti Kulkarni Kaujalgi**
HOD, IT&CS Dept,
ICLES' Motilala Jhunjhunwala College Vashi, Navi Mumbai

: **Ms Nabila Kazi**
Assistant Professor,
Bunts Sangha's S.M.Shetty College of Science, Commerce & Management Studies,
Powai, Mumbai.

September 2022, Print - 1

Published by : Director,
Institute of Distance and Open Learning,
University of Mumbai,
Vidyanagari,Mumbai - 400 098.

DTP composed and Printed by: Mumbai University Press

CONTENTS

Chapter No.	Title	Page No.
	Unit 1:	
1.	The Internet of Things: An Overview	01
2.	Design Principles for Connected Devices	12
3.	Internet Principles	19
	Unit 2:	
4.	Thinking About Prototyping	32
5.	Prototyping Embedded Devices	45
	Unit 3:	
6.	Prototyping the Physical Design	65
7.	Prototyping Online Components	87
	Unit 4:	
8.	Techniques For Writing Embedded Code	102
9.	Business Models	112
	Unit 5:	
10.	Moving To Manufacture	127
11.	Ethics	150

T.Y.B.SC (I.T)
SEMESTER - V

INTERNET OF THINGS

Unit	Details	Lectures
I	**The Internet of Things: An Overview**: The Flavour of the Internet of Things, The "Internet" of "Things", The Technology of the Internet of Things, Enchanted Objects, Who is Making the Internet of Things? **Design Principles for Connected Devices:** Calm and Ambient Technology, Magic as Metaphor, Privacy, Keeping Secrets, Whose Data Is It Anyway? Web Thinking for Connected Devices, Small Pieces, Loosely Joined, First-Class Citizens on The Internet, Graceful Degradation, Affordances. **Internet Principles:** Internet Communications: An Overview, IP, TCP, The IP Protocol Suite (TCP/IP), UDP, IP Addresses, DNS, Static IP Address Assignment, Dynamic IP Address Assignment, IPv6, MAC Addresses, TCP and UDP Ports, An Example: HTTP Ports, Other Common Ports, Application Layer Protocols, HTTP, HTTPS: Encrypted HTTP, Other Application Layer Protocols.	**12**
II	**Thinking About Prototyping:** Sketching, Familiarity, Costs versus Ease of Prototyping, Prototypes and Production, Changing Embedded Platform, Physical Prototypes and Mass Personalisation, climbing into the Cloud, Open Source versus Closed Source, Why Closed? Why Open? Mixing Open and Closed Source, Closed Source for Mass Market Projects, Tapping into the Community. **Prototyping Embedded Devices:** Electronics, Sensors, Actuators, Scaling Up the Electronics, Embedded Computing Basics, Microcontrollers, System-on-Chips, Choosing Your Platform, Arduino, developing on the Arduino, Some Notes on the Hardware, Openness, Raspberry Pi, Cases and Extension Boards, Developing on the Raspberry Pi, Some Notes on the Hardware, Openness.	**12**
III	**Prototyping the Physical Design:** Preparation, Sketch, Iterate, and Explore, Nondigital Methods, Laser Cutting, Choosing a Laser Cutter, Software, Hinges and Joints, 3D Printing, Types of 3D Printing, Software, CNC Milling, Repurposing/Recycling. **Prototyping Online Components:** Getting Started with an API, Mashing Up APIs, Scraping, Legalities, writing a New API, Clockodillo, Security, implementing the API, Using Curl to Test, Going Further, Real-Time Reactions, Polling, Comet, Other Protocols, MQ Telemetry Transport, Extensible Messaging and Presence Protocol, Constrained Application Protocol.	**12**

IV	**Techniques for Writing Embedded Code:** Memory Management, Types of Memory, Making the Most of Your RAM, Performance and Battery Life, Libraries, Debugging. **Business Models:** A Short History of Business Models, Space and Time, From Craft to Mass Production, The Long Tail of the Internet, Learning from History, The Business Model Canvas, Who Is the Business Model For? Models, Make Thing, Sell Thing, Subscriptions, Customisation, be a Key Resource, Provide Infrastructure: Sensor Networks, take a Percentage, Funding an Internet of Things Startup, Hobby Projects and Open Source, Venture Capital, Government Funding, Crowdfunding, Lean Startups.	**12**
V	**Moving to Manufacture:** What Are You Producing? Designing Kits, Designing Printed circuit boards, Software Choices, The Design Process, Manufacturing Printed Circuit Boards, Etching Boards, Milling Boards. Assembly, Testing, Mass-Producing the Case and Other Fixtures, Certification, Costs, Scaling Up Software, Deployment, Correctness and Maintainability, Security, Performance, User Community. **Ethics:** Characterizing the Internet of Things, Privacy, Control, Disrupting Control, Crowdsourcing, Environment, Physical Thing, Electronics, Internet Service, Solutions, The Internet of Things as Part of the Solution, Cautious Optimism, The Open Internet of Things Definition.	**12**

Books and References:

Sr. No.	Title	Author/s	Publisher	Edition	Year
1.	Designing the Internet of Things	Adrian McEwen, Hakim Cassimally	WILEY	First	2014
2.	Internet of Things – Architecture and Design	Raj Kamal	McGraw Hill	First	2017
3.	Getting Started with the Internet of Things	Cuno Pfister	O'Reilly	Sixth	2018
4.	Getting Started with Raspberry Pi	Matt Richardson and Shawn Wallace	SPD	Third	2016

1

THE INTERNET OF THINGS:

AN OVERVIEW

Unit Structure

1.0 Objective
1.1 The Flavor of the Internet of Things
1.2 The "Internet" of "Things"
    1.2.1 IoT − Key Features
    1.2.2 IoT - Characteristics
    1.2.3 IoT − Advantages
    1.2.4 IoT − Disadvantages
1.3 The Technology of the Internet of Things
1.4 Enchanted Objects
1.5 Who is making the Internet of Things?
1.6 Summary
1.7 Question
1.8 References

1.0. OBJECTIVE

1. The fundamental objective of IoT is to obtain and analyze data from things.
2. IoT covers a wide spectrum of applications, including the detailed real-time sensing of our environment and the embedding of connected intelligence into everyday objects.
3. By connecting all kinds of objects and systems, Internet stuff can offer new ways to research and learn.

1.1 THE FLAVOR OF THE INTERNET OF THINGS

1. My Train-Schedule Alarm Clock
2. Vitality GlowCap
3. Umbrella with Weather-Forecasting Handle
4. Nike + Health App
5. Transport for London
6. Wheredial

1. My Train-Schedule Alarm Clock:

The alarm rings. As we open our eyes dim, we see that it's five minutes later than our usual wake-up time. It was just a simple state machine that slept until it got close to wake-up time, then it started

monitoring the departure boards. The clock has checked the train times online, and your train must be delayed, so it lets you sleep a little longer.

2. **Vitality GlowCap:**

   Vitality GlowCap is an easy-to-use, comprehensive medication adherence system consisting of a smart cap and bottle, Vitality Mobile Application, and the Vitality Medication Adherence Portal. Automated visual and audible alerts are scheduled during dosage windows which signals that it is time for the user to take his or her medications. A blinking light reminds you it's time to take your tablets. If you forget, the medicine bottle cap goes online and emails your doctor, care managers, and trusted family and friends to let them know.

3. **Umbrella with Weather-Forecasting**

   A model for appropriate embedded technology, the Forecast umbrella provides information about the likelihood of rain so that users can make a simple decision about whether to take their umbrella with them as they leave for home. Using existing Wi-Fi technology to wirelessly pull information from the internet, Forecast's lighted umbrella handle is lit up, which means that it has checked the BBC weather reports and predicted rain.

4. **Nike + Health App:**

   A pedometer in your training shoes and a heart monitor in your wrist band help track your run around the block. 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.

5. **Transport for London:**

   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. 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.

6. **Wheredial:**

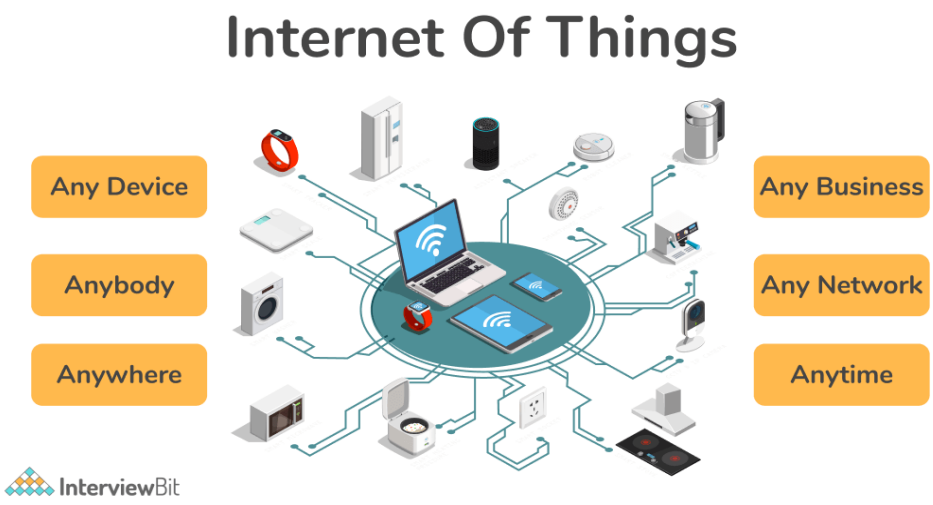
   Your phone checks you in automatically to a location-based service (such as Foursquare). On your mantelpiece at home, an ornament with a dial notices the change and starts to turn so that the text on it points to the word "Traveling". Your family will also see later that you've arrived at "Work" safely.

1.2. THE "INTERNET" OF "THINGS"

The Internet of Things (IoT) is the network of physical devices, vehicles, home appliances, and other items embedded with electronics, software, sensors, actuators, and connectivity which enables these things to connect and exchange data, creating opportunities for more direct integration of the physical world into computer-based systems, resulting in efficiency improvements, economic benefits, and reduced human exertions.

Connecting everyday things embedded with electronics, software and sensors to the internet enabling them to collect and exchange data. IoT (Internet of Things) is an advanced automation and analytics system which exploits networking, sensing, big data, and artificial intelligence technology to deliver complete systems for a product or service. These systems allow greater transparency, control, and performance when applied to any industry or system.

For example: A temperature sensor sends temperature data to a process which determines that the room temperature is too hot and so sends a signal to turn on the air conditioning.



Equation of Internet of Things:

Physical Object
+
Controllers, Sensors and Actuators
+
Internet
=
Internet of Things

Physical Object:
Devices, vehicles, buildings and other items which are embedded with electronics, software, sensors, and network connectivity, which enables these objects to collect and exchange data.

Controllers:

A controller, in a computing context, is a hardware device or a software program that manages or directs the flow of data between two entities. In a general sense, a controller can be someone that interfaces between two systems and manages communications between them.

Sensors:

Sensor is a device, module, or subsystem whose purpose is to detect events or changes in its environment and send the information to other electronics, frequently a computer processor. A sensor is always used with other electronics.

Actuators:

An actuator is a component of a machine that is responsible for moving and controlling a mechanism or system.

Internet:

The internet is a globally connected network system that uses TCP/IP to transmit data via various types of media. The internet is a network of global exchanges – including private, public, business, academic and government networks – connected by guided, wireless and fiber-optic technologies.

1.2.1. IoT − Key Features

The most important features of IoT include artificial intelligence, connectivity, sensors, active engagement, and small device use. A brief review of these features is given below:

1. **Artificial Intelligence** – IoT essentially makes virtually anything "smart", meaning it enhances every aspect of life with the power of data collection, artificial intelligence algorithms, and networks. This can mean something as simple as enhancing your refrigerator and cabinets to detect when milk and your favorite cereal run low, and to then place an order with your preferred grocer.
2. **Connectivity** – New enabling technologies for networking, and specifically IoT networking, mean networks are no longer exclusively tied to major providers. Networks can exist on a much smaller and cheaper scale while still being practical. IoT creates these small networks between its system devices.
3. **Sensors** – IoT loses its distinction without sensors. They act as defining instruments which transform IoT from a standard passive network of devices into an active system capable of real-world integration.
4. **Active Engagement** – Much of today's interaction with connected technology happens through passive engagement. IoT introduces a new paradigm for active content, product, or service engagement.
5. **Small Devices** – Devices, as predicted, have become smaller, cheaper, and more powerful over time. IoT exploits purpose-built small devices to deliver its precision, scalability, and versatility.

1.2.2. IOT - Characteristics

The fundamental characteristics of the IoT are as follows:

1. **Interconnectivity:** With regard to the IoT, anything can be interconnected with the global information and communication infrastructure.
2. **Things-related services:** The IoT is capable of providing thing-related services within the constraints of things, such as privacy protection and semantic consistency between physical things and their associated virtual things. In order to provide thing-related services within the constraints of things, both the technologies in the physical world and information world will change.
3. **Heterogeneity:** The devices in the IoT are heterogeneous as based on different hardware platforms and networks. They can interact with other devices or service platforms through different networks.
4. **Dynamic changes:** The state of devices changes dynamically, e.g., sleeping and waking up, connected and/or disconnected as well as the context of devices including location and speed. Moreover, the number of devices can change dynamically.
5. **Enormous scale:** The number of devices that need to be managed and that communicate with each other will be at least an order of magnitude larger than the devices connected to the current Internet. Even more critical will be the management of the data generated and their interpretation for application purposes. This relates to semantics of data, as well as efficient data handling.
6. **Safety:** As we gain benefits from the IoT, we must not forget about safety. As both the creators and recipients of the IoT, we must design for safety. This includes the safety of our personal data and the safety of our physical well-being. Securing the endpoints, the networks, and the data moving across all of it means creating a security paradigm that will scale.
7. **Connectivity:** Connectivity enables network accessibility and compatibility. Accessibility is getting on a network while compatibility provides the common ability to consume and produce data.

1.2.3. IoT − Advantages

1. **Improved Customer Engagement:** Current analytics suffer from blind-spots and significant flaws in accuracy; and as noted, engagement remains passive. IoT completely transforms this to achieve richer and more effective engagement with audiences.
2. **Technology Optimization:** The same technologies and data which improve the customer experience also improve device use, and aid in more potent improvements to technology. IoT unlocks a world of critical functional and field data.

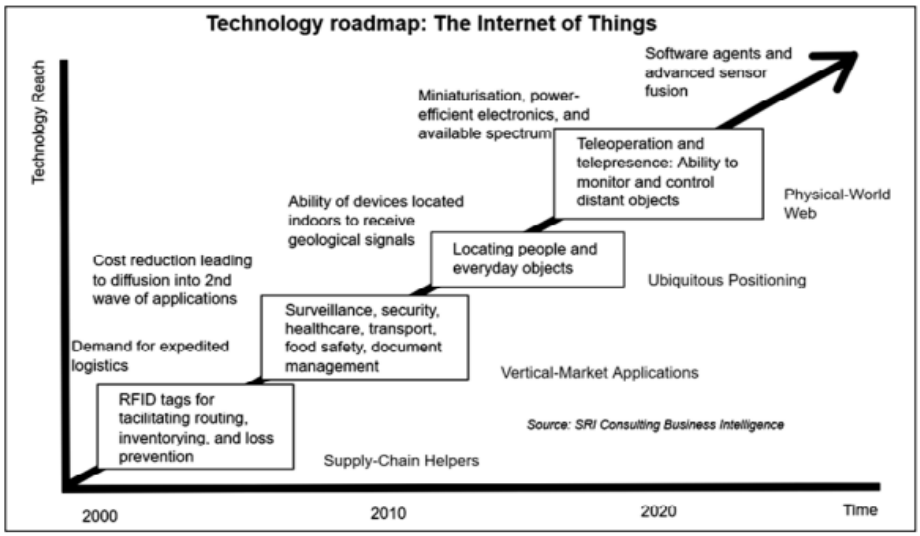
3. **Reduced Waste:** IoT makes areas of improvement clear. Current analytics give us superficial insight, but IoT provides real-world information leading to more effective management of resources.
4. **Enhanced Data Collection:** Modern data collection suffers from its limitations and its design for passive use. IoT breaks it out of those spaces, and places it exactly where humans really want to go to analyze our world. It allows an accurate picture of everything.

1.2.4. IoT − Disadvantages

1. **Security:** IoT creates an ecosystem of constantly connected devices communicating over networks. The system offers little control despite any security measures. This leaves users exposed to various kinds of attackers.
2. **Privacy:** The sophistication of IoT provides substantial personal data in extreme detail without the user's active participation.
3. **Complexity:** Some find IoT systems complicated in terms of design, deployment, and maintenance given their use of multiple technologies and a large set of new enabling technologies.
4. **Flexibility:** Many are concerned about the flexibility of an IoT system to integrate easily with another. They worry about finding themselves with several conflicting or locked systems.
5. **Compliance:** IoT, like any other technology in the realm of business, must comply with regulations. Its complexity makes the issue of compliance seem incredibly challenging when many consider standard software compliance a battle.

1.3 THE TECHNOLOGY OF THE INTERNET OF THINGS

The major enabling technologies and protocols of IoT are RFID, NFC, low-energy Bluetooth, low-energy wireless, low-energy radio protocols, LTE-A, and WiFi-Direct.



Technology's great drivers have initially been fundamental needs, such as food and water, warmth, safety, and health. Technological advances progress with enabling and controlling the movement of people, their possessions, livestock, and other resources. 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. But the other human needs we looked at haven't ceased to exist, nor will they. We still need to eat and drink. We still need light and warmth. We still need love and friendship. We still need chairs, clothes, and shoes; means of transport and communication; and ways to entertain ourselves. As technology has progressed, new categories of objects have been created: in the electronic age, they have included telephones, radios, televisions, computers, and smartphones. Demand drives down prices, and research leads to optimization and miniaturization. A television screen would originally have physically dominated a living room, not only are today's flat-screen panels more compact, but the technology is so ubiquitous that a high solution screen capable of displaying television content can be embedded into a door frame or a kitchen unit, and of course, even smaller screens can find their way into music players and mobile phones. 8 Internet of Things Computers have become so cheap to produce a general purpose microchip in devices that your washing machine may contain a computer running Linux, the cash register at the supermarket may run on Windows, and your video player may run a version of Apple's OS X. Let's consider the computers that exist in modern cars: they have myriad sensors to determine how well the car is running—from oil gauge and the pressure to the internals of your engine. As well as diagnostics, computerized brakes may assist the driver when the processor spots conditions such as the wheels locking or spinning out of control. All this is local information, and although the processing and analysis of this data may be highly sophisticated, it will be limited to whatever your car manufacturer has programmed. But perhaps your car also tracks your location using

GPS: this is external data. High-end cars may communicate the location back to a tracking service for insurance and anti-theft purposes. GPS can be created in real time by "social route planning" based on the data aggregated from what other connected drivers nearby are doing. When the Internet moved out of academia and the military, with the first commercial Internet service providers (ISPs) opening for business in the late 1980s, the early adopters of the consumer Internet may have first gone online with a computer running an Intel 486 chip, costing around £1500, or around the price of a small car. A microchip with equivalent power might set you back around £0.50, or the price of a chocolate bar. The rapid rise of processing power, and the consequent cost decrease, is not a new insight: it is widely known as Moore's law (the rule of thumb, suggested by the cofounder of Intel, which says the number of transistors you can fit on a silicon chip will double every 18 months). Price is a qualitative as well as a quantitative change. The price of computing power has come down to affordable levels. The wealth of programming and debugging

resources available for these platforms has made them attractive to hobbyists and the prototyping market. Internet connectivity is also cheaper and more convenient than it used to be. 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. Another factor at play is the maturity of online platforms. This provides a ready ecosystem for other websites to "mash up" a number of services into something new, enables mobile phone "Apps", and now makes it easy for connected devices to consume. The online services mature, so too do the tools used to build and scale them. Web services frameworks such as Python and Django or Ruby on Rails allow easy prototyping of the online component. Cloud services such as Amazon Web Services mean that such solutions can scale easily with use as they become more popular.

1.4 ENCHANTED OBJECTS:

Media Lab researcher and Technologist David Rose has talked about Enchanted Objects and coined the term "enchanted objects" as to describe ordinary objects with extraordinary functions and has categorised various objects drawn from fairy tales and fantasy literature in ways that apply as much to technological objects.

1. **For Protection,**

   Example:

   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 has been driven by the need for military superiority, for the purpose of security.

2. 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.

   Example:

   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.

3. 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.

   Example:

   In story: for Effortless Mobility invented flying carpets, and even teleportation.

In reality: Through technology, we have invented cars and railways, bicycles, and aeroplanes.

4. The need for Creative Expression

Example:
In story: by the enchanted paintbrushes and magic flutes

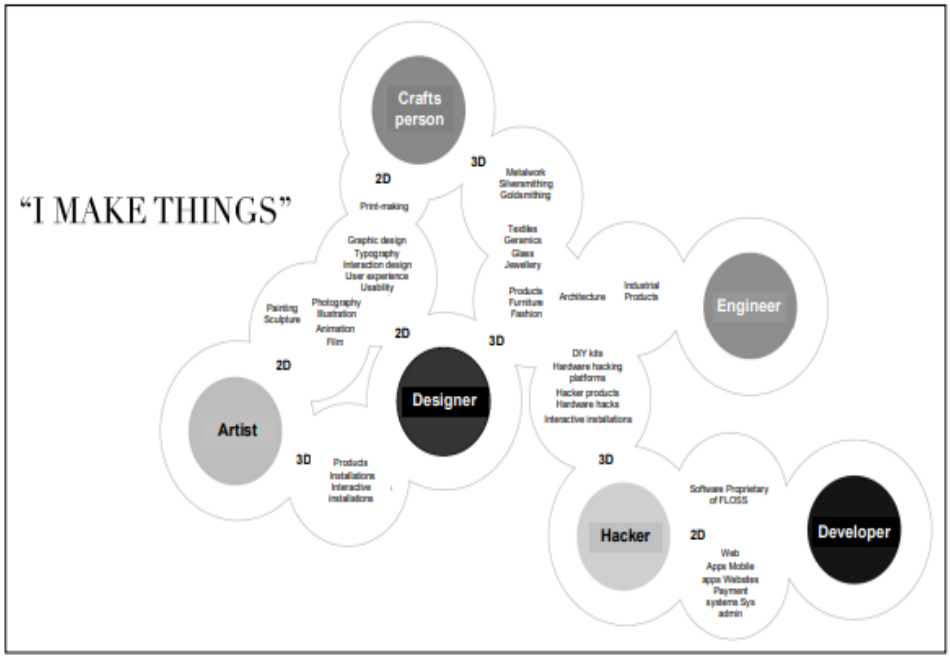
In reality: from charcoal to paint to computer graphics, or from drums to violins and electronic synthesisers.

Unlike IOT, the era of enchanted objects is about making the everyday objects in our world smarter, rather than making technology requiring a high cost of ownership. The enchanted objects learn on their own, have low or no cost of ownership, and don't require a host of technical people to program or maintain them. 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.

1.5. WHO IS MAKING THE INTERNET OF THINGS?

Internet of Things thought leader and entrepreneur Alexandra Deschamps-Sonsino noted at the Victoria and Albert Museum's Power of Making Symposium, both these words mean many things to different people.



Many persons are involved and they gave their contribution to develop an IOT platform are as follows:

1. Craftsperson
2. Artist
3. Designer
4. Engineer
5. Hacker
6. Developer

Artists may collaborate with the designers on installations or with traditional craftspeople on printmaking. Designers and engineers work closely to make industrial products, and hobbyist "hackers" by their nature, are a diverse group encompassing various technical and artistic interests and skills. The Internet of Things straddles all these disciplines: 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.

1.6 SUMMARY:

We began by looking at some examples of the Internet of Things in action. The Internet of Things can be characterized as joining the physical object, the computer embedded into it, and communication and code on the Internet itself. We compared Internet-connected devices to enchanted objects. However, this means that the playing field for making such a connected device is astonishingly level. There really is no better time to enter the exciting world of the Internet of Things.

1.7 QUESTION:

1) What are the flavors of the Internet of Things?

2) Write an equation of the "Internet" of "Things". And explain the purpose of IOT.

3) Explain the technology of the Internet of Things.

4) Explain how and where IOT fits with the help of history of technology.

5) What are enchanted objects? Explain with examples how technology has always been associated with magic.

6) Who is making the Internet of Things?

1.8 REFERENCES:

1. Designing the Internet of things by Adrian McEwen, Hakim Cassimally Wiley Publication.

2. Internet of Things by Prof. Sandeep Vishwakarma, Prof. Kiran Gurbani, Prof. Nitesh N. Shukla Himalaya Publishing House

3. Getting Started with the Internet of Things by Cuno Pfister O'Reilly

❄❄❄❄❄❄❄

2

DESIGN PRINCIPLES FOR CONNECTED DEVICES

Unit Structure

2.1 Calm and Ambient Technology
2.2 Magic as Metaphor,
2.3 Privacy, Keeping Secrets
2.3.1 Keeping Secrets
2.4 Who's Data Is It Anyway?
2.5 Web Thinking for Connected Devices
2.6 Small Pieces, Loosely Joined,
2.7 First-Class Citizens on the Internet
2.8 Graceful Degradation
2.9 Affordances
2.10 Summary
2.11 Question
2.12 References

2.1 CALM AND AMBIENT TECHNOLOGY

The IoT has its roots in the work done by Mark Weiser at Xerox PARC in the 1990s. • His work didn't assume that there would be network connectivity but was concerned with what happens when computing power becomes cheap enough that it can be embedded into all manners of everyday objects. He coined the term ubiquitous computing or ubicomp. Ubicomp is an ambient technology and through his research and writing sought to explore what that would mean for the people living in such a world.

Calm and Ambient technology means technology which acts in background, not something to which we actively pay attention i.e. Ambient noise in background recording. The term Calm technology means a system that doesn't seek your attention.

Example: 1) Live Wire: Live wire is one of the first IOT devices. Created by artist Natalie Jeremijenko. Live wire also known as Dangling String. It 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 (make a sudden movement) whenever a packet of information is

sent across the network. Under normal, light network load, the string twitches 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 (who is used to the normal behavior) that something is amiss and lets him investigate further. 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. On a computer, updating the screen is purely visual, so any additional senses must be engaged explicitly. Like Live Wire, Bubblino—Adrian's Internet of Things bubble machine which searches Twitter and blows bubbles when it finds new tweets matching a search phrase is a good example in which the side effect of the motor is to generate an audible notification that something is happening. With their Olly device, agency Mint Digital combines the motor with a deliberate olfacto

Example: 2) Split-flap display: Split-flap displays have been phased in and out and are replaced by dot-matrix LED displays. The newer displays are much easier to update with new destinations. Split-flap displays are at airports and railway stations.

Example: 3) Airtunes Wi-Fi speakers: Which anyone plays music through. Users will often wonder exactly what a particular track is but had no way of finding out who was in charge of the music at that moment and what was playing right now.

2.2. MAGIC AS METAPHOR

One of the main issues with introducing any new technology or service that is radically different from the norm is getting people to understand and accept it. Early adopters are generally happier looking a bit strange or doing things somewhat awkwardly to take the benefits of the new gadgets; however, for the technology/service to catch on, you need to persuade the majority to take it up. In addition to technology becoming capable of a particular action, we often need society, for want of a better term, 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 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, although the big advances sneak up on us over time, hidden in plain sight. Rao called this concept the manufactured normalcy 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.... So, what of the Internet of Things? As we saw in the last chapter, 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. 28 Designing the Internet of Things 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 Weasley clock could use magic to divine the whereabouts of each family member and was therefore also aware of when they were in mortal peril. 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 traveling, or at a restaurant.

