UNIT – I

1

The Internet of Things

Definition of Internet of Things (IoT) :

Internet :

The internet is a vast global network of connected servers, computers, tablets and mobiles that is governed by standard protocols for connected systems. It enables sending, receiving or, communication of information, connectivity with remote servers, cloud and analytics platforms.

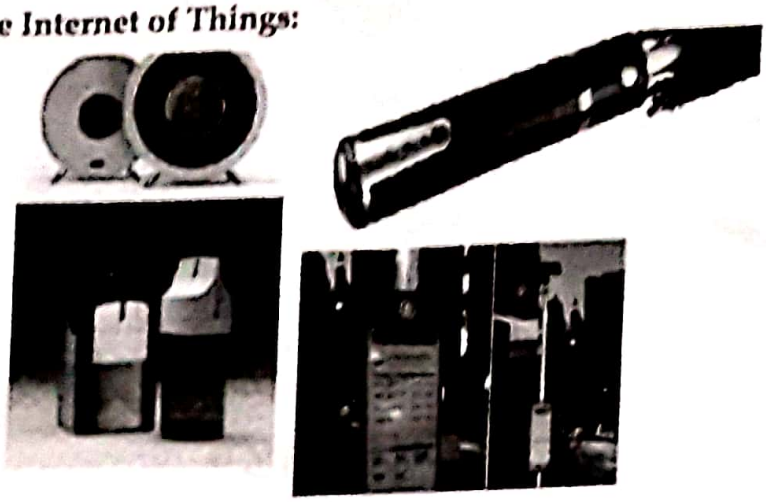
- **Things :** Thing is a word used to refer a physical object, an action or idea, a situation or activity in case when one does not wish to be precise.
- **Internet of things :** Internet of things is a network of physical objects or things embedded with electronics, software, sensors and connectivity to enable it to achieve greater value and service by exchanging data with the manufacturer, operator and/or other connected devices. Each thing is uniquely identifiable through its embedded computing system but is able to interoperate within the existing internet infrastructure.

Relation between IoT and Ubiquitous Computing :

- Ubiquitous or pervasive computing is a concept in computer science and engineering where computing is made to appear everywhere and anywhere. Ubiquitous Computing was designed to make objects intelligent and create richer interaction.
- Internet of Things (IoT) is a new paradigm that includes a network of smart objects, which are embedded sensors, communicating using the Internet. IoT is ubiquitous computing with data being sent over internet. For IoT internet must be involved.
- So ubiquitous is basically superset of IoT.

Scanned by CamScanner

Flavour of The Internet of Things:



- **Examples :**
  - i) The alarm rings. As you open your eyes blearily, you see that it's five minutes later than your usual wake-up time. The clock has checked the train times online, and your train must be delayed, so it lets you sleep in a little longer.
  - ii) 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. It arrives when you turn the next corner. Every bus has GPS tracking its location, they simply connect to the bus company's online service and always give the updated information.
  - iii) On your lunch break, 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.
- All the cases we saw used the *internet* to send, receive, or communicate information. And in each case, the gadget that was connected to the internet wasn't a computer, tablet, or mobile phone but an object, a *thing*.
- A bus display has to be readable to public transport users, including the elderly and partially sighted and be able to survive poor weather conditions and the risk of vandalism. The sports bracelet is easy to wear while running, has a display that is large enough and bright enough to read even when you are moving, and will survive heat, cold, sweat, and rain.
- Many of the use cases could be fulfilled, and often are, by general-purpose computers. Although we don't carry a desktop PC around with us, many

Scanned by CamScanner

people do carry a laptop or tablet. More to the point, in almost every country now, most people do carry a mobile phone, and in many cases this is a smartphone that easily has enough power for any task one could throw at a computer.

- So the idea of the internet of things suggests that rather than having a small number of very powerful computing devices in your life (laptop, tablet, phone, music player), you might have a large number of devices which are perhaps less powerful (umbrella, bracelet, mirror, fridge, shoes). An earlier buzzword for roughly the same concept was "ubiquitous computing".
- The Thing is present, physically in the real world, in your home, your work, your car, or worn around your body.
- This means that it can receive/sense inputs from your world and transform those into data which is sent onto the Internet for collection and processing.
- The presence of the Thing also means that it can produce outputs into your world with what we call "actuators". Some of these outputs could be triggered by data that has been collected and processed on the Internet.
- From all the above points the definition of IoT in equation form can be re-written as below:

Physical Object

+

Controller, Sensors and Actuators

+

Internet

=

Internet of Things

The Technology of IoT:

- Technology's great drivers have initially been fundamental needs, such as food and water, warmth, safety, and health. Hunting and foraging, fire, building and fortifications, and medicine grow out of these needs.
- Then, because resources for these things are not always distributed where and when one might like, technological advances progress with enabling and controlling the movement of people, their possessions, livestock, and other resources. Trade develops as a movement of goods from a place where they

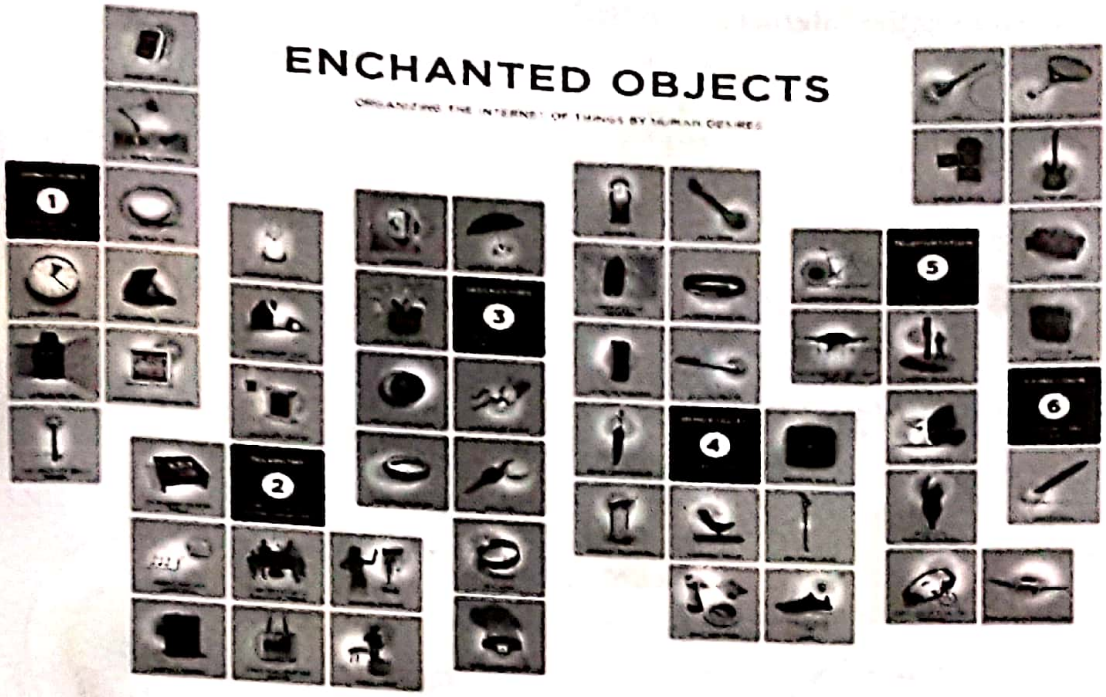
Scanned by CamScanner

are plentiful and cheap to one where they are rare and valuable. Storage is a form of movement in time.

- Information becomes key, too—hence, the development of language to communicate technology to others.
- Travellers might pass on messages as well as goods and services, and an oral tradition allows this information to pass through time as well as space.
- The invention of writing makes this communication ever more important and allows, to some extent, human lives to be preserved in words by and about writers, from the ancient philosophers and poets to the present day.
- From writing, via the telegraph, radio, and television, to digital information, more and more technology has been about enabling the movement of information or doing interesting things with that information.
- Manufacturers of electronic products have started to incorporate general-purpose computer CPUs into their products, from washing machines to cars, as they have seen that it has become, in many cases, cheaper to do this than to create custom chips.
- Internet connectivity is also cheaper and more convenient than it used to be. Whereas in the past, we were tied to expensive and slow dial-up connections, nowadays we have broadband subscriptions, providing always-on connectivity to the Net. Wired Ethernet provides a fairly plug-and-play networking experience, but most home routers today also offer Wi-Fi, which removes the need for running cables everywhere.
- Another factor at play is the maturity of online platforms. Whereas early web apps were designed to be used only from a web browser, the much heralded "Web 2.0", as well as bringing us "rich web apps", popularized a style of programming using an Application Programming Interface (API), which allows other programs, rather than just users, to interact with and use the services on offer.
- 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.
- As 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. Similarly, cloud services such as Amazon Web Services mean that such solutions can scale easily with use as they become more popular.

Scanned by CamScanner

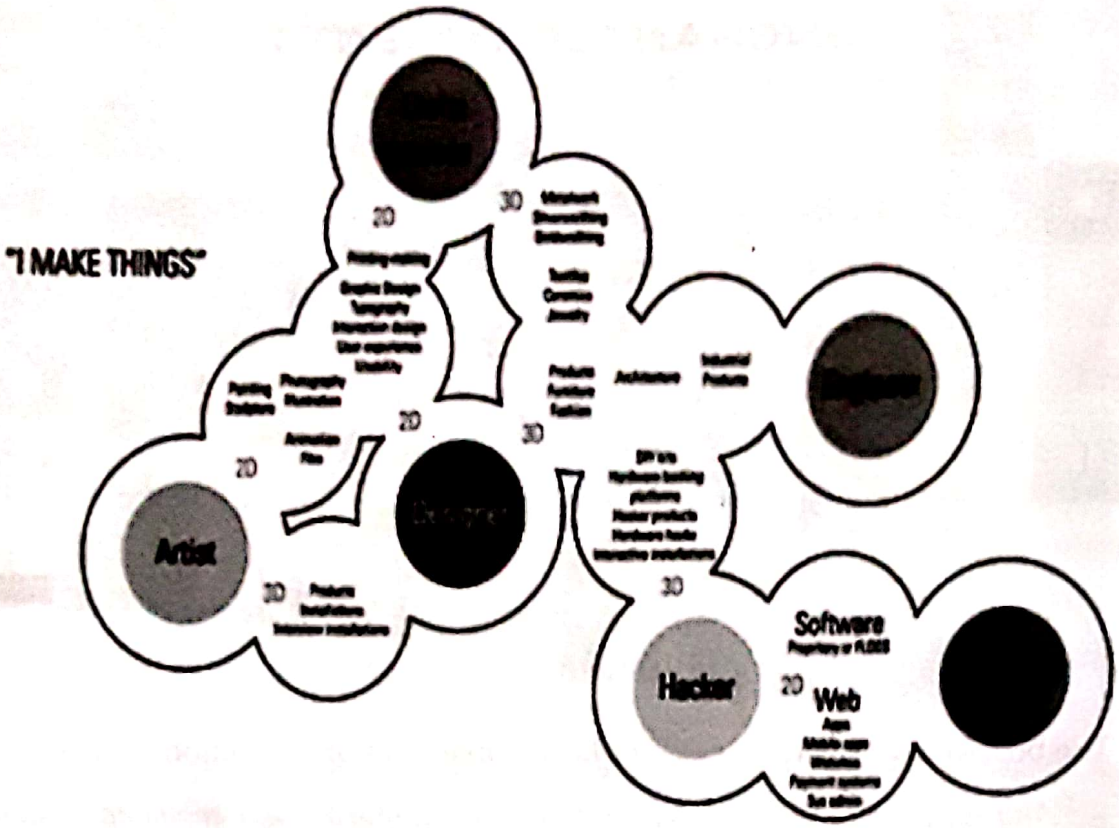
Enchanted Objects:



- The best known of Arthur C. Clarke's "three laws of prediction" states
  - *Any sufficiently advanced technology is indistinguishable from magic.*
  - *http://en.wikipedia.org/wiki/Clarke's_three_laws*
- Magic like technology has evolved to meet our needs and desires.
- The objects in folktales and fairy tales are often wish-fulfilment fantasies to fill the deepest desires.
- **Examples :**
  - For *Protection*, just as magical swords and helmets protected the protagonists of fairy tales from their enemies, so has much of the development of science and technology throughout history been driven by the need for military superiority, for the purpose of security or conquest.
  - Humans have always desired *Omniscience*, from Snow White's wicked stepmother asking "Mirror mirror on the wall, who's the fairest of them all?" to the friends settling an argument of fact by looking up articles from Wikipedia on their smartphones
  - The need for *Creative Expression* is fulfilled in stories by the enchanted paintbrushes or magic flutes and harps, while we have always used technology to devise such creative outlets, from charcoal to paint to computer graphics, or from drums to violins and electronic synthesisers.

