IoT Solutions

1.	Attempt any three of the following:
a.	**Define and explain the Internet of Things.**
Ans:	The Internet of things is defined as a paradigm in which objects equipped with sensors, actuators, and processors communicate with each other to serve a meaningful purpose. **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 thought of as something or 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. Example of Internet of Things: In your kitchen, 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 to let her know. 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.
b.	**"Any sufficiently advanced technology in indistinguishable from magic". Discuss.**
	This is the statement of Clarke's We have seen that technology has evolved to meet our needs and desires. The parallel invention of magic serves similar goals. Any new innovation feels like a magic. This is very old truth of humans. Whatever new thing gets developed using advanced technology, it feels like magic. e.g.

	i. Ringing of telephone and talking to someone distant on it. ii. Invention of bulb At least 5 examples expected on new innovations using advanced technology and explanation.
c.	**Explain calm and ambient technology using example of Live Wire.**
Ans:	The Internet of Things 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 manner of everyday objects. He coined the term ubiquitous computing, or ubicomp for short, to describe it, and through his research and writing sought to explore what that would mean for the people living in such a world. With its focus on computing power being embedded everywhere, ubicomp 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. **Example of Live Wire:** A great example of this approach is Live Wire, one of the first Internet of Things devices. Created by artist Natalie Jeremijenko when she was in residence at Xerox PARC under the guidance of Mark Weiser, Live Wire (also sometimes called Dangling String) is a simple device: an electric motor connected to an eight-foot long piece of plastic string. The power for the motor is provided by the data transmissions on the Ethernet network to which it is connected, so it twitches whenever a packet of information is sent across the network. Under normal, light network load, the string twitches 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 behaviour) 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 olfactory indicator to provide a smelly notification of one of a number of social media events.
d.	**What is manufactured normalcy field? Explain.**
Ans:	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.

e. **Differentiate between static IP address and Dynamic IP address.**

Ans:

	Dynamic IP	**Static IP**
Full form	Dynamic Internet Protocol	Static Internet Protocol
Definition	The internet protocol will constantly change	The internet protocol will remain the same
Cost Effective	More cost effective	Less cost effective
Security Risk	Lower	Higher
Upload/Download Speed	Slower	Faster
Good for	Good for residential user and small business owners	Web servers, email servers and other Internet servers. Also, for VOIP, VPN, playing online games or game hosting

f. Define protocol. Explain the following application layer protocols: HTTP, HTTPS, SMTP, and FTP.

Ans: **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.

HTTP: The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, hypermedia information systems.[1]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.

HTTPS: Hypertext Transfer Protocol Secure (HTTPS) 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.
2.	**Attempt any three of the following.**
a.	**Discuss the tradeoffs between cost versus ease of prototyping.**
Ans:	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 a 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. 1. 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 for about £3. This amount is just for the chip, so you would have 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. 2. Stepping upwards to the approximately £20 mark, 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. But, of course, you are still programming in C++, for reasons of performance and memory. 3. For more money again, approximately £30, you could look at the BeagleBone, 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.
b.	**What are the challenges when we move from prototype to mass production? Explain.**
Ans:	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. This issue is something to be aware of. If the first prototype you built on a PC, iPhone, BeagleBone, or whatever has helped you get investment or collaborators, you may be well placed to go about replicating that compelling functionality on your final target. 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. However, while the

	technique might change—injection moulding in place of 3D printing, for example—in most cases, it won't change what is possible. 3. **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 (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.
c.	**Discuss open source versus closed source hardware and software. State its advantages and disadvantages.**
Ans:	Open source software refers to the where the source code is kept open. People can modify the code and redistribute it. Generally open source is governed by GNU GPL license where the code can be modified, redistributed and sold but the source code has to be kept open. The same applies for the hardware. With open source, there is an issue or intellectual property rights. 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. 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. Although getting good legal information on what to protect and how best to enforce those rights is hard and time-consuming, larger companies may well be geared up to take this route. If you are developing an Internet of Things device in such a context, working within the culture of the company may simply be easier, unless you are willing to try to persuade your management, marketing, and legal teams that they should try something different. 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 Thingiverse. Advantages for open source: • 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. Disadvantages of open source: • People may steal data. • If we decide to cut corners, we may have to go back and fix everything for final release. • After you release something as open source, you may still have a perceived duty to maintain and support it