2.3. PRIVACY AND KEEPING SECRETS

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.

2.3.1 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 license 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 license 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 license plate search string. That meant he could download a full list of license 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.

2.4. WHO'S 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.

2.5. 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.

2.6. 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 generalized 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.

2.7. 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.

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.

2.8. 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.

2.9. AFFORDANCES:

Donald Norman defines affordances as 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 realize 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 behaviors.

2.10 SUMMARY

This chapter will have given you a deeper understanding of the emerging field of the Internet of Things and some ways to direct your thinking when you are designing something to fit into the landscape.
The examples have shown how you need to think not just about the technical details of how the device will work, but also of how it will fit into the wider context of the user's life.
You also need to take care not to divulge any information that users wouldn't expect you to. It is a new field full of expanding possibilities which gives us opportunities for delighting and enriching people's lives. Even you will already have encountered different aspects of the network bleeding through—for example, in the description of Natalie Jeremijenko's Live Wire or the mention of HTTP protocol versions in the section on graceful degradation.

2.11 QUESTION

1. What is CALM and AMBIENT technology? Explain with the example of Live Wire.
2. Explain the terms "Magic as Metaphor" and "Manufactured Normalcy Field" along with a few examples.
3. Describe "KEEPING SECRETS" with the example of "An Instrumented Car Park".

4. Define and explain the following terms :

a. web thinking for connected devices

b. small pieces, loosely joined

c. first-class citizens on the internet

5. What is graceful degradation? Also explain affordance with respect to technology.

2.12 REFERENCES

1. Designing the Internet of things by Adrian McEwen, Hakim Cassimally Wiley Publication.

2. Internet of Things by Prof. Sandeep Vishwakarma, Prof. Kiran Gurbani, Prof. Nitesh N. Shukla Himalaya Publishing House

3. Getting Started with the Internet of Things by Cuno Pfister O'Reilly

❄❄❄❄❄❄❄

3

INTERNET PRINCIPLES

Unit Structure

3.1 Internet Communications: An Overview
3.2 The IP Protocol Suite (TCP/IP)
3.3 User Datagram Protocol
3.4 IP Addresses
3.5 Domain Name Server
3.6 Static IP Address Assignment
3.7 Dynamic IP Address Assignment
3.8 IPv6
3.9 MAC Addresses
3.10 TCP and UDP Ports
3.11 Application Layer Protocols
- 3.11.1. HTTP
- 3.11.2. HTTPS
- 3.11.3. SNMP
- 3.11.4. SMTP
- 3.11.5. DNS
- 3.11.6. Telnet
- 3.11.7. FTP

3.12 Summary
3.13 Question
3.14 References

3.1. INTERNET COMMUNICATIONS: AN OVERVIEW:

The Internet is the network that connects computers all over the world. It works according to a set of agreed-upon protocols. TCP (*Transmission Control Protocol*) and IP(*Internet Protocol*) are the most commonly-used protocols for using the Internet. (But there are others at lower levels.) The combination is simply known as TCP/IP. The Internet is a packet switching system. Any message is broken into packets that are transmitted independently (sometimes by different routes). These packets are called datagrams. The route chosen for each datagram depends on the traffic at any point in time. Each datagram has a header of between 20 and 60 bytes, followed by the payload of up to 65,515 bytes of data. The header consists of, amongst other data:

1. The version number of the protocol in use
2. The IP address of the sender (or source, or origin)
3. The IP address of recipient (or destination)

TCP breaks down a message into packets. At the destination, it reassembles packets into messages. It attaches a checksum to each packet. If the checksum doesn't match the computed checksum at the destination, the packet is retransmitted.

Thus TCP ensures reliable transmission of information. In summary, TCP:

1. Provides retransmission of lost data

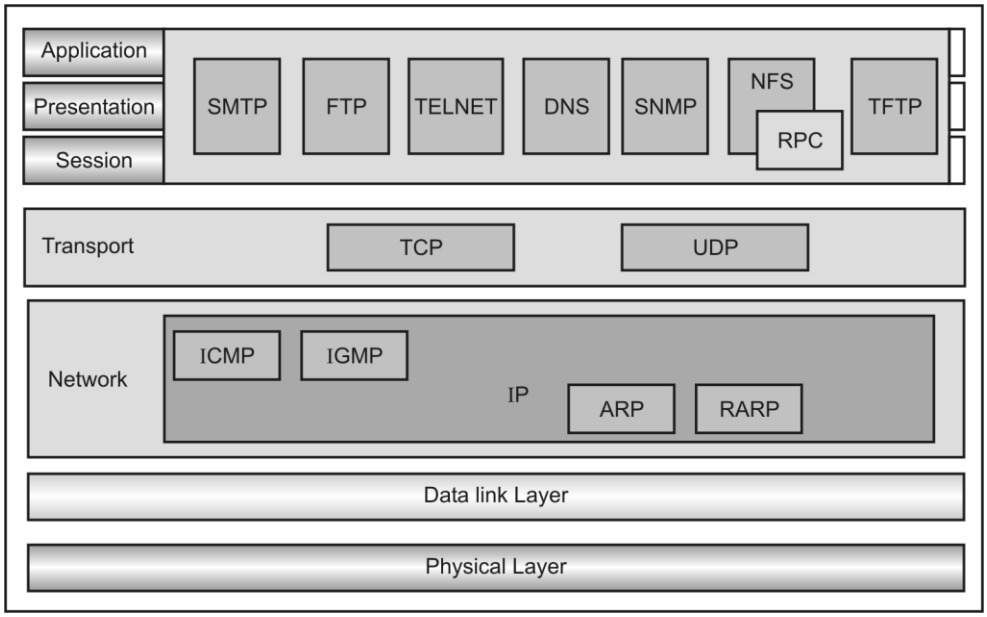
2. Ensures delivery of data in the correct order

IP is concerned with routing. IP attaches the address of the destination of each packet. IP ensures that packets get to the right place. TCP is the higher-level protocol that uses the lower-level IP. When an application is written, the general principle is to use the highest level protocol that you can, provided that it provides the functionality and performance that is required. Many applications can be written using TCP/IP. For example, a Web browser can be written in Java using only URLs, without any explicit mention of sockets. On each machine an application program makes calls on procedures in the transport layer (normally TCP). In turn the transport layer makes calls on the Internet layer (normally IP). In turn the Internet layer makes calls on the physical layer, which is different depending on the technology of the communication link.

At the destination machine, information is passed up through the layers to the application program. Each application program acts as if it is communicating directly with the application on another machine. The lower levels of the communication software and hardware are invisible. This four-layer model is sufficient for understanding Internet software. But there are other models that use a different number of layers, like the ISO seven-layer model.

3.2. THE IP PROTOCOL SUITE (TCP/IP):

The TCP/IP model was developed prior to the OSI model. The TCP/IP model is not exactly similar to the OSI model. The TCP/IP model consists of five layers: the application layer, transport layer, network layer, data link layer and physical layer. The first four layers provide physical standards, network interface, internetworking, and transport functions that correspond to the first four layers of the OSI model and these four layers are represented in the TCP/IP model by a single layer called the application layer. TCP/IP is a hierarchical protocol made up of interactive modules, and each of them provides specific functionality. Here, hierarchical means that each upper-layer protocol is supported by two or more lower-level protocols.



Network Access Layer

A network layer is the lowest layer of the TCP/IP model. A network layer is the combination of the Physical layer and Data Link layer defined in the OSI reference model. It defines how the data should be sent physically through the network. This layer is mainly responsible for the transmission of the data between two devices on the same network. The functions carried out by this layer are encapsulating the IP datagram into frames transmitted by the network and mapping of IP addresses into physical addresses. The protocols used by this layer are Ethernet, token ring, FDDI, X.25, frame relay.

Internet Layer

An internet layer is the second layer of the TCP/IP model. An internet layer is also known as the network layer. The main responsibility of the internet layer is to send the packets from any network, and they arrive at the destination irrespective of the route they take. Following are the protocols used in this layer are:

IP Protocol: IP protocol is used in this layer, and it is the most significant part of the entire TCP/IP suite. Following are the responsibilities of this protocol

1. **IP Addressing**: This protocol implements logical host addresses known as IP addresses. The IP addresses are used by the internet and higher layers to identify the device and to provide internetwork routing.
2. **Host-to-host communication:** It determines the path through which the data is to be transmitted.
3. **Data Encapsulation and Formatting**: An IP protocol accepts the data from the transport layer protocol. An IP protocol ensures that the data is sent and received securely; it encapsulates the data into a message known as IP datagram.

4. **Fragmentation and Reassembly** : The limit imposed on the size of the IP datagram by data link layer protocol is known as Maximum Transmission Unit (MTU). If the size of the IP datagram is greater than the MTU unit, then the IP protocol splits the datagram into smaller units so that they can travel over the local network. Fragmentation can be done by the sender or intermediate router. At the receiver side, all the fragments are reassembled to form an original message.
5. **Routing:** When IP datagram is sent over the same local network such as LAN, MAN, WAN, it is known as direct delivery. When source and destination are on the distant network, then the IP datagram is sent indirectly. This can be accomplished by routing the IP datagram through various devices such as routers.

ARP Protocol: ARP stands for Address Resolution Protocol. ARP is a network layer protocol which is used to find the physical address from the IP address. The two terms are mainly associated with the ARP Protocol :

ARP request : When a sender wants to know the physical address of the device, it broadcasts the ARP request to the network.

ARP reply : Every device attached to the network will accept the ARP request and process the request, but only the recipient recognizes the IP address and sends back its physical address in the form of ARP reply. The recipient adds the physical address both to its cache memory and to the datagram header

ICMP Protocol: ICMP stands for Internet Control Message Protocol. It is a mechanism used by the hosts or routers to send notifications regarding datagram problems back to the sender.

A datagram travels from router-to-router until it reaches its destination. If a router is unable to route the data because of some unusual conditions such as disabled links, a device is on fire or network congestion, then the ICMP protocol is used to inform the sender that the datagram is undeliverable. An ICMP protocol mainly uses two terms:

ICMP Test: ICMP Test is used to test whether the destination is reachable or not.

ICMP Reply: ICMP Reply is used to check whether the destination device is responding or not.

The core responsibility of the ICMP protocol is to report the problems, not correct them. The responsibility of the correction lies with the sender.

ICMP can send the messages only to the source, but not to the intermediate routers because the IP datagram carries the addresses of the source and destination but not of the router that it is passed to.

Transport Layer

The transport layer is responsible for the reliability, flow control, and correction of data which is being sent over the network.

The two protocols used in the transport layer are User Datagram protocol and Transmission control protocol.

3.3 UDP

Most applications use TCP. However, an example of a situation in which it is desirable to use a lower-level protocol is the case of audio *streaming*. If you want to download a sound file, it can take some time, even though it may be compressed. You have to wait (perhaps some considerable time, relatively speaking) for the complete file to download, before it can be played. An alternative is to listen to the sound as it is being downloaded - which is called *streaming*. One of the most popular technologies is called RealAudio. RealAudio does not use TCP because of its overhead. The sound file is sent in IP packets using the UDP (*User Datagram Protocol*) instead of TCP. UDP is an unreliable protocol, since: It doesn't guarantee that a packet will arrive and the packet arrives in the right order. UDP doesn't re-send a packet if it is missing or there is some other error, and it doesn't assemble packets into the correct order. But it is faster than TCP. In this application, losing a few bits of data is better than waiting for the re-transmission of some missing data. The application's major mission is to keep playing the sound without interruption. (In contrast, the main goal of a file transfer program is to transmit the data accurately.) The same mechanism is used with video streaming. UDP is a protocol at the same level as TCP, above the level of IP.

3.4 IP ADDRESSES

An *IP address* is a unique address for every host computer in the world. Consists of 4 bytes or 32 bits. This is represented in *quad notation* (or *dot* notation) as four 8-bit numbers, each in the range 0 to 255, e.g. 131.123.2.220.

IP addresses are registered so that they stay unique.

You can find the IP address of the local machine under Windows NT by typing the following command at the DOS prompt in a console window:

ipconfig

Under Unix or Linux, this command is:

ifconfig

Exercise: Type these commands at the Windows DOS command prompt and/or the Unix/Linux prompt.

The IP address 127.0.0.1 is a special address, called the *local loopback address*, that denotes the local machine. A message sent to this address will simply return to the sender, without leaving the sender. It is useful for testing purposes.

3.5. DOMAIN NAMES

A *domain name* is the user-friendly equivalent of an IP address. It is used because the numbers in an IP address are hard to remember and use. It is also known as a *host name*.

Example:

cs.stmarys.ca

Such a name starts with the most local part of the name and is followed by the most general. The whole name space is a tree, whose root has no name. The first level in the tree is something like com, org, edu, ca, etc. The parts of a domain name don't correspond to the parts of an IP address. Indeed, domain names don't always have 4 parts - they can have 2, 5 or whatever. All applications that use an address should work whether an IP address or a domain name is used. In fact, a domain name is converted to an IP address before it is used.

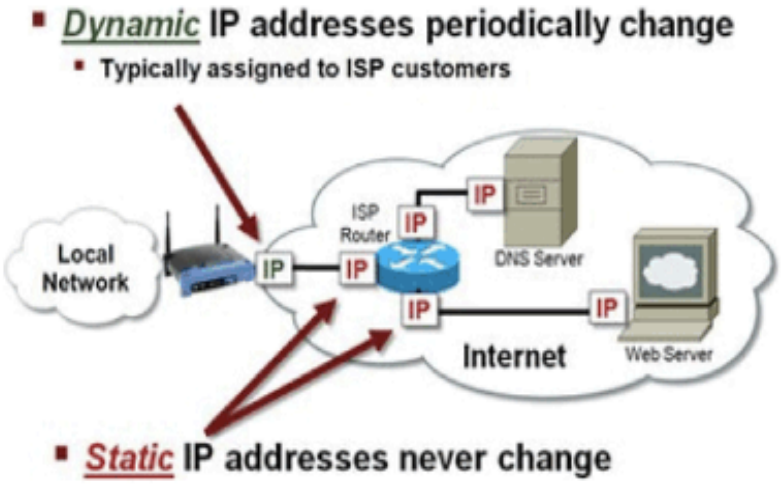
The Domain Name System: A program, say a Web browser, that wants to use a domain address usually needs to convert it into an IP address before making contact with the server. The *domain name system* (DNS) provides a mapping between IP addresses and domain names. All this information cannot be located in one place, so it is held in a *distributed database*.

3.6. STATIC IP ADDRESSES

A static IP address is an IP address that always stays the same. If you have a web server, FTP server, or other Internet resource that must have an address that cannot change, you can get a static IP address from your Internet Service Provider (ISP). A static IP address is usually more expensive than a dynamic IP address, and some ISPs do not supply static IP addresses. You must configure a static IP address manually.

3.7. DYNAMIC IP ADDRESSES

A dynamic IP address is an IP address that an ISP lets you use temporarily. If a dynamic address is not in use, it can be automatically assigned to a different device. Dynamic IP addresses are assigned using either DHCP or PPPoE. Dynamic Host Configuration Protocol (DHCP) is an Internet protocol that computers on a network use to get IP addresses and other information such as the default gateway. Some ISPs assign IP addresses through Point-to-Point Protocol over Ethernet (PPPoE). PPPoE adds some of the features of Ethernet and PPP to a standard dial-up connection. This network protocol allows the ISP to use the billing, authentication, and security systems of their dial-up infrastructure with DSL modem and cable modem products.



3.8. INTERNET PROTOCOL VERSION 6 (IPV6)

IP address is your digital identity. It's a network address for your computer so the Internet knows where to send you emails, data, etc. Internet Protocol version 6 (IPv6) is the most recent version of the Internet Protocol (IP), the communications protocol that provides an identification and location system for computers on networks and routes traffic across the Internet. IPv6 was developed by the Internet Engineering Task Force (IETF) to deal with the long-anticipated problem of IPv4 address exhaustion, and is intended to replace IPv4. IPv6 is an Internet Layer protocol for packet-switched internetworking and provides end-to-end datagram transmission across multiple IP networks, closely adhering to the design principles developed in the previous version of the protocol, Internet Protocol Version 4 (IPv4).

3.9. MAC ADDRESS:

MAC address is the physical address, which uniquely identifies each device on a given network. To make communication between two networked devices, we need two addresses: IP address and MAC address. It is assigned to the NIC (Network Interface card) of each device that can be connected to the internet.

- It stands for Media Access Control, and also known as Physical address, hardware address, or BIA (Burned In Address).
- It is globally unique; it means two devices cannot have the same MAC address. It is represented in a hexadecimal format on each device, such as 00:0a:95:9d:67:16.
- It is 12-digit, and 48 bits long, out of which the first 24 bits are used for OUI(Organization Unique Identifier), and 24 bits are for NIC/vendor-specific.

- It works on the data link layer of the OSI model.
- It is provided by the device's vendor at the time of manufacturing and embedded in its NIC, which ideally cannot be changed.
- The ARP protocol is used to associate a logical address with a physical or MAC address.

3.10. TCP AND UDP PORTS:

Transportation layer conventions utilize the idea of ports and multiplexing/demultiplexing to convey information to individual services listening on network nodes. These ports are spoken to by a solitary 16-bit number, implying that they can constitute a range of numbers 0-65535. This range has been separated by the IANA (Internet Assigned Numbers Authority) into several different segments :

- Port 0 is not used for internet/network traffic, but it's sometimes utilized in communications going down between different programs on identical computers.
- Ports 1-1023 are suggested as system ports. These ports speak to official ports for most notable system administrations and many common network services. HTTP usually communicates at port 80, while FTP at port 21. In most working frameworks/OS, administrative-level access is expected to begin a program that tunes in on a system port.
- Ports 1024-49151 are known as registered ports. These ports are utilized for a bunch of different network functions and services that probably won't be very frequently used as ones that are on system ports. On most operating systems, any client of any entry-level can run a program tuning in on a Registered port.
- Port 49152-65535 are known as Ephemeral ports (or private ports). Ephemeral ports can't be enlisted with IANA and are commonly utilized for setting up outbound network traffic and connections. All TCP traffic needs one destination port and one source port to establish a connection. At the point when a client needs to speak with a server, the client will be appointed an ephemeral port to be utilized for simply that one connection, while the server tunes in on a static system framework or registered port.Not every single working operating system follows ephemeral port proposals of IANA. The ephemeral ports utilized for outbound associations comprise ports 49152 through 65535. Yet, this range of ports can change as per the operating system and framework you're dealing with. Quite a few times registered ports are utilized, however, no modern operating system will ever utilize a system port for outbound connections.

Here are some commonly used ports for reference:

PORT	Service	Description	Transport Protocol
7	Echo	Port just echoes whatever is sent to it. This feature can be used in many attacks, such as Smurf/Fraggle.	TCP and UDP
20 /21	File Transfer Protocol (FTP)	Port used by FTP protocol to send data to the client	TCP
22	Secure Shell (SSH)	Used as secure replacement protocol for Telnet	TCP and UDP
23	Telnet	Port used by Telnet to remotely connect to a workstation or server(unsecured)	TCP
25	Simple Mail Transfer Protocol (SMTP)	Used to send E-Mail over internet	TCP
53	Domain Name System (DNS)	Port for DNS requests, network routing, and zone transfers	TCP and UDP
67 /68	Dynamic Host Configuration Protocol (DHCP)	Used on networks that do not use static IP address assignment.	UDP
80	Hypertext Transfer Protocol (HTTP)	Used for browsing web-pages on a browser	TCP
110	Post Office Protocol (POP3)	Port used to retrieve complete contents of a server mailbox	TCP
143	Internet	IMAP4 is a new protocol to read	TCP and UDP

PORT	Service	Description	Transport Protocol
	Message Access Protocol (IMAP4)	an email with a wider range of operations	
194	Internet Relay Chat Protocol(IRC)	allows communication in the form of text between multiple parties, one or more clients can connect to a centralized server.	TCP and UDP
443	HTTP with Secure Sockets Layer (SSL)	Port used for secure web traffic	TCP and UDP
3389	Remote Desktop Protocol(RDP)	Port used by remote desktop to remotely manage other windows system	TCP and UDP

The transport layer has a lot of ports to accommodate lots of various applications protocols simultaneously. The ports are dictated by the type of transport layer connectivity.

Also, remember that there are lots of protocols in the application layer but not all of them require port numbers (like TCP or UDP). Internet Control Message Protocol (ICMP) is one of them.

3.11 APPLICATION LAYER PROTOCOL

A protocol is a set of rules and guidelines for communicating data. Rules are defined for each step and process during communication between two or more computers. Networks have to follow these rules to successfully transmit data.

An application layer is the topmost layer in the TCP/IP model. It is responsible for handling high-level protocols, issues of representation. This layer allows the user to interact with the application. When one application layer protocol wants to communicate with another application layer, it forwards its data to the transport layer. There is an ambiguity in the application layer. Every application cannot be placed inside the application layer except those who interact with the communication system. For example: a text editor cannot be considered in the application layer while a web browser using HTTP protocol to interact with the network where HTTP protocol is an application layer protocol.

Following are the main protocols used in the application layer

3.11.1. HTTP: HTTP stands for Hypertext transfer protocol. HTTP is an application protocol for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web, where hypertext documents include hyperlinks to other resources that the user can easily access, for example by a mouse click or by tapping the screen. HTTP was developed to facilitate hypertext and the World Wide Web. It transfers the data in the form of plain text, audio, video.

3.11.2. HTTPS: Hypertext Transfer Protocol Secure is an extension of the Hypertext Transfer Protocol (HTTP) for secure communication over a computer network, and is widely used on the Internet. In HTTPS, the communication protocol is encrypted using Transport Layer Security (TLS), or, formerly, its predecessor, Secure Sockets Layer (SSL). The protocol is therefore also often referred to as HTTP over TLS, or HTTP over SSL. SMTP: Simple Mail Transfer Protocol (SMTP) is an Internet standard for electronic mail (email) transmission. FTP: The File Transfer Protocol (FTP) is a standard network protocol used for the transfer of computer files between a client and server on a computer network.

3.11.3. SNMP: SNMP stands for Simple Network Management Protocol. It is a framework used for managing the devices on the internet by using the TCP/IP protocol suite. It gathers data by polling the devices on the network from a management station at fixed or random intervals, requiring them to disclose certain information. It is a way that servers can share information about their current state, and also a channel through which an administrator can modify predefined values. The Port number of SNMP is 161(TCP) and 162(UDP).

3.11.4. SMTP: SMTP stands for Simple mail transfer protocol. It stands for Simple Mail Transfer Protocol. It is a part of the TCP/IP protocol. Using a process called “store and forward,” SMTP moves your email on and across networks. It works closely with something called the Mail Transfer Agent (MTA) to send your communication to the right computer and email inbox. The Port number for SMTP is 25.

3.11.5. DNS: DNS stands for Domain Name System. An IP address is used to identify the connection of a host to the internet uniquely. But, people prefer to use the names instead of addresses. Therefore, the system that maps the name to the address is known as Domain Name System. It stands for Domain Name System. Every time you use a domain name, therefore, a DNS service must translate the name into the corresponding IP address. For example, the domain name www.abc.com might translate to 198.105.232.4. The Port number for DNS is 53.

3.11.6. TELNET: It is an abbreviation for Terminal Network. It establishes the connection between the local computer and remote computer in such a way that the local terminal appears to be a terminal at

the remote system. Telnet stands for the **TEL**etype **NET**work. It helps in terminal emulation. It allows Telnet clients to access the resources of the Telnet server. It is used for managing files on the internet. It is used for the initial setup of devices like switches. The telnet command is a command that uses the Telnet protocol to communicate with a remote device or system. Port number of telnet is 23.

3.11.7. FTP: FTP stands for File transfer protocol. FTP is a standard internet protocol provided by TCP/IP used for transmitting the files from one host to another. It is mainly used for transferring the web page files from their creator to the computer that acts as a server for other computers on the internet. It is also used for downloading the files to the computer from other servers.

3.12 SUMMARY

The TCP/IP protocol suite is the foundation of data communication over the Internet. Each layer represents a set of rules for how to communicate, and every higher layer builds on the lower ones, offering a higher level of abstraction. Most development on the Internet, then, will involve the very highest layers of abstraction—the application layer, which includes the HTTP protocol which enables the world wide web, as well as the APIs. To really understand how communication works, and to make the best technological choices, it is important to understand the lower levels, too.
TCP offers great reliability on the transport layer; however, sometimes you may decide that the lightweight UDP protocol suits your needs better.
IP addresses, an important concept from the Internet layer, and are key for accessing devices across the Internet. The next generation of this protocol, IPv6, is used heavily in infrastructure and will be critical in preventing the exhaustion of addresses which is coming, hastened by the massive growth in mobile telephony, tablet computing, and the Internet of Things.
At the lowest level, the link layer, devices are identified not by IP addresses (as this isn't defined until the layer above) but rather by MAC addresses. This layer includes common types of local network, such as wired Ethernet and WiFi, as well as new personal area network (PAN) protocols, such as the 802.15.4 standard.

3.13 QUESTION

1. Describe use of following protocols: IP, TCP and UDP.
2. Write a note on DNS.
3. Compare static and dynamic IP address assignment.
4. Differentiate between TCP and UDP protocols.
5. What are the benefits of using IPV6 in IOT?
6. Write a note on the MAC address.
7. Explain the use of TCP and UDP ports. Give example.

3.14 REFERENCES

1. Designing the Internet of things by Adrian McEwen, Hakim Cassimally Wiley Publication.
2. Internet of Things by Prof. Sandeep Vishwakarma, Prof. Kiran Gurbani, Prof. Nitesh N. Shukla Himalaya Publishing House
3. Getting Started with the Internet of Things by Cuno Pfister O'Reilly

❄❄❄❄❄❄❄

4

THINKING ABOUT PROTOTYPING

Unit Structure

4.0 Objectives
4.1 Introduction
4.2 Benefits of prototyping
4.3 Sketching
4.4 Familiarity
4.5 Costs Versus Ease of Prototyping
4.6 Prototypes and Production
4.6.1 Changing Embedded Platform
4.6.2 Physical Prototypes And Mass Personalisation
4.6.3 Climbing Into The Cloud
4.7 Open Source Versus Closed Source
4.7.1 Why Closed Source?
4.7.2 Why Open Source?
4.7.3 Disadvantages of Open Source
4.7.4 Open Source as a Competitive Advantage
4.7.5 Open Source as a Strategic Weapon
4.7.6 Mixing Open and Closed Source
4.7.7 Closed Source For Mass Market Projects
4.8 Tapping into the Community
4.9 Summary
4.10 Chapter End Questions
4.11 References

4.0 OBJECTIVE:

- Students will be able to explain the importance of sketching a prototype before actually implementing.
- Student will get the knowledge of sketching method of a prototype product.
- Student will be able to explain the importance of prototype and production.
- Students will be able to differentiate between Open Source and Closed Source

4.1 INTRODUCTION

You may want to create an IOT device just for yourself or may want to produce millions of products, in both the cases the most sensible thing to do is to start making a prototype.

Making a prototype first has many benefits.

You will inevitably come across problems in your design that you need to change and iterate. Doing this with a single object is trivial compared to modifying hundreds or thousands of products.

The prototype is optimized for ease and speed of development and also the ability to change and modify it.

This prototype is relatively inexpensive, but you will most likely end up with something that is serviceable rather than polished and that will cost more than someone would be willing to pay for it in a shop.