Scanned by CamScanner

Who is Making the Internet of Things?



- The disciplines involved in making IoT are as follows :
  - Artist
  - Craftsperson
  - Designer
  - Developer
  - Engineer
  - Hacker
- There are many crossover points between all the disciplines listed.
  - Artists may collaborate with designers on installations or with traditional craftspeople on printmaking.
  - Designers and engineers work closely to make industrial products.
  - Hobbyist "hackers" (in the sense of tinkerers and amateur engineers), by their nature, are a diverse group encompassing various technical and artistic interests and skills.
  - The map isn't complete, and one could raise issue with omissions: no role of "architect" is listed, only the discipline of architecture, straddling the roles of engineer, designer, and craftsperson.

Scanned by CamScanner

- 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
  - An engineer might be required to solve difficult technical challenges, especially in scaling up to production.
  - Every object that is being crafted, designed, and engineered object, is, or could be, the work of an artist also.

QUESTIONS

1. Define Internet Of Things (IOT). Explain any 5 Flavor of IOT.
2. Explain the advantages & disadvantages of an IOT.
3. What is Internet Of Things? Explain the application of IOT.
4. What is IOT Components? Explain each in details.
5. Explain the features of IOT.
6. What do you mean by Enchanted Objects? Explain.
7. Who is Making Internet of Things? Justify it.

Scanned by CamScanner

2 Design Principles for Connected Devices

Calm & Ambient Technology

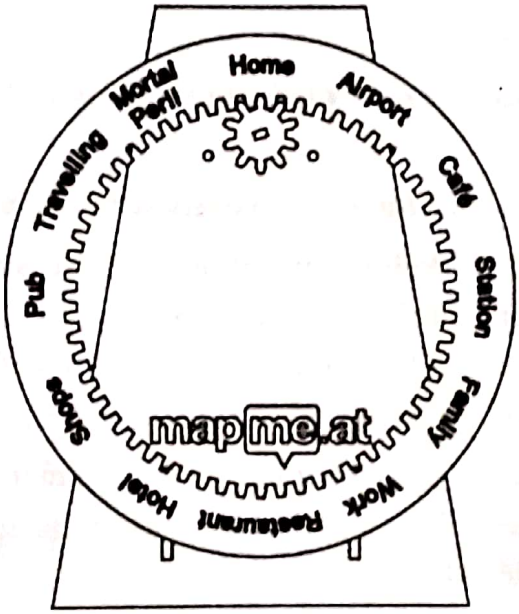
- *Mark Weiser* coined the term "ubiquitous computing" for when computing power becomes cheap enough that it can be embedded into all manner of everyday objects.
- With focus on computing power being embedded everywhere, ubiquitous computing is often also referred to as *ambient computing*.
- However, the term "ambient" also has connotations of being merely in the background, not something to which we actively pay attention and in some cases as something which we seek to remove.
- The term *calm technology – as Mark Weiser defined* —systems which don't vie for attention yet are ready to provide utility or useful information when we decide to give them some attention.
- This rapid increase in computing devices led to new challenges which includes configuration, how to provide power to all these items, how they talk to each other, and how they communicate with us.
- The power and networking challenges are purely technical and are driving developments such as 6LoWPAN.
- Configuration and user interaction, however, obviously involve people and so are difficult problems to solve with just technical solutions.
- According to Eliel Saarinen "Always design a thing by considering it in its next larger context—a chair in a room, a room in a house, a house in an environment, anenvironment in a city plan"
- We should also think of how the connected device will interact as one of a wealth of connected devices.

Scanned by CamScanner

- For connected devices which are just sensing their world, or generally acting as *inputs*, there shouldn't be any issues. They will happily collect information and deposit it into some repository online for processing or analysis.

Magic as Metaphor

- Arthur C. Clarke has claimed that "any sufficiently advanced technology is indistinguishable from magic" and the Internet of Tings commonly bestows semi-hidden capabilities onto everyday objects.
- Some Internet of Things projects draw their inspiration directly from magic. For example
  - John McKerrell's WhereDial



  - Ambient Orb

  - Enchanted Umbrella

Privacy

- The Internet of Things devices that we own aren't the only ones that should concern us when it comes to matters of trust.
- With more sensors and devices watching us and reporting data to the Internet, the privacy of third parties who cross our sensors' paths (either by accident or design) is an important consideration.

Scanned by CamScanner

Keeping Secrets

- For certain realms, such as health care, privacy concerns are an obvious issue, even seemingly innocuous applications can leak personal information, so you should be alert to the danger and take measures to avoid it.
  - *Don't share more than you need to provide the service.*
  - If you can avoid gathering and/or storing the data in the first place, you need not worry about disclosing it accidentally.
- In this day and age, it is standard practice to never store passwords as clear text. You could also consider applying the standard mechanisms for password encryption, such as the one-way hash, to other pieces of data.

Whose Data is it Anyway?

- With the number of sensors being deployed, it isn't always clear whose data is being gathered.
  - Consider the case of a camera deployed in an advertising hoarding.
- 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.
- When convening to debate such issues in the summer of 2012, the participants at the Open Internet of Things Assembly coined the term data subjects—those people to whom the data pertains.
- There's no clear understanding of what rights, if any, such "data subjects" will enjoy, but it is an area that deserves more debate and attention.

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.
- It is good to bear the Postel's law in mind when designing or building anything which must interact with other services — particularly when you aren't the one building the other components with which your system interacts.

Small Pieces, Loosely Joined

- What this means for the architects of a service is that each piece should be designed to do one thing well and not rely too much on tight integration with the separate components it uses.

Scanned by CamScanner

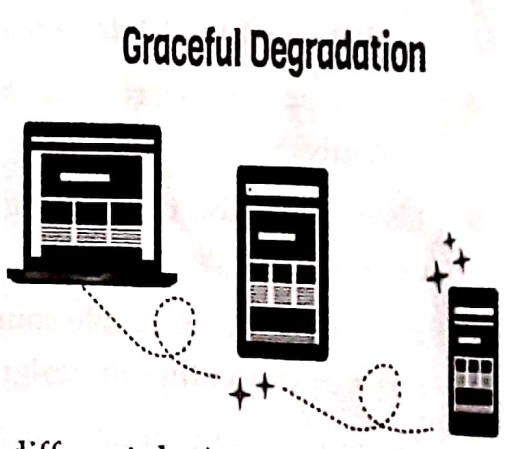
- Strive also to make the components more generalised.
  - That will help you, and others, to reuse and repurpose the components to build new capabilities.
- Where possible, use existing standards and protocols rather than inventing your own.
- Any loss of elegance or efficiency of code size or electronics will be outweighed by the availability of standard libraries and skills for people to interact with, and build on, your system.

First-Class Citizens on the Internet

- An extension of the concept of loose coupling is to strive to make your devices 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.
- In the few cases where the existing protocols don't work, such as in extremely low-powered sensors, a better solution is to work with your peers to amend existing standards or create new open standards which address the issue within the conventional standards groups.
- The evolution of the mobile web serves as a good cautionary example.

Graceful Degradation

- Because the Internet is so welcoming and tolerant of all sorts of devices and services, 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.



- The first is to acknowledge that the wealth of different devices is likely to be a problem and design your system to expect it.
- 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*,
- 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.

Scanned by CamScanner

- This capability can even span technologies and does so in its most common usage.
- Aside from using the same techniques when designing our connected devices, we might also be able to apply that approach to the devices themselves to give a degree of fault tolerance.
- This rapid increase of devices and the likelihood that some of them will break in some way means that it is important that their technology continues to add what value it can, as parts cease to function.
- When your early-adopter Internet Fridge can no longer talk to your WiFi because it's only IPv4 and the world has moved to IPv6, you would still be able to use its touchscreen to write messages and view the photos stored in the USB stick stuck in it. And if the touchscreen breaks, you should still be able to keep the food inside it cold.

AFFORDANCES

- Donald Norman defines affordances as follows:
  - *Affordances provide strong clues to the operations of things. Plates are for pushing. Knobs are for turning. Slots are for inserting things into. 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.*
- An important start is to keep the existing affordances of the object being enhanced.
- Users who don't realise that a device has any extra capabilities should still be able to use it as if it hasn't.
- Although this principle sounds like common sense, it is often discarded due to costs or difficulties in design.

  Example : Light dimmer in home automation system
- Things get trickier with interactions that are invisible – either because they use wireless communication or because they are invoked through learned gestures.

  Example: RFID tags
- Similar rules apply when designing physical interfaces. Don't overload familiar connectors with unfamiliar behaviours.

  Example: 3.5mm audio jacks

Scanned by CamScanner

QUESTIONS

1. Explain calm and ambient technology
2. What do you mean by magic as metaphor?
3. Who invent the Wheredial? Explain the working of Wheredial
4. How to keep or maintain privacy of IOT devices?
5. Explain Hashes in detail.
6. Whose Data Is It Anyway? Justify it.
7. Define the following :
   a) Small Pieces.
   b) Loosely Joined.
   c) First-Class Citizens On The Internet.
   d) Graceful degradation
8. What do you mean by affordances?

Scanned by CamScanner

3 Internet Principles

Internet Protocol (IP) :

- Data is sent from one machine to another in a packet, with a destination address and a source address in a standardised format (a "protocol").
- The identifier used in the IP layer of the TCP/IP protocol suite to identify each device connected to the Internet is called the Internet address or IP address.
- An IPv4 address is a 32-bit address that uniquely and universally defines the connection of a host or a router to the Internet; an IP address is the address of the interface.

Transmission Communication Protocol (TCP) :

- To confirm the delivery of message packet to authentic receiver we use TCP. In this the sender checks whether the receiver received the message and also whether the message is complete or there is some missing data.
- The simplest transport protocol on the Internet, TCP is built on top of the basic IP protocol and adds sequence numbers, acknowledgements, and retransmissions.
- This means that a message sent with TCP can be arbitrarily on and give the sender some assurance that it actually arrived at the destination intact.
- Combination of TCP and IP is so useful, many services are built on it in turn, such as email and the HTTP protocol that transmits information across the World Wide Web.

IP Protocol Suite(TCP/IP) :

- In the current networking system the combination of TCP/IP is observed everywhere and it is therefore simply referred as TCP/IP suite or stack of protocols layered on top of each other, each layer building on the capabilities of one below.
- The low-level protocols at the *link layer* manage the transfer of bits of information across a network link. This could be by an Ethernet cable,by WiFi,

Scanned by CamScanner

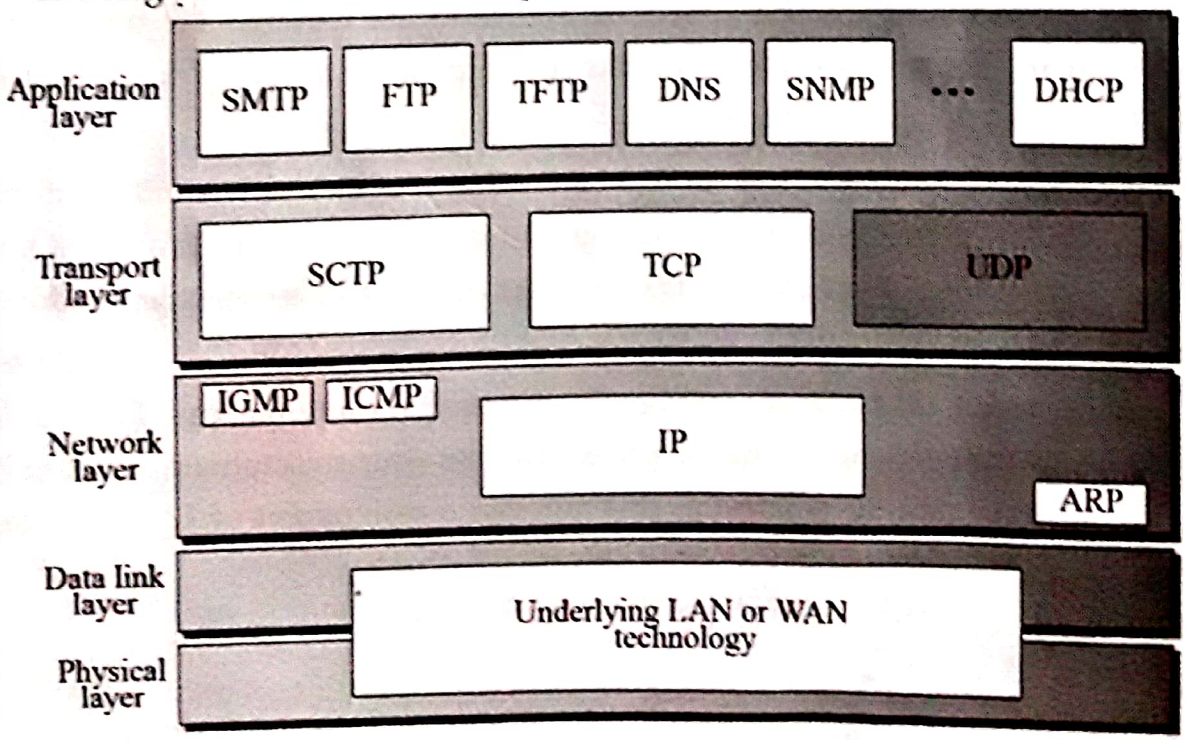
or across a telephone network, or even by short-range radio designed to carry data over the Personal Area Network (PAN).