	• Although you may not have paying customers, your users are a community that you may want to maintain. It is true that, if you have volunteered your work and time, you are entirely responsible for choosing to limit that whenever you want. But abandoning something before you've built up a community around it to pass the reins to cannot be classed as a successful open source project.
d.	**Explain the following with respect to prototyping embedded devices: Processor Speed, RAM, Networking, USB, Power Consumption and physical size and form factor.**
Ans:	**Processor Speed:** The processor speed, or clock speed, of your processor tells you how fast it can process the individual instructions in the machine code for the program it's running. Naturally, a faster processor speed means that it can execute instructions more quickly. The clock speed is still the simplest proxy for raw computing power, but it isn't the only one. You might also make a comparison based on millions of instructions per second (MIPS), depending on what numbers are being reported in the datasheet or specification for the platforms you are comparing. Some processors may lack hardware support for floating-point calculations, so if the code involves a lot of complicated mathematics, a by-the-numbers slower processor with hardware floating-point support could be faster than a slightly higher performance processor without it. **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. **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. **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. **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. **Physical Size and Form Factor:** Physical Size and Form Factor The continual improvement in manufacturing techniques for silicon chips means that we've long passed the point where the limiting factor in the size of a chip is the amount of space required for all the transistors and other components that make up the circuitry on the silicon. Nowadays, the size is governed by the number of connections it needs to make to the surrounding components on the PCB.
e.	**How is development done for Arduino? Explain.**
Ans:	More than just specs, the experience of working with a board may be the most important factor, at least at the prototyping stage.

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: 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: 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. However, for many purposes, you can achieve reasonable results with a simpler task-dispatching library.

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" (a way of doing multitasking, at a very low level).

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.

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.

f. Compare Raspberry Pi and Arduino.

Ans:

SL	Raspberry Pi	Arduino
1	It is a mini computer with Raspbian OS.It can run multiple programs at a time.	Arduino is a microcontroller, which is a part of the computer. It runs only one program again and again.
2	It is difficult to power using a battery pack.	Arduino can be powered using a battery pack.

	3	It requires complex tasks like installing libraries and software for interfacing sensors and other components	It is very simple to interface sensors and other electronic components to Arduino.
	4	It is expensive	It is available for low cost.
	5	Raspberry Pi can be easily connected to the internet using Ethernet port and USB Wi-Fi dongles.	Arduino requires external hardware to connect to the internet and this hardware is addressed properly using code.
	6	Raspberry Pi did not have storage on board. It provides an SD card port.	Arduino can provide onboard storage.
	7	Raspberry Pi has 4 USB ports to connect different devices.	Arduino has only one USB port to connect to the computer.
	8	The processor used is from ARM family.	Processor used in Arduino is from AVR family Atmega328P
	9	This should be properly shutdown otherwise there is a risk of files corruption and software problems.	This is a just plug and play device. If power is connected it starts running the program and if disconnected it simply stops.

	10	The Recommended programming language is python but C, C++, Python, ruby are pre-installed.	Arduino uses Arduino, C/C++.
3.	**Attempt any three of the following.**		
a.	**Explain the non-digital methods of prototyping.**		
Ans:	**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. Plasticine doesn't suffer from this problem, but as it remains malleable, it isn't ideal for prototypes which are going to be handled. Modelling clay is best used for short-term explorations of form, rather than longer-term functional prototypes. **Epoxy putty:** You might have encountered this 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:** We've already mentioned the ubiquitous LEGO sets, but you might also consider Meccano (or Erector Sets in the United States) and plenty of others. If you're lucky, you already have some gathering dust in the attic or that you can borrow from your children. 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, but there's no requirement for you to use them. Many hackers combine an Arduino for sensing and control with LEGO for form and linkages, as this provides an excellent blend of flexibility and ease of construction. **Cardboard:** Cardboard is cheap and easy to shape with a craft knife or scissors, and available in all manner of colours and thicknesses. In its corrugated form, 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, although it is more rigid than corrugated cardboard. There are also specialist foamboard craft knives which allow easy 45-degree cuts for mitred edges and have two blades—spaced 3mm apart—which make it trivial to cut slots into which you can insert another sheet of foamboard to generate three-dimensional shapes. **Extruded polystyrene:** 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.		