At the end of this stage, you'll have a demonstrable product that you can use to convince yourself, your business partners, and your investors that your idea has legs and is worth trying to sell.

Finally, the process of manufacture will iron out issues of scaling up and polish.

4.2 BENEFITS OF PROTOTYPING :

One can come across problems in design that need to change and iterate. Doing this with a single object is trivial compared to modifying hundreds or thousands of products. Prototyping is optimized for ease and speed of development and also the ability to change and modify it

With the Internet of Things, we need to build three things in parallel:

i. Physical Thing

ii. Electronics to make the Thing smart

iii. Internet service that we'll connect to.

At the end of this stage, you'll have an object that works. And if you are planning to move to production, 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. The final product will be cheaper per unit and more professional, but will be much more expensive to change.

For one thing, your project has its own goals and requirements. For another, new microcontrollers are constantly coming onto the market; best practice and popularity of server software stacks is ever changing; and rapid prototyping continues to evolve.

4.3 SKETCHING :

When you are working on your prototype, the first step you will be doing is to jot down some ideas or draw out some design ideas with pen and paper. Then extend it to also include sketching in hardware and software. With a sketch, the designer is able to communicate a potential solution that could be built on later to form a refined design.

Sketching enables you to brainstorm, explore multiple ideas, define flows, communicate with team members all why being quick and cheap

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. The focus isn't on fidelity of the prototype but rather on the ease and speed with which you can try things out.

First, jot down some ideas or draw out some design ideas with pen and paper. It is also necessary to include sketching in hardware and software.

Explore the problem space: iterate through different approaches and ideas to find out what works and what doesn't.

The focus shouldn't be on the fidelity of the prototype but rather on the ease and speed with which you can try things out.

For the physical design, you can use LEGO, cogs, three dimensional forms, foam core or cardboard.

Three things to keep in mind while designing

i. Graphic design of the device

ii. Physical Hardware

iii. Server Software to tie the rest of the system

4.4 FAMILIARITY

Familiarity involves knowledge or know-how about tools and techniques for creation, designing and development.

For example

If you can already program like a whiz 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.

4.5 COSTS VERSUS EASE OF PROTOTYPING

Although familiarity with a platform may be attractive in terms of 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.

This trade-off is not hard and fast, but it is beneficial if you can choose a prototyping platform in performance/capabilities bracket similar to a final production solution. That way, you will be less likely to encounter any surprises over the cost, or even the wholesale viability of your project, down the line.

For example, the cheapest possible way of creating an electronic device might currently be an AVR microcontroller chip, which you can purchase from a component supplier. However this would require you to sweat the details of how to connect the pins to other components and how to flash the chip with new code. For many people, this platform would not be viable for an initial prototype.

Stepping upwards, you could look at an Arduino or similar. It would have exactly the same chip, but it would be laid out on a board with labelled headers to help you wire up components more easily, have a USB port where you could plug in a computer, and have a well-supported IDE to help make programming it easier.

For more money again, you could look at the Beagle Bone, which runs Linux and has enough processing power and RAM to be able to run a high-level programming language: libraries are provided within the concurrent programming toolkit Node.js for JavaScript to manipulate the input/output pins of the board.

If you choose not to use an embedded platform, you could think about using a smartphone instead. Smartphones might cost more than an embedded platform, they have many of the same features that make the cheaper platforms attractive: connection to the Internet, input capabilities and output capabilities. You can often program them in a choice of languages of high or low level, from Objective C and Java, to Python or HTML and JavaScript. Finally, a common PC might be an option for a prototype. These PCs cost most among all the embedded options and have a host of Internet connection and I/O possibilities. You can program them in whatever language you already know how to use.

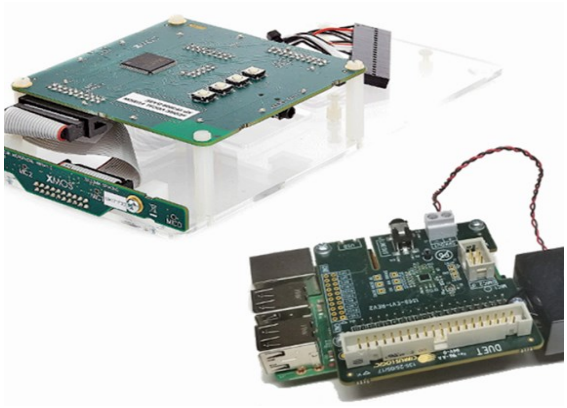
For the first prototype, the cost is probably not the most important issue: the smartphone or computer options are particularly convenient if you already have one available. Although prototyping a “thing” using a piece of general computing equipment might seem like a sideways step, depending on your circumstances, it may be exactly the right thing to do to show whether the concept works and get people interested in the project, to collaborate on it, or to fund it.

If your device has physical interactions, you will find that a PC is not optimized for this kind of work. It doesn’t expose GPIO pins. An electronics prototyping board, 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. As always, there is no single “right answer” but a set of trade-offs. The prototype is something that will get you started, and the experience of making it will teach you much more about the final best platform for your device than any book.

4.6 PROTOTYPES AND PRODUCTION

4.6.1 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 you've started with a free-form, powerful programming platform, you may find that porting the code to a more restricted, cheaper, and smaller device will bring many challenges.

If you port to a more powerful platform, you may be able to rewrite your code in a more modern, high-level way or simply take advantage of faster processor speed and more RAM. 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.

4.6.2 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.

An aspect that may be of interest is in the way that digital fabrication tools can allow each item to be slightly different, letting you personalise each device in some way.

There are challenges in scaling this to production, as you will need to keep producing the changeable parts in quantities of one, but mass personalisation, as the approach is called, means you can offer something unique with the accompanying potential to charge a premium.

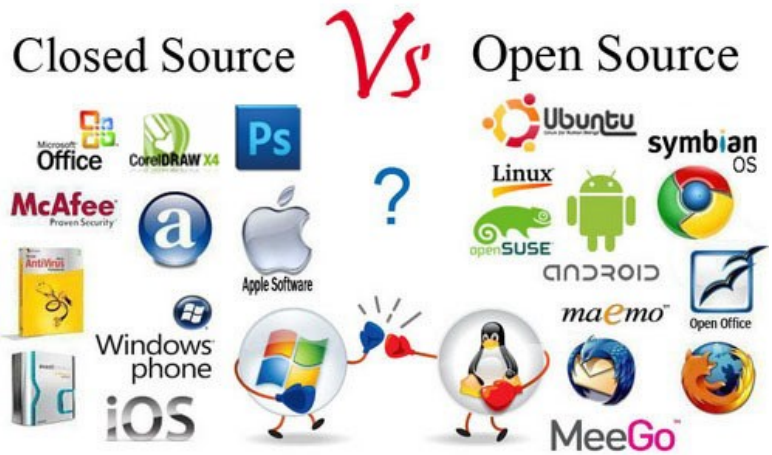
4.6.3 CLIMBING INTO THE CLOUD

The server software is the easiest component to take from prototype into production.

It might involve switching from a basic web framework to something more involved (particularly if you need to add user accounts and the like), but you will be able to find an equivalent for whichever language you have chosen. That means most of the business logic will move across with minimal changes. Beyond that, 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.

4.7 OPEN SOURCE VERSUS CLOSED SOURCE



From the perspective of Internet of Things, we look at two issues

i. Your assertion, as the creator, of your Intellectual Property rights.

ii. Your users' rights to freely tinker with your creation.

As a creative person, you may be torn between your own desire to learn how things work and modify and re-use them and the worry that if other people were to use that right on your own design/invention/software, you might not get the recognition and earnings that you expect from it.

This tension between the closed and open approaches is rather interesting, especially when applied to a mix of software and hardware, as we find with Internet of Things devices.

Open Source	Closed Source
Provides the original source code so that advanced user can modify it to make it work better	No one can duplicate or distribute without permission of the company
Open Source software available to all free of cost	Closed source software is protected by copyrights
Any individual can develop or modify	Development is centralised, undertaken by expert developers
Quality assurance or bug removal is done by small team	Large number of individuals are available for improving quality and bug removal
Android	Apple

4.7.1 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.

That person would have to instead reverse-engineer the functionality of the hardware and software. In addition, simply copying the design slavishly would also infringe copyright.

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 (whereas after you've licensed something as open source, you can't simply revoke that licence). Closed-source software is built with users in mind.

Closed-source software offers ongoing technical support for its users. Closed-source software is usually 1 or 2 steps ahead of open-source software

4.7.2 WHY OPEN?

- In the open source model, you release the sources that you use to create the project to the whole world.

You might publish the software code to GitHub, the electronic schematics using Fritzing or SolderPad, and the design of the housing/shell to Tingiverse

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.

This is also a gift economy:

- you can use other people's free and open source contributions within your own project.
- Forums and chat channels exist all over the Internet.
- 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.
- economy: you can use other people's free and open source contributions within your own project.
- Faster time to market. Because open source solutions are openly available and can be explored for free, it's often much faster to investigate options and get solutions off the ground.
- Open source solutions should be thought of as more than just free software, the fact that they require no licensing fees remains a decisive advantage when looking at the total cost of deploying a solution

4.7.3 Disadvantages of Open Source

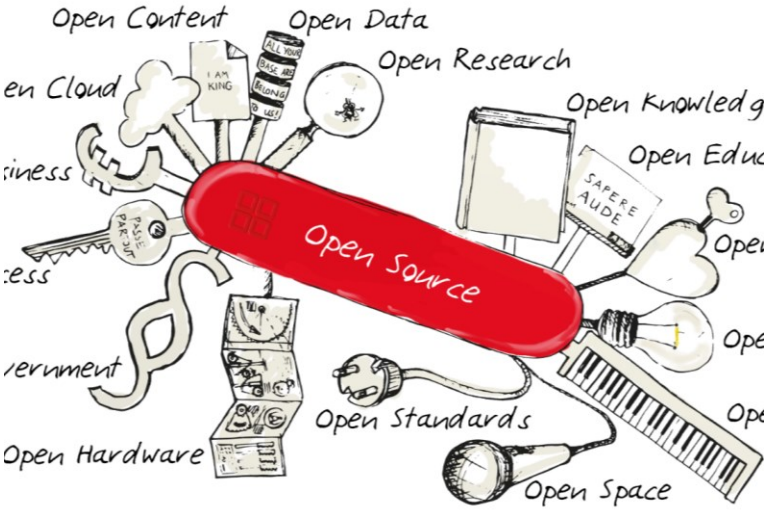
- People will steal my idea"
- People don't want to listen to your idea but tell you about their great idea.
- People use your contribution in a way that interests them.
- Deciding to release as open source may take more resources

- 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.

4.7.4 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.
- 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.
- The "geek" community often choose a product because, rather than being a commercial "black box", it, for example, exposes a Linux shell or can communicate using an open protocol such as XML. This community can be your biggest ally.

4.7.5 Open Source as a Strategic Weapon



- One step further in the idea of open source used aggressively is the idea of businesses using open source strategically to further their interests (and undermine their competitors).
- Companies can use improvements in open source versions of complementary products to increase demand for their products.
- 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.
- This dynamic is fascinating 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.

4.7.6 Mixing Open and Closed Source

While many businesses can exist as purely one or the other, you shouldn't discount having both coexist.
As long as you don't make unfounded assertions about how much you use open software, it's still possible to be a "good citizen" who contributes back to some projects whether by contributing work or simply by helping others in forums while also gaining many of the advantages of open source.

4.7.7 CLOSED SOURCE FOR MASS MARKET PROJECTS

One edge case for preferring closed source when choosing a licence may be when you can realistically expect that a project might be not just successful but huge, that is, a mass market commodity.
While "the community" of open source users is a great ally when you are growing a platform by word of mouth, if you could get an existing supply and distribution chain on your side, the advantage of being first to market and doing so cheaper may well be the most important thing.
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.

4.8 TAPPING INTO THE COMMUNITY

- Community is not an exclusive feature of open source projects.
- While thinking about which platform you want to build for, having a community to tap into may be vital or at least useful.
- Communities may be large, like support for Arduino platform or small, like community support for Chumby Hacker Board.
- 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.
- The perceived danger of sharing an idea or an implementation or a question with other people is looking like an idiot in public.
- While many parts of many Internet communities are much more sympathetic to this fear than one might expect, the mask of anonymity on the Internet can seem to permit people to be less supportive or simply more rude than you might hope for.

4.9 SUMMARY

- With the Internet of Things, we need to build three things in parallel:
  1. Physical Thing
  2. Electronics to make the Thing smart
  3. Internet service that we'll connect to.

- When you are working on your prototype, the first step you will be doing is to jot down some ideas or draw out some design ideas with pen and paper. Then extend it to also include sketching in hardware and software. With a sketch, the designer is able to communicate a potential solution that could be built on later to form a refined design.
- Three things to keep in mind while designing
  1. Graphic design of the device
  2. Physical Hardware
  3. Server Software to tie the rest of the system

- If you choose not to use an embedded platform, you could think about using a smartphone instead. Smartphones might cost more than an embedded platform, they have many of the same features that make the cheaper platforms attractive: connection to the Internet, input capabilities and output capabilities. You can often program them in a choice of languages of high or low level, from Objective C and Java, to Python or HTML and JavaScript.
- Although prototyping a "thing" using a piece of general computing equipment might seem like a sideways step, depending on your circumstances, it may be exactly the right thing to do to show whether the concept works and get people interested in the project, to collaborate on it, or to fund it.
- When you scale up, you may well have to think about moving to a different platform, for cost or size reasons.

- There are challenges in scaling this to production, as you will need to keep producing the changeable parts in quantities of one, but mass personalisation, as the approach is called, means you can offer something unique with the accompanying potential to charge a premium.
- As a creative person, you may be torn between your own desire to learn how things work and modify and re-use them and the worry that if other people were to use that right on your own design/invention/software, you might not get the recognition and earnings that you expect from it.

4.10 CHAPTER END QUESTIONS

1. Explain benefits of prototyping.
2. Differentiate between open source and closed source
3. What are the advantages of open source softwares?
4. Why should you mix open and closed source for the Internet of Things?

4.11 REFERENCE

Books and References:

Sr. No.	Title	Authors	Publisher	Edition	Year
1	Designing the Internet of Things	Adrian McEwen, Hakim Cassimally	Wiley	First	2014
2	Internet of Things – Architecture and Design	Rajkamal	McGraw Hill	First	2017
3	Getting Started with the Internet of Things	Cuno Pfister	O'Reilly	Sixth	2018
4	Getting Started with Raspberry Pi	Matt Richardson and Shawn Wallace	SPD	Third	2016

Website:

www.youtube.com

https://internetofthingswiki.com

www.edurekha.co/blog/iot-tutorial

❄❄❄❄❄❄❄

5

PROTOTYPING EMBEDDED DEVICES

Unit Structure

5.0 Objectives
5.1 Introduction
5.2 Electronics
5.3 Sensors
5.4 Actuators
5.5 Scaling up the Electronics
5.6 Embedded Computing Basics
5.6.1 Microcontroller
5.6.2 System-On-Chips
5.6.3 Choosing Your Platform
5.7 Arduino
5.8 Raspberry Pi
5.8.1 Developing on the Raspberry Pi
5.9 Summary
5.10 Chapter End Questions

5.0 OBJECTIVE:

- Students will be able to understand different types electronic components such as sensors and actuators and their importance in IoT.
- Student will get the knowledge on what aspects needs to be looked into while selecting different platforms for building IoT projects..
- Student will be able to differentiate Arduino and Raspberry Pi

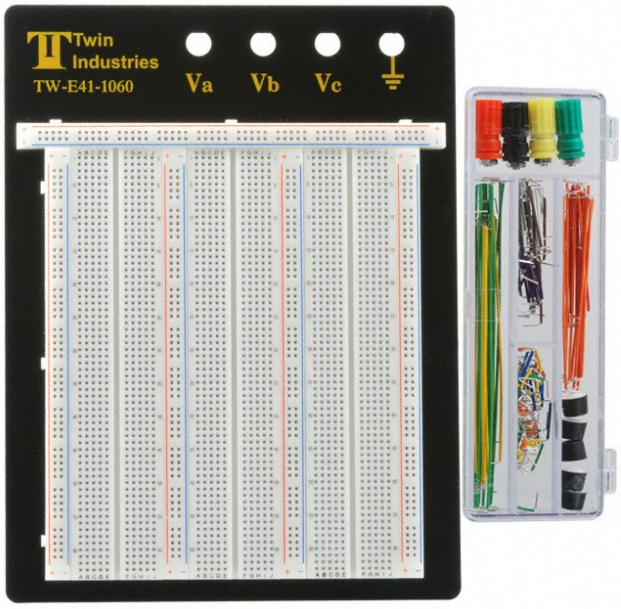
5.1 INTRODUCTION

You may want to create an IOT device just for yourself or may want to produce millions of products, in both the cases the most sensible thing to do is to start making a prototype. Making a prototype first has many benefits. While making the IoT product one should understand what kind of electronics devices one should select and the type of sensors and actuators that will be required to enhance the product.

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, which also means you can experiment with different options quickly and easily.

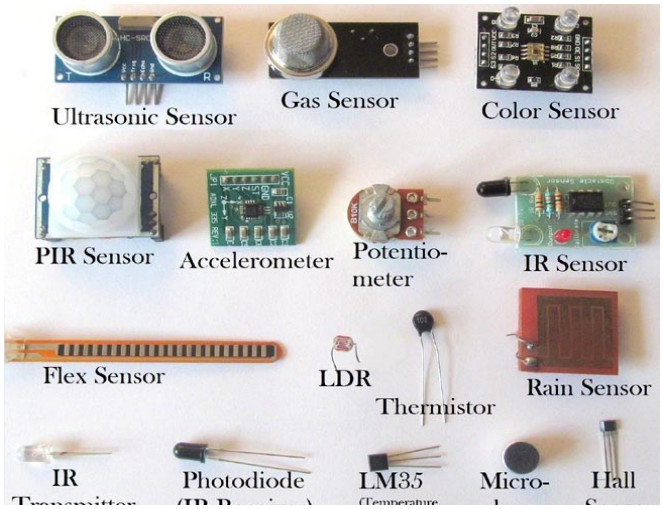
5.2 ELECTRONICS :



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, which also means you can experiment with different options quickly and easily.
When it comes to thinking about the electronics, it's useful to split them into two main categories:

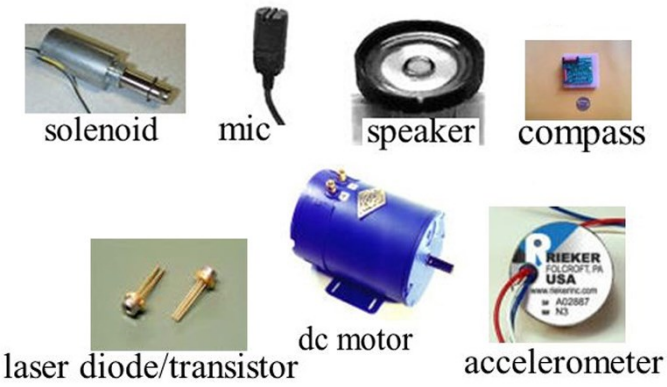
- **Sensors:** Sensors are the ways of getting information into your device, finding out things about your surroundings.
- **Actuators:** Actuators are the outputs for the device — the motors, lights, and so on, which let your device do something to the outside world.

5.3 SENSORS



- Pushbuttons and switches, which are probably the simplest sensors, allow some user input.
- Potentiometers (both rotary and linear) and rotary encoders enable you to measure movement.
- Sensing the environment is another easy option. Light-dependent resistors (LDRs) allow measurement of ambient light levels, thermistors and other 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, but piezo elements (used in certain types of microphones) can also be used to respond to vibration.
- Distance-sensing modules, which work by bouncing either an infrared or ultrasonic signal off objects, are readily available and as easy to interface to as a potentiometer.

5.4 ACTUATORS

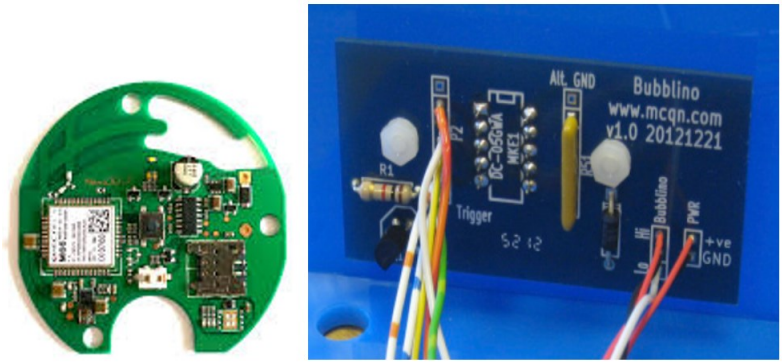


- One of the simplest and yet most useful actuators is light, because it is easy to create electronically and gives an obvious output. Light-emitting diodes (LEDs) typically come in red and green but also white and other colours.
- More complicated visual outputs also are available, such as LCD screens to display text or even simple graphics.
- Piezo elements, as well as responding to vibration, can be used to create it, so you can use a piezo buzzer to create simple sounds and music. Alternatively, you can wire up outputs to speakers to create more complicated synthesised sounds.
- Solenoids can by used to create a single, sharp pushing motion, which could be useful for pushing a ball off a ledge or tapping a surface to make a musical sound.
- More complicated again are motors.
- Stepper motors can be moved in steps. 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.

- Alternatively, if you want a motor that will turn to a given angle, you would need a servo. Although a servo is more controllable, it tends to have a shorter range of motion, often 180 or fewer degrees.

5.5 SCALING UP THE ELECTRONICS



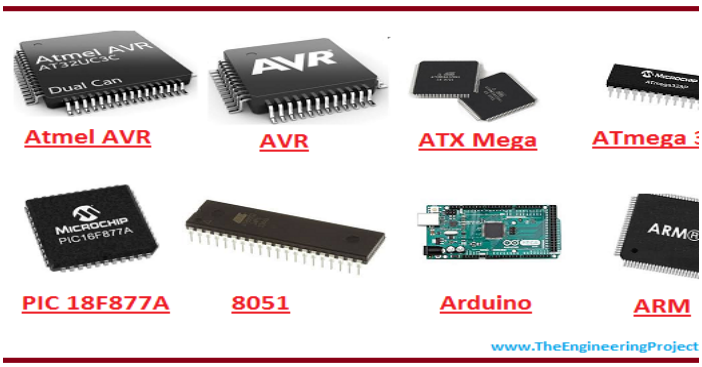
From the perspective of the electronics, the starting point for prototyping is usually a "breadboard". This lets you push-fit components and wires to make up circuits without requiring any soldering and therefore makes experimentation easy. Moving beyond the protoboard option tends to involve learning how to lay out a PCB. This task isn't as difficult as it sounds, for simple circuits at least, and mainly involves learning how to use a new piece of software and understanding some new terminology.

For small production runs, you'll likely use through-hole components, so called because the legs of the component go through holes in the PCB and tend to be soldered by hand. You will often create your designs as companion boards to an existing microcontroller platform.

When you want to scale things even further, moving to a combined board allows you to remove any unnecessary components from the microcontroller board, and switching to surface mount components — where the legs of the chips are soldered onto the same surface as the chip — eases the board's assembly with automated manufacturing lines.

5.6 EMBEDDED COMPUTING BASICS

5.6.1 Microcontroller



Internet of Things devices take advantage of more tightly integrated and miniaturised solutions—from the most basic level of microcontrollers to more powerful system-on-chip (SoC) modules. These microcontrollers are the engines of countless sensors and automated factory machinery. They are the last bastions of 8-bit computing in a world that's long since moved to 32-bit and beyond.

Unlike the market for desktop computer processors, which is dominated by two manufacturers (Intel and AMD), the microcontroller market consists of many manufacturers (Atmel, Microchip, NXP, Texas Instruments, to name a few), each with a range of chips for different applications. The ubiquitous Arduino platform is based around Atmel's AVR ATmega family of microcontroller chips.

The on-board inclusion of an assortment of GPIO pins and ADC circuitry means that microcontrollers are easy to wire up to all manner of sensors, lights, and motors. Because the devices using them are focused on performing one task, they can dispense with most of what we would term an operating system, resulting in a simpler and much slimmer code footprint than that of a SoC or PC solution.

In these systems, functions which require greater resource levels are usually provided by additional single-purpose chips.

5.6.2 System-On-Chips

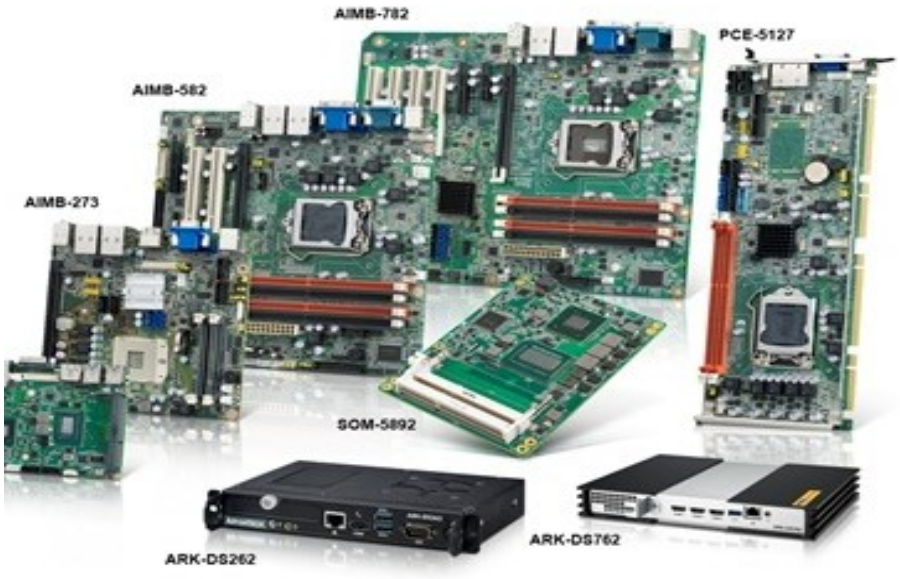
In between the low-end microcontroller and a full-blown PC sits the SoC (for example, the BeagleBone or the Raspberry Pi).

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, nudging into the gigahertz for top-end solutions, and include RAM measured in megabytes. 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 specialized embedded providers and the big OS players, such as Microsoft and Linux.

5.6.3 Choosing Your Platform



How to choose the right platform for your Internet of Things device?
This isn't to say that it's an impossible question—more that there are almost as many answers as there are possible devices.

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.

This involves some of the factors that you need to weigh—and possibly play off against each other—when deciding how to build your device.

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.
- Generally, you will use the processor speed as one of a number of factors when weighing up similar systems.
- 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—for example, processing video in real time—then you'll be looking at a SoC platform.

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. If you're handling large datasets on the device, that could govern how much space you need.
- You can often find ways to work around memory limitations, either in code or by handing off processing to an online service.
- It is difficult to give exact guidelines to the amount of RAM you will need, as it will vary from project to project.

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.
- ZigBee and Bluetooth LE protocol have a very low power-consumption profile.
- For remote or outdoor deployment, little beats simply using the mobile phone networks.

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, so choosing one of them reduces the need for an extra chip in your circuit.
- Instead of the microcontroller presenting itself as a device, some can also act as the USB "host". This configuration lets you connect items that would normally expect to be connected to a computer—devices such as phones, for example, using the Android ADK, additional storage capacity, or WiFi dongles.
- Devices such as WiFi dongles often depend on additional software on the host system, such as networking stacks, and so are better suited to the more computer-like option of SoC.