- The *Internet layer* then sits on top of these various links and abstracts away the gory details in favour of a simple destination address.
- Then TCP, which lives in the *transport layer*, sits on top of IP and extends it with more sophisticated control of the messages passed.
- Finally, the *application layer* contains the protocols that deal with fetching web pages, sending emails, and Internet telephony. Of these, HTTP is the most ubiquitous for the web, and indeed for communication between Internet of Things devices.

User Datagram Protocol(UDP) :

- It is part of the TCP/IP suite of protocols used for data transferring. UDP is a known as a "stateless" protocol, meaning it doesn't acknowledge that the packets being sent have been received.
- These limitations make TCP preferable for many of the tasks that Internet of Things devices will be used for.
- Since there is no overhead, the UDP protocol is typically used for streaming media.

  Example : Voice over IP (VoIP)—computer-based telephony, such as Skype
- UDP is also the transport for some very important protocols which provide common, low-level functionality, such as DNS and DHCP, which relate to the discovery and resolution of devices on the network.
- The diagram below shows UDP position in TCP/IP suite



Scanned by CamScanner

IP Addresses :

- An Internet Protocol address (IP address) is a logical numeric address that is assigned to every single computer, printer, switch, router or any other device that is part of a TCP/IP-based network.
- The IP address is the core component on which the networking architecture is built; no network exists without it.
- An IP address is a logical address that is used to uniquely identify every node in the network. Because IP addresses are logical, they can change.
- IP address gives the network node an address so that it can communicate with other nodes or networks, just like mail is sent to friends and relatives.
- The numerals in an IP address are divided into 2 parts:
  i) The network part specifies which networks this address belongs "to" .
  ii) The host part further pinpoints the exact location.
- An IPv4 address is a 32-bit address that uniquely and universally defines the connection of a host or a router to the Internet
- Following are the types of IP addresses"
  i) **Private IP Address** : It is reserved for internal use behind a router or other Network Address Translation (NAT) device, apart from the public. Private IP addresses are in contrast to public IP addresses which are public and can *not* be used within a home or business network.
  ii) **Public IP Address** : It is an IP address that your home or business router receives from your ISP. Public IP addresses are required for any publicly accessible network hardware, like for your home router as well as for the servers that host websites.
  iii) **Static IP Address** : It is an IP address that was manually configured for a device, versus one that was assigned via a DHCP server. It's called *static* because it doesn't change. It's the exact opposite of a dynamic IP address, which *does* change.
  iv) **Dynamic IP Address** : It is an IP address that's automatically assigned to each connection, or *node*, of a network, like your smartphone, desktop PC, wireless tablet... whatever. This automatic assignment of IP addresses is done by what's called a DHCP server.

Scanned by CamScanner

- The table below shows the rules for IP addressing:

Rule	Purpose	Example
Host ID cannot be all binary 1s	This address represents a network broadcast	131.107.255.255
Host ID cannot be all binary 0s	This address identifies a network	131.107.0.0
Network ID cannot be all binary 0s	This address represents "on this network"	0.0.145.23
Network ID cannot be all binary 1s	The address represents " on all networks"	255.255.1.142
Network ID cannot be decimal 127	This address range is reserved for the loopback address	127.0.0.1
IP address cannot be all binary 0s	This address is used to represent the default route	0.0.0.0
IP address cannot be all binary 1s	This address is used to represent a broadcast	255.255.255.255
Network IDs of 224 and above in the first octet cannot be assigned to hosts	Class D addresses are reserved for multicasting, while Class E addresses represent an experimental range	224.0.0.1

Question : Find the error, if any, in the following IPv4 addresses:

a) 111.56.045.78

b) 221.34.7.8.20

c) 75.45.301.14

d) 11100010.23.14.67

Solution

a) There should be no leading zeroes (045).

b) We may not have more than 4 bytes in an IPv4 address.

c) Each byte should be less than or equal to 255.

d) A mixture of binary notation and dotted-decimal notation.

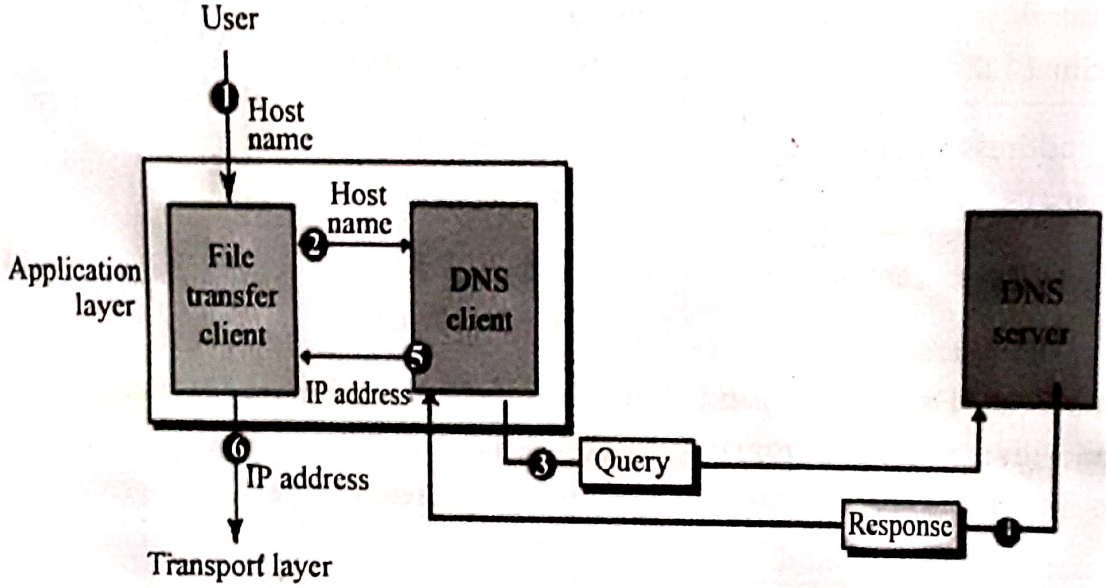
Scanned by CamScanner

Domain Name Server(DNS) :

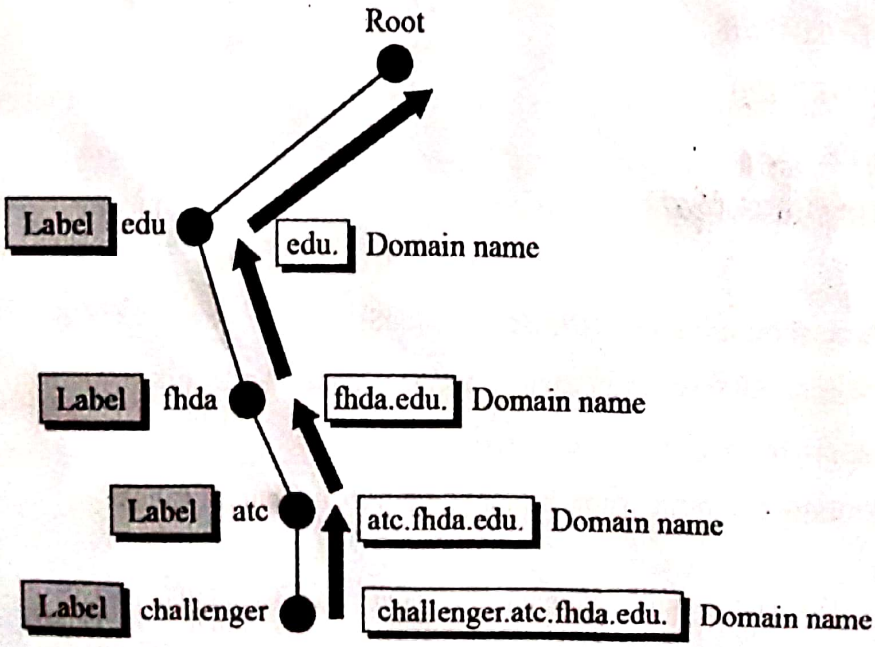
- The domain name system (DNS) is the way that internet domain names are located and translated into internet protocol (IP) addresses.
- The domain name system maps the name people use to locate a website to the IP address that a computer uses to locate a website. For example, if someone types TechTarget.com into a web browser, a server behind the scenes will map that name to the IP address 206.19.49.149

- **Need for DNS :**

To identify an entity, TCP/IP protocols use the IP address, which uniquely identifies the connection of a host to the Internet. However, people prefer to use names instead of numeric addresses. Therefore, we need a system that can map a name to an address or an address to a name.



- **Domain Names and Labels**

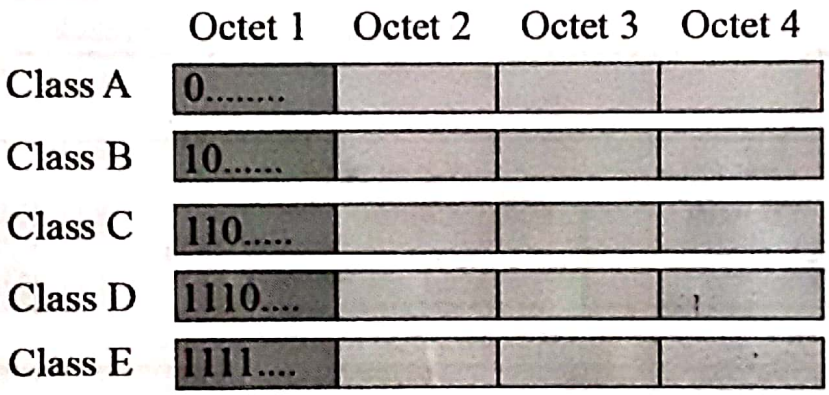


Scanned by CamScanner

Static IP Address Assignment :