b.	**What are laser cutter? Explain the main features to consider while choosing a laser cutter.**
Ans:	Three-dimensional printers can produce more complicated parts, but the simpler design process (for many shapes, breaking it into a sequence of two-dimensional planes is easier than designing in three dimensions), greater range of materials which can be cut, and faster speed make the laser cutter a versatile piece of kit. Laser cutters range from desktop models to industrial units which can take a full 8' by 4' sheet in one pass. Most commonly, though, they are floorstanding and about the same size as a large photocopier. Most of the laser cutter is given over to the bed; this is a flat area that holds the material to be cut. The bed contains a two-axis mechanism with mirrors and a lens to direct the laser beam to the correct location and focus it onto the material being cut. It is similar to a flatbed plotter but one that burns things rather than drawing on them. **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. You don't need to think just about the biggest item you might create; 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. For example, the laser cutter at our workplace has a 40W laser, which can cut up to 10mm-thick acrylic. Moving a few models up in the same range, to one with a 60W laser, would allow us to cut 25mmthick acrylic.
c.	**Explain the different methods used for 3D printing.**
Ans:	**Fused filament fabrication (FFF):** Also known as fused deposition modeling (FDM), this is the type of 3D printer you're most likely to see at a maker event. The RepRap and MakerBot designs both use this technique, as does the Stratasys at the industrial level. It works by extruding 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. However, the surface can have a visible ridging from the thickness of the filament. **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. MakieLab uses laser-sintered nylon to 3D print the physical versions of its dolls. **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. The Z Corp. machines use this technique and use a print medium similar in texture to plaster. After the printing process, the models are quite brittle and so need post- processing 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. The Mcor IRIS is an example of this sort of printer. **Stereolithography and digital light processing**: Stereolithography is possibly the oldest 3D printing technique and has a lot in common with digital light processing, which is enjoying a huge surge in popularity and experimentation at the time of this writing. Both approaches build their models from a vat of liquid polymer resin which is cured 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. Whilst these approaches are limited to printing with resin, the resultant models are produced to a fine resolution. The combination of this with the relatively low cost of DLP projectors makes this a fertile area for development of more affordable high-resolution printers.
d.	**Discuss the different standards that must be considered while implementing APIs.**
Ans:	An API defines the messages that are sent from client to server and from server to client. Ultimately, 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. 1.Representational State Transfer (REST) 2. JSON-RPC 3. XML-RPC: This standard is just like JSON-RPC but uses XML instead of JSON 4. 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.
e.	**Explain POLLING and COMET.**
Ans:	**POLLING :** If you want the device or another client to respond immediately, how do you do that? You don't know when the event you want to respond to will happen, so you can't make the request to coincide with the data becoming available. Consider these two cases: ▪ The WhereDial should start to turn to "Work" the moment that the user has checked into his office. ▪ The moment that the task timer starts, the client on the user's computer should respond, offering the opportunity to type a description of the task. The traditional way of handling this situation using HTTP API requests was to make requests at regular intervals. This is called polling. You might make a call every minute to check whether new data is available for you. However, this means that you can't start to respond until the poll returns. So this might mean a delay of (in this example) one minute plus the time to establish the HTTP connection. You could make this quicker, polling every 10 seconds, for example. But this would put load on the following: ▪ The server: If the device takes off, and there are thousands of devices, each of them polling regularly, you will have to scale up to that load. ▪ The client: This is especially