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.
- Even with access to mains electricity, the power consumption may be something to consider because lower consumption may be a desirable feature.
- 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.
- Therefore, a more powerful processor may not be a disadvantage even in a low-power embedded device.

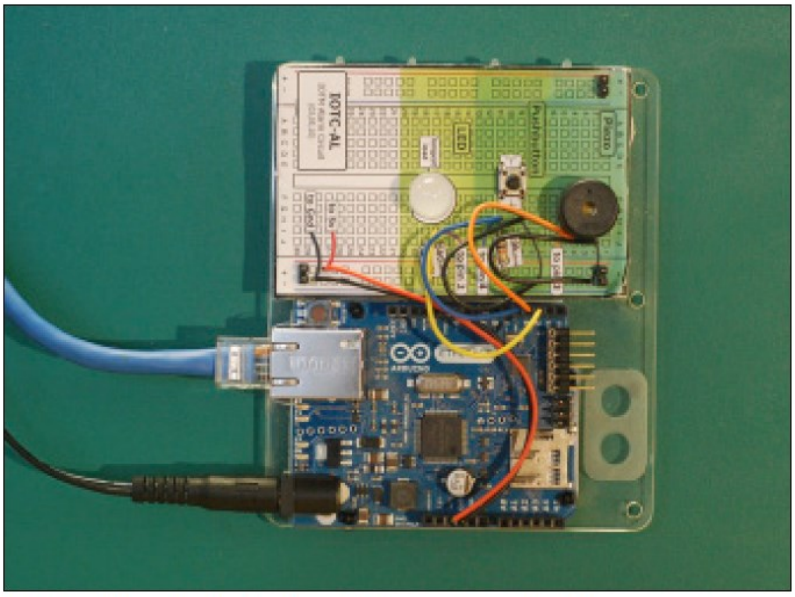
Interfacing with Sensors and Other Circuitry

- 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.
- Different microcontrollers or SoC solutions offer different mixtures of these interfaces in differing numbers.

Physical Size and Form Factor

- Earlier the limiting factor in the size of a chip was 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.

5.7 ARDUINO



An Arduino Ethernet board, plugged in, wired up to a circuit and ready for use.

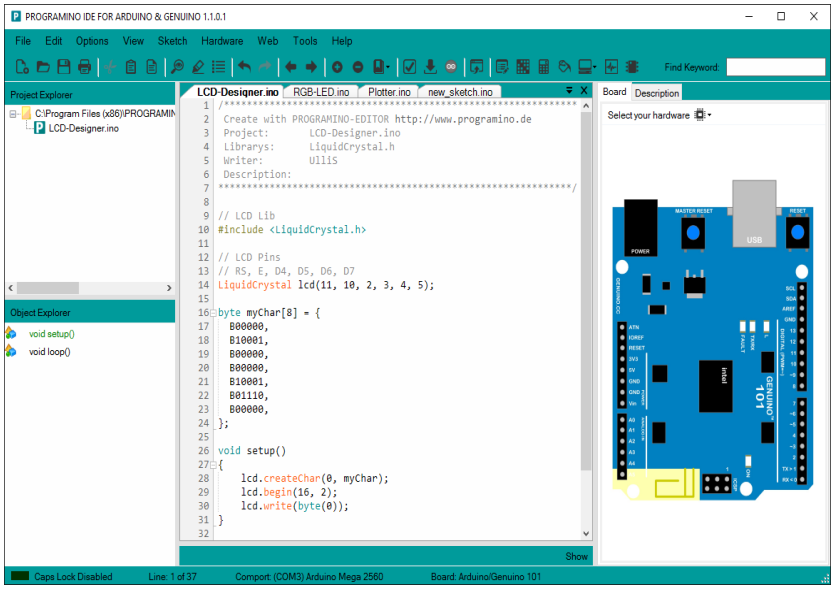
The poster child for the Internet of Things. Its birth was in Ivrea in Northern Italy in 2005. A group from the Interaction Design Institute Ivrea (IDII) wanted a board for its desig students to use to build interactive projects. An assortment of boards was around at that time, but they tended to be expensive, hard to use, or both. So, the team put together a board which was cheap to buy—around £20— and included an onboard serial connection to allow it to be easily programmed. Combined with an extension of the Wiring software environment, it made a huge impact on the world of physical computing.

A decision early on to make the code and schematics open source meant that the Arduino board could outlive the demise of the IDII and flourish. It also meant that people could adapt and extend the platform to suit their own needs. As a result, an entire ecosystem of boards, add-ons, and related kits has flourished. 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, and the open source ethos has encouraged the community to share circuit diagrams, parts lists, and source code. The "standard" Arduino board has gone through a number of iterations: Arduino NG, Diecimila, Duemilanove, and Uno.

Developing on the Arduino

- More than just specs, the experience of working with a board may be the most important factor, at least at the prototyping stage.
- The Arduino is optimised for simplicity, and this is evident from the way it is packaged for use.
- 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

Integrated Development Environment



- You usually develop against the Arduino using the integrated development environment (IDE)
- Although this is a fully functional IDE, based on the one used for the Processing language, it is very simple to use.
- 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

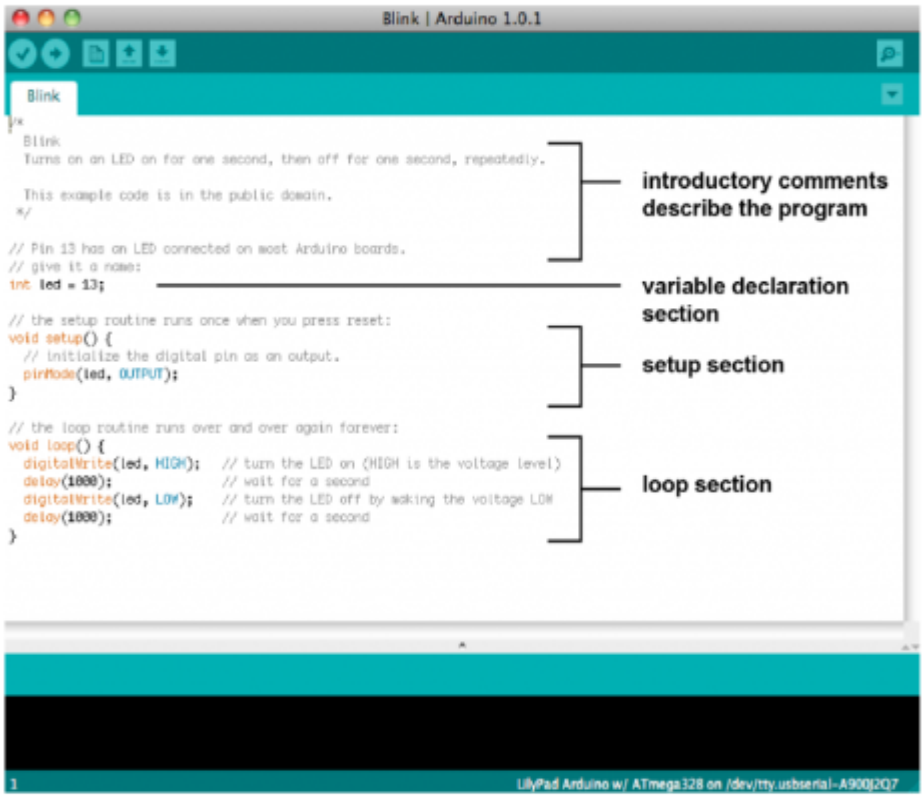
- Connecting to the board should be relatively straightforward via a USB cable.
- Sometimes you might have issues with the drivers or with permissions on the USB port, but they are usually swiftly resolved once and for good.
- After this, you need to choose the correct serial port and the board type.
- 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.

- 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 — the avr-gcc toolset.
- The avr-gcc toolset 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 dialect of C++ derived from the Wiring platform.
- 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".
- This variant of C++ tries to be forgiving about the ordering of code; for example, it allows you to call functions before they are defined.

- This alteration is just a nicety, but it is useful to be able to order things in a way that the code is easy to read and maintain, given that it tends to be written in a single file.
- 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.

Example:

```
// 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 this happens on your computer, you have ample opportunity to get detailed and possibly helpful information from the compiler about what the problem is.
- When the code is pushed to the Arduino, the rules of the game change.
- 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.
- Or perhaps your program leaks memory and eventually just stops working.
- Or (and worse) a programming error might make the code continue to work dutifully but give entirely the wrong results.
- 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.

Hardware

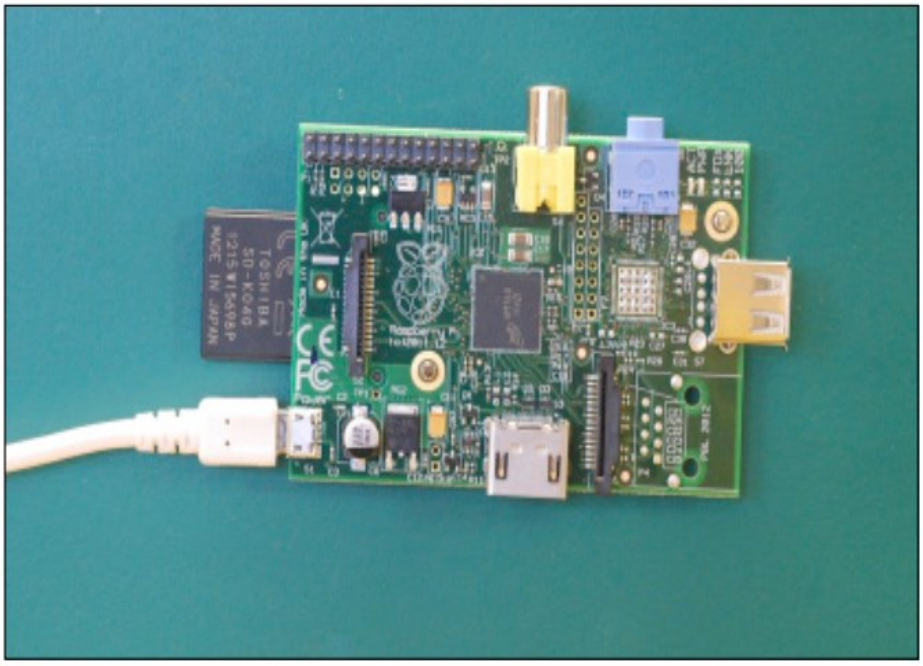
- The Arduino exposes a number of GPIO pins and is usually supplied with "headers". The headers are optimised for prototyping and for being able to change the purpose of the Arduino easily.
- Each pin is clearly labelled on the controller board. The details of pins vary from the smaller boards such as the Nano, the classic form factor of the Uno, and the larger boards such as the Mega or the Due.
- You can power the Arduino using a USB connection from your computer. This capability is usually quite convenient during prototyping because you need the serial connection in any case to program the board.
- The Arduino also has a socket for an external power supply, which you might be more likely to use if you distribute the project.
- Choosing one of the specialist boards isn't the only way to extend the capabilities of your Arduino. You can also 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; and all manner of other possibilities — so much so that an entire website, http://shieldlist.org/, is dedicated to comparing and documenting them.

Openness

- The Arduino project is completely open hardware and an open hardware success story. The only part of the project protected is the Arduino trademark, so they can control the quality of any boards calling themselves an Arduino.
- In addition to the code being available to download freely, the circuit board schematics and even the EAGLE PCB design files are easily found on the Arduino website.
- This culture of sharing has borne fruit in many derivative boards being produced by all manner of people.
- Some are merely minor variations on the main Arduino Uno, and in some cases, such as with the wireless-focused Arduino Fio board, what starts as a third-party board (it was originally the Funnel IO) is later adopted as an official Arduino-approved board

5.8 RASPBERRY PI



A Raspberry Pi Model B board. The micro USB connector only provides power to the board; the USB connectivity is provided by the USB host connectors (centre-bottom and centre-right).

The Raspberry Pi, unlike the Arduino, wasn't designed for physical computing at all, but rather, for education. The vision of Eben Upton, trustee and cofounder of the Raspberry Pi Foundation, was to build a computer that was small and inexpensive and designed to be programmed and experimented with, like the ones he'd used as a child, rather than to passively consume games on.

The Foundation gathered a group of teachers, programmers, and hardware experts to thrash out these ideas from 2006. The project has always taken some inspiration from a previous attempt to improve computer literacy in the UK: the "BBC Micro" built by Acorn in the early 1980s.The model names of the Raspberry Pi, "Model A" and "Model B", hark back to the different versions of the BBC Micro.

Difference between Arduino and Raspberry Pi

	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

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

So, 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 Arduino has a fraction of the raw processing power, memory, and storage required for it to run a modern OS.

Importantly, the Pi Model B has built-in Ethernet and can also use cheap and convenient USB Wi-Fi dongles, rather than having to use an extension "shield".

Note that although the specifications of the Pi are in general more capable than even the top-of-the-range Arduino Due, we can't judge them as "better" without considering what the devices are for!

Cases and Extension Boards

Because the Pi can be useful as a general-purpose computer or media centre without requiring constant prototyping with electronic components, one of the first demands enthusiasts have had was for convenient and attractive cases for it. Many makers blogged about their own attempts and have contributed designs to Tingiverse, Instructables, and others.

The Foundation has deliberately not authorised an "official" one, to encourage as vibrant an ecosystem as possible.

Beyond these largely aesthetic projects, extension boards and other accessories are already available for the Raspberry Pi. Many interesting kits are in development, such as the Gertboard, designed for conveniently playing with the GPIO pins. A lot of people are doing interesting things with their Pis, but as the platform is so much more high level and capable, the attention may be spread more thinly — from designing cases to porting operating systems to working on media centre plug-ins.

Physical computing is just one of the aspects that attention may be paid to.

5.8.1 Developing on the Raspberry Pi

The number of variables on the Raspberry Pi are much greater, and there is much more of an emphasis on being able to do things in alternative ways. However, "best practices" are certainly developing. It's worth checking on the Raspberry Pi websites, IRC channels, and so on, later to see how certain practices will have evolved.

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, Raspbian is a distro based on Debian.
- Occidentalis: This is Adafruit's customised Raspbian.
- For Internet of Things work, the Adafruit distro is recommended.

The main tweaks that interest us are

The sshd (SSH protocol daemon) is enabled by default, so you can connect to the console remotely. The device registers itself using zero-configuration networking (zero-conf) 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.

With the Raspberry Pi, you've already had to make decisions about the distro and download it. Now that distro needs to be unpacked on the SD card, which you purchase separately. Note that "Class 10" cards work best. After you boot up the Pi, you can communicate with it just as you'd communicate with any computer—that is, either with the keyboard and monitor that you've attached, or with the Adafruit distro, via ssh.

From Windows, you can use an SSH client such as PuTTY.
Programming Language
The Foundation suggests Python as a good language for educational programming.
Python makes the following tasks easier:

- Handling strings of character data
- Completely avoiding having to handle memory management
- Making calls to Internet services and parsing the data received
- Connecting to databases and more complex processing
- Abstracting common patterns or complex behaviours

Also, being able to take advantage of readily available libraries on PyPi may well allow simple reuse of code that other people have written, used, and thoroughly tested.
As always, you have to be aware of a few trade-offs, related either to the Linux platform itself or to the use of a high-level programming language
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
```

Programming Language:

- We specifically contrast Python with C++, as the low-level language used for Arduino programming.
- Python, as with most high-level languages, compiles to relatively large (in terms of memory usage) and slow code, compared to C++.
- Python is likely to be "fast enough" for most tasks, however, if the electronics of the sensors and actuators you are working with require split-second timing, Python might be too slow.
- Python handles memory management automatically.
- Manual memory management causes memory leaks which may build up and eventually end up with the device running out of memory and crashing.
- Linux itself arguably has some issues for "real-time" use.
- Many processes 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, they may cause problems that are entirely unrelated to your code.
- The trade-offs that may or may not be important to you, or rather more or less important than the features of the Pi and the access to a high-level programming language. The most important issue, again, is probably the ease of use of the environment.

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.
- 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.

- Python and other high-level languages also have mature testing tools which allow you to assert expected behaviours of your routines and test that they perform correctly.

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.
- Unlike those in the Arduino, the pins in the Raspberry Pi aren't individually labelled.
- The intention is rather that you will plug a cable (IDC or similar) onto the whole block, which leads to a "breakout board" where you do actual work with the GPIO.
- Alternatively, you can connect individual pins using a female jumper lead onto a breadboard.
- The Raspberry Pi doesn't have any analogue inputs (ADC), which means that options to connect it to electronic sensors are limited, out of the box, to digital inputs (that is, on/off inputs such as buttons).
- 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:

- Because one of the goals of the Raspberry Pi is to create something "hackable", it is no surprise that 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, and they have released only a partial datasheet for the BCM2835 chipset.
- These team members of Raspberry Pi core team have been able to publish certain materials, such as PDFs of the Raspberry Pi board schematics, and so on.
- However, the answer to the question "Is it open hardware?" is currently "Not yet"

5.9 SUMMARY

- 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, which also means you can experiment with different options quickly and easily.
- Sensors are the ways of getting information into your device, finding out things about your surroundings

- Actuators are the outputs for the device — the motors, lights, and so on, which let your device do something to the outside world.
- Distance-sensing modules, which work by bouncing either an infrared or ultrasonic signal off objects, are readily available and as easy to interface to as a potentiometer
- From the perspective of the electronics, the starting point for prototyping is usually a “breadboard”. This lets you push-fit components and wires to make up circuits without requiring any soldering and therefore makes experimentation easy.

5.10 UNIT END QUESTIONS.

1. Differentiate between Arduino Due and Raspberry Pi Model B.
2. Which parameter are important for changing embedded platform ?
3. What are the things need to consider while developing arduino project?
4. Compare between Python and C++ as development language for Internet of Things.
5. Write short note on microcontrollers and system-on-chips.
6. Explain hardware of Raspberry pi.

❄❄❄❄❄❄❄

6

PROTOTYPING THE PHYSICAL DESIGN

Unit Structure

6.0 Objectives
6.1 Introduction
- 6.1.1 Preparation
- 6.1.2 Sketch, Iterate and Explore
- 6.1.2 Nondigital Methods
- 6.1.3 Laser Cutting
- 6.1.4 Choosing a Laser Cutter
- 6.1.5 Software
- 6.1.6 Hinges and Joints
- 6.1.7 3D Printing
- 6.1.8 Types of 3D Printing
- 6.1.9 Software
- 6.1.10 CNC Milling
- 6.1.11 Repurposing/Recycling
- 6.1.12 A Case Study: Ackers Bell

6.2 Let us Sum Up
6.3 List of References
6.4 Bibliography
6.5 Unit End Exercises

6.0 OBJECTIVES

After going through this unit, you will be able to:

- Define Internet of Things.
- Understand Sensors, Actuators & Internet
- Describe the basic elements of IOT.
- Design the IOT devices.
- Connect the physical devices & sensors with network.
- Enabling the interconnection and integration of the physical world and the cyber space.
- Understand the trend of future networking.
- Learn the new technologies.

6.1 INTRODUCTION

The internet of things, or IoT, is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers (UIDs) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.

An IoT ecosystem consists of web-enabled smart devices that use embedded systems, such as processors, sensors and communication hardware, to collect, send and act on data they acquire from their environments. IoT devices share the sensor data they collect by connecting to an IoT gateway or other edge device where data is either sent to the cloud to be analyzed or analyzed locally.

6.1.1 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) hack days(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).

6.1.2 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 optimizing 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.

6.1.3 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.

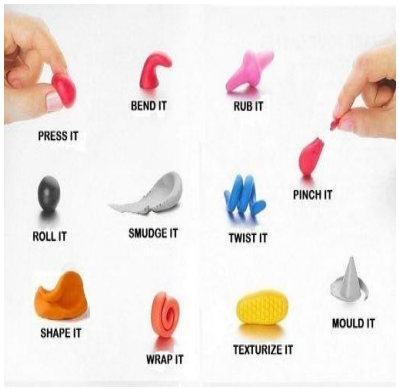
Epoxy putty:

- 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.

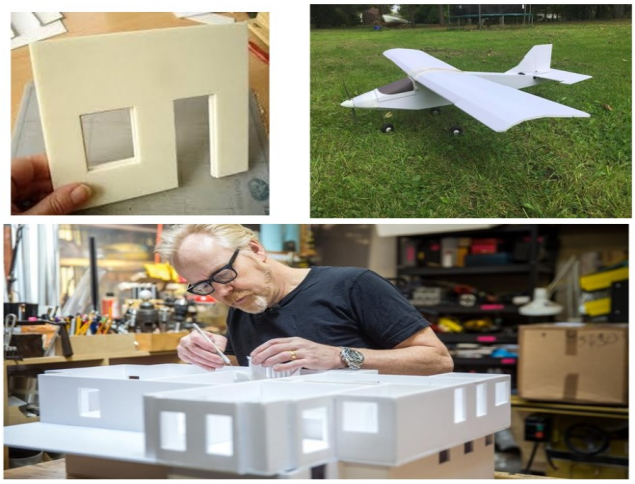
- You can purchase systems to control LEGO sets from a computer.
- Many hackers combine an Arduino for sensing and control with LEGO for form.

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.

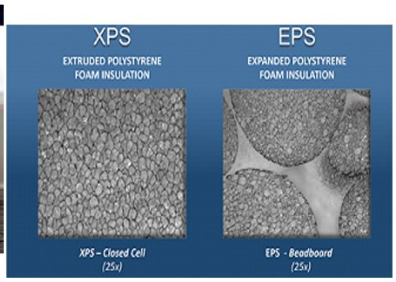
Foamcore or foamboard:

- 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.



6.1.4 LASER CUTTING

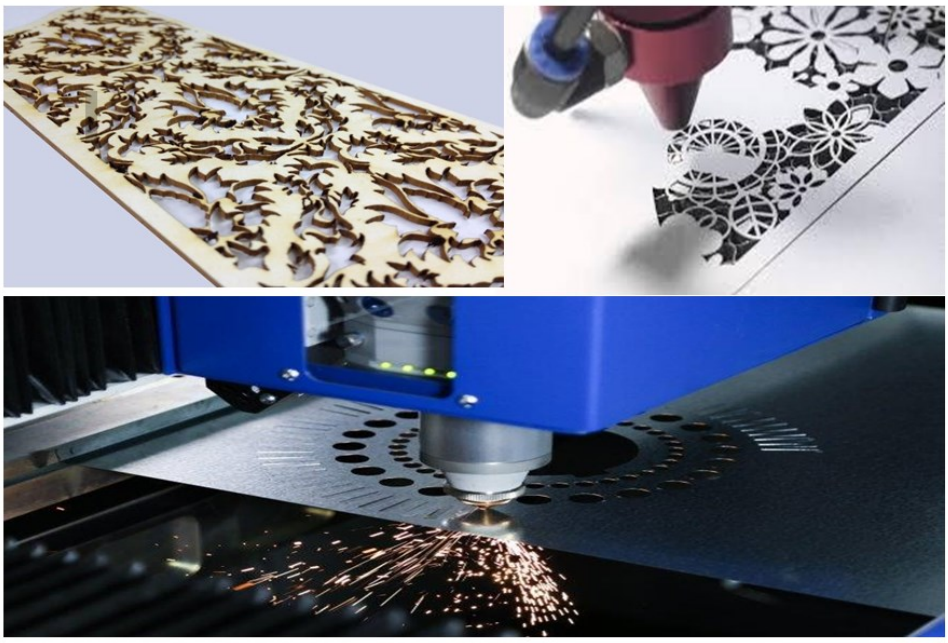
Three-dimensional printers can produce more complicated parts, but the simpler design process greater range of materials which can be cut, and faster speed make the laser cutter a versatile piece of kit.

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 floor standing 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.



6.1.5 CHOOSING A LASER CUTTER

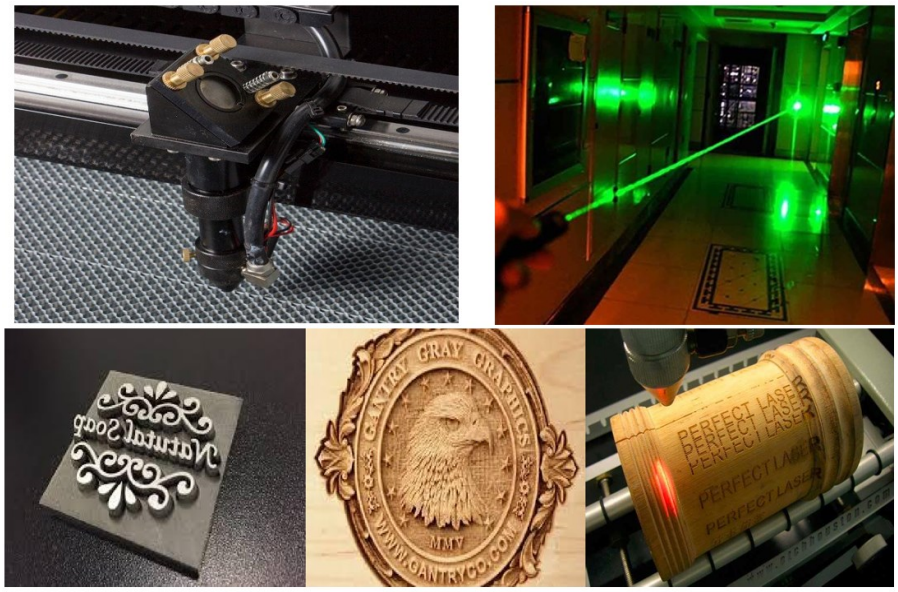
When choosing a laser cutter, you should **consider two main features:**

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.



6.1.6 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.

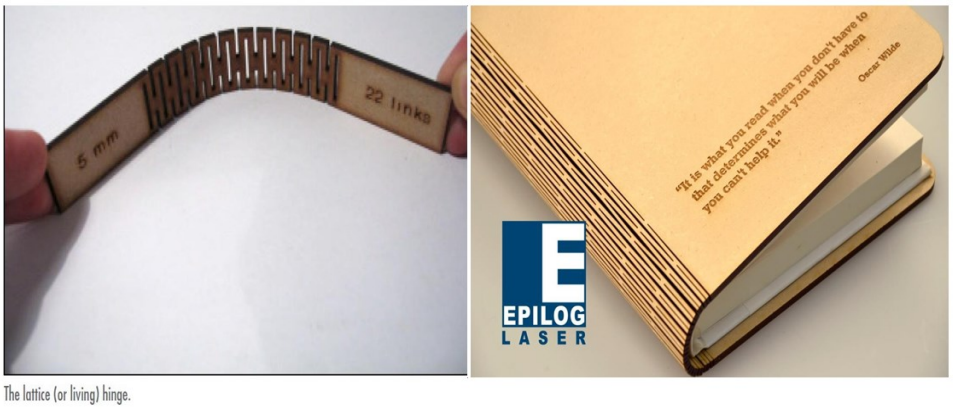
The kerf, the width of the cut made by the laser, is about 0.2mm but isn't something you need to include in the design. A thinner stroke width is better, as it will stop the laser cutter from misinterpreting it as two cuts when you need only one.

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.

6.1.7 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.