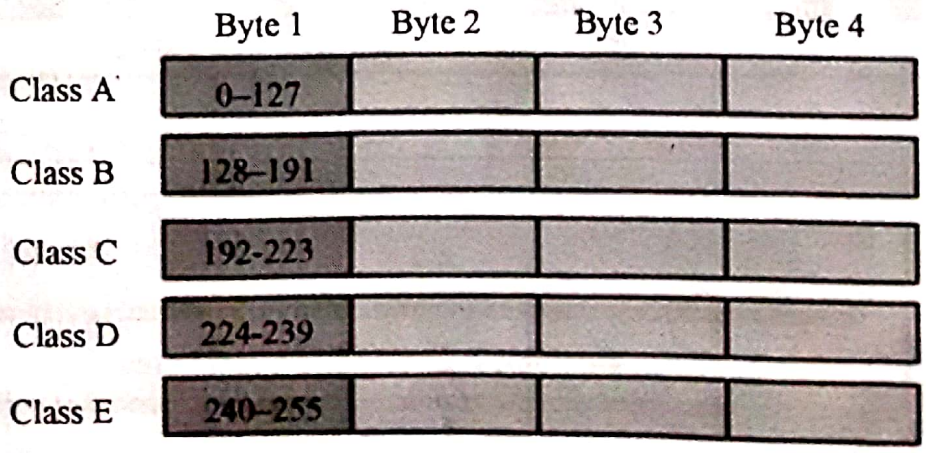
- IP addresses, when started a few decades ago, used the concept of classes. This architecture is called class full addressing. In the mid-1990s, a new architecture, called classless addressing, was introduced that supersedes the original architecture. In this section, we introduce class full addressing because it paves the way for understanding classless addressing and justifies the rationale for moving to the new architecture.
- In many cases, however, the system administrator simply assigns server numbers in order. The administrator makes a note of the addresses and updates DNS records and so on to point to these addresses. We call this kind of address *static* because once assigned it won't change again without human intervention.
- Starting Address of the different classes :

i) In Binary Notation



Binary notation

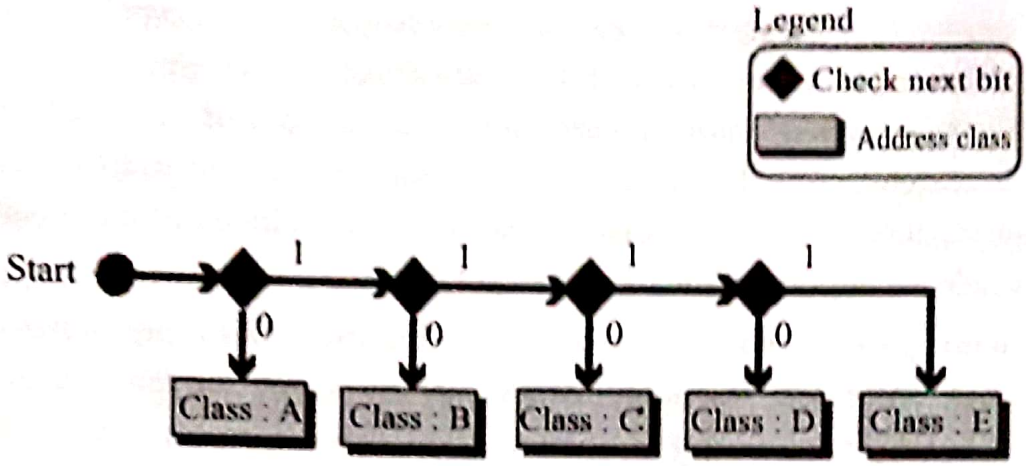
ii) In Decimal Notation



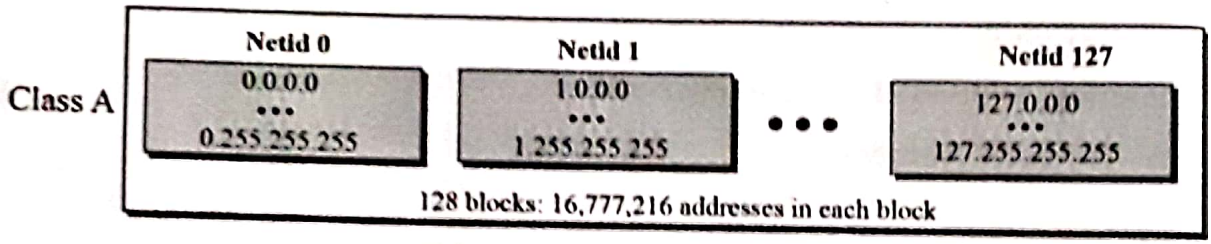
Dotted-decimal notation

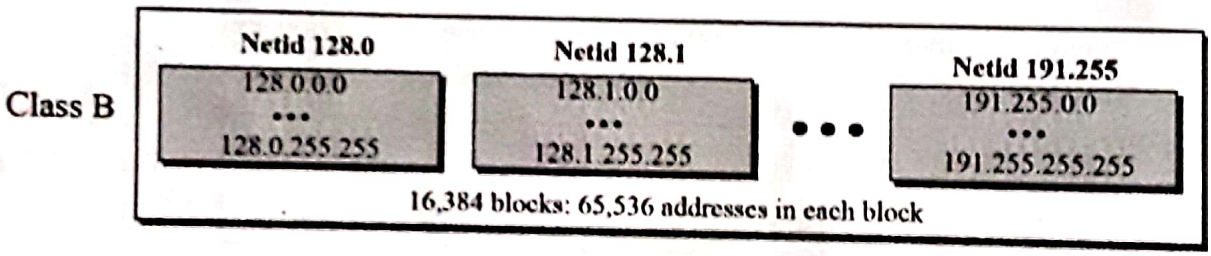
Scanned by CamScanner

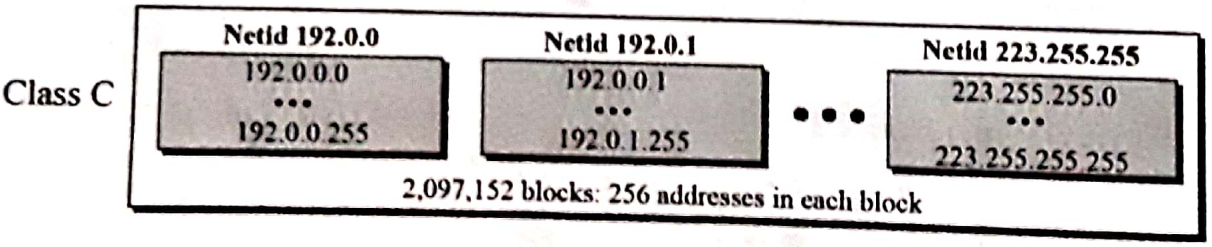
- Finding the address using continuous checking:

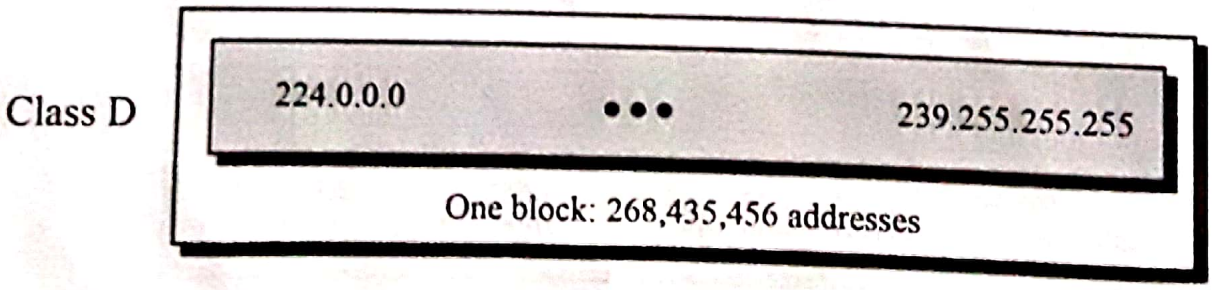


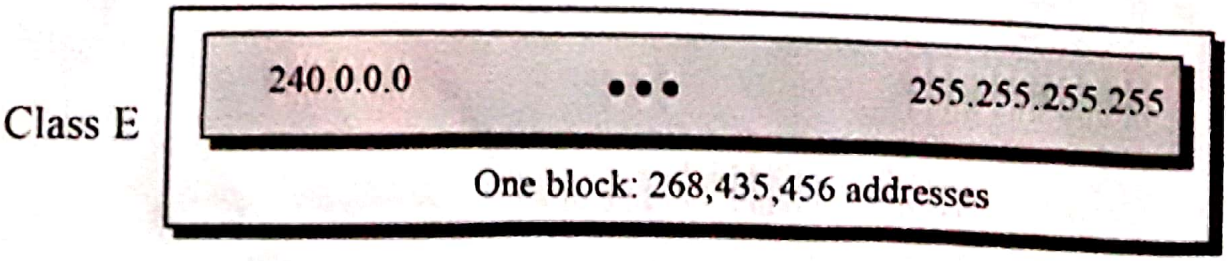
- Address range of Classes:











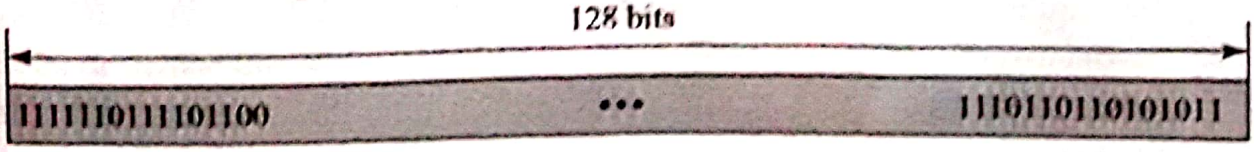
Scanned by CamScanner

Dynamic IP Address Assignment:

- A *dynamic IP address* is an *IP address* that is assigned automatically by the system to a device, account or user when it is connected to the network; that is, it is assigned as needed rather than in advance.
- Dynamic IP addresses are assigned by the *dynamic host configuration protocol* (DHCP), which is one of the key protocols in the TCP/IP protocol suite.
- Dynamic IP addresses contrast with *static IP addresses*, which are assigned manually and semi-permanently to a device, account or user. With dynamic addressing, a computer, account, etc. will typically have a different IP address every time it connects to the network. In some systems, the device's IP address can change even while it is still connected to the network.
- The main advantage of dynamically assigning IP addresses is that it allows them to be reused, thereby greatly increasing the total number of computers and other devices that can use the Internet or other network.
- Another advantage is enhanced security for individual users because their IP address is different every time they log into the network.
- Still another benefit is simplification of network administration because the software keeps track of IP addresses and thus relieves the administrator from the very tedious task of having to manually assign a unique IP address to every computer as it enters the network.
- Even the simplest computing devices such as the Arduino board can use DHCP.
- Although the Arduino's Ethernet library allows you to configure a static IP address, you can also request one via DHCP.
- Using a static address may be fine for development (if you are the only person connected to it with that address), but for working in groups or preparing a device to be distributed to other people on arbitrary networks, you almost certainly want a dynamic IP address.

IPv6

- An IPv6 address is 128 bits or 16 bytes (octet) long as shown in figure below. The address length in IPv6 is four times of the length address in IPv4

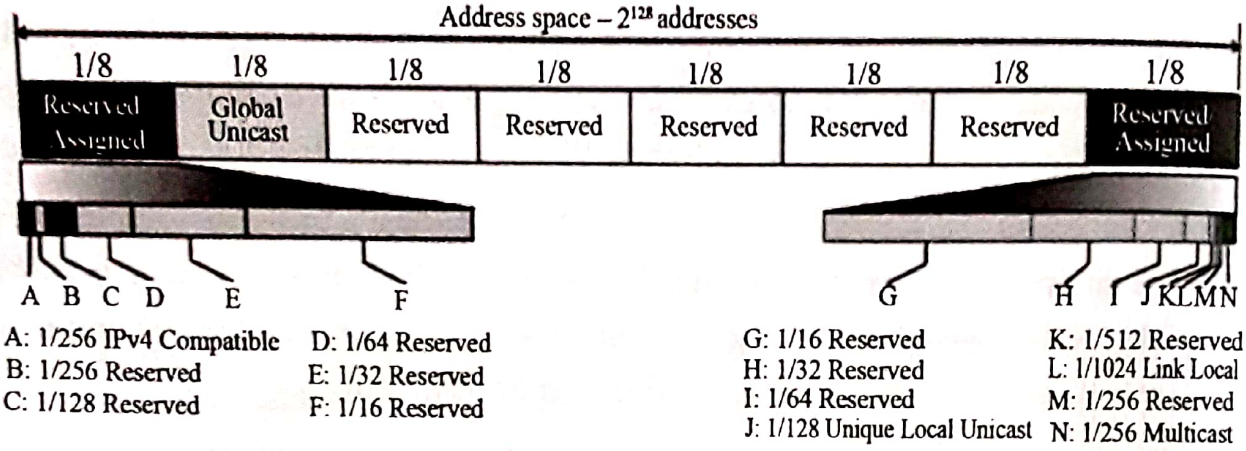


- Colon Hexadecimal Notation :

  FDEC : BA98 : 7654 : 3210 : ADBF : BBFF : 2922 : FFFF

Scanned by CamScanner

- Like the address space of IPv4, the address space of IPv6 is divided into several blocks of varying size and each block is allocated for special purpose. Most of the blocks are still unassigned and have been left aside for future use.
- To better understand the allocation and the location of each block in address space, we first divide the whole address space into eight equal ranges. This division does not show the block allocation, but we believe it shows where each actual block is located.