	important if, as per the earlier Arduino example, the microcontroller blocks during each connect! **COMET**: Comet is an umbrella name for a set of technologies developed to get around the inefficiencies of polling. As with many technologies, many of them were developed before the "brand" of Comet was invented; however, having a name to express the ideas is useful to help discuss and exchange ideas and push the technology forward.
f.	**Write a short note on Message Queuing Telemetry Transport Protocol.**
Ans:	MQTT is a lightweight messaging protocol, designed specifically for scenarios where network bandwidth is limited or a small code footprint is desired. 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. A sister protocol, MQTT for Sensors (MQTT-S), is also available for extremely constrained platforms or networks where TCP isn't available, allowing MQTT's reach to extend to sensor networks such as ZigBee.
4.	**Attempt any three of the following.**
a.	**Discuss the limitations of memory in embedded devices. How is it managed? Explain.**
Ans:	Page 208, 209
b.	**What are the concerns regarding performance and battery life while writing code for embedded systems?**
Ans:	When it comes to writing code, performance and battery life tend to go hand in hand—what is good for one is usually good for the other. Whether either or both of these are things that you need to optimise depends on your application. A device which is tethered to one place and powered by an AC adaptor plugged into the wall isn't as reliant on energy conservation, for example. However, consuming less energy is something to which all devices should aspire. Similarly, if you're building something which doesn't have to react instantly— maybe an ambient notifier for a weather forecast, which doesn't have any ill effect if it updates a few seconds later—or if it doesn't have an interactive user interface which needs to respond promptly to the user's actions, maximising performance might not be of much concern. For items which run from a battery or which are powered by a solar cell, and those which need to react instantaneously when the user pushes a button, it makes sense to pay some attention to performance or power consumption. Although there is a lot of truth in the Donald Knuth.

	A lot of the biggest power-consumption gains come from the hardware design. In particular, if your device can turn off modules of the system when they're not in use or put the entire processor into a low-power sleep mode when the code is finished or waiting for something to happen, you have already made a quick win. That said, it is still important to optimize the software, too! After all, the quicker the main code finishes running, the sooner the hardware can go to sleep. One of the easiest ways to make your code more efficient is to move to an event-driven model rather than polling for changes. The reason for this is to allow your device to sit in a low power state for longer and leap into action when required, instead of having to regularly do busywork to check whether things have changed and it has real work to do. Setting up this model is trickier to do with the networking code if you are acting as a client, rather than waiting as a server. On the hardware side, look to use processor features such as comparators or hardware interrupts to wake up the processor and invoke your processing code only when the relevant sensor conditions are met. If your code needs to pause for a given amount of time to allow some effect to occur before continuing, use calls which allow the processor to sleep rather than wait in a busy-loop.
c.	**Write a short note on libraries for embedded systems.**
Ans:	l**wIP:** 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. This means that some of the retransmission logic for the TCP layer bleeds into the application code, making your code more tightly coupled and more complex. 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 (not quite literally, but 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.
d.	**What is a business model? Who is the business for? Explain.**
Ans:	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. **WHO IS THE BUSINESS MODEL FOR?** Primarily, the reason to model your business is to have some kind of educated hypothesis about whether it might deliver what you want from it. Even if you don't use a semi-formal method like the canvas we just discussed, anyone who starts up any business will have thought, at least briefly, about whether she can afford to do it, what the business is, and whether she'll get paid. As a programmer or a maker, you might believe it counterintuitive to think of a piece of paper with nine boxes in it as a "tool", but when you have a well-tested separation of factors to consider, the small amount of structure the canvas provides should help you think about the business and give you ways to brainstorm different ideas: ▪ What if we target the product at students instead of businesses? ▪ What if we outsource our design to an agency? ▪ What if we sell at low volume/high value instead? Many great product ideas turn out to be impractical, ahead of their time, or unprofitable. Being able to analyze how the related concepts mesh will help you challenge your product idea and either make it stronger or know when to abandon it. The model is also useful if you want to get other people involved. This could be an employee or a business partner…or an investor. In each of these cases, the other parties will want to know that the business has potential, has been thought out, and is likely to survive and perhaps even go places. With a new business startup, you have no track record of success to point to. Although what will sell the business is primarily the product itself, and of course your passion for it, being able to defend a well-thought-out model of the business is an important secondary consideration for someone who is planning to sink time and perhaps money into your business.
e.	**Explain the following business models: Make Thing Sell Thing, Subscriptions, Customisation.**
Ans:	**MAKE THING, SELL THING:** The simplest category of models, "make a Thing and sell it," is, of course, valid for the Internet of Things. Adrian sells custom-built Bubblini, and the startup Good Night Lamp is preparing to ramp up production of its eponymous lamps as an off-the-shelf product. As you will see in Chapter 10, electrical products sold in shops (physical or online) may be subject to legislation and certification (RoHS, Kitemarks, 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. **SUBSCRIPTIONS:** A Thing would be a dumb object if it weren't for the important Internet component which allows the device to remain up to date with useful and current content. But, of course, this ongoing service implies costs to the provider—development, maintenance of servers, hosting costs, and in some cases even connection costs. A subscription model might be appropriate, allowing you to recoup these costs and possibly make ongoing profit by charging fees for your service. Many products could legitimately use this method, but