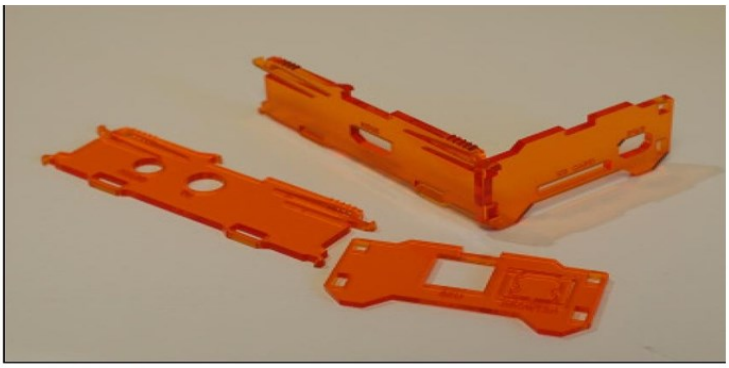


The lattice (or living) hinge.

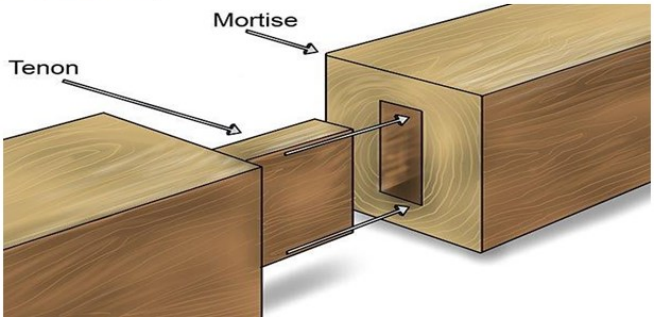
Integrated Elastic Clips

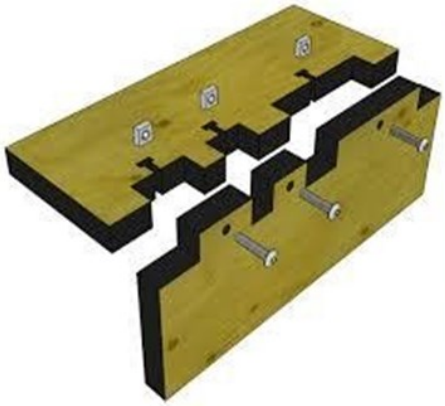
- This jointing technique is used in situations similar to a through mortise-and- tenon joint, when joining two sheets of material at 90 degrees.

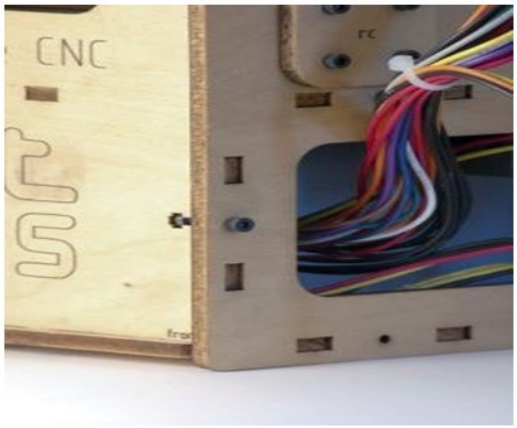
- The tenon is replaced with two hooks for holding the mortise sheet tight to the tenon sheet without any need for glue or additional fixings.
- To provide the required flexibility in the tenon to fit it through the mortise during assembly, additional, deeper cuts are made into the tenon side.



Integrated elastic clips.







Bolted Tenon (or T-Slot) Joints

- It is a modified version of the standard mortise-and-tenon joint which adds a T- or cross shaped 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.

6.1.8 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.)

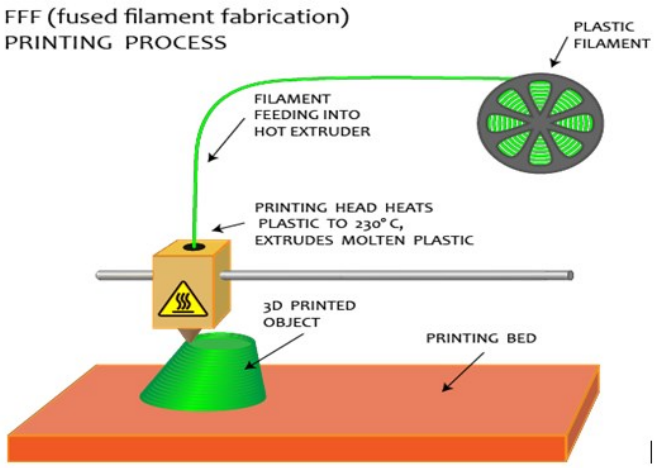
Interlocking rings without any joins

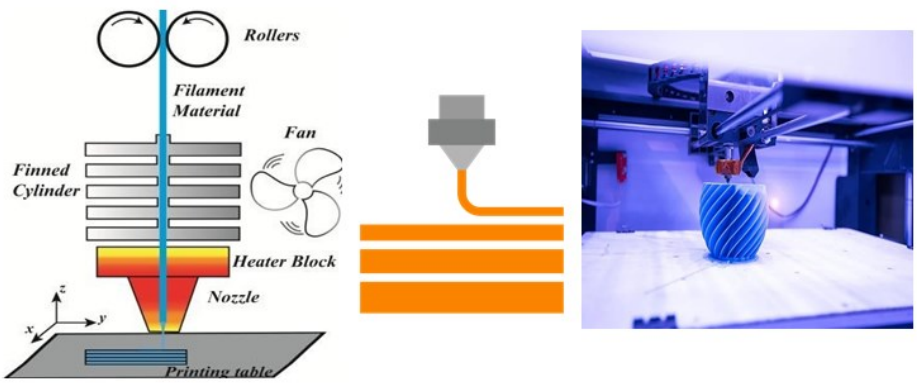
Medieval Knight

6.1.9 TYPES OF 3D PRINTING

Fused filament fabrication (FFF)

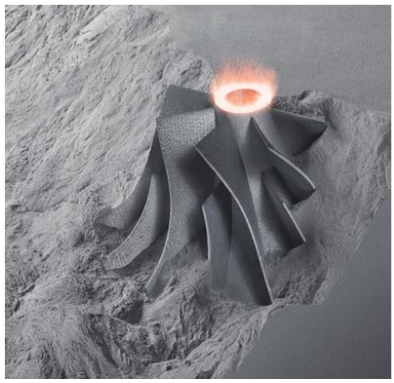
- Also known as fuseddeposition modeling (FDM).
- It works by extruding (**Plastics extrusion** is a high-volume manufacturing process in which **raw plastic is melted and formed into a continuous profile.**) a fine filament of material (usually plastic) from a heated nozzle.
- The nozzle can be moved 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.

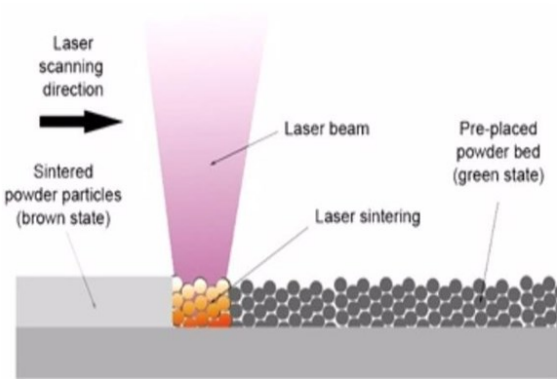




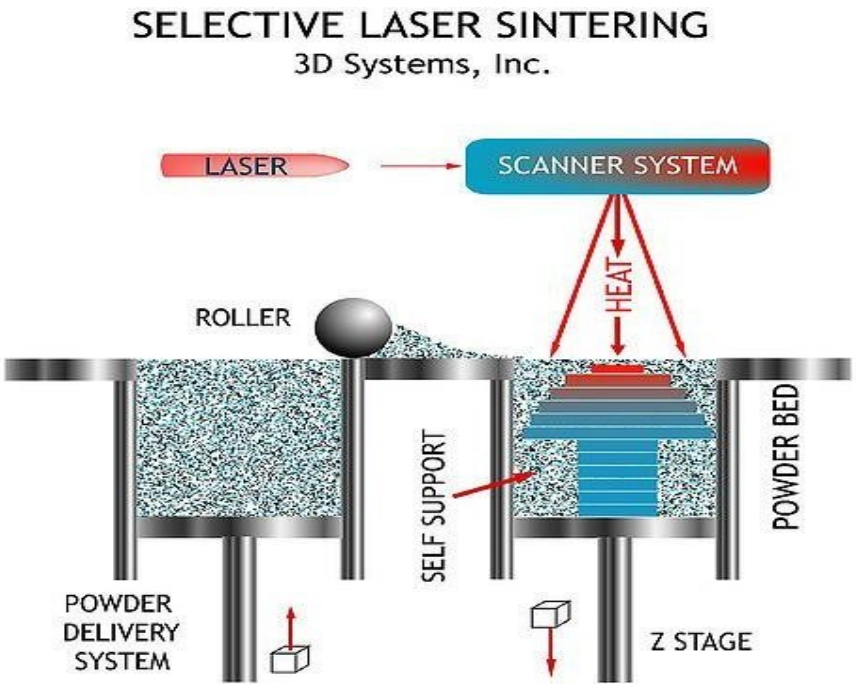
Laser sintering:

- 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.





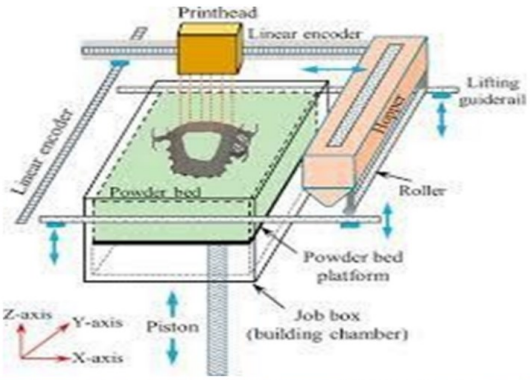
- **Selective LASER Sintering**



Powder bed:

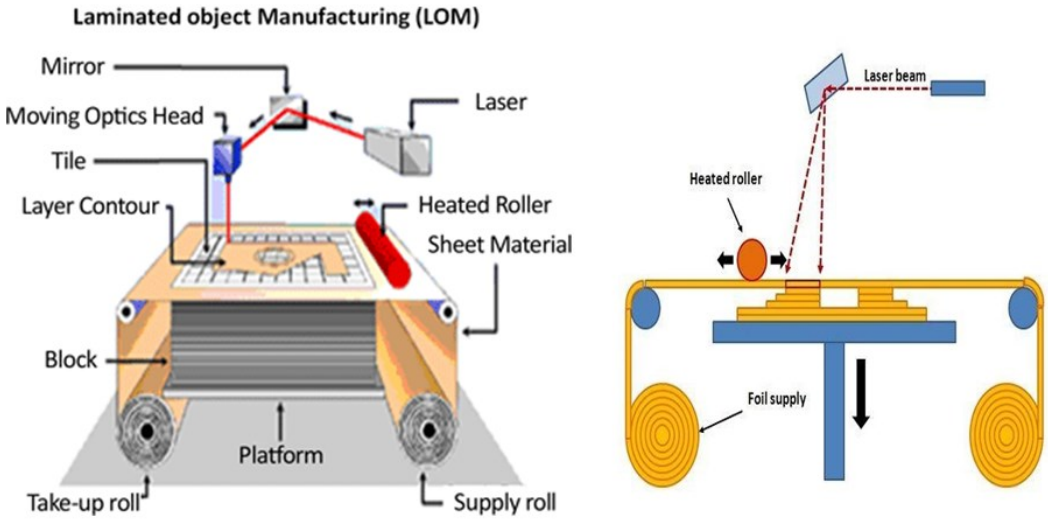
- Like laser sintering, the powder-bed printers start with a raw material in a powder form, but rather than fusing it together with a laser, the binder is more like a glue which is dispensed by a print head similar to one in an inkjet printer.

- 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.



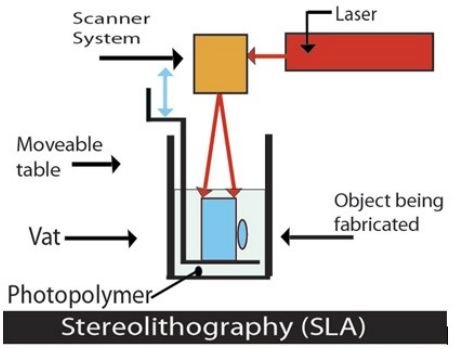
Laminated object manufacturing (LOM):

- 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.



3D PRINTING (Websites)

Shapeways (http://www.shapeways.com/),
i.materialise

(http://i.materialise.com/), and Ponoko (https://www.ponoko.com/) have all been offering similar services for a while now.

You upload your design online, choose how you want it printed, and a few days later receive it in the post.

6.1.10 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.

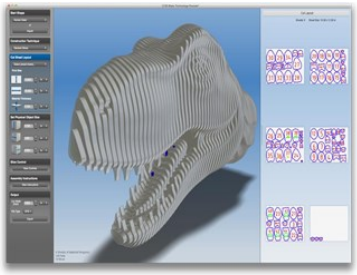
In the open source camp, the main contenders are OpenSCAD (http://www.openscad.org) and FreeCAD (http://free-cad.sourceforge.net).

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.

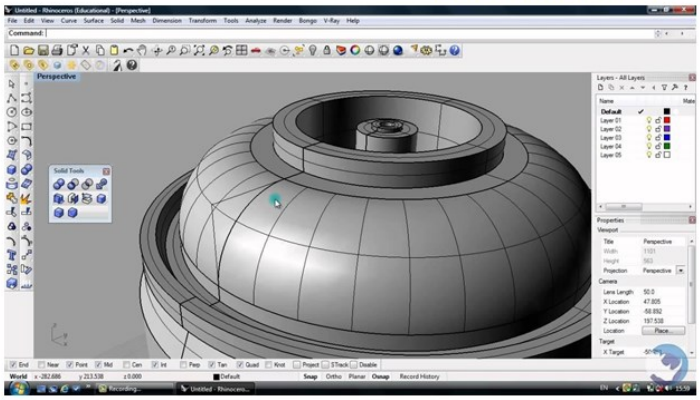
This is usually known as the slicing algorithm because its most important function is to carve the model into a series of layers and work out how to instruct the printer to build up each layer. Skeinforge was the first slicing software used by the open source printers, but it has been largely overtaken 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.

TINKER CAD



Autodesk's 123D



Blender

6.1.11 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.

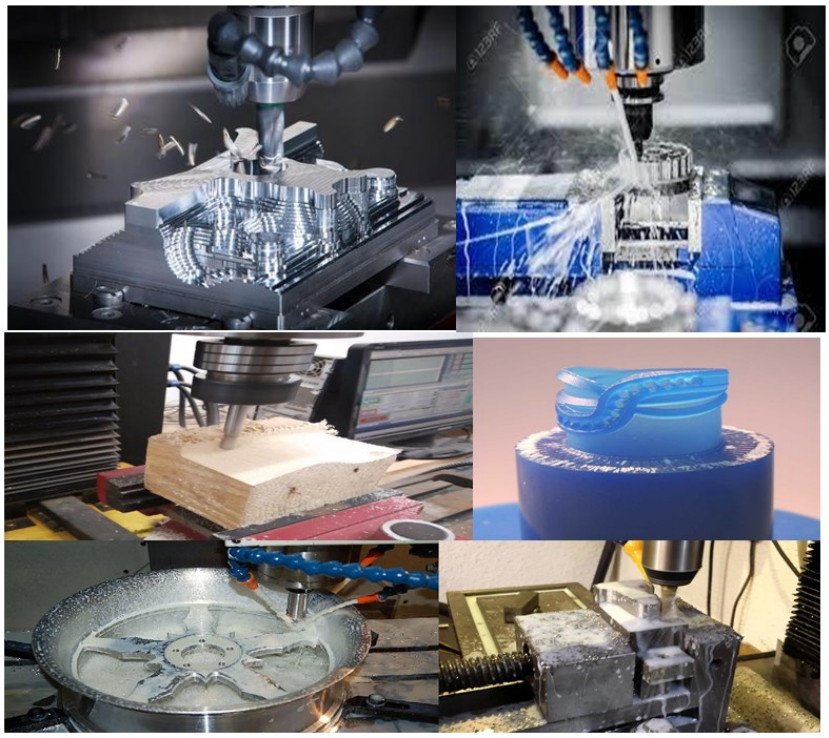
It cuts away the parts which aren't needed—much like a sculptor chips away at a block of stone to reveal the statue, except that milling uses a rotating cutting bit (similar to an electric drill)

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.

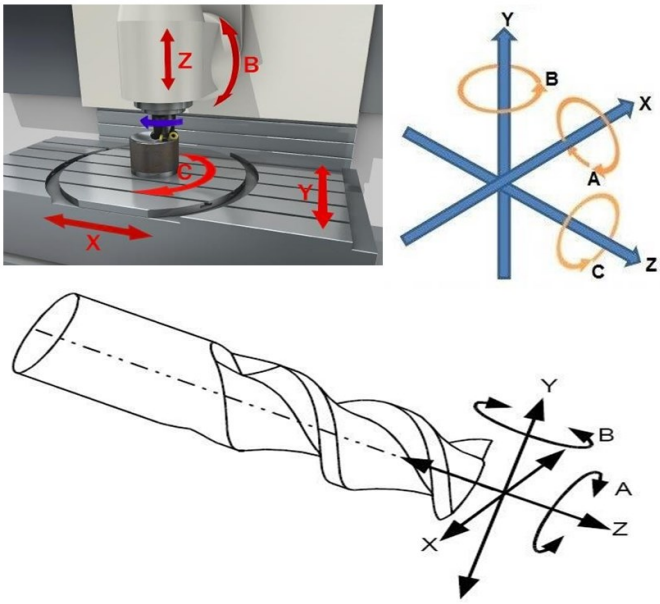
An advantage of milling over etching the board is that you can have the mill drill any holes for components or mounting at the same time, saving you from having to do it manually afterwards with your drill press.

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.



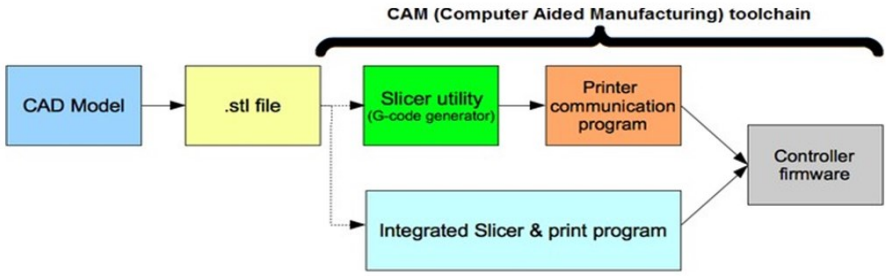
Beyond size and accuracy, the other main attribute that varies among CNC mills is the number of axes of movement they have:

- 2.5 axis: This type has three axes of movement—X, Y, and Z—it can move only any two at one time.
- 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
- 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).
- axis: This machine adds a second rotary axis—normally around the Y— which is known as the B axis.
- 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:



1 CAD (Computer-Aided Design) software lets you design the model.

2 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.



6.1.12 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.

6.1.13 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.

6.2 Let us Sum Up

- We have seen prototyping the physical design, Sketch, Iterate and Explore, Nondigital Methods.
- Also seen 3D PRINTING and their types.
- We have seen software used in designing too.
- We have seen Case Study of Ackers' bell.

6.3 List of References

https://ieeexplore.ieee.org/abstract/document/7123563
https://ieeexplore.ieee.org/abstract/document/7842420

6.4 Bibliography

1. Designing the Internet of Things, Adrian McEwen, Hakim Cassimally WILEY publisher.
2. Internet of Things – Architecture and Design, Raj Kamal, McGraw Hill

6.5 Unit End Exercises

1. Explain Laser cutting & the criteria for selecting the Laser cutter.
2. List all non-digital methods for prototyping. Explain any 3 methods.
3. What are hinges & joints? Explain any two.
4. Explain software used for 3D printing.
5. What is 3D printing? State its type.
6. Explain CNC milling.
7. Explain repurposing.

❄❄❄❄❄❄❄

7

PROTOTYPING ONLINE COMPONENTS

Unit Structure

7.0 Objectives

7.1 Introduction

- 7.1.1 Getting Started with an API
- 7.1.2 Mashing Up APIs
- 7.1.3 Scraping
- 7.1.4 Legalities
- 7.1.5 Writing a New API
- 7.1.6 Clockodillo
- 7.1.7 Security
- 7.1.8 Implementing the API
- 7.1.9 Using Curl to Test
- 7.1.10 Going Further
- 7.1.11 Real-Time Reactions
- 7.1.12 Polling
- 7.1.13 Comet
- 7.1.14 Other Protocols :MQ Telemetry Transport
- 7.1.15 Extensible Messaging and Presence Protocol
- 7.1.16 Constrained Application Protocol

7.2 Let us Sum Up

7.3 List of References

7.4 Bibliography

7.5 Unit End Exercises

7.0 OBJECTIVES

After going through this chapter, you will be able to:

- Connect the physical devices & sensors with network.
- Enabling the interconnection and integration of the physical world and the cyber space.
- Understand the trend of future networking.
- Learn the new technologies.
- Understand the importance of security in IoT field
- Learn the protocols used in IoT.

7.1 INTRODUCTION

The internet of things, or IoT, is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers (UIDs) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.

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

7.1.1 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.

7.1.2 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 orbuy.
- Showing Twitter trends on a global map or in a timeline or a chartingAPI.
- Fetching Flickr images that are related to the top headlines retrieved fromnewspaper's API.

7.1.3 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 representative's 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

7.1.4 LEGALITIES

Screen-scraping may break the terms and conditions of a website.

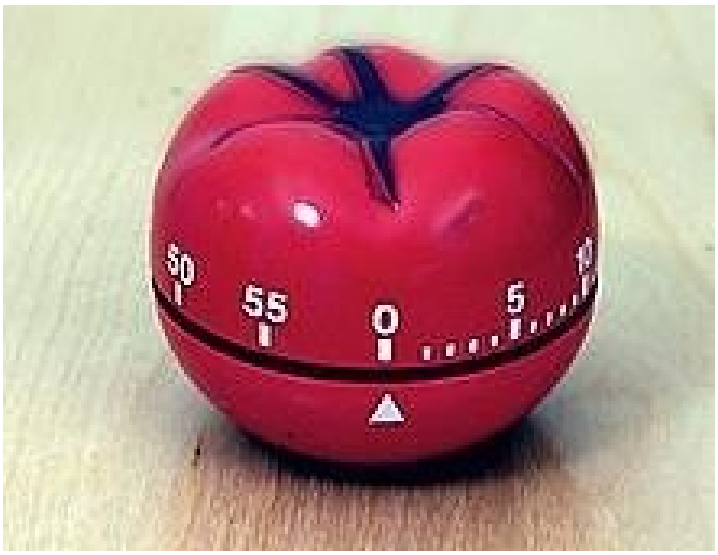
For example, Google doesn't allow you to screen-scrape.

Alternative sources of information often are available.

For example, you could use Open Street Map instead of Google Maps.

7.1.5 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).



7.1.6 Clockodillo

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 newtimer
- Change the duration of an existingtimer
- Mark a timercompleted
- Cancel atimer

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 beable to see historical data:
- Previous timers, in alist
- 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.

7.1.7 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'swired 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

7.1.8 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

7.1.9 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.

7.1.10 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.

7.1.11 Real-Time Reactions

- 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.

7.1.12 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!

7.1.13 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 XML Http Request (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.

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 Web Sockets 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.

7.1.14 Other Protocols: MQ TELEMETRY TRANSPORT

MQTT (http://mqtt.org) is a lightweight messaging protocol, designed specifically for scenarios where network bandwidth is limited.

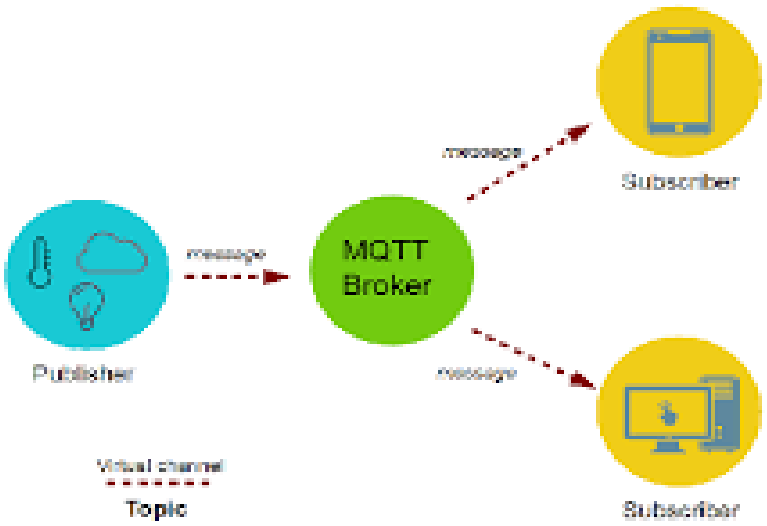
It was developed initially by IBM but has since been published as an open standard, and a number of implementations, both open and closed source, are available, together with libraries for many different languages.

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.

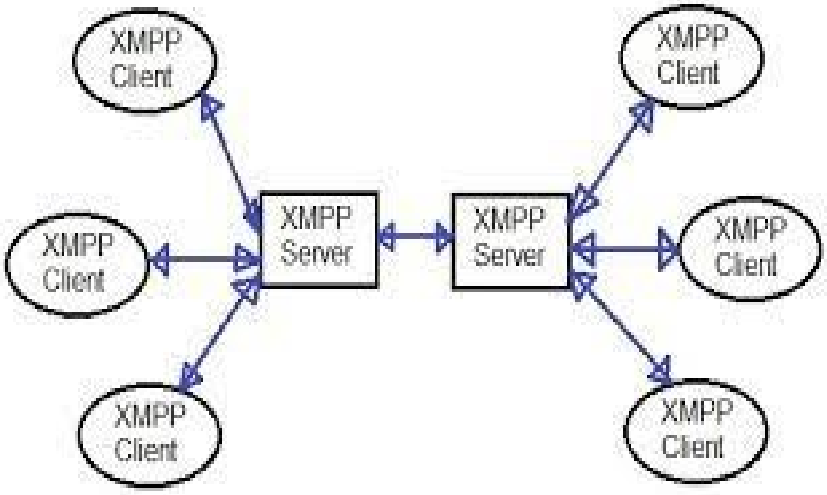
Recipients subscribe to whichever topics interest them, and whenever a new message is published on that topic, the message broker delivers it to all interested recipients.

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.5



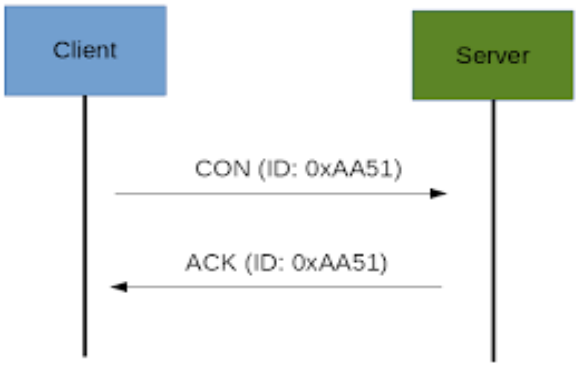
7.1.15 Extensible Messaging and Presence Protocol

- 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.



7.1.16 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.



7.2 LET US SUM UP

- We have learnt prototyping Online Component, Getting Started with an API, Writing a New API and some examples as well.
- We have also learnt about protocols used in IoT.