A: 1/256 IPv4 Compatible
B: 1/256 Reserved
C: 1/128 Reserved
D: 1/64 Reserved
E: 1/32 Reserved
F: 1/16 Reserved
G: 1/16 Reserved
H: 1/32 Reserved
I: 1/64 Reserved
J: 1/128 Unique Local Unicast
K: 1/512 Reserved
L: 1/1024 Link Local
M: 1/256 Reserved
N: 1/256 Multicast

IPv6 and Powering Devices

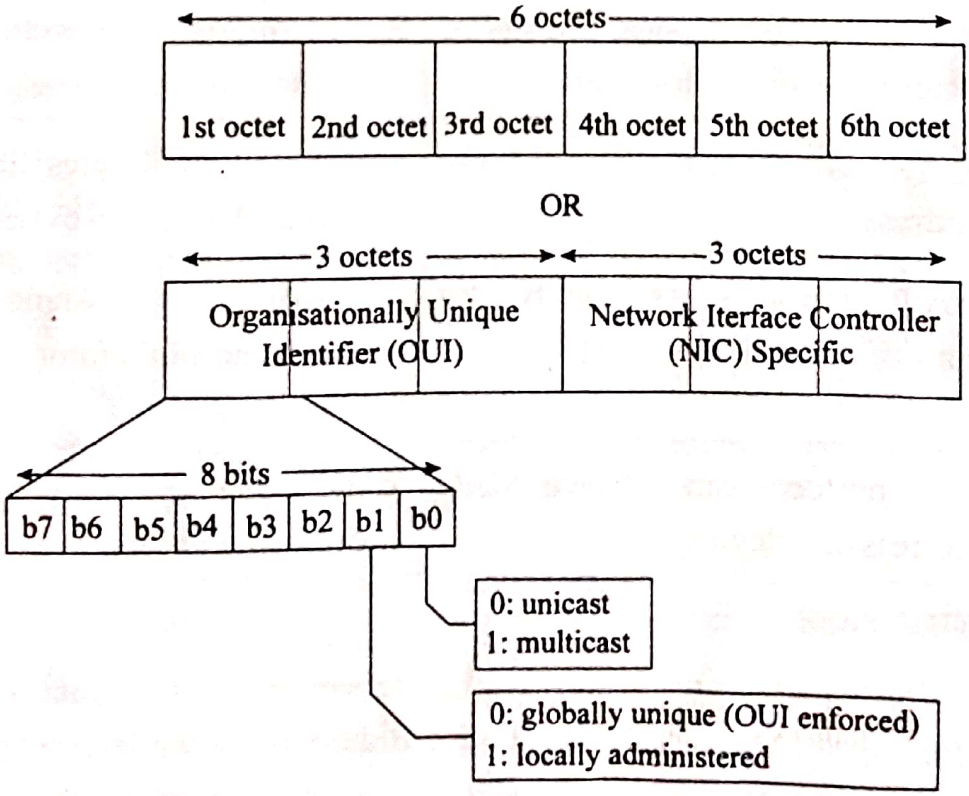
- It was originally expected that mobile phones connected to the Internet (another huge growth area) would push this technology over the tipping point. In fact, mobile networks are increasingly using IPv6 internally to route traffic.
- Although this infrastructure is still invisible to the end user, it does mean that there is already a lot of use below the surface which is stacked up, waiting for a tipping point.
- But as we are moving to various of product that can be made in IOT one of the important aspects that needs to be taken into consideration is power consumption of the end product.
- We know that we can regularly charge and maintain a small handful of devices. At any one moment, we might have a laptop, a tablet, a phone, a camera, and a music player plugged in to charge.
- The constant juggling of power sockets, chargers, and cables is feasible but fiddly.
- The requirements for large numbers of devices, however, are very different. The devices should be low power and very reliable, while still being capable of connecting to the Internet. But to accomplish this, these devices will team together in a mesh network.

Scanned by CamScanner

Media Access Control (MAC) ADDRESSES :

- A media access control address (MAC address) of a device is a unique identifier assigned to a network interface controller (NIC) for communications at the data link layer of a network segment.
- MAC addresses are used as a network address for most IEEE 802 network technologies, including Ethernet and Wi-Fi. In this context, MAC addresses are used in the medium access control protocol sublayer.
- Apart from having IP address, every network-connected device also has a MAC address, which is like the final address on a physical envelope in our analogy.
- It is used to differentiate different machines on the same physical network so that they can exchange packets. This relates to the lowest-level "link layer" of the TCP/IP stack.
- Though MAC addresses are globally unique, they don't typically get used outside of one Ethernet network (for example, beyond your home router). So, when an IP message is routed, it hops from node to node, and when it finally reaches a node which knows where the *physical* machine is, that node passes the message to the device associated with that MAC address.
- MAC stands for *Media Access Control*. It is a 48-bit number, usually written as six groups of hexadecimal digits, separated by colons—for example:

18 : 25 : 49 : 77 : 59 : BC



Scanned by CamScanner

- Most devices, such as laptop, come with the MAC address burned into their Ethernet chips.
- Some chips, such as the Arduino Ethernet's WizNet, don't have a hard-coded MAC address, though. This is for production reasons: if the chips are mass produced, they are, of course, *identical*. So they can't, physically, contain a distinctive address. The address could be stored in the chip's firmware, but this would then require every chip to be built with custom code compiled in the firmware.
- Most consumer devices use some similar process to ensure that the machine always starts up with the same unique MAC address.
- The Arduino board, as a low-cost prototyping platform for developers, doesn't bother with that nicety, to save time and cost. Yet it does come with a sticker with a MAC address printed on it. Although this might seem a bit odd, there is a good reason for it: that MAC address is reserved and therefore is guaranteed unique if you want to use it.
- For development purposes, you can simply choose a MAC address that is known not to exist in your network.

Comparison between MAC Address and IP Address:

Sr. No	MAC Address	IP Address
1.	Media Access Control Address.	Internet Protocol Address
2.	It identifies the physical address of a computer on the internet	It identifies connection of a computer on the internet.
3.	It is 48 bits (6 bytes) hexadecimal address.	IPv4 is a 32-bit (4 bytes) address, and IPv6 is a 128-bits (16 bytes) address.
4.	MAC address is assigned by the manufacturer of NIC card	IP address is assigned by the network administrator or Internet Service Provider.
5	ARP protocol can retrieve MAC address of a device.	RARP protocol can retrieve IP address of a device

TCP and UDP Ports

- Both TCP and UDP are protocols used for sending bits of data — known as packets — over the Internet. They both build on top of the Internet protocol.
- In other words, whether you are sending a packet via TCP or UDP, that packet is sent to an IP address. These packets are treated similarly, as they are

Scanned by CamScanner

forwarded from your computer to intermediary routers and on to the destination.

- TCP and UDP are not the only protocols that work on top of IP. However, they are the most widely used port. The widely used term "TCP/IP" refers to TCP over IP. UDP over IP could just as well be referred to as "UDP/IP", although this is not a common term
- **So what's the difference between these two port types?**
  - UDP is a connectionless protocol that runs on top of IP (UDP/IP), and TCP is a connection-oriented protocol that runs on top of IP (TCP/IP). *Connectionless* means that a host can send a message to another host without first establishing a connection with the recipient. The host simply puts a message onto the network with a destination address and hopes that the message arrives. In addition, the transmission or receipt of a UDP packet doesn't guarantee any further communication in either direction. Because a UDP packet doesn't require an existing connection, network systems use UDP primarily for broadcasting messages (i.e., a one-to-many sending, much like unsolicited junk email). The most common UDP packets—DNS registrations and name-resolution queries—are sent to port 53.
  - In contrast, TCP ports support only connection-oriented protocols. A connection-oriented protocol requires that network endpoints establish a channel between them before they transmit messages. Whereas incoming UDP traffic indicates an unsolicited message that doesn't necessarily require a response, incoming TCP messages arrive because someone is trying to establish a point-to-point connection with your system and, in turn, is expecting a response.

An Example: HTTP Ports

- Whenever there is a request from the browser for an HTTP page, the request is the usually send to port 80 as web server is responds to this port.
- However if the message is send to some other ports, then following conditions could encountered
  - There is no port to respond to the request hence machine will with an "RST" packet(a control sequence resetting the TCP/IP connection) to complain about the same.
  - There is no port to respond to the request, but the firewall lets the request simply hang instead of replying. The purpose of this (lack of) response is to

Scanned by CamScanner

discourage attackers from trying to find information about the machine by scanning every port.

- The port/client may decide that replying to the port is a bad idea hence refuses to do it.

  Eg. Google Chrome does this for a fairly arbitrary list of "restricted ports"

- The message arrives at a port that is expecting something other than an HTTP message. The server reads the client's response, decides that it is garbage, and then terminates the connection (or, worse, does a nonsensical operation based on the message).

- Ports 0–1023 are "well-known ports", and only a system process or an administrator can connect to them.
- Ports 1024–49151 are "registered", so that common applications can have a usual port number. However, most services are able to bind any port number in this range.
- The Internet Assigned Numbers Authority (IANA) is responsible for registering the numbers in these ranges. People can and do abuse them, especially in the range 1024–49151, but unless you know what you're doing, you are better off using either the correct assigned port or (for an entirely custom application) a port above 49151.

OTHER COMMON PORTS

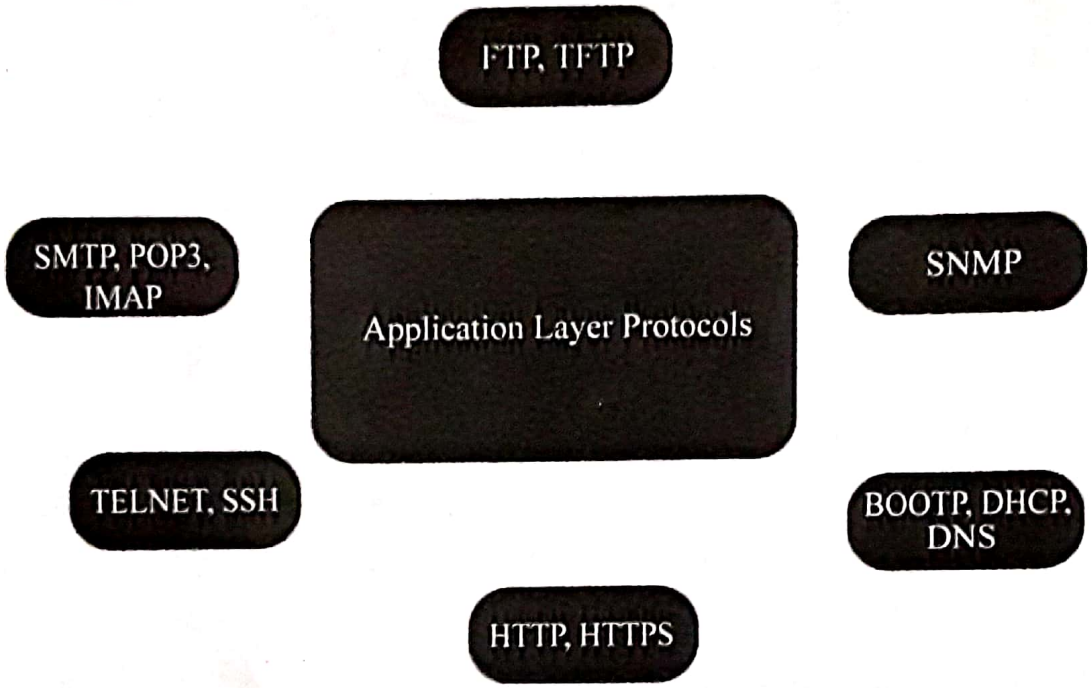
- 80 HTTP
- 8080 HTTP (for testing servers)
- 443 HTTPS
- 22 SSH (Secure Shell)
- 23 Telnet
- 25 SMTP (outbound email)
- 110 POP3 (inbound email)
- 220 IMAP (inbound email)

Application Layer Protocols

- An application layer is an abstraction layer that specifies the shared communications protocols and interface methods used by hosts in a communications network.