	perhaps the more complex, content-driven services would find it more convincing. Paying Bubblino a monthly fee to blow bubbles might seem steep, but the BERG Cloud, which delivers nicely formatted news and entertainment to its Little Printer, might have seemed an ideal product for this model. As it stands, content consumers do not pay for either BERG Cloud or for any content subscriptions. **CUSTOMISATION:** We touched on the improvements to mass production whereby the process of buying a car can be tweaked to the buyer's requirements. For an Internet of Things device, at the intersection between solid thing and software, there are options for customisation that we believe may lead to new business models. For a mass-produced item, any customisation must be strictly bounded to a defined menu: a selection of different colours for the paintwork, options for fittings such as tyres, the trimmings and upholstery inside, and for features like the onboard computer control and display. Fordian logic dictates that all these components must be optimised for manufacture and fit well together.
f.	**Write a short note on venture capital.**
Ans:	getting funding for a project from an external investor presents its own work and risks. The process of applying for funding takes time, and although much of this time can be justified as thrashing out the business model, it's not directly related to the work you actually want to be doing on the product itself. Startups often concentrate their fundraising activities into rounds, periods in which they dedicate much of their effort into raising a target amount of money, often for a defined step in their business plan. Before any official funding round comes the informal idea of the friends, family, and fools (FFF) round. This stage may be the one in which you've contributed your life savings, and persuaded your aunt, your best friend, and a local small business to pitch in the rest, on the basis of your reputation. Although it's important to consider the possible impact on your personal relationships, this round of funding may be the most straightforward to get hold of. A common next step would be an angel round. The so-called angels are usually individual investors, often entrepreneurs themselves, who are willing to fund some early-stage startups which a more formal investor (such as venture capitalists that we look at shortly) might not yet touch. The reason might be that these angels have a technical or business background in your product or simply that, as individual investors, they may have more scope to go with their own intuition about your worth. Angels typically disburse sums that are significant for early-stage startups—in the region of tens or possibly hundreds of thousands of pounds. However, the personal interest and experience that angels can bring to your company means that their advice, contacts, and other help may well be as useful as any money they provide. A good place to find an angel in the US could be AngelList, a long tail aggregator where investors can meet startups. The venture capital (VC) round is similar, but instead of your courting individual investors, the investor is a larger group with significant funds, whose sole purpose is to discover and fund new companies with a view to making significant profit. VCs may be interested if angels have already funded you and will certainly be interested if other VC companies are already looking at funding you. VCs will certainly want equity, probably a significant amount of it, and a position on your board of directors. Again, this last role may be as much to help fill gaps that your management team don't cover as much as it is