7.3 LIST OF REFERENCES

https://ieeexplore.ieee.org/abstract/document/7123563
https://ieeexplore.ieee.org/abstract/document/7842420

7.4 BIBLIOGRAPHY

1. Designing the Internet of Things, Adrian McEwen, Hakim Cassimally WILEY publisher.
2. Internet of Things – Architecture and Design, Raj Kamal, McGraw Hill

7.5 UNIT END EXERCISES

1. Explain Mashing of API.
2. Explain Clockodillo.
3. Explain Polling.
4. Explain Comet.
5. Explain MQ Telemetry Transport.
6. Explain Constrained Application Protocol (CoAP)
7. Explain Extensible Messaging and Presence Protocol.
8. Explain Scraping with examples.

❄❄❄❄❄❄❄

8

TECHNIQUES FOR WRITING EMBEDDED CODE

Unit Structure

8.1 Memory Management
8.2 Types of Memory
8.2.1 Primary Memory
8.2.1.1 Random Access Memory (RAM)
8.2.1.1 a) Static RAM (SRAM)
8.2.1.1 b) Dynamic RAM (DRAM)
8.2.1.2 Read Only Memory (ROM)
8.2.1.2 a) PROM (Programmable Read Only Memory)
8.2.1.2 b) EPROM (Erasable Programmable Read Only Memory)
8.2.1.2 c) EEPROM (Electrically Erasable Programmable Read Only Memory)
8.2.2. Secondary Memory
8.3 Making the Most of Your RAM
8.3.1 Organizing RAM: Stack vs Heap
8.4 Performance and Battery Life
8.5 Libraries
8.6 Debugging.
8.7 Lets Sum Up
8.8 List of References..
8.9 Bibliography
8.10 Unit End Exercises.

8.1 MEMORY MANAGEMENT

- Memory is like the human brain. It is used to store data and instructions. Computer memory is the storage space in a computer where data is processed and the instructions needed for processing are stored.
- Memory management is the process of controlling and coordinating a computer's memory, assigning sections called blocks to different running programs to optimize the overall performance of the system.
- Memory management resides in the hardware, in the operating system (OS), programs and applications.

- One of the biggest differences between embedded system platforms and conventional computing platforms is the lack of available resources. When we talk about laptops and servers, they come with memories with a storage capacity of gigabytes. Microcontroller Resources are measured in kilobytes.

8.2 TYPES OF MEMORY:

- Memory is the most essential element of a computer system because without it the computer cannot perform simple tasks. Computer memory comes in two basic types -

8.2.1 Primary Memory

- Primary / Main memory only contains data and instructions that are currently processed by the computer. Its capacity is limited and data will be lost if the power is turned off. It usually consists of semiconductor components.
- These memories are not as fast as registers. Processed data and commands are in main memory. It is divided into two subcategories: RAM (mostly volatile memory) and ROM (mostly non-volatile memory).

Characteristics of Main Memory

- These are semiconductor memories.
- It is known as the main memory.
- Usually volatile memory.
- Data is lost in case power is switched off.
- It is the working memory of the computer.
- Faster than secondary memories.
- A computer cannot run without the primary memory.

Types of Primary Memory

8.2.1.1 Random Access Memory (RAM)

- RAM (Random Access Memory) is the internal memory of the CPU, used to store data, programs, and program results. It is a read/write memory used to store data until the machine works. Once the device is turned off, the data will be deleted. The access time in the RAM does not depend on the address, that is, each storage location in the memory can be accessed as easily as other locations and requires the same time.
- The data in RAM can be accessed at will, but it is very expensive. RAM is volatile, i.e., data stored in it is lost when we switch off the computer or if there is a power failure. Hence, a backup Uninterruptible Power System and is often used with computers.

- RAM is small, both in terms of its physical size and in the amount of data it can hold. RAM is further classified into two types SRAM (Static Random-access Memory) and DRAM (Dynamic Random-Access Memory).

8.2.1.1 a) Static RAM (SRAM)

- The word static indicates that memory retains its contents as long as it is energized. However, data is lost when the power is turned off due to instability. The SRAM uses an array of 6 transistors with no capacitors. The transistors don't need a power supply to prevent leakage, so SRAM doesn't need to be updated regularly.
- There is extra space in the die, so SRAM uses more chips than DRAM for the same amount of memory, which increases production costs. Thus, SRAM is used as a cache and has very fast access.

Characteristic

- Long life
- No need to refresh
- Faster
- Used as cache memory
- Large size
- Expensive
- High power consumption

8.2.1.1 b) Dynamic RAM (DRAM)

- DRAM, unlike SRAM, needs to be constantly updated to preserve data. This is done by placing memory in the update circuit, which overwrites the data several hundred times per second. Most system memories use DRAM because it is cheap and small.

All DRAMs are made up of memory cells, which are made up of one capacitor and one transistor.

Characteristics

- Short data lifetime
- Needs to be refreshed continuously
- Slower as compared to SRAM
- Used as RAM
- Smaller in size
- Less expensive
- Less power consumption

8.2.1.2 Read Only Memory (ROM)

ROM stands for Read Only Memory. A memory from which we can only read, but not write. This type of memory is non-volatile. Information is

permanently stored in such storage devices during production. The ROM contains instructions for starting the computer. This operation is called bootstrapping. ROM chips are used not only in a computer, but also in other electronic systems or where programming does not require changes. Examples: washing machines and microwaves, calculators and peripherals.

Types of Read Only Memory (ROM)

8.2.1.2 a) PROM (Programmable Read Only Memory) - can be programmed by the user. Once programmed, the data and instructions contained therein cannot be changed.

8.2.1.2 b) EPROM (Erasable Programmable Read Only Memory) - Can be reprogrammed. To erase data from it, expose it to ultraviolet light. To reprogram it, delete all previous data.

8.2.1.2 c) EEPROM (Electrically Erasable Programmable Read Only Memory) - data can be erased by applying an electric field, without the need for UV light. We can only erase parts of the chip. d) Mask ROM (MROM): This is a type of read-only memory (ROM) whose contents are programmed by the chip manufacturer (not the user). The earliest ROMs were hardware devices that contained a pre-programmed set of data or instructions. These types of ROMs are known as masked ROMs and are inexpensive.

Advantages of ROM

- Non-volatile in nature
- Cannot be accidentally changed
- Cheaper than RAMs
- Easy to test
- More reliable than RAMs
- Static and do not require refreshing
- Contents are always known and can be verified

8.2.2 Secondary Memory

This type of storage is also called external storage or non-volatile storage. It is slower than the main memory. These are used to store data/information permanently. The CPU does not directly access this memory, but through input-output routines. The contents of the auxiliary storage are first transferred to the main storage, and then the CPU can access them. For example, floppy disks, CDROM, DVD, etc.

Characteristics of Secondary Memory

- These are magnetic and optical memories.
- It is known as the backup memory.
- It is a non-volatile memory.

- Data is permanently stored even if power is switched off.
- It is used for storage of data in a computer.
- Computers may run without the secondary memory.
- Slower than primary memories.

8.3 MAKING THE MOST OF YOUR RAM

- When you only have a few kilobytes or tens of kilobytes of RAM, it causes your device to malfunction or crash and it is easier to make the best compromise between maximum RAM utilization and reliability if 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 not to allocate memory dynamically, that is, while the program is running.
- In the deterministic model, instead of reserving space for the entire page, you reserve space for important information to be extracted, as well as a memory buffer that can be used as a work area when downloading and processing files on the page.
- is not a whole Pages are downloaded into memory, but in chunks-fill the buffer each time, and then process the chunk of data before proceeding to next?
- However, pre-installed programs consume a lot of system memory, so disabling unnecessary startup programs can help maximize memory usage. To switch programs, right-click on the taskbar and select Task Manager from the context menu, then open the "Start" tab.

8.3.1 Organizing RAM: Stack vs Heap

What is Stack?

- A stack is a special area of computer memory that stores temporary variables created by a function. On the stack, variables are declared, stored, and initialized at runtime.
- This is temporary storage. After the end of the calculation task, the variable memory will be automatically cleared. The stack section mainly contains methods, local variables, and reference variables.

What is Heap?

- The heap is the memory used by programming languages to store global variables. By default, all global variables are stored in the heap space. It supports dynamic memory allocation.
- Heap is not automatically managed for you, nor is it closely managed by the CPU. It is more like a floating memory area.

Stack vs Heap

i. Stack is a linear data structure whereas Heap is a hierarchical data structure.

ii. Stack memory will never become fragmented whereas Heap memory can become fragmented as blocks of memory are first allocated and then freed.

iii. Stack accesses local variables only while Heap allows you to access variables globally.

iv. Stack variables can't be resized whereas Heap variables can be resized.

v. Stack memory is allocated in a contiguous block whereas Heap memory is allocated in any random order.

vi. Stack doesn't require to deallocate variables whereas in Heap deallocation is needed.

vii. Stack allocation and deallocation are done by compiler instructions whereas Heap allocation and deallocation is done by the programmer.

8.4 PERFORMANCE AND BATTERY LIFE

- When it comes to writing code, performance and battery life usually goes hand in hand which is good for one, usually good for the other.
- What needs to be optimized depends on the application. For battery or solar powered items, or items that need to respond immediately when the user presses a button, it's worth paying attention to performance or energy consumption.
- The greatest energy savings come from hardware design, especially if the device can shut down system modules when not in use, or put the entire CPU into a low-power hibernation state when code is ready or waiting for something to happen.
- However, software optimization is still important! After all, the sooner the main code completes execution, the faster the hardware goes to sleep.
- One of the easiest ways to increase code performance is to move to an event-driven model instead of requesting changes.
- The reason for this is to allow the device to remain in a low-power state longer and start operating when needed, rather than performing routine activities on a regular basis to check if something has changed and if it has actual work to be done.

8.5 LIBRARIES

- Nowadays, when you develop software for servers or desktops, you are used to having access to a huge number of possible libraries and frameworks that make your task easier. In an embedded world, tasks are often a bit more complex.
- With the increasing supply of system chips and their use of embedded Linux, where most server packages can be included in the same way as in "normal" Linux systems.
- On the other hand, microcontrollers still have too limited resources to simply use the libraries and code of the main operating system.
- We 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, you probably are better
- off starting from scratch or finding one that's already written with microcontroller limitations in mind.
- Here are a few libraries which might be of interest:
- lwIP: lwIP, or LightWeight IP, is a full TCP/IP stack which runs in low-resource conditions. It requires only tens of kilobytes of RAM and around 40KB of ROM/flash. The official Arduino WiFi shield uses a version of this library.
- uIP: uIP, or micro IP, is a TCP/IP stack targeted at the smallest possible systems. It can even run on systems with only a couple of kilobytes of RAM. It does this by not using any buffers to store incoming packets or outgoing packets which haven't been acknowledged. It's quite common on Arduino systems which don't use the standard Ethernet shield and library, such as the Nanode board, using the Ethercard port for AVR.
- uClibc: uClibc is a version of the standard GNU C library (glibc) targeted at embedded Linux systems. It requires far fewer resources than glibc and should be an almost drop-in replacement. 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 (the scheduler switches between the tasks quickly enough that it looks that way, just like the multitasking on your PC).
- 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.

8.6 DEBUGGING

- One of the most frustrating parts of writing software is knowing there is an error in your code.
- In embedded systems, this can be doubly frustrating, as there are usually fewer ways to find out what is happening so that a problem can be tracked down.
- The creation of devices for the Internet of Things further complicates the situation due to the introduction of both specialized electronic circuits (which may be malfunctioning or improperly designed) and communication with servers over the network.
- Modern IDEs for desktops have excellent support for delving into what is happening as code is running.
- You can set breakpoints that stop execution when a predefined set of conditions is met, in which case you can search the memory to see what's in it, evaluate the expression to see if your definition is correct, then traverse the code line by line to see what happens.
- You can even change the contents of memory or variables on the fly to affect the rest of the code execution, and in more advanced systems, rewrite the code while the program is running.
- Debugging environments for embedded systems are generally more primitive. If your embedded platform is running a more complete operating system such as Linux, then you are in a better position than if you are developing on a small microcontroller.
- Systems like embedded Linux usually support remote debugging with tools like gdb, the GNU debugger.
- This utility allows a debugger from a desktop computer to be connected to the built-in motherboard, usually via a serial connection, but sometimes also via Ethernet or a similar network channel.
- Once connected, you have access to many desktop-like debugging features - the ability to set breakpoints, step through code, and inspect variables and memory contents.

8.7 LETS SUM UP

To run embedded code we need some storage space. In this unit we have seen different times of memories like primary and secondary memory and different ways to manage yem. Making best use of RAM and organizing it with the concept of stack & heap has been also illustrated with an appropriate diagram. To enhance the performance we need good battery life and we have some inbuilt libraries like uClibc,LwIp that works well with the embedded platform and takes less storage space to run the code. In embedded systems, it is frustrating to identify the exact line of code or connection that is giving an error, however there are different ways to find out what is happening so that a problem can be tracked down. One of the common ways to find errors is with the GNU debugger.

8.8 LIST OF REFERENCES.

Designing the Internet of Things Adrian McEwen, Hakim Cassimall Internet of Things – Architecture and Design Raj Kamal

Getting Started with the Internet of Thing--Cuno Pfister

8.9 BIBLIOGRAPHY

https://www.collegenote.net/curriculum/introduction-to-information-technology

https://www.tutorialspoint.com/computer_fundamentals/computer_ram.html

https://www.guru88.com/stack-vs-heap.html

https://www.c-sharpcorner.com/article/stack-vs-heap-memory-c-sharp/

8.10 UNIT END EXERCISES.

Q.1 stores or retains data after power off is called-

a. Volatile storage

b. Non-volatile storage

c. Sequential storage

d. Direct storage

Q.2 A permanent memory, which requires data and instruction for start-up the computer and does not erase data after power off.

a. Network interface card

b. HDD

c. RAM

d. ROM

Q.3 The Chips where information is stored as hard-coded are called as:

a. RAM

b. ROM

c. Flash

d. Cache

Q.4. In which of the following data structures, the first thing to be removed is the last item that was placed onto it.

a. heap

b. Queue

c. stack

d. Linked List

Q.5. When a program starts execution, which type of variables are allocated first on the stack?

a. local

b. global

c. static

d. Constant

Q.6. The maximum memory usage on the stack depends on which of the following criteria?

a. No. of local variables that code has

b. No. of instructions written in your code

c. No. of static variables that code has

d. The execution path that code can take through your code

Q.7. Debugging is _____?

a. creating program code?

b. finding and correcting errors in the program code

c. identifying the task to be computerized

d. creating the algorithm

Q.8. EPROM stands for

a. Electrically Erasable Programmable Memory

b. Electrically Erased Programmable Memory

c. Electrically Erasing Programmable Memory

d. Electrically Erase Programmable Memory

Q.9. Memory available in heap memory are in

a. Chunks

b. Fragments

c. Bundles

d. Isolation

Q.10 One of the most prominent reasons an IoT device goes on frequent rebooting is

a. Self-maintenance

b. Low Memory

c. Memory Defragmentation

d. Process Automation

❄❄❄❄❄❄❄

9

BUSINESS MODELS

Unit Structure

9.0 What is the Business Model?
9.1 A Short History of Business Models
9.2 Learning from History:
9.2.1 Space and Time.
9.2.2 From Craft to Mass Production.
9.2.3 The Long Tail of the Internet.
9.3 The Business Model Canvas
9.4 Who Is the Business Model For?
9.4.1 Models
9.4.2 Make Thing, Sell Thing
9.5 Subscriptions
9.6 Customization
9.7 Be a Key Resource
9.8 Provide Infrastructure: Sensor Networks
9.9 Take a Percentage
9.10 Funding an Internet of Things Startup,
9.10.1 Hobby Projects and Open Source,
9.11 Venture Capital
9.12 Government Funding
9.13 Crowd funding
9.14 Lean Startups.
9.15 Lets Sum Up
9.16 List of References.
9.17 Bibliography
9.18 Unit End Exercises.

9.0 WHAT IS A BUSINESS MODEL?

A business model can be defined as "assuming what customers want, what they want, and how the company is organized to best meet these needs, get paid and profit for them."

This definition brings together many factors:

- A group of people (customers).
- Identifying customer needs.
- One thing your company can do to meet these needs.
- Organizational practices that help achieve that goal-and can achieve the success criteria in a sustainable way.

9.1 SHORT HISTORY OF BUSINESS MODELS

From the earliest times, and for the great majority of human existence, we have gathered in tribes, with common property and shared resources.

This is an almost universal pattern amongst hunter-gatherers, as it means that every member of the tribe can find food and shelter even if they have not been lucky foraging or hunting that day.

We could describe this form of collectivism as a basic gift economy. Gift economies develop where those with the appropriate skills can provide their products or services — hunting, pottery, livestock, grain, childcare — and expect repayment of this obligation not immediately but with a gift of comparable worth later. This is not a written debt but a social obligation, which the recipient will repay in due course.

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

9.2 LEARNING FROM HISTORY

First, we've seen that some models are ancient, such as Make Thing Then Sell It. 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.

Finally, new business models have the power to change the world, like the way branded soap ushered in mass consumerism and mass production changed the notion of work itself.

9.2.1 Space and Time

Technological advancements such as waterway navigation and portage of boats over land opened up new possibilities, as did the domestication of animals such as the camel, which unlocked trade routes through the Western Arabian deserts. The preservation of food was done, either through salting or smoking, or simply better storage technology such as grain silos.

As well as facilitating transportation through space, preservation is also a way of transporting goods through time. A farmer or trader who cannot eat or sell all his produce during the glut of harvest can fetch a better price months later at a higher price. So, a merchant trader's business is transporting goods through space and time, and his suppliers, the producers, benefit from that by being able to sell a bulk of their produce in one go, after which they can continue with their daily life and work.

Money, then, abstracted trade further, setting an easy-to-calculate exchange rate between a fixed currency and the product being exchanged.

9.2.2 From Craft to Mass Production

When Gutenberg demonstrated his printing press circa 1450, books changed from being priceless treasures, hand-crafted by monks and artisans, to a commodity that could be produced. Soon every bourgeois family could afford their own books, at least a copy of the Gutenberg Bible, the first mass-produced book.

It is no exaggeration to suggest that the invention laid the foundations for an information culture which is currently exemplified by the Internet and the World Wide Web. Trade routes would play their part here too, as the printing press spread to the New World and India via the sea routes that would be discovered by the end of the century. The cost of printing would become ever smaller as the technology spread, leading to new business models with the rise of newspapers and pamphlets.

In 1884, the British company Lever Brothers launched Sunlight Soap, the first household soap to be sold not by weight, to be cut in the shop by the grocer, but packaged in bars and branded with a logo. This was an innovation in mass consumerism, whereby the brand established a link of trust directly with the consumer, relegating the middleman, the grocer, to becoming just a way to deliver the product to the consumer.

Mass production, perfected by Ford Motor Company, was another major change in business model, driven not by how Henry Ford sold his cars but by how he made them. Ford moved away from the "craft production" of cars sold by commission to highly custom requirements and made by skilled craftsmen. Rather he insisted on standard gauges for parts so that the cars could be assembled and fitted together, ending up identical. This approach made it simple to maintain and repair a Ford car, so the average person could afford to buy one without employing a mechanic to keep it working. The fact that mass production also drove down the costs to produce these cars also helped keep them affordable.

The transition to mass production had its own cost, not least that semiskilled factory labor may not be as fulfilling as the more varied craftsman role that it displaced. As well as social cost, the typical operation can reach bottlenecks in efficiency.

The method of lean production pioneered by Toyota in the 1850s retains many aspects of mass production (efficiency, automation, and high volume of production) but instead of producing masses of a single part, assembly, or finished product, can be run to produce them to order, at a specified date. This approach allows the company a much greater degree of customisation than does mass production, and the emphasis on continuous improvement of efficiency is believed to lead to a more fulfilling and varied environment for the factory worker.

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 1830s and exploded with McDonald's and Burger King in the 1850s.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 (local tastes, laws, and religious observances mean that menus are tweaked globally).

In an interesting turn, many new fast-food chains are fighting against this movement by favoring sauces made by hand from freshly sourced ingredients instead of mass-produced ones.

9.2.3 The Long Tail of the Internet

As we have seen, huge changes in business practice are usually facilitated by, or brought about as a consequence of, technological change. One of the greatest technological paradigm shifts in the twentieth century was the Internet.

From Tim Berners-Lee's first demonstration of the World Wide Web in 1880, 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.

Long tail Internet giants help this process by aggregating products from smaller providers, as with Amazon Marketplace or eBay's sellers. This approach helps thousands of small third-party traders exist, but also makes money for the aggregator, who doesn't have to handle the inventory or delivery at all, having outsourced it to the long tail.

9.3 THE BUSINESS MODEL CANVAS

One of the most popular business model templates is the business model canvas of Alexander Osterwalder and his startup Business Model Foundry.

At first glance, each cell appears to be just a form item that can be replaced with a nine- point checklist. However, the boxes are designed to be the right size for your notes, emphasizing that you can play around with the ideas you have and move them around. In addition, the layout gives meaning and context to each element.

At the bottom right, we have Revenue Streams, which is more or less the question of "how are you going to make money?

The central box, Value Propositions, is, in plainer terms, what you will be producing—that is, your Internet of Things product, service, or platform.

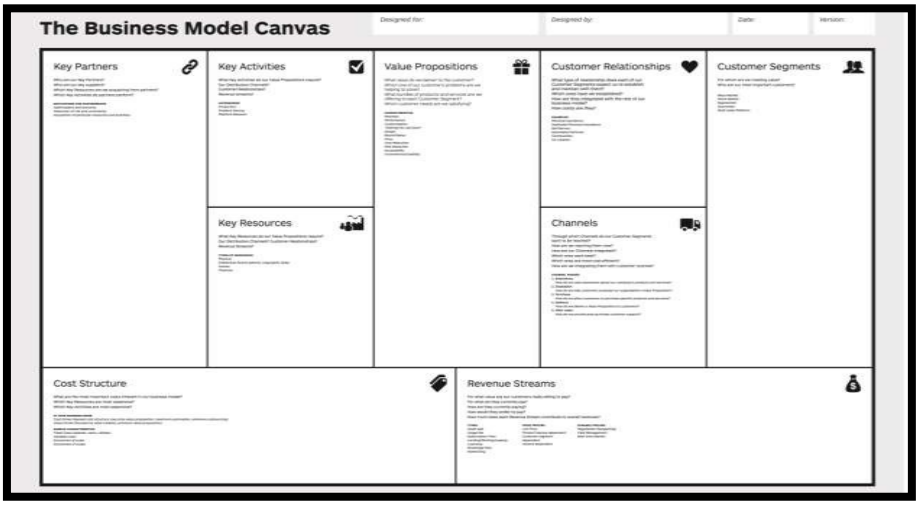
The Customer Segments are the people you plan to deliver the product to. That might be other makers and geeks, the general public, families, or businesses.

The Customer Relationships might involve a lasting communication between the company and its most passionate customers via social media.

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.

Key Resources include the raw materials that you need to create the product but also the people who will help build it. The intellectual resources you have are also valuable, as are the finances required to pay for all this! Of course, few companies can afford the investment in time and money to do all the Key Activities themselves or even marshal all the Key Resources. You will need Key Partners, businesses that are better placed to supply specific skills or resources, because that is their business model, and they are geared up to do it more cheaply or better than you could do yourself. The Cost Structure requires you to put a price on the resources and activities you just defined. Which of them are most expensive? Given the costs you will have, this analysis also 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).



Business Model Canvas Template

9.4 WHO IS THE BUSINESS MODEL FOR?

Primarily, the reason to model your business is to have some kind of educated hypothesis about whether it might deliver what you want from it.

Even if you don't use a semi-formal method like the canvas, anyone who starts up any business will have thought, at least briefly, about whether you can do it, what the business is, and whether you'll get paid.

As a programmer or a maker, you might believe it counterintuitive to think of a piece of paper with nine boxes in it as a "tool", but when you have a well-tested separation of factors to consider, the small amount of structure

the canvas provides should help you think about the business and give you ways to brainstorm different ideas.

Many great product ideas turn out to be impractical, ahead of their time, or unprofitable.

Being able to analyze how the related concepts mesh will help you challenge your product idea and either make it stronger or know when to abandon it. The model is also useful if you want to get other people involved. This could be an employee or a business partner or an investor.

Perhaps to a lesser extent, your customers will also be considering whether to invest their time and money in your product. They will ask themselves certain questions about it.

The business model provides, among other things, a useful tool for understanding what plans there are to keep the service running in both cases. 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”. This formulation was popularized by Andrew Lewis in 209.

Derek Powazek, CEO of social startup Cute-Fight, challenges several assumptions often made:

- 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.

9.4.1 Models

- The Business Model Canvas is a tool for generating and analyzing models.
- It is a good idea to have a look at some of the models that Internet of Things companies have used or might use and consider some of the parameters these models relate to on the canvas.

9.4.2 Make Thing, Sell Thing

The simplest category of models, “make a Thing and sell it”, valid for the Internet of Things.

Electrical products sold in shops (physical or online) may be subject to legislation and certification (RoHS, Kite marks, and so on), which is an additional factor and cost to consider. 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.

9.5 SUBSCRIPTIONS

Subscription is a relatively new business model where a customer agrees to pay companies for products or services over a period of time. For example, a customer may consent to the purchase of an annual magazine subscription, which he receives on a regular basis (monthly, weekly, etc.).

Subscription business models rely on the idea of selling a product or service to generate monthly or annual subscription revenue.

People are eager to pay for music subscriptions, corporate collaboration software, and of course cell phones, so perhaps IoT products in these areas will find subscriptions more attractive to their consumers. It is also recommended that users pay a premium to get additional features or remove restrictions.

This model can be combined with our first two models: the purchase of a physical device gives you free lifetime access to the corresponding internet service, but additional paid services are also available.

9.6 CUSTOMIZATION

Personalization is the adaptation of a service or product to specific people, sometimes related to groups or segments of people.

Mass Customization is a marketing and manufacturing technique that combines the flexibility and personalization of custom products with the low unit costs associated with mass production. Other names for bulk customization are made to order or built to order.

For IoT devices, there are some customization options on the interface between entities and software, which we think can bring new business models.

9.7 BE A KEY RESOURCE

Not every IoT company sells products to the mass market. Some sell components or expertise to other companies or component manufacturing or consulting services.

In this type of business, you effectively position yourself as a "key resource" or "partner" in the business model of others.

Small businesses such as Adafruit and Oomlout sell electronic components to hobbyists. Manufacturers produce printed circuit boards (PCBs) and other custom electronic products for device manufacturers such as Things.

9.8 PROVIDE INFRASTRUCTURE: SENSOR NETWORKS

The sensor network consists of a group of small powered devices and a wireless or wired network infrastructure. Record conditions in any number of environments, including industrial plants, farms, and hospitals.

The sensor network connects to the Internet or computer networks to transfer data for analysis and use. Nodes of the sensor network together define and control the environment. They provide interaction between people or computers and the environment.

Sensor data is a fascinating topic on the Internet of Things: while official data sources exist, often very precisely calibrated and expensive to create, they can be difficult to obtain, and of course, only exist if a government agency or company decides to apply. its large but limited resources.

The long tail of third-party data sensor enthusiasts can complement and sometimes surpass official information streams. You need a platform to aggregate this data, and one of the companies competing for this role is Xively. They allow each consumer to get a real-time data stream from the sensors and data from multiple channels can be displayed, graphically represented and compared.

From the outset, Xively has been committed to providing a free open-source public data infrastructure as well as delivering advanced business offerings with enhanced privacy options and options, and formal service level agreements.Sensor data is information that can be freely shared or simply sold. Many energy suppliers are implementing smart meters that promise higher efficiency and thus lower bills, but also collect huge amounts of information.