Scanned by CamScanner

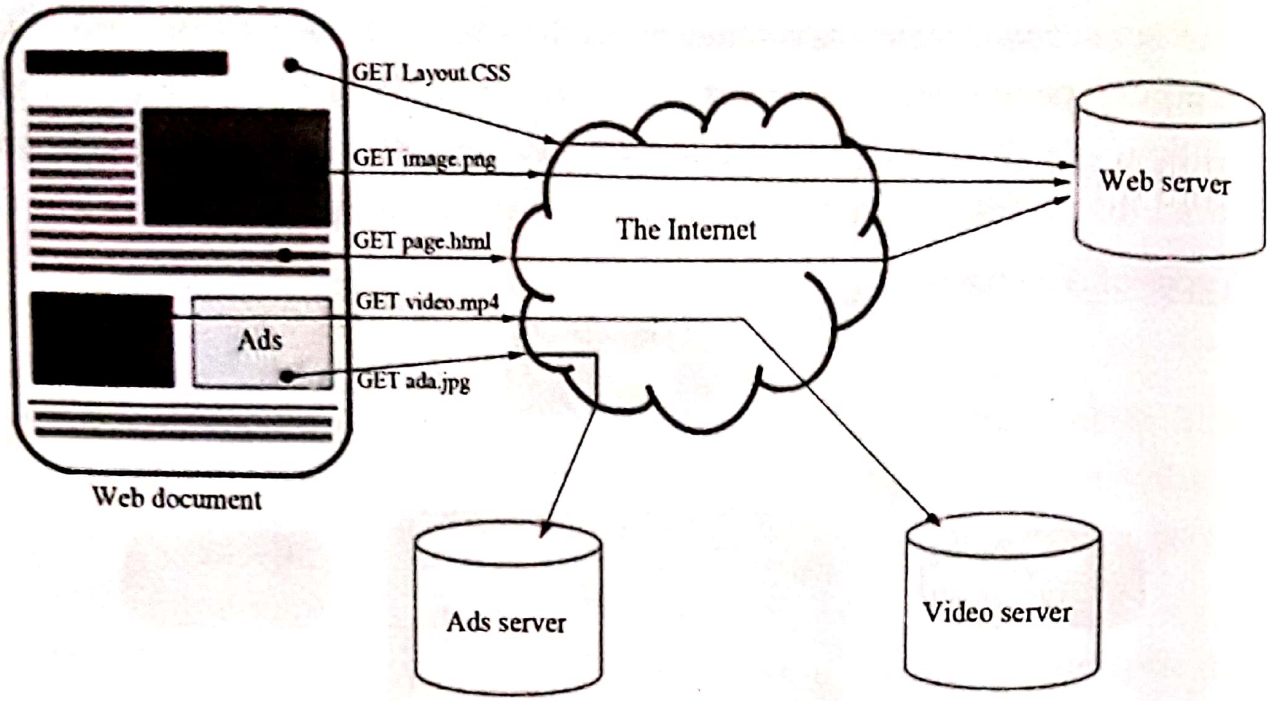
- The application layer abstraction is used in both of the standard models of computer networking : the Internet Protocol Suite (TCP/IP) and the OSI model. Although both models use the same term for their respective highest level layer, the detailed definitions and purposes are different.
- Types of Application Layer Protocol



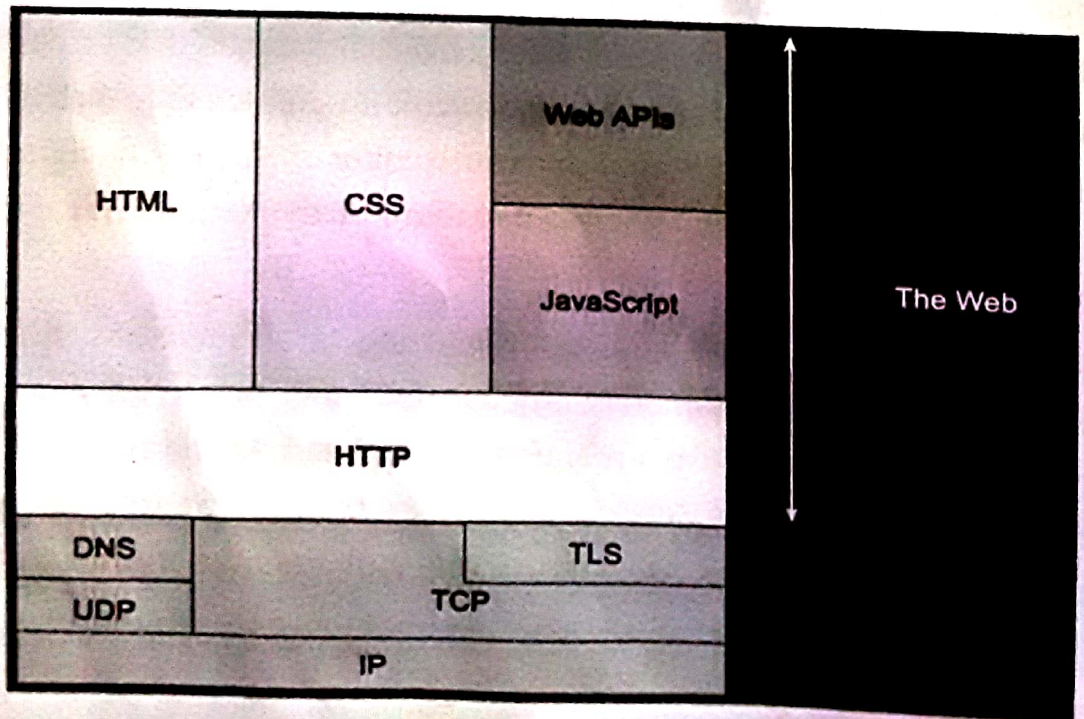
Hyper Text Transfer Protocol (HTTP) :

- HTTP is the underlying protocol used by the World Wide Web and this protocol defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands.
- HTTP is called a *stateless* protocol because each command is executed independently, without any knowledge of the commands that came before it. This is the main reason that it is difficult to implement Web sites that react intelligently to user input. This shortcoming of HTTP is being addressed in a number of new technologies, including ActiveX, Java, JavaScript and cookies.
- **HTTP** is a protocol which allows the fetching of resources, such as HTML documents. It is the foundation of any data exchange on the Web and a client-server protocol, which means requests are initiated by the recipient, usually the Web browser. A complete document is reconstructed from the different sub-documents fetched, for instance text,layout description, images, videos, scripts, and more.

Scanned by CamScanner



- Clients and servers communicate by exchanging individual messages (as opposed to a stream of data). The messages sent by the client, usually a Web browser, are called *requests* and the messages sent by the server as an answer are called *responses*.
- HTTP is an extensible protocol which has evolved over time. It is an application layer protocol that is sent over TCP, or over a TLS-encrypted TCP connection, though any reliable transport protocol could theoretically be used. Due to its extensibility, it is used to not only fetch hypertext documents, but also images and videos or to post content to servers, like with HTML form results. HTTP can also be used to fetch parts of documents to update Web pages on demand.



Scanned by CamScanner

HTTP FLOW :

When the client wants to communicate with a server, either being the final server or an intermediate proxy, it performs the following steps :

1. Open a TCP connection : The TCP connection will be used to send a request, or several, and receive an answer. The client may open a new connection, reuse an existing connection, or open several TCP connections to the servers.
2. Send an HTTP message : HTTP messages (before HTTP/2) are human-readable. With HTTP/2, these simple messages are encapsulated in frames, making them impossible to read directly, but the principle remains the same.

```
GET / HTTP/1.1
Host: developer.mozilla.org
Accept-Language: fr
```

3. Read the response sent by the server:

```
HTTP/1.1 200 OK
Date: Sat, 09 Oct 2010 14:28:02 GMT
Server: Apache
Last-Modified: Tue, 01 Dec 2009 20:18:22 GMT
ETag: "51142bc1-7449-479b075b2891b"
Accept-Ranges: bytes
Content-Length: 29769
Content-Type: text/html

<!DOCTYPE html... (here comes the 29769 bytes of the requested w
```

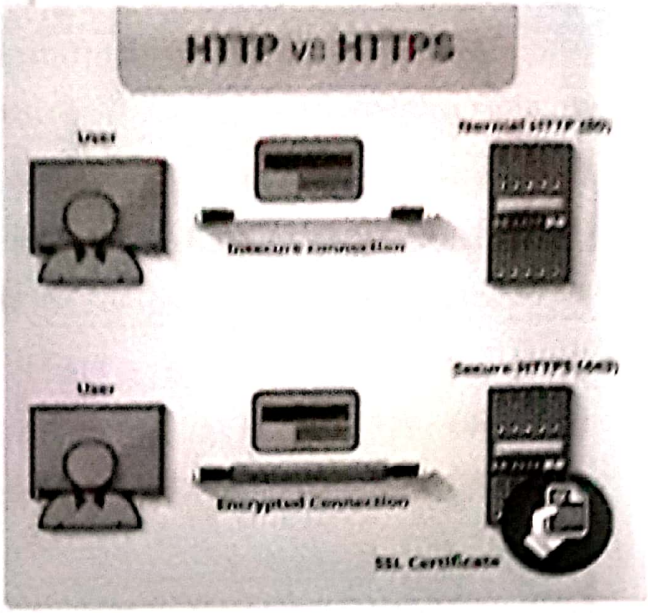
4. Close or reuse the connection for further requests.

HTTPS: Encrypted HTTP

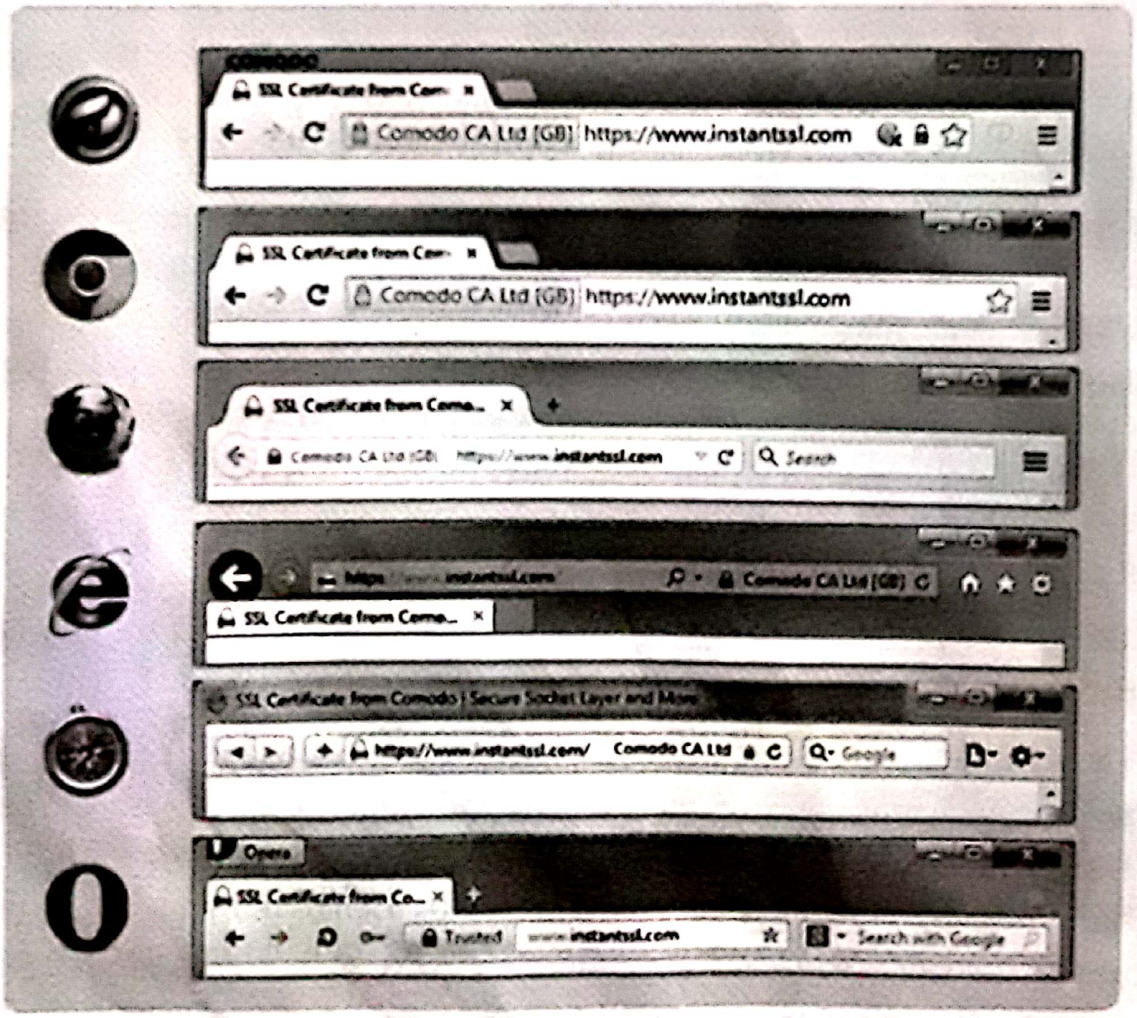
- Hyper Text Transfer Protocol Secure (HTTPS) is the secure version of HTTP, the protocol over which data is sent between your browser and the website that you are connected to.
- The 'S' at the end of HTTPS stands for 'Secure'. It means all communications between your browser and the website are encrypted.
- The principal motivation for HTTPS is authentication of the accessed website and protection of the privacy and integrity of the exchanged data while in transit. It protects against man-in-the-middle attacks. The bidirectional encryption of communications between a client and server protects against eavesdropping and tampering of the communication. In practice, this provides a reasonable assurance that one is communicating without interference by attackers with the website that one intended to communicate with, as opposed to an impostor.