	to keep an eye on you and their money. Typically, VC funding will be larger chunks of money, from half a million pounds up.
5.	**Attempt any three of the following.**
a.	**What are the different software options for designing PCB? Explain.**
Ans:	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. **Fritzing:** 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. **KiCad:** 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. **EAGLE:** The reason for its popularity most likely comes down to its long having a free version for noncommercial 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. So EAGLE is a good choice to learn PCB design, although in addition to its noncommercial licence, the free version is also restricted to two layers and a maximum board size of 100mm × 80mm.
b.	**Explain the steps for manufacturing PCBs.**
Ans:	**ETCHING BOARDS:** The most common PCB-making technique for home use is to etch the board. Some readily available kits provide all you need. The first step is to get the PCB design onto the board to be etched. This process generally involves printing out the design from your PCB design software onto a stencil. If you're using photo-resist board, it will be onto a stencil which masks off the relevant areas when you expose it to UV light; or if you're using the toner-transfer method, it will be for your laser printer to print onto glossy paper ready to transfer. **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. **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. **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. **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. 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.
c.	**What is the importance of Certification for IoT devices? Explain.**
Ans:	**1.**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. **2.** For the main part, these regulations are there for good reason. They make the products you use day in, day out, safer for you to use; make sure that they work properly with complementary products from other suppliers; and ensure that one product doesn't emit lots of unwanted electromagnetic radiation and interfere with the correct operation of other devices nearby. **3.** The regulations that your device needs to pass vary depending on its exact functionality, target market (consumer, industrial, and so on), and the countries in which you expect to sell it. 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. **4.** 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. This is sometimes noticeable in the "dit, dit-dit-dit" picked up by a poorly insulated stereo just before your mobile phone starts ringing.
d.	**Explain privacy with respect to Internet of Things.**
Ans:	**1.**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. **2.** 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). **3.**Even innocuous photos can leak data. With GPS coordinates (produced by many cameras and most smartphones) embedded into the picture's EXIF metadata, an analysis of your Flickr/Twitpic/Instagram feed can easily let an attacker infer where your house, your work, or even your children's school is. Even if you stripped out the data, photo-processing technology enables searching of similar photos, which may include these coordinates or other clues. **4.** 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. **5.** 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.
e.	Discuss the five critical requirements for sensor commons project.
Ans:	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 (for example, local WiFi bandwidth usage), 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 (both the physical sensor itself and the ongoing costs of keeping it powered and connected to the network) and if the community already trusts the project. If the sensors are complicated to set up or require massive lengths of cabling, they will get much less take-up! The Xively air-quality project led to the creation of the "air-quality egg", a simple, inexpensive sensor with precisely these features. **Be highly visible**: Being visible involves explaining why the project's sensors are occupying a public space. We've already discussed the ethics of hidden sensors. Being honest and visible about the sensor will help to engender trust in the project and also advertise and explain the project further. This may reduce the probability of vandalism too. Advertising not just the sensors but the data (both online and in real life) and the ways that data has helped shape behaviour will also generate a positive feedback loop.

	Be entirely open: Being open is perhaps what distinguishes the sensor commons from a government project the most. Government data sets are often entirely closed, but the data that is released from them will be given a lot of attention because of the rigour and precision that (we expect) their sensor projects will have. A community sensor network may have uncalibrated devices—that is, the readings for a device may be consistently out from the "correct" value and may have additional noise at the extremes of the scale. **Be upgradable:** Finally, the project should be designed to be upgradable, to enable the network to remain useful as the needs change or hardware gets to the end of its working life. This requirement interplays with the dispersibility and openness of the project, and the up-front thought to managing the project long term will feed back into the trust in the project.
f.	Write a short note on cautious optimism.
Ans:	**1.**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. **2.** It is true that any technological advance could be co-opted by corporations, repressive governments, or criminals. But (we hope) technology can be used socially, responsibly, and (if necessary) 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. **3.** As a massively interdisciplinary field, practitioners of Internet of Things may have an opportunity (or perhaps responsibility) to contribute to providing moral leadership in many of the upcoming ethical challenges we have looked at. Before we let that get to our heads though, we should remember an important lesson on humility from Laura James's keynote at the OpenIoT assembly **4.** 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.