In terms of the business model, you need to consider the legitimacy and whether it is in line with your company's values

9.9 TAKE A PERCENTAGE

In the example of a sensor network, if the value of the collected data exceeds the cost of the physical sensor device, you can make that physical product available for free. In fact, energy companies do this quite often with smart meters.

Even without charging the end user of an IoT device, there are many opportunities to make a profit somewhere (advertising revenue, data fees from companies or government agencies, data bandwidth fees, etc.).

In the rapidly evolving field of IoT, what exactly a "product for sale" consists of is an area that still needs to be explored.

9.10 FUNDING AN INTERNET OF THINGS STARTUP

Finance is the fuel that drives business. A business may use a variety of routes and channels to obtain funding, and there are often multiple channels. The choice of the type of financing depends on the type of business, the current business situation and the direction in which the owners intend to develop.

The challenge of securing your initial funding is critical and there are several options worth considering. If you have enough personal money to devote yourself entirely to building a new IoT startup without taking up any extra work, you can of course finance your own business.

Apart from the risk of investing in a personal project that has no real chance of success, it will be a very fortunate situation. Even more fortunate is the surplus of money to cover material and personnel costs.

9.10.1 Hobby Projects and Open Source

If your project is also your hobby, you may not incur any additional costs, apart from those you would spend in your spare time anyway. This is completely true, and in order to transform your product into a successful business or community, you may want to operate at a less leisurely pace than a normal hobby would entail.

One way to speed up the development of a project is to publish all the details as open source and try to build a community around it.

9.11 VENTURE CAPITAL

Startups with growth potential require a certain amount of investment. Wealthy investors love to invest in such companies with long-term growth prospects. This capital is known as venture capital and investors are called venture capital investors.

Venture capital is money invested in a small business; or they only exist as an initiative but have enormous growth potential. Venture capital investments are also called venture capital or patient venture capital as they come with the risk of losing money if the venture fails and the investment requires a medium to long term return.

Venture capital is the most appropriate financing option for an expensive source of capital for companies and, in most cases, for companies with high seed capital requirements and no other low-cost alternatives.

- High Risk
- Lack of Liquidity
- Long term horizon
- Equity participation and capital gains
- Venture capital investments are made in innovative projects
- Suppliers of venture capital participate in the management of the company

9.12 GOVERNMENT FUNDING:

A government grant is a financial contribution made by a federal, state, or local government agency to a useful project of some kind.

It is indeed a gift; it does not include technical or financial assistance, such as a loan or loan guarantee, direct allocation of interest, or income sharing.

The fellow is not expected to repay the money. Government grants help fund ideas and projects that provide public services and stimulate the economy. Governments often want to promote industrial and technological development in their country and can provide funding to achieve specific goals.

Although governments can and do create their own venture capital funds or partner with existing funds in various ways, they generally manage most of their funds differently.

Every government funding has two achievable attributes such as

1. **Outputs:** this metric is used to determine if they are doing what the agency wants to fund. This metric may simply be proof that you are managing money well, or it may be related to the goals that the corporation itself is trying to promote. You may need to write reports on a regular basis or meet certain defined milestones on time.

2. **Spending constraints:** Some financings may require you to spend some of the money on business consulting or web development, for example, possibly with the company or partners of the fund broker. Of course, this requirement can be very valuable, but the practice seems easily misleading; It would be better to illustrate this as financing in kind than in cash.

9.13 CROWDFUNDING VS CROWDSOURCING

Crowdfunding	Crowdsourcing
Crowdfunding is the process of sourcing money or funds from a group or groups of people.	Crowdsourcing is the process of sourcing information or skills or end products from a group or groups of people.
In a crowdfunding campaign, a person, business, or organization raises a relatively small amount of money from a large group of people. Crowdfunding is a method of raising capital through the collective effort of friends, family, customers, and individual investors.	The term "crowdsourcing" comes from two other words: crowd and outsourcing. Outsourcing is defined as "obtain (goods or a service) from an outside or foreign supplier, especially in place of an internal source."
Crowdfunding is an excellent way for groups and people who don't have access to sources of or need for large amounts of capital — like big investors or bank loans — to raise money.	While crowdfunding is pretty straightforward, crowdsourcing is more varied. At the most basic level, crowdsourcing is using the knowledge and power of the crowd to speed up a process.
The three primary type's are donation-based, rewards-based, and equity crowdfunding.	That process could be anything from a survey to a website to a certain skill that a company or individual is seeking. While any "crowd" could theoretically be used for crowdsourcing, the term is generally used to mean sourcing from crowds online.
Examples: Kickstarter Kickstarter is one of the first crowdfunding sites in India. The platform is most known for creative projects. Films, books, research, and innovation are some of the kinds of campaigns that are often seen on this platform. Wishberry For those looking to fund creative projects, Wish berry is the platform to go to. The website is strictly for creative projects only. Art, comics, publishing, theater, music and dance, the platform supports all	Examples: Wikipedia is a crowdsourced encyclopedia, with thousands of people from around the world contributing every day. Asking a Facebook community for opinions on which bank to use is an example of crowdsourcing. The hugely popular technology development site GitHub often utilizes crowdsourcing to solve technology problems.

projects, creative. It also helps campaigners with consulting and marketing services. Like Kickstarter, Wish berry is also a rewards-based platform. Funders and donors receive rewards from the projects that they fund. 3)Milaap – crowdfunding for personal and social causes Most preferred for raising funds for personal and social causes, Milaap is a platform for individuals, NGOs and social entrepreneurs to raise funds for causes. The platform offers micro-loans to people in rural and underserved regions in the country to support projects related to education, water, sanitation, health, and energy. Focus area: low-income borrowers, small funds for social causes and enterprises.	
With crowdfunding, people are looking specifically to raise money from the crowd.	With crowdsourcing, people are seeking pretty much anything else: information, work, talent, skills, etc.

9.14 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).
- A lean start-up is a method of starting a new business or introducing a new product on behalf of an existing company. The Lean Startup method advocates the development of products whose desire has already been demonstrated by consumers so that a market already exists when the product is launched.
- Lean reduces the first two constraints by helping start-ups launch the products customers really need faster and cheaper than traditional methods, while the third makes startups less risky.
- Lean Startup is a business and product development methodology aimed at shortening product development cycles and quickly determining the profitability of the proposed business model; it does this by combining experimentation with business hypotheses, iterative product releases, and validated learning.

Example: 1. Dropbox 2. ZAPPOS

9.15 LETS SUM UP

Success of any business is incomplete without proper planning and execution. In this unit we have studied Business Model as a reference model for all those who have intention to do business and need a roadmap for executing each and every step systematically. There are 8 steps for implementing a Business Model which has been described and illustrated with a diagram. Every business model has a key characteristic of making things and selling them. However there are different ways that have emerged to promote business like subscriptions & customization. The only challenge faced by businesses nowadays is funding. In this unit we have studied different types of funding like venture capitalist, government funding, crowdfunding & crowdsourcing.

9.16 LIST OF REFERENCES.

- Designing the Internet of Things Adrian McEwen, Hakim Cassimall
- Internet of Things – Architecture and Design Raj Kamal
- Getting Started with the Internet of Thing--Cuno Pfister

9.17 BIBLIOGRAPHY

https://www.startups.com/library/expert-advice/crowdfunding-vs-crowdsourcing

9.18 UNIT END EXERCISES.

Q.1 what are the main success criteria for business?

a. Loss

b. Profit

c. Failure

d. Quality

Q.2 Who said this "It is no exaggeration to suggest that the invention laid the foundations for an information culture which is currently exemplified by the Internet and the World Wide Web.

a. Kevin James

b. Gutenberg

c. Bill gates

d. Alexander

Q.3 Ethic of mass production resulted in new business models which pioneered both "self-service shopping" and the sale of a whole range of products under one roof. Example is_____

a. Supermarkets

b. Fast-food franchising

c. Hypermarket

d. Wholesale

Q.4 Who wrote the popular templates for working on a business model is the Business Model Canvas?

a. Alexander Osterwalder

b. Gutenberg

c. Eric Anderson

d. Matthew Henry

Q. 5 What are the ways of reaching the customer segments in Business Model Canvas.

a. Reference

b. Bridges

c. Channels

d. Invitation

Q.6 _____is a Creative Commons–licensed single-page planner.

a. Template

b. Design

c. Boxes

d. Canvas

Q.7 A_____model might be appropriate, allowing you to recoup model costs and possibly make ongoing profit by charging fees for your service

a. Publication

b. Advertisement

c. Distribution

d. Subscription

Q.8 In the _____model, a smaller or larger part of your product is free, while the users are also encouraged to pay a premium to get additional features or remove limits.

a. Subscription

b. Freemium

c. Free

d. Advertisement

Q.9 _____ are the people you plan to deliver the product to.

a. Customer Segments

b. People

c. Client

d. Customer relationship

Q.10 _____is the practice of funding a project or venture by raising small amounts of money from a large number of people, typically via the Internet.

a. Government funding

b. Kickstarter

c. Lean start-ups

d. Crowdfunding

❄❄❄❄❄❄❄

10

MOVING TO MANUFACTURE

Unit Structure

10.0 Objectives
10.1 Introduction
10.2 What are you producing
10.3 Designing Kits
10.4 Designing Printed Circuit Boards
10.5 Software Choices
10.6 The Design Process
10.7 Board
10.8 Manufacturing Printed Circuit Boards
10.8.1 Etching Boards
10.8.2 Milling Boards
10.9 Third Party Manufacturing
10.10 Assembly
10.11 Testing
10.12 Mass Producing the Case and Other Fixtures
10.13 Certification
10.14 Cost
10.15 Scaling up Software
10.16 Deployment
10.17 Correctness and Maintainability
10.18 Security
10.19 Performance
10.20 User Community
10.21 Summary
10.22 Chapter End Questions
10.23 References

10.0 OBJECTIVE:

- In this chapter we dig into both the obvious and the less obvious steps in turning your prototype into a finished product.
- Student will get knowledge about the PCB that can be used for developing and embedded product.
- Students will be able to understand the ethics that are related in making and selling finish product.

10.1 INTRODUCTION

After creating and showcasing your prototype to friends or on the Internet, one to most frequent questions they ask is “Can I get one?” and you start to wonder about creating the product and selling it.
How many devices are you looking to sell? Even if you’re just going to make a few in your spare time, what would happen if your idea turns out to be a huge hit and you have half the Internet beating a path to your order page? It’s a nice problem to have, but if your costings expect your labour to be free because you enjoy soldering up a few boards, you might need to rework those costs more realistically. Either that or you will soon find out just how much you enjoy soldering boards when you’re doing it all day every day to meet demand.

If you don’t want to make each item by hand, you will need to find someone who can do the job for you. Hiring out work is an extra cost, naturally, but at the same time you’ll usually be buying the components in greater bulk, which will be cheaper, so things often balance out. If you are making enough, the economies of scale of automation kick in, which means you can either lower your prices or increase your profits—or, ideally, both. Then there are the less obvious things. Will you need to design some packaging so that the products aren’t damaged during shipping or to make them appealing when sat on a shelf alongside a host of other gadgets? And what about certification?

If you’re selling the product in the United States, the Federal Communications Commission (FCC) will want to make sure it is safe and won’t be throwing out a lot of unwanted electromagnetic interference. In Europe, you’ll need to meet similar requirements to use the CE mark, and there are additional regulations such as the Restriction of Hazardous Substances (RoHS), a directive which bans the sale of products containing certain substances (such as lead and cadmium) above given levels

10.2 WHAT ARE YOU PRODUCING?

Before we get into the specifics of how to scale up production of your device, it’s useful to pause for a moment and consider what exactly you’ll be producing. Depending on your motives and desires for the product, not to mention 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 Instructible. Aside from maybe a bit of promotion on sites such as Make or Hack – a - Day to help people find it, you are basically done. A huge maker community on- and offline share their projects and experience this way. Joining in is a great way to meet fellow makers and to give something back to the community if they helped you get started.

You might find that you become the go-to person for that sort of project, and people might commission you to build related projects for them or want to hire you to work in their company.

When you decide that you want to get your invention into the hands of many more people, you can generally split your choices into three categories :

i. A kit that your customers can assemble themselves.

ii. A ready - made electronics board which users can use as a sub - assembly in their own projects.

iii. 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.

Each of these options builds on a lot of the work you would already have done for the previous one. A complete device will contain an assembled electronics board, and the electronics board would need a printed circuit board (PCB), much like the one users would need to solder components to in a kit.

10.3 DESIGNING KITS:

The first step towards selling your idea as a product is to provide it as a kit. Although you might think it is simple to order the relevant parts and then lay them out to follow your schematic, you are likely to be

underestimating how much you have learned in working out how to make it.

For every person like yourself who gets to the end of building something, there are many more who would have given up halfway through or found it too daunting to even start. Most kits tend to provide only the electronics and software for a particular application rather than any physical housing.

The reason partly comes down to the difficulty, for a cottage kit-building industry, of sourcing custom plastic components.

Also, because the main market for such kits is others in the maker community, these makers will perhaps prefer to combine the kit into a project of their own. However, with the growing accessibility of 3D printers and laser - cutters, it is becoming vastly more feasible to provide housing and other physical components even for kits.

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 ecosystem or a cape on the Beagle Bone.

This makes sense because it reduces the support overhead for the kit provider; either users will already be familiar with the platform, or, if not, plenty of assistance will be available elsewhere to cover the basics of getting up and running. The kit's documentation can then focus on just what is specific to building the project. 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. 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.

The Instructible website has a number of guides to help out, and another good approach is to copy the procedures used by other kit makers that you've found useful. Working out what price to charge is harder and more of an art form than a science. Some rules of thumb can help, though. The most obvious is to understand what your costs are.

You should create a spreadsheet or 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. The BOM gives you the marginal cost for your product and doesn't include any of the fixed costs in setting up for production or developing the software, as they'll be spread out across all the items you sell.

A good starting point for working out your price 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. This means that if you end up with a hit on your hands, you can enlist some intermediaries to help scale up distribution without losing money on each kit you sell.

The most important cost to drive down is that of the BOM. That will give you the most flexibility in setting your price so that you can maximise the profits that you make. By looking at the prices of similar products, you can get a flavour of the market and use that to guide your pricing. You don't need to necessarily find products which solve the same problem; you can also look at ones that are delivered in a similar form.

For example, if your kit is a shield for Arduino, you can compare prices of other shield kits to get a feel for the range of prices that people expect to pay. Because the customer is responsible for assembling and soldering the kit together, usually the only thing you'll need to get custom made just for your application is the bare PCB. That tends to keep your costs down, as you're offloading the labour required to build the circuit to the kit's user.

For the target demographic for a kit, the need for assembly can be seen as a benefit because it adds to the sense of achievement and ownership in getting the kit working. However, it can also add to the support overhead, because you will need to deal with remotely debugging your customers' issues, which may be down to their poor soldering rather than defects in your work.

These problems can all be resolved by moving on a step towards a consumer product and selling fully assembled PCBs, populated with all the components. If you aren't selling too many and your design doesn't include any especially fine-pitched or complicated surface-mount components, you might decide to assemble them yourself; otherwise, you'll need to find an assembly house to do the work on your behalf. Either option will also benefit from your working out a way for you to test the finished board, checking that the components are soldered in properly and, ideally, that they function correctly.

Obviously, 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. This incorporates the assembled PCBs and adds more components or processes.

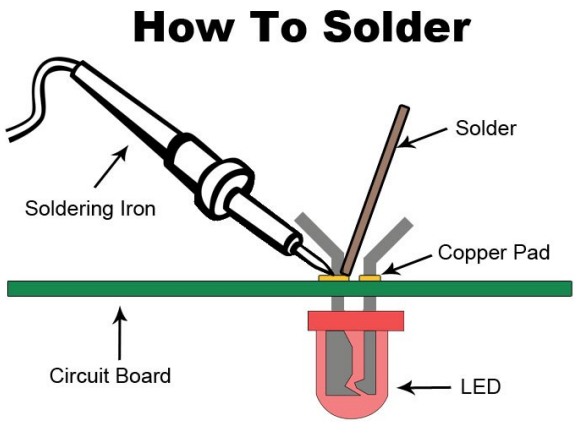
10.4 DESIGNING PRINTED CIRCUIT BOARDS

Soldering things up is a good step towards making your prototype more robust, because the connections should, if your soldering skills are up to par, mean that you have a solid electrical connection that won't shake loose. 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.

It also lets you use components that don't easily fit into the stripboard grid pattern of holes, including some of the simpler surface-mount components. 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.

The range of options for building a custom PCB runs from etching (or milling) boards yourself, through using one of the many mail-order batch PCB services, to having them made and populated for you.



Whichever of those options you choose, the first step in creating your PCB is going to involve designing it. Before we investigate the available software for PCB design, we should look at what makes up a PCB and some of the terms you are likely to encounter. The PCB is made up of a number of layers of fibreglass and copper, sandwiched together into the board. The fibre glass 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, 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.

For surface-mount components, they are just an area of copper on the top or bottom of the board, which provides the place for the component to be soldered. For through-hole connections, they also have a hole drilled through the board for the leg to poke through.

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, predictably, have two layers of copper: one on the top and one on the bottom. More complicated circuits, particularly as you start to use more advanced processors which come in smaller packages, may need even more layers to allow room to route all the traces round to where they are needed. Three-or five-layer boards aren't uncommon, and even some seven-layer boards are used for really complicated circuits.

- 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.

- This is a similar, though generally 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; however, because of this, they complicate the PCB manufacturing process and are best avoided unless absolutely necessary.

- 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.

An added advantage of this approach is that the ground plane provides a way to "catch" some of the stray electromagnetic signals that can result, particularly from high-frequency signal lines. This reduces the amount of electromagnetic interference (EMI) given out by the circuit, which helps prevent problems with other parts of the circuit or with other nearby electronic devices. 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, giving the traditional colour of most circuit boards, though other colours are also available. 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.

This last piece of information is vital; it is more likely than not that you'll end up with a few iterations of the circuit design as you flush out bugs. Being able to tell one version from the other among the boards on your workbench, or, more importantly, knowing exactly which design is in a product with a user reported fault, is essential

10.5 SOFTWARE CHOICES

As you might expect, 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 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. Fritzing even offers a fabrication service to make it simple to make your design a reality. You also can export the breadboard design view as an image or a PDF.

KiCad 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. 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. That led to a wealth of how-to guides and other helpful resources for EAGLE being developed and shared by the user community.

A more recent rival to EAGLE in the commercial pro-sumer market is Design Spark PCB. It is provided by electronics distributor RS Components as part of the company's Design Spark community and, as a result, is free of charge. Unlike EAGLE,

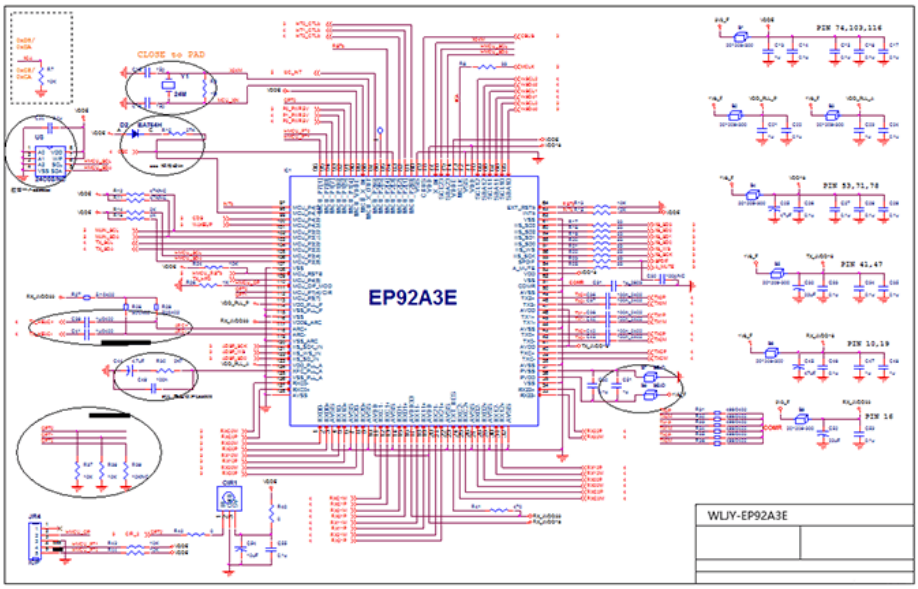
Design Spark PCB does not restrict the size of boards you can design (up to 1m2) or the number of layers (it supports up to 14 layers).

However, it is the only program described here which isn't available for Linux or Mac. Compatibility is an issue which RS Components acknowledges and does endeavour to ensure that it works under the Windows compatibility layers Wine on Linux and CrossOver for Mac; however, in practice it's a little clunky.

10.6 THE DESIGN PROCESS

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.

You usually start the design in the schematic view.



Schematic View of PCB

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 standardized 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. Adding one of those is merely a case of finding the relevant part and sometimes adding the exact part number.

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. The manufacturer's datasheet lists the available packages for that part and how they differ. You usually have a choice of packages depending on the soldering style (through-hole or surface-mount) and several other criteria.

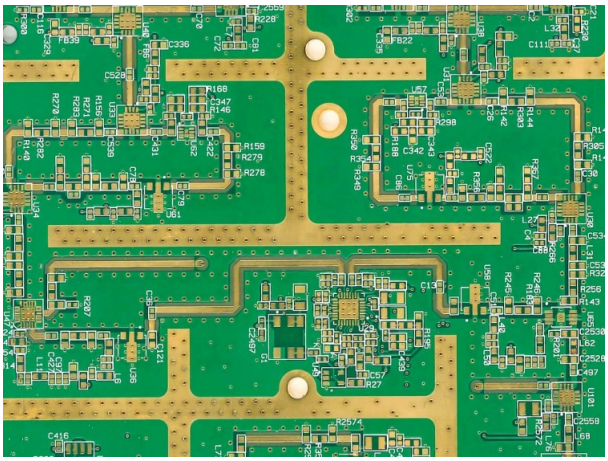
The choice of package doesn't actually affect the schematic because at this point you are interested only in the functionality of the component rather than its physical attributes. However, making the right choices when initially choosing the component will save you having to rework that down the line when you come to the board view.

It's not uncommon to find that you want to use a part which isn't included in your PCB software's component library. If you're lucky, you might find that someone else has provided it in a third-party library. Otherwise, you'll have to design your own.

Creating the design is not difficult because all the information you need is in the datasheet, but attention to detail here will pay dividends with how well the part sits on the finished PCB. When you are happy with the schematic design of your board, you can proceed to laying out the physical board. Although it makes sense to approach the PCB design in this manner, it is not a rigid procedure.

You can flip back and forth between the two views of the design, but finishing the schematic as best you can before moving to the physical aspect will minimize the amount of rework you'll have to perform if the design changes

10.7 BOARD



Finished view of PCB

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. And you might find 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. After that, it's just a case of arranging components in a way that makes the routing of wires easiest.

Start by placing the components which have the most connections first and then fitting the remainder around those constraints. As you move the

components around, you may notice a criss-crossing of fine lines joining some of them together, which seem as messy as cheese strings trailing from a slice of pizza. These connections join the various pads on the components to each other. As you position the components, you should try to reduce how tangled they are, but don't worry about this issue excessively; you're just aiming for a good starting point for the work of routing the connections properly.

Your PCB software has an auto-route function, which you can use to route all the tracks for you. However, such functions are far from perfect, and in practice, you will find it best to at least lay out the more important tracks first by hand, if not to do all the routing manually. 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 such as nonobvious connectors and to identify the board itself.

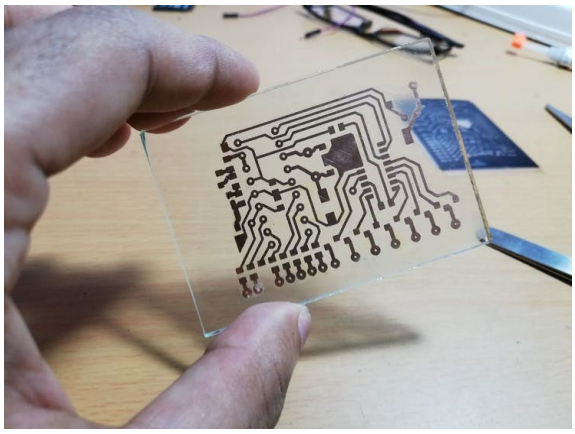
Also include a version number so that you can differentiate any future revisions, and also add maybe your name or the company name and a website for people to find out more. Then give your PCB a final once-over.

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

10.8 MANUFACTURING PRINTED CIRCUIT BOARDS

Now that you've designed your PCB, the next step is to make one or lots of them. 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.

10.8.1 Etching Boards



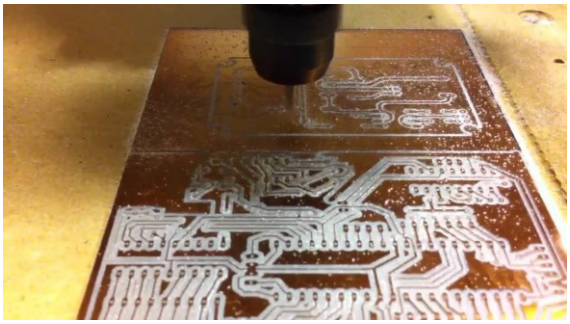
- The most common PCB-making technique for home use is to etch the board. The first step is to get the PCB design onto the board to be etched. This process generally involves printing out the design from your PCB design software onto a stencil.

- 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; or if you're using the toner-transfer method, it will be for your laser printer to print onto glossy paper ready to transfer. 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; and for the toner- transfer method, you'll use a super-hot iron. 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. You can do this by hand, or, if you have access to a CNC mill, you can export the drill file from your PCB design package to provide the drill locations for your mill.

10.8.2 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 (to save time milling it away).

10.9 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, so it's worth getting a few quotes before deciding which one to use.

If you need the boards quickly, a local firm is your best bet and generally has a lead time measured in days. If you have the luxury of more time, you can cast your net further, including to China, which might reduce your costs but could mean a few weeks' wait before you receive your order. 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 (which gives the PCBs their colour and stops any solder adhering to areas where it shouldn't), silkscreen (for the labels, versioning info, and so on) and drill files.

10.10 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. Surface-mount assembly is a little more involved but quite achievable if you don't have any components with particularly complicated package types.

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. The solder for surface-mount work comes as a paste, supplied in tubs or tubes. Using a squeegee and the solder paste stencil, you need to put down an even layer of solder over all the component locations and then carefully lift the stencil off the board.

Now comes the tricky bit. Using tweezers and ideally a loupe or magnifying glass, place each component onto the relevant spot on the PCB. The paste holds the parts in place to a degree, but take care not to knock the board at this point in case some of the parts get displaced. 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. 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. The maker-community retailer Sparkfun Electronics has a good tutorial covering some of the techniques that it has used for surface-mount soldering.

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. The components to feed into the machine are supplied in a form known as tape and reel. Considering all this, if you are running your own pick-and-place machine. This complexity can be reduced if you 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 PC 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. But offloading the work to someone who specialises in it has other benefits, too. The extra throughput that they deal with means they will naturally employ dedicated staff to run the production lines. Paradoxically, those staff will most likely be both more skilled than you are at running the pick-and-place machines and doing any hand soldering and cost less per hour than you do. This then frees you up for the many other tasks involved in bringing a product to market or to working on your next idea. 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. If the manufacturer has dealt with the supplier before, its reputation might let you negotiate a better deal.