Scanned by CamScanner

- HTTPS is often used to protect highly confidential online transactions like online banking and online shopping order forms.



- When you request a HTTPS connection to a webpage, the website will initially send its SSL certificate to your browser. This certificate contains the public key needed to begin the secure session. Based on this initial exchange, your browser and the website then initiate the 'SSL handshake'. The SSL handshake involves the generation of shared secrets to establish a uniquely secure connection between yourself and the website.
- When a trusted SSL Digital Certificate is used during a HTTPS connection, users will see a padlock icon in the browser address bar. When an Extended Validation Certificate is installed on a web site, the address bar will turn green.



Scanned by CamScanner

- The major benefits of a HTTPS certificate are :
  - Customer information, like credit card numbers, is encrypted and cannot be intercepted
  - Visitors can verify you are a registered business and that you own the domain
  - Customers are more likely to trust and complete purchases from sites that use HTTPS.

Other Application Layer Protocols:

- All protocols work in a roughly similar way. Some cases involve more than just a two-way request and response. For example, when sending email using SMTP, you first need to do the "HELO handshake" where the client introduces itself with a cheery "hello" (SMTP commands are all four letters long, so it actually says "HELO") and receives a response like "250 Hello example.org pleased to meet you!"
- In all cases, it is worth spending a little time researching the protocol on Google and Wikipedia to understand in overview how it works.
- You can usually find a library that abstracts the details of the communication process, and we recommend using that wherever possible.
- Bad implementations of network protocols will create problems for you and the servers you connect to and may result in bugs or your clients getting banned from useful services.
- So, it is generally better to use a well-written, well-debugged implementation that is used by many other developers. In general, the only valid reasons for you, the programmer, to ever speak to any application layer protocol directly (that is, without using a library) are
  - There is no implementation of the protocol for your platform (or the implementation is inefficient, incomplete, or broken).
  - You want to try implementing it from scratch, for fun.
  - You are testing, or learning, and want to make a particular request easily.

QUESTIONS

1. Explain internet Communication and list its different kinds of protocols
2. Give the brief introduction about Internet Protocol (IP), TCP.

Scanned by CamScanner

3. Define the following term with respect to IOT:
   a) IP addresses.
   b) DNS.
   c) Static IP address assignment.
   d) Dynamic IP address assignment.
4. Explain TCP/IP protocol suite with diagram.
5. Explain the role of UDP and MAC Address in IOT.
6. What is the use of IPv6 in IOT?
7. What do you mean by HTTP ports? List out other common ports.
8. List and Explain Protocol presented at application layer.

Scanned by CamScanner

UNIT – II

4

Thinking About Prototyping

Prototyping is the most important process of product development. The core of prototyping is to test the product and usually used for demonstration, testing, communication and so on

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.

Scanned by CamScanner

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.

FAMILIARITY

Another option to consider is familiarity. 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. The same applies to the server software, obviously.

COSTS VERSUS EASE OF PROTOTYPING

It is 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 based on performance/capabilities.

For the first prototype, the cost is probably not the most important issue: the smartphone or computer options are particularly convenient if you already have one available, at which point they are effectively zero-cost.

Of course, if your device has physical interactions (blowing bubbles, turning a clock's hands, taking input from a dial), you will find that a PC is not optimized for this kind of work. 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.

For many beginners to hardware development, the Arduino toolkit is a surprisingly good choice. Yes, the input/output choices are basic and require an ability to follow wiring diagrams and, ideally, a basic knowledge of electronics.

Scanned by CamScanner

If you already know how to develop an application in C#, in Python, or in JavaScript, you have a great place to start. But if you don't know, you first have to evaluate and choose a language and then work out how to write it, get it going, and make it start automatically.

Another option is to marry the capabilities of a microcontroller to connect to low-level components such as dials, LEDs, and motors while running the hard processing on a computer or phone. A kit such as an Arduino easily connects to a computer via USB, and you can speak to it via the serial port in a standard way in any programming language.

PROTOTYPES AND PRODUCTION

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.

Of course, if you've used a constrained platform in prototyping, you may find that you have to make choices and limitations in your code. Dynamic memory allocation on the 2K that the Arduino provides may not be especially efficient, so how should that make you think about using strings or complex data structures? 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. But will the new platform have the same I/O capabilities? And you have to consider the ramping-up time to learn new technologies and languages. In practice, you will often find that you don't need to change platforms.

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.

Climbing into the Cloud

The server software is the easiest component to take from prototype into production. As we saw earlier, it might involve switching from a basic web framework to something more involved but you will be able to find an equivalent

Scanned by CamScanner

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.

OPEN SOURCE VERSUS CLOSED SOURCE

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

Why Closed?

- If you declared copyright on some source code or a design, someone who wants to market the same project cannot do so by simply reading your instructions and following them
- You might also be able to protect distinctive elements of the visual design with trademarks and of the software and hardware with patents
- Starting a project as closed source doesn't prevent you from later releasing it as open source
- 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

Why Open?

- 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.

Scanned by CamScanner

- 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

Disadvantages of Open Source

- Some open source applications may be tricky to set up and use. Others may lack user-friendly interfaces or features that your staff may be familiar with.
- There may also be less support available for when things go wrong – open source software tends to rely on its community of users to respond to and fix problems.
- "but people will steal my idea!" In general, if you talk to people about an idea, it's hard enough to get them to listen because they are waiting to tell you about *their* great idea.

Open Source as a Competitive Advantage

- Using open source work is often a no-risk way of getting software that has been tested, improved, and debugged by many eyes
- Commercial equivalents may be available for all these examples, but then you have to factor in the cost and rely on a single company's support forums instead of all the information available on the Internet.
- Using open source aggressively gives your product the chance to gain mindshare

Mixing Open and Closed Source

- Software companies are taking a "best of both worlds" approach by creating products that use a combination of OS and proprietary software code.
- Open sourcing many of libraries and keeping core business closed

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.

Scanned by CamScanner

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?

Scanned by CamScanner

5

Prototyping Embedded Devices

In order to create an IoT product, we should study electronics because whatever platform we choose, the rest of the circuitry that we will build to connect it to will be pretty much the same.

ELECTRONICS

Most of the prototyping can be done on what are called *solderless breadboards*. They enable you to build components together into a circuit with just a push-fit connection,

When it comes to thinking about the electronics, it's useful to split them into two main categories:

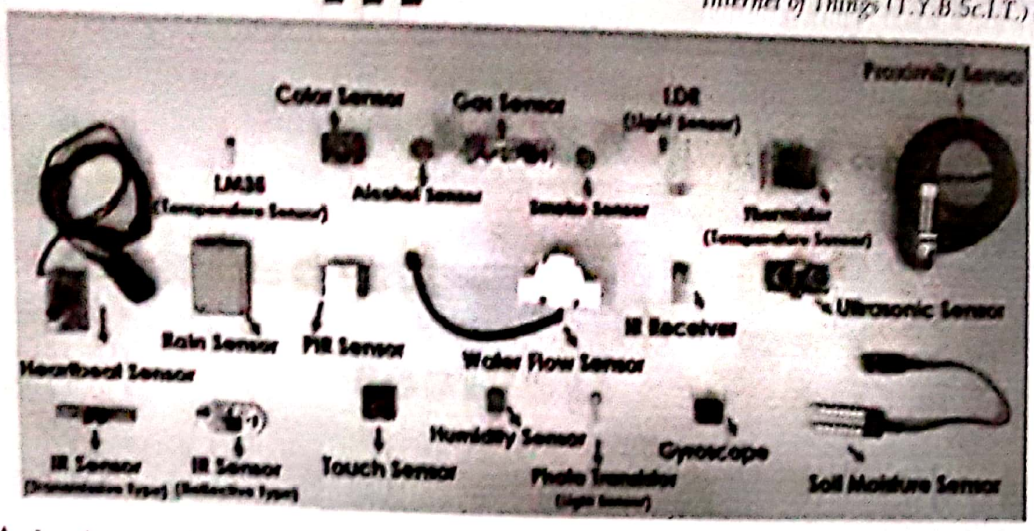
- **Sensors :** A sensor is a device that detects and responds to some type of input from the physical environment. The specific input could be light, heat, motion, moisture, pressure etc.

Pushbuttons and switches, which are probably the simplest sensors, allow some user input. Potentiometers (both rotary and linear) and rotaryencoders 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.

Scanned by CamScanner



- **Actuators :** An **actuator** is a machine or part of a machine which moves or controls another part in response to an input—the motors, lights, and so on.

One of the simplest and yet most useful actuators is light, because it is easy to create electronically and gives an obvious output. 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.

More complicated again are motors. Stepper motors can be moved in *steps*, as the name implies. Usually, a fixed number of steps perform a full rotation. DC motors simply move at a given speed when told to. Both types of motor can be one-directional or move in both directions. 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 (whereas steppers and DC motors turn indefinitely).

For all the kinds of motors that we've mentioned, you typically want to connect the motors to gears to alter the range of motion or convert circular movement to linear, and so on.

- **Analog to Digital converter :** It is a device that converts analog signals (usually voltage) obtained from environmental (physical) phenomena into digital format. It can be expressed as A/D or A-to-D or A-D or ADC.

- **Digital to analog convertor :** It is an electronics device in form of IC, which converts digital signal to its equivalent analog signal. It can be expressed as DAC, D/A, D2A, or D-to-A

- **Serial Peripheral Interface (SPI) :** It is a synchronous serial communication interface specification used for short distance communication. SPI devices communicate in full duplex mode using a master-slave architecture with a single

Scanned by CamScanner

master. The master device originates the frame for reading and writing. Multiple slave devices are supported through selection with individual slave select (SS) lines.

- **I²C (Inter-Integrated Circuit)** : pronounced *I-squared-C*, is a synchronous, multi-master, multi-slave bus. It is widely used for attaching lower-speed peripheral ICs to processors and microcontrollers in short-distance, intra-board communication.

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. When you're happy with how things are wired up, it's common to solder the components onto some protoboard, which may be sufficient to make the circuit more permanent and prevent wires from going astray.

Moving beyond the protoboard option tends to involve learning how to lay out a PCB.

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.

EMBEDDED COMPUTING BASICS

MICROCONTROLLERS

A microcontroller is a single chip microcomputer made through VLSI fabrication. A microcontroller also called an embedded controller because the microcontroller and its support circuits are often built into, or embedded in, the devices they control. A microcontroller is available in different word lengths like microprocessors (4bit,8bit,16bit,32bit,64bit and 128-bit microcontrollers are available today).

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 and include RAM measured in megabytes

Scanned by CamScanner

rather than kilobytes. Storage for SoC modules tends not to be included on the chip, with SD cards being a popular solution.

The greater capabilities of SoC mean that they need some sort of operating system to marshal their resources. A wide selection of embedded operating systems, both closed and open source, is available and from both specialised embedded providers and the big OS players, such as Microsoft and Linux.

CHOOSING YOUR PLATFORM

Choosing the *right* platform for Internet of Things device is based on following parameters.

- **Processor Speed :** The processor speed decides how fast it can process the individual instructions. A faster processor speed means that it can execute instructions more quickly. Microcontrollers tend to be clocked at speeds in the tens of MHz, whereas SoCs run at hundreds of MHz or possibly low GHz.
- **RAM :** The overall performance of system is determined by how well processor and RAM work together. RAM provides the working memory for the system. More RAM means more flexibility. The amount of RAM needed is vary from project to project.
- **Networking :** Wired Ethernet is often the simplest for the user—generally plug and play—and cheapest, but it requires a physical cable. Wireless solutions required 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.

ZigBee is one technology, aimed particularly at sensor networks and scenarios such as home automation. The recent Bluetooth LE protocol (also known as Bluetooth 4.0) has a very low power-consumption profile similar to ZigBee's and could see more rapid adoption due to its inclusion into standard Bluetooth chips included in phones and laptops. SMS can be used for low-bandwidth, higher-latency communication; for higher data rates, you will use the same data connections, like 3G, as a smartphone.

- **USB :** USB can provide both power and networking. Some of the microcontrollers have inbuilt 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—devices such as phones, WiFi dongles. Devices such as WiFi dongles often depend on additional software on the host system, such as networking stacks.

Scanned by CamScanner

■ **Power Consumption** : Faster processors, more power consumption. For portable devices rely on an unconventional power supply (batteries, solar power) depending on where they are installed, power consumption may be an issue.

However, processors may have a minimal power-consumption sleep mode. In this mode, power consumption is less. Therefore, a more powerful processor may *not* be a disadvantage even in a low-power embedded device.

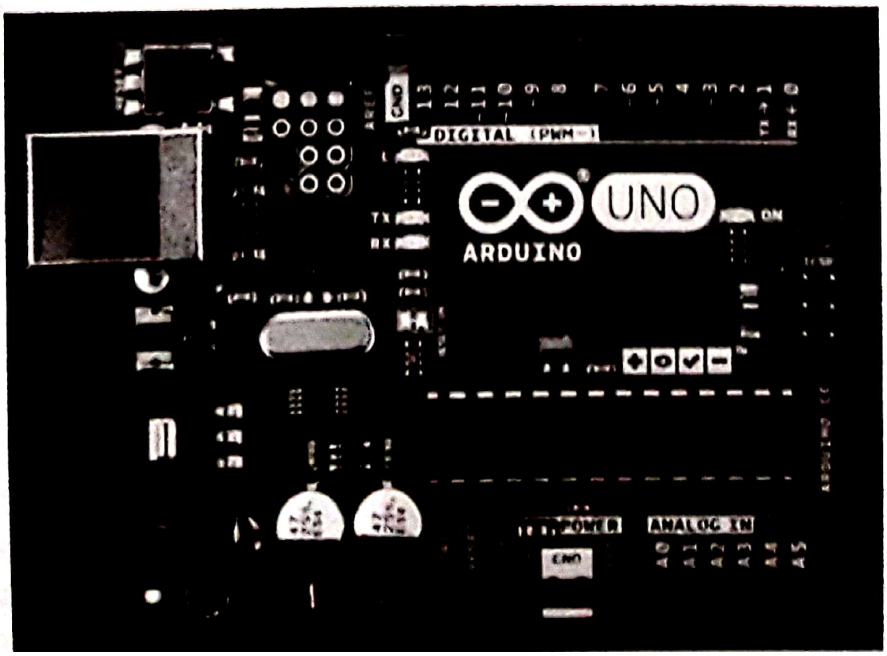
■ **Interfacing with Sensors and Other Circuitry** : Devices needs to interact with—either sensors to gather data about its environment; or motors, LEDs, screens, and so on, to provide output. These devices could connect to the circuitry through some sort of peripheral bus—SPI and I2C being common ones. The circuitry may involve ADC or DAC modules to read or write varying voltages. Generic GPIO pins, which provide digital on/off inputs or outputs..

■ **Physical Size and Form Factor** : Nowadays, the size is governed by the number of connections with surrounding components on the PCB. With surface-mount technology, the size is reduced to the great extent because it doesn't require holes to be drilled in the board for connections.

The limit to the size that each connection can be reduced to is then governed by the capabilities and tolerances of your manufacturing process. Some surface-mount designs are big enough for home-etched PCBs and can be hand-soldered. Others require professionally produced PCBs and accurate pick-and-place machines to locate them correctly.

Due to these trade-offs in size versus manufacturing complexity, many chip designs are available in a number of different form factors, known as *packages*. This lets the circuit designer choose the form that best suits his particular application.

ARDUINO



Scanned by CamScanner

Arduino is an open-source electronics platform based on easy-to-use hardware and software. These days the Arduino project covers a number of microcontroller boards, but its birth was in Ivrea in Northern Italy in 2005. A group from the Interaction Design Institute Ivrea (IDII) wanted a board for its design 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 L20—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.

Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

Features of different Arduino boards are as follows :

Arduino Board	Processor	Memory	Digital I/O	Analogue I/O
Arduino Uno	16Mhz ATmega328	2KB SRAM, 32KB flash	14	6 input, 0 output
Arduino Due	84MHz AT91SAM3X8E	96KB SRAM, 512KB flash	54	12 input, 2 output
Arduino Mega	16MHz ATmega2560	8KB SRAM, 256KB flash	54	16 input, 0 output
Arduino Leonardo	16MHz ATmega32u4	2.5KB SRAM, 32KB flash	20	12 input, 0 output

DEVELOPING ON THE ARDUINO

Integrated Development Environment

You usually develop against the Arduino using the integrated development environment (IDE) that the team supply at http://arduino.cc. Although this is a fully functional IDE, based on the one used for the Processing language (http://processing.org/), 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

Scanned by CamScanner

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. 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 :

- Code is checked and compiled, with any compilation errors reported to you.
- If code compiles successfully, it gets transferred to the Arduino and stored in its flash memory.
- Arduino reboots and starts running the new code.

Operating System

In Arduino, the bootloader, which simplifies the code-pushing process described previously. When the board switched on, it simply runs the code 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.

Language

The language for Arduino is a merely set 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 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.

Scanned by CamScanner

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.

SOME NOTES ON THE HARDWARE

- Number of GPIO pins and is usually supplied with "headers" (plastic strips that sit on the pin holes). Each pin is clearly labelled on the controller board.
- Power outputs such as 5 volts or 3.3 volts (usually labelled 5V and 3V3, or perhaps just 3V),
- One or more electric ground connections (GND), numbered digital pins, and numbered analog pins prefixed with an *A*.
- Arduino can be powered using a USB connection from your computer. The Arduino also has a socket for an external power supply, which you might be more likely to use if you distribute the project.
- Outside of the standard boards, a number of them are focused on a particular niche application—for example, the Arduino Ethernet has an on-board Ethernet chip and trades the USB socket for an Ethernet one, making it easier to hook up to the Internet. This is obviously a strong contender for a useful board for Internet of Things projects.
- 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.

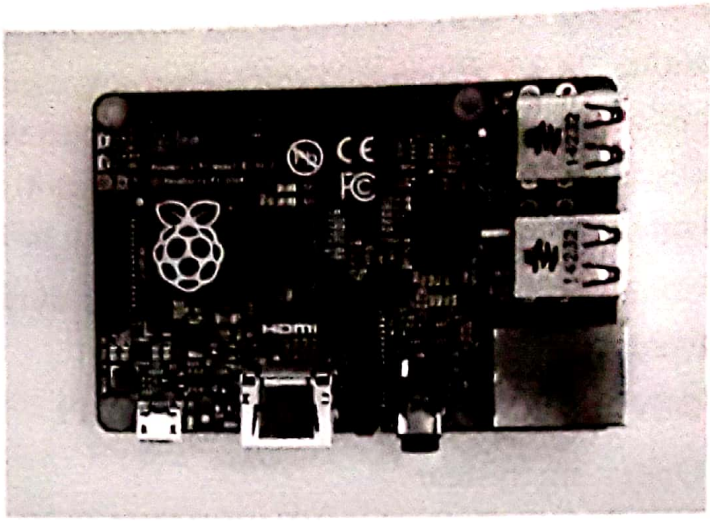
OPENNESS

The Arduino project is completely open hardware. The only part of the project protected is the Arduino trademark, so they can control the quality of any boards calling themselves an Arduino. 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, but many others introduce new features or form factors that the core Arduino team have overlooked.

Scanned by CamScanner

RASPBERRY PI



The **Raspberry Pi** is a tiny and affordable computer that you can use to learn programming through fun, practical projects. Unlike the Arduino, it 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.

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

Arduino Due Raspberry Pi Model B

	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
GPIO	54 GPIO pins	8 GPIO pins
PWM Output	12	1
UART	4	1
Audio/Video output	---	HDMI out Component video and audio out
Connections	SPI bus I²C bus USB 16U2 + native host 12 analogue inputs (ADC) 2 analogue outputs (DAC)	SPI bus with two chip selects I²C bus 2 USB host sockets Ethernet

Scanned by CamScanner

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 WiFi dongles, rather than having to use an extension "shield".

Note that although the specifications of the Pi are in general more than capable even the top-of-the-range Arduino Due,

DEVELOPING ON THE RASPBERRY PI

Operating System

Although many operating systems can run on the Pi, we recommend using a popular Linux distribution, such as

- **Raspbian :** Released by the Raspbian Pi Foundation, Raspbian is a distro based on Debian. This is the default "official" distribution and is certainly a good choice for general work with a Pi.
- **Occidentalis :** This is Adafruit's customised Raspbian. Unlike Raspbian, the distribution assumes that you will use it "headless"—not connected to keyboard and monitor—so you can connect to it remotely by default.

  (Raspbian requires a brief configuration stage first.)

For Internet of Things work, we recommend something such as the Adafruit distro. You're most probably not going to be running the device with a keyboard and display, so you can avoid the inconvenience of sourcing and setting those up in the first place. The main tweaks that interest us are that :

- The sshd (SSH protocol daemon) is enabled by default, so you can connect to the console remotely.
- The device registers itself using zero-configuration networking (zeroconf) with the name raspberrypi.local, so you don't need to know or guess which IP address it picks up from the network in order to make a connection.

Programming Language

One choice to be made is which programming language and environment you want to use. There is some guidance from the Foundation, which suggests Python as a good language for educational programming

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++. :** The Pi has more

Scanned by CamScanner

than enough memory. The speed of execution may or may not be a problem: Python is likely to be "fast enough" for most tasks, and certainly for anything that involves talking to the Internet, the time taken to communicate over the network is the major slowdown. 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 :** Automatic memory management generally results in fewer bugs and performs adequately. However, this automatic work has to be scheduled in and takes some time to complete. Depending on the strategy for garbage collection, this may result in pauses in operation which might affect timing of subsequent events.
- **Linux itself arguably has some issues for "real-time" use :** Due to its being a relatively large operating system, with many processes that may run simultaneously, precise timings may vary due to how much CPU priority is given to the Python runtime at any given moment. This hasn't stopped many embedded programmers from moving to Linux, but it may be a consideration for your case.
- **An Arduino runs only the one set of instructions, in a tight loop, until it is turned off or crashes :** The Pi constantly runs a number of processes. If one of these processes misbehaves, or two of them clash over resources (memory, CPU, access to a file or to a network port), they may cause problems that are entirely unrelated to your code. This is unlikely but may result in occasional, possibly intermittent, issues which are hard to identify and debug.

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, perhaps days or months later.

Python code on Linux gives you the advantages of both the language and the OS.

Because the Pi is a general-purpose computer, without the strict memory limitations of the SOME NOTES ON THE HARDWARE

- The Raspberry Pi has 8 GPIO pins, which are exposed along with power and other interfaces in a 2-by-13 block of male header pins.
- A cable (IDC or similar) can be connected onto the whole block, which leads to a "breakout board" where you do actual work with the GPIO. Alternatively,

Scanned by CamScanner

you can connect individual pins using a female jumper lead onto a breadboard.

- The block of pins provides both 5V and 3.3V outputs. However, the GPIO pins themselves are only 3.3V tolerant.
- The Pi doesn't have any over-voltage protection, so you are at risk of breaking the board if you supply a 5V input! The alternatives are to either proceed with caution or to use an external breakout board that has this kind of protection
- 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.
- It is powered by a standard USB cable, the voltage transmitted over USB from a laptop computer, a powered USB hub, or a USB charger varies greatly. If you're not able to power or to boot your Pi, check the power requirements and try another power source.

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. However, many of the Raspberry Pi core team are Broadcom employees and have been active in creating drivers and the like for it, which are themselves open source.

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.

Scanned by CamScanner