10.11 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. Actually, through the automated assembly process, you might have had some testing steps included already. Assembly lines can include automatic optical inspection (AOI). In this process, a high-resolution camera inspects some aspect of the board and its components; for example, it could check that the solder paste is laid properly before the board goes into the pick-and-place machine and compare it to a known good version. Any boards which vary from the “golden” reference version by too high a margin are flagged for further checks from a skilled human operator.

After the boards pass the AOI, the next step is to run them through a functional test. This step is something that you can, and should, be doing even with boards that you've soldered by hand. The functional test just involves powering up the board as it will be used in the finished product and ensuring that it does what it is supposed to. However, that might take a nontrivial amount of time. The focus here is not on ensuring it will run through all normal operations, but just that the PCB and its components are soldered correctly, that none of the components are faulty, and that there aren't any manufacturing defects in the PCB itself.

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.

10.12 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. However, whilst minimising labour costs is a good target, it's not the only factor you need to consider in your production recipe; production rates are also important.

To give you a flavour of the sorts of issues involved, 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. 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. Any blemishes in the surface of the mould will be transferred to every part produced using them. 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 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. In assembled products, the parts are naturally removed from the sprue during production.

Like any production technique, injection moulding has its own design considerations. 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. If you need the thicker walls for strength, an alternative is to use ribs to add rigidity without lots of additional plastic.

Some of the common techniques for achieving maximum strength with a minimum amount of material and also ways to mould mounting points for PCBs or screw holes for holding the assemblies together. 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. If your parts are small enough, you can replicate many of them on one mould or collect lots of different parts together.

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. Obviously, there is a section of the mould cavity where the different colours mix, but with careful design, that is just part of the sprue and so discarded.

10.13 CERTIFICATION

One of the less obvious sides of creating an Internet of Things product is the issue of certification. If you forget to make the PCB or write only half of the software for your device, it will be pretty obvious that things aren't finished when it doesn't work as intended. Fail to meet the relevant certification or regulations, and your product will be similarly incomplete—but you might not realise that until you send it to a distributor, or worse still, after it is already on sale.

If you take a closer look at any gadget that's near to hand, you will find a cluster of logos on it somewhere…CE, FCC, UL.… Each of these marks signifies a particular set of regulations and tests that the item has passed: the CE mark for meeting European standards; FCC for US Federal Communications Commission regulations; and UL for independent testing laboratory UL's tests.

The regulations that your device needs to pass vary depending on its exact functionality, target market, and the countries in which you expect to sell it. Negotiating through all this isn't for the faint of heart, and 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 a barrage of tests (hopefully) far beyond anything it will encounter in general use. 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. All the tests are performed on a DUT, which needs to be the final specification for the entire product. As a result, the testing will most likely be a critical point in your delivery schedule, and any problems discovered will delay shipment while you iterate through a new revision to fix the issues.

For the EMC tests, the device is isolated in an anechoic radio frequency (RF) chamber to minimise the chance of external electromagnetic interference confusing the tests. The resultant test report is added to your technical file, which is referenced by your declaration of conformity. Assembling this is a requirement for certification and documents the key information about your device and the testing it has undergone. 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 in case the authorities need to inspect it.

For certain regulations you must also notify a specific, named body; for example, circuits that perform radio communication and so intentionally emit electromagnetic interference must be registered with the FCC when sold in the US. Such registration is in addition to issuing the declaration of conformity for self- certification. Because of the added complexity and overhead—both administrative and financial—of some of the more involved directives, it is often wise to use pre- approved modules. You therefore can include a WiFi module (chips, antenna, and associated circuitry), for example, or a mains power adaptor, without having to resubmit to all the relevant testing. As long as you don't modify the module in any way, the certification done by its manufacturer is deemed sufficient, and you just need to include that in your technical file.

10.14 COST

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. However, the upside is that you'll be able to get your awesome connected device into the hands of many more people and that's much more important. If you've raised enough money upfront, the task of managing the "manufacturing project" will be easier, but as with all projects, there will still be time scales and dependencies to look after.

You don't need to be able to wield a Gantt chart or have a PRINCE2 project management qualification, but 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.The time between ordering an item and its being ready to use can come as a bit of a shock. You are likely to encounter this situation first with PCB manufacture, where it can often take a couple of weeks from sending off the Gerber files to having something to solder up. Faster turnaround times are possible, but you tend to pay a premium.

The setup and tooling for processes such as injection moulding can be worse. Getting the tool machined and tested and then fine-tuned is likely to take a month or two. Sometimes during that process you're involved, but there also will be big chunks of time while you sit waiting for the work to be done. With careful planning, you'll be able to work on other parts of the project while you are waiting—maybe finishing the server coding or preparing the marketing materials—but you will still need to block out the time for it in your project plan.

Working out the likely cost for your project is a bit of a black art and obviously depends greatly on the complexity of what you're trying to achieve. 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. When buying components in small quantities, you will often notice that the price list also shows the cost if buying in greater bulk. And to get through certification, 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 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.

10.15 SCALING UP SOFTWARE

Producing a physical thing as a prototype or as a manufactured product turn out to be two entirely different propositions. The initial prototype may well be of different size, shape, colour, materials, finish, and quality to what ends up on the shelf.

Yet 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. Yet, as with the physical form and electronics of the device, software needs to be polished before it can be exposed to the real world.

After looking at what is involved in deploying software—both on the embedded device and for any online service you have built—we look at the various factors that require this polish: correctness, maintainability, security, performance, and community.

10.16 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 be done once only, during the manufacture of the device. The software will be flashed to the device in a similar way to how you updated your prototype device during development.

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.

Finally, as a malleable and user-facing software product, there is always the possibility of updating the code to add value by introducing new features or improving the design. As a result of all of this, it is not merely possible, but 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.

10.17 CORRECTNESS AND MAINTAINABILITY

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. As it lives in a central place, the server software is easy to update, either to fix bugs or to introduce new features. This is a real boon for web applications, as it removes an entire class of support issues. 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, and that is one of the selling points of the Electric Imp platform, for example. However, it should be approached with some caution: what if a firmware update itself caused a failure of a home heating system? And as Internet of Things products become more integrated into our lives and our homes, they become a tempting target for hackers. If your device can update its own code, the channels for delivering and authenticating any updates must be rock solid, lest they be compromised.

10.18 SECURITY

Apart from securing the communication between client device and web server, you should consider the server itself; as an always-on, always-connected device visible to the whole Internet, it is the most obvious target. Following are some of the more important guidelines :

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.

User passwords should never be stored in plain text. If your database were ever compromised, an attacker could easily log in as any user. Passwords should be encrypted with a secure algorithm which is not known to be trivially cracked, and "salted" for additional security.

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, or including unfiltered user input in your HTML pages, as this could allow a cross-site scripting (XSS) attack. Strip out all HTML tags or escape the output.

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.

10.19 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, but in fact the other factors are usually far more critical. 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.

In general, 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.

10.20 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. While few large companies can boast Apple's

famous commitment to “insanely great customer service”, a small, focused startup can often match them for responsiveness and enthusiasm.

At a minimum, you need a support email or a bug-reporting tool, 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.

10.21 SUMMARY

- Building physical products is hard, and products that connect with the Internet take up the complexity another notch. However, the finished items elicit a reaction that is hard to match with a website or an app.
- Moving to manufacturing involves many skills and tasks that are quite different from those you used to bring the idea to life, and it can feel quite daunting when you realise all that needs to be done.
- However, this process is something that the world has been perfecting for over a hundred years and even in the electronics industry is a few decades old. So if you find the right partners, it needn’t be as difficult as you fear.

10.22 CHAPTER END QUESTIONS:

i. Write a short note on designing kits.

ii. Explain the process of designing printed circuit boards.

iii. What are the software choices when designing printed circuit boards?

iv. Explain the design process for printed circuit boards.

v. What are the different ways in which printed circuit boards can be manufactured?

vi. Explain the process of testing for a printed circuit board.

vii. Explain the issues faced during custom casing or other sub-assemblies with an example.

viii. What different certifications and tests are required for Internet of Things products?

ix. What various factors require to be polished when scaling up the software?

10.23 REFERENCES:

Books and References:

Sr. No.	Title	Authors	Publisher	Edition	Year
1	Designing the Internet of Things	Adrian McEwen, Hakim Cassimally	Wiley	First	2014
2	Internet of Things – Architecture and Design	Rajkamal	McGraw Hill	First	2017
3	Getting Started with the Internet of Things	Cuno Pfister	O'Reilly	Sixth	2018
4	Getting Started with Raspberry Pi	Matt Richardson and Shawn Wallace	SPD	Third	2016

Websites:

www.youtube.com

www.wikipedia.com

www.google.com

https://internetofthingswiki.com

❄❄❄❄❄❄❄

11

ETHICS

Unit Structure

11.0 Objective
11.1 Introduction
11.2 Characterizing the Internet of Things
11.3 Privacy
11.4 Control
11.5 Disrupting Control
11.6 Crowdsourcing
11.7 Physical Thing
11.8 Electronics
11.9 Internet Service
11.10 Solution
11.11 The Internet of Things as part of the solution
11.12 Cautious Optimism
11.13 The Open Internet of Things Definition
11.14 Summary
11.15 Chapter End Questions
11.16 References

11.0 OBJECTIVE:

- In this chapter we dig into ethics that need to be considered while developing any Internet of Things product
- Student should understand how technology permits crowdsourced sensor networks which tips the balance of civic power to let citizens make their own informed decisions about issues that concern them.
- Strong understanding of the ethical decisions we need to make as technologists, we should be ready to help guide the Internet of Things in such a way that it need not become an invasive, oppressive instrument but instead a framework in which we can learn better what it is to be human.

11.1 INTRODUCTION

Technology often creates new possibilities with one hand and new problems with the other. The society has generally two kinds of people,

pessimistic and optimistic. We find more people to be more pessimistic rather than optimistic. The reason may be that the rejection of the new and the different is deeply embedded in societies and can be phrased in more visceral terms.

Take a moment to reflect on your own stance to technology in general. It may be that you are generally anti- or generally pro-technology. It is also possible that you are optimistic about some classes of technology (Internet, space travel) but pessimistic about others (genetically modified food, wind energy).

In many traditional disciplines, there are courses on ethics—for example, in university courses on engineering or computer science. The practitioners of the Internet of Things, as we saw in the beginning of the book, may come from varied backgrounds, from these technical fields, through design, to the fine arts. In this chapter, we do not present an entire course on formal ethics. Rather, we look at several aspects, many of which are related to such an ethics syllabus and consider them in relation to the Internet of Things. We look at many extreme and challenging ideas, some of which may seem like science fiction or may be politically uncomfortable.

11.2 CHARACTERIZING THE INTERNET OF THINGS

Let us start by summarizing what the Internet of Things is, to get a handle on what particular changes it can bring to humanity's relationship to the "good life". We've stressed the fact that a certain powerful technology (computer chips) is suddenly cheap and plentiful.

This is by no means a new observation, nor the first technology that has undergone this explosive transition. This availability of technology brings certain abilities within the reach of not just the powerful but the ordinary citizen. Examples can be found in the fields of 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 adage that "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 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 Where Dial

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, and we refer to them in the sections that follow.

11.3 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: from minutiae about what you ate for breakfast to blog posts about your work, from articles about your hobbies to Facebook posts about your parties with friends.

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.

But do you always want people to be able to see that data? Do you want not just your family and friends but also companies, the government, and the police to be able to see information about you, forever?

A common argument is "if you've got nothing to hide, then you've got nothing to fear". There is some element of truth in this, but it omits certain important details, some of which may not apply to you, but apply to someone:

- You may not want your data being visible to an abusive ex-spouse.
- You might be at risk of assassination by criminal, terrorist, or state organizations.
- You might belong to a group which is targeted by your state (religion, sexuality, political party, journalists).

More prosaically, you change and your persona changes. Yet your past misdemeanor's (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.

Let's consider the mobile phone, in particular an Internet-connected phone with on-board camera. Although we don't typically consider phones as Internet of Things devices, the taking of a photo with a camera phone is a quintessential task for a Thing: whereas in the past you would have had to take a photo, develop it, take the printed photo to your computer, scan it, and then upload it or take your digital camera to the computer and transfer the photo across via USB, now you can upload that compromising photo, in a single click, while still drunk.

The ability to do something is present in a defined context (the personal) rather than locked in a set of multiple processes, culminating in a general-purpose computer.

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 personal details. Similar issues exist with sports-tracking data, whether produced by an actual Thing, such as Nike+ or a GPS watch, or a pseudo-Thing, like the RunKeeper app on your smartphone. This data is incredibly useful to keep track of your progress, and sharing your running maps, speed, heartbeat, and the like with friends may be motivating. But again, it may be trivial for an attacker to infer where your house is and get information about the times of day that you are likely to be out of the house.

The idea of people knowing where you are can evoke strong emotions. Yet the idea of knowing that your loved ones are safe is a similarly deep-seated human emotion. 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.

But the decision to give your mother a Good Night Lamp might seem less sensible months later when you arrive home late at night. Or you might regret giving your partner a WhereDial if later she becomes jealous and suspicious of your innocent (or otherwise) movements.

Even if these devices are themselves respectful of your privacy, their security or lack thereof might allow an attacker to get information. For example, if it were possible to read an IP packet going from the goodnightlamp.com servers to a household, could you find out that the associated "big lamp" had been switched off? Even if this packet is encrypted, could an attacker infer something by the fact that a packet was sent at all? These risks are to be considered very carefully by responsible makers of Internet of Things devices.

So far, we've looked at devices that you, as an individual, choose to deploy. But as sensor data is so ubiquitous, it inevitably detects more than just the data that you have chosen to make public.

For a start, we saw previously that many "things" have little in their external form that suggests they are connected to the Internet. When you grab an Internet-connected scarf from the coat rack or sit on an Internet-connected chair, should you have some obvious sign that data will be transmitted, or an action triggered?

Urbanist and technologist Adam Greenfield has catalogued interactive billboards with hidden cameras which record the demographics of the people who look at them and vending machines which choose the products to display on a similar basis. He comments that, as well as being intrusions into public space, these objects are not just what they seem to be. Rather, they have an agenda of "predicting behaviours and encouraging normative behaviours", which is not transparent.

Perhaps a Thing that is implemented digitally will not give the subtle cues that its analogue counterpart used to give.

Let us consider the electricity smart meter. The real-time, accurate measurement of electricity has many admirable goals.

Understanding usage patterns can help companies to produce electricity at the right times, avoiding overproduction and optimizing efficiency. The aggregate data collected by the companies is useful for the noble environmental goals like conserving energy...but how about individual data?

If you could mine the data to see subtle peaks, associated with kettles being switched on for tea or coffee, perhaps you could infer what television programmes a household watches. If there are four longer peaks in the morning, this might suggest that four family members are getting up for an electric shower before going to school or work. Now what if you triangulate this data with some other data—for example, the water meter readings?

The idea of analysing multiple huge datasets is now a reality. There are smart algorithms, and there is the computing power to do it.

By combining both ends of the long tail (the cheap, ubiquitous Internet of Things devices on the one hand and the expensive, sophisticated, powerful data-mining processors on the other), it is possible to process and understand massive quantities of data. How powerful this ability will be may well depend on what data you have available to compare.

It is very important to note that even aggregate data can "leak" information. 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?

At the Open Internet of Things Assembly 2011 in London, there was a great deal of discussion about who owns sensor data. The electricity

companies? The householder who has signed the electricity contract? Or all the stakeholders in the house? If you visit a friend's house, data about you is collected and "owned" by someone else. This might mean that in future you would be a stakeholder in this valuable data too.

As sensors such as CCTV cameras, temperature meters, footfall counters, 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.

While futurology is fun, it's also hard. As the visionary computer scientist Alan Kay famously said in 1971, "The best way to predict the future is to invent it".

It is clear that we are leaking ever more data onto the online world, and some of this data will be vital to dealing with human crisis of the near future. Rob van Kranenburg, founder of the think tank Council, predicts that the requirement to smoke out inefficiency, for the survival of our species, will lead to a state of post- privacy. Do we have the "courage to live in the light"?

11.4 CONTROL

- Some of the privacy concerns we looked at in the preceding sections really manifest only if the "data subject" is not the one in control of the data.
- The example of the drunken photo is more sinister if it was posted by someone else, without your permission. This is a form of cyberbullying, which is increasingly prevalent in schools and elsewhere.
- 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, in more or less sinister ways.
- In the case of an employer, we are bound to see cases in the future in which a contractual obligation is needed to share data collected by some Internet of Things device. 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. High-end cars

already have Internet connected tracking and security systems which may even be a requisite in getting insurance at all.

- 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, the simple logistical difficulty of an absolute dystopia vanishes.
- Commentary on the mourning at Kim Jong-Il's death noted that the, no doubt genuine, grief of many may also have been amplified by the social and political compulsion to express the "histrionics of grief ".
- In a world where pervasive body-blogging is becoming feasible, it is conceivable that not only the display of grief but also the physiological symptoms of it could be verified.
- 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, and every month one hears news of other suggested legislation which, to a technical specialist, may seem not just badly thought out and unworkable but also immensely dangerous.
- Just as the printing press gave the state a greater degree of control via propaganda, the Internet gives hitherto unknown possibilities for propaganda and monitoring.
- Even in a democratic state, with the readily accessibly technology in place, it is up to the institutions that safeguard democracy and the will of the people to become the main bulwark against the threat of authoritarian control.
- 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.

11.5 DISRUPTING CONTROL

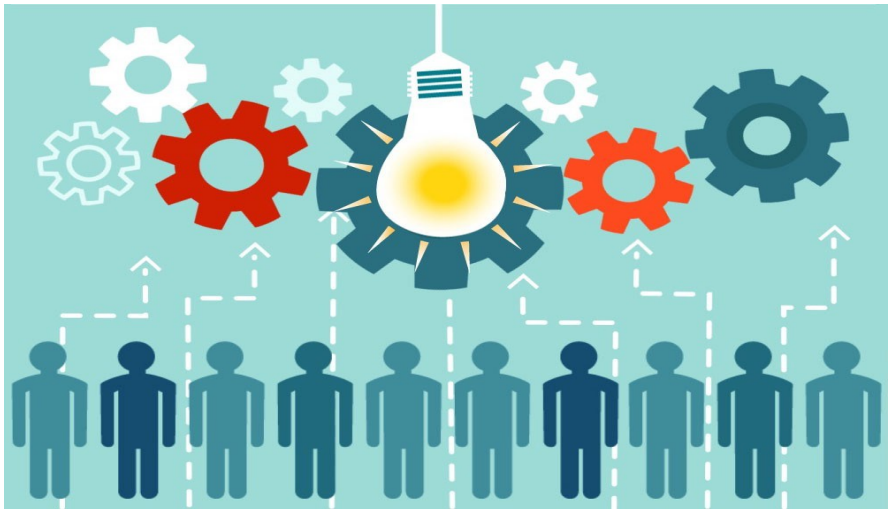
The other major possibility that Eaves suggests is that "The Internet Destroys the State". This is also a hard and uncomfortable scenario to imagine. However, toning down this idea a little, we can see a more likely one of "the Internet" fighting back against an attempt by the state or corporations to co-opt it.

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.

11.6 CROWDSOURCING

One fascinating feature of modern Internet life is "crowdsourcing", from knowledge (Wikipedia, et al.) to funding projects (Kickstarter, Indiegogo) to work (Mechanical Turk). 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. After the Fukushima Daiichi nuclear disaster, there were fears that insufficient information was available to track the spread of the leaked radioactive materials.

Many hackers around the world built Geiger counters, and Xively was a focal point for Japanese engineers to publish their data to. Perhaps the Japanese government or the management of the Fukushima plant would have provided that kind of accurate, widespread data if they could. But power or financial interests might have worked against this.

Andrew Fisher, a technologist with interests in big data and ubiquitous computing, has written persuasively about a quiet revolution of the "sensor commons", his term for this collaborative voluntary effort to provide environmental data.

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.

- **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.

- **Be entirely open** : Being open is perhaps what distinguishes the sensor commons from a government project the most. 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. The project should also have an API and permissive licensing so that the community can choose to do different, complementary things with the data from the network.

- **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.

While Fisher writes specifically on sensor networks, the principles he proposes are, relevant to consider for any ethical project in the field of the Internet of Things.

11.7 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, 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. Calculating the energy costs for manufacture is harder. You may need to consider other environmental factors, such as emissions produced during normal operation or during disposal of the object.

11.8 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. Considering the potential cost savings, even a responsible manufacturer may find it reasonable to offset the extra carbon emissions.

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. Perhaps little can be done to reduce this environmental cost currently, but as companies in other countries start to mine REMs, being aware of the parameters may allow improvements in purchasing choices.

Every electronically enhanced Thing that you produce will incur these costs and will also need to be powered to run. Speaking to the Internet (via WiFi or 3G) is a major part of the power cost of running an Internet of Things device. Anything that can be done to reduce this cost will make the devices more efficient. Choosing suppliers of WiFi chips wisely and following the low-power IPv6 developments (6LoWPAN) closely will be helpful here.

11.9 INTERNET SERVICE

As Nicholas Negroponte (founder of MIT's Media Lab) preaches, "Move bits, not atoms" (Being Digital, Vintage, 1996). 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, parchment scrolls, and libraries of paper books or microfiche

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. Many server hosting specialists now offer carbon-neutral hosting, where you pay extra to offset your emissions. Running inefficient code or services may cause higher power usage. So, of course, will having more customers, although of course we won't go as far as suggesting that you cut down on those!

11.10 SOLUTION

Compared to a simple, physical object, an instrumented Internet of Things device does seem to use vastly more resources in its production, daily use, and waste disposal. Considering our starting point—that this kind of instrumentation is now cheap enough to put everywhere—it seems as though the mass rollout of the Internet of Things will only contribute to environmental issues!

From a more optimistic point of view, it's also true that the realisation that the number of Internet-connected devices will be exploding in the coming years is spurring massive research into low-power efficient chips and communications.

11.11 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.

A UN Environment Programme report warns that the lack of reliable and consistent time-series data on the state of the environment is a major barrier to increasing the effectiveness of policies and programmes. 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.

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.

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. Capitalism's great success has always been how it routes around problems and finds a steady state which is the most efficient to the market. There is no reason why capitalism as-could-be should not be part of the process of striving towards efficiency on an environmental as well as monetary level.

Van Kranenburg also makes alternative, starker proposals: not only may privacy become obsolete, but even those currently personal possessions such as cars might also become communal, through the increasing move from ownership to rental models. As resources become ever scarcer, a greater percentage of income might be spent on covering rental of all goods—cars, food, possibly even housing.

This kind of futurology leads to scenarios such as the death of money itself: a fixed proportion of income to rent needed services from a commercial supplier is more or less indistinguishable from taxation to pay for communal services.

Whether the death of privacy and of money sounds like utopia or dystopia to you, it is worth considering the impact tomorrow of the technologies we implement to deal with the problems of today.

As a counterpoint to these messages of doom, Russell Davies of London's Really Interesting Group (RIG) often tries to bring the discussion of Things back to fun. Although this may not sound as engaged or political an attitude, by looking for the unintended uses for technologies, the end

users, rather than the political elites, can turn them into platforms for human expression.

Similarly, the World Wide Web was originally conceived to share academic papers but has taken on the roles of brokering business on the one hand and publishing pictures of kittens on the other without breaking its step.

The Internet of Things will also, if we let it, become a platform for whatever people want it to be. Although this may be less important than saving our species from environmental disaster, perhaps it is no less ethical in terms of asserting our humanity through, and not simply in spite of, the technology that we might have feared would dehumanise us.

11.12 CAUTIOUS OPTIMISM

Between the tempting extremes of technological Luddism and an unquestioning positive attitude is the approach that we prefer: one of cautious optimism. Yes, the Luddites were right—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.

It is true that any technological advance could be co-opted by corporations, repressive governments, or criminals. But technology can be used socially, responsibly, and subversively, to mitigate against this risk.

Although the Internet of Things can be, and we hope will always be, fun, being aware of the ethical issues around it, and facing them responsibly, will help make it more sustainable and more human too.

As a massively interdisciplinary field, practitioners of Internet of Things may have an opportunity or responsibility to contribute to providing moral leadership in many of the upcoming ethical challenges we have looked at.

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.

This tension underscores the difficulty of trying to figure out overriding concerns about any complex area, such as the Internet of Things!

11.13 THE OPEN INTERNET OF THINGS DEFINITION

The Open IoT Assembly 2011 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. This is a very strong provision (and most likely unworkable for data which is inherently anonymous in the first instance) but one which would provide real individual protection if it were widely followed. As with any information gathering, “reasonable efforts” should be made to retain privacy and confidentiality.

- **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.

11.14 SUMMARY

- Technology often creates new possibilities with one hand and new problems with the other.
- We have looked at the capabilities of connected devices and noted how new technology fosters new business models. Yet just as these new possibilities were born of the defining characteristics of the Internet of Things, so the ethical concerns raised spring from the same essence.
- Inherent in the physical "thing" produced are environmental concerns: the carbon cost of manufacture and of the raw materials required for its complex electronics. Then there is a moral question over whether the workers tasked to construct it are treated fairly.
- Taking complex electronics off the desk and putting them into the fabric of the world and your personal possessions allows sensors to read sensitive data on an unprecedented scale.
- Combined with the ability to aggregate this data via the Internet, it becomes possible for companies to monitor their customers through their telephones, electricity meters, or their air fresheners.
- Similarly, government bodies can monitor their citizens in locations and ways that might not be obvious. This raises questions of transparency and privacy and the issue of our right to have access to data that concerns us.
- As technologists, we don't have to see these concerns as prophecies of doom but as a reminder to consider our responsibilities while we are engaged in designing and building our magical objects to delight our customers and bring us profit or amusement.
- We have also touched on how the technology of the Internet of Things can act as counterbalance to its own potential flaws.
- The widespread sensors and powerful processing can lead to better information and possible mitigations to our impact on the environment.
- The accessibility of the technology permits crowdsourced sensor networks which tips the balance of civic power to let citizens make their own informed decisions about issues that concern them.
- As the field of the Internet of Things develops, we will see more clearly the next most immediate steps to take. Perhaps some of the futures we've envisaged will come to pass, and perhaps they won't.
- With a strong understanding of the ethical decisions we need to make as technologists, we should be ready to help guide the Internet of Things in such a way that it need not become an invasive, oppressive instrument but instead a framework in which we can learn better what it is to be human.

11.15 CHAPTER END QUESTIONS

1. Why is privacy important for Internet of Things?
2. What are the reasons for giving up some control for IoT devices data.
3. Explain disrupting control and crowdsourcing with suitable examples.
4. What is "sensor commons" project? What are the critical requirements for a sensor commons project.
5. What are the environmental concerns about producing and running an IoT device?
6. How can Internet of Things be a part of the Solution to reduce environmental waste?
7. Write a short note on cautious optimism as solution for IoT.
8. What is "Open Internet of Things Definition"? What are the main goals of the definition?

11.16 REFERENCES

Books and Reference:

Sr. No.	Title	Authors	Publisher	Edition	Year
1	Designing the Internet of Things	Adrian McEwen, Hakim Cassimally	Wiley	First	2014
2	Internet of Things – Architecture and Design	Rajkamal	McGraw Hill	First	2017
3	Getting Started with the Internet of Things	Cuno Pfister	O'Reilly	Sixth	2018
4	Getting Started with Raspberry Pi	Matt Richardson and Shawn Wallace	SPD	Third	2016

Website:

www.youtube.com

www.wikipedia.com

https://internetofthingswiki.com

❄❄❄❄❄❄❄