Unit III

Contents:

Prototyping the Physical Design: Preparation, Sketch, Iterate, and Explore, Nondigital Methods, Laser Cutting, Choosing a Laser Cutter, Software, Hinges and Joints, 3D Printing, Types of 3D Printing, Software, CNC Milling, Repurposing/Recycling.

Prototyping Online Components: Getting Started with an API, Mashing Up APIs, Scraping, Legalities, Writing a New API, Clockodillo, Security, Implementing the API, Using Curl to Test, Going Further, Real-Time Reactions, Polling, Comet, Other Protocols, MQ Telemetry Transport, Extensible Messaging and Presence Protocol, Constrained Application Protocol.

Recommended Books:

Designing the Internet of Things by Adrian McEwen, Hakim Cassimally, WILEY publication

Internet of Things Architecture and Design by Raj Kamal, McGraw Hill Publication

Prerequisites

Unit II	Sem. II	Sem. III	Sem. IV	Sem. V	Sem. VI
Prototyping	Microprocessor Architecture		Embedded Sytems	Internet of Things	

PROTOTYPING THE PHYSICAL DESIGN

One of the reasons that the Internet of Things is such an exciting area is that it cuts across so many different disciplines.

It involves software, electronics, experience design, and product design.

It's this last field, product design, that seems to be the stumbling block for many a nascent Internet of Things project, judging by the number of technically brilliant creations gracing the pages of the Hackaday website or Kickstarter that are left as bare circuit boards or crammed into whatever box came to hand.

If the Internet of Things is to succeed in reaching mass appeal, this lack of design is something we need to change.

PREPARATION

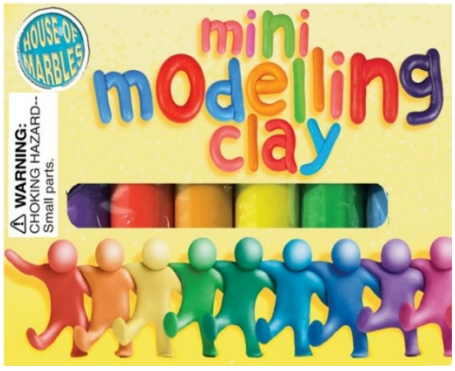
- The other side to a (very) basic design education is an understanding and appreciation of the tools available and the materiality of the processes involved in making things in the physical, rather than the digital, realm.
- Items crafted in the digital world can focus exclusively on appearances and take on any form they like; in the real world, those pesky laws of physics and the limitations of the tools for fabrication come into play.
- As with most things, the more experience you gain with different materials and tools, the better your understanding will be when you come to design something.
- This is another great argument for seeking out and joining your local hackspace or makerspace. Most of them have tools that you wouldn't necessarily be able to afford on your own, such as laser cutters or 3D printers, and many have other machines, such as CNC mills, lathes, or vacuum formers. All of them have a wealth of expertise that you can tap into, in the form of the community of members.

SKETCH, ITERATE, AND EXPLORE

- Doing a lot of preparatory work doesn't mean that you won't do additional exploration and gathering of possible starting points when you do finally sit down with a specific project in mind.
- Arguably, in the early stages of a design, you can never do too much iterating through ideas and trying out different approaches to solving the problem.
- Your first idea is unlikely to be the best, so you should be optimising for speed of iteration rather than quality of prototype.
  - You could iterate through designs with a 3D printer, but doing so with a pen and paper is much quicker.
- Begin with a broad search across the problem area.
- The objective is to get to grips with as many aspects of the design as possible, rather than drilling down into one specific possible result.
- Use whatever tools make most sense to help with the idea generation and exploration.
- The key lesson is to use these techniques to experiment with different possibilities and learn which features of which designs are best.
- This approach allows you to synthesize the results into a coherent final design.

NONDIGITAL METHODS

- Pen and paper remain an essential tool in the designer's arsenal, but they aren't the only ones to have survived the digital revolution.
- One of the key advantages that these techniques have over the newer digital fabrication methods is their immediacy.
- **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 have a tendency to dry out and crack if left exposed to the air. Some remain malleable, and aren'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 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.

- **Toy construction sets**
  - **T**he ubiquitous LEGO sets, you might also consider Meccano (or Erector Sets in the United States) and plenty of others. The interesting feature of these sets is the availability of gears, hinges, and other pieces to let you add some movement to your model. 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. There are also specialist foamboard craft knives.

- **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. It is often referred to as "blue foam", although it's the density rather than the colour which is important. Light yet durable, it can be easily worked: you can cut it with a craft knife, saw it, sand it, or, for the greatest ease in shaping it, buy hot-wire cutter.

LASER CUTTING

https://www.youtube.com/watch?v=SIjUVCho_xU

- Although the laser cutter doesn't get the same press attention as the 3D printer, it is arguably an even more useful item to have in your workshop.
- Three-dimensional printers can produce more complicated parts, but the simpler design process, greater range of materials which can be cut, and faster speed make the laser cutter a versatile piece of kit.
- Laser cutters range from desktop models to industrial units which can take a full 8' by 4' sheet in one pass.
- Most of the laser cutter is given over to the bed; this is a flat area that holds the material to be cut.
- The bed contains a two-axis mechanism with mirrors and a lens to direct the laser beam to the correct location and focus it onto the material being cut. It is similar to a flatbed plotter but one that burns things rather than drawing on them.
- The computer controls the two-axis positioning mechanism and the power of the laser beam.
- This means that not only can the machine easily cut all manner of intricate patterns, but it can also lower the power of the laser so that it doesn't cut all the way through.

CHOOSING A LASER CUTTER

- When choosing a laser cutter, you should consider two main features:
  - **The size of the bed:** This is the place where the sheet of material sits while it's being cut, so a larger bed can cut larger items.
  - **The power of the laser:** More powerful lasers can cut through thicker material.
- Depending on what you're trying to create, you can cut all sorts of different materials in a laser cutter.
  - Whilst felt, leather, and other fabrics are easy to cut, for Internet of Things devices you will probably be looking at something more rigid.
  - Card and, particularly, corrugated cardboard are good for quick tests and prototyping, but MDF, plywood, and acrylic are the most common choices.
- Whilst you are able to get laser cutters which can cut metal, they tend to be the more powerful and industrial units.
- If you don't have a laser cutter of your own, there is a good chance that your local makerspace or hackspace will have one that you could use.
- You might even be able to obtain access to one at a local university or college.
- Failing that, laser-cutting bureau services somewhat like copy shops are becoming increasingly common. Architects often use these services to help them build architectural models.
- If that approach proves fruitless, a number of online providers, such as Ponoko, let you upload designs that they cut and then post back to you.

SOFTWARE

- The file formats or software which you need to use to provide your design vary across machines and providers.
- Although some laser-cutting software will let you define an engraving pattern with a bitmap, typically you use some type of vector graphics format.
- Vector formats capture the drawing as a series of lines and curves, which translate much better into instructions for moving the laser cutter than the grid-like representation of a bitmap. There's also no loss in fidelity as you resize the image.
- **CorelDRAW** is a common choice for driving the laser cutters themselves, and you can use it to generate the designs too. Other popular options are **Adobe Illustrator**, as many designers already have a copy installed, and **Inkscape**, largely because it's an open source alternative and therefore freely available. The *best* choice is the one you're most comfortable working with, or failing that, either the one your laser cutter uses or the one you can afford.
- When creating your design, you use the stroke (or outline) of the shapes and lines rather than the filled area to define where the laser will cut and etch.
- Different types of operation—cut versus etch or even different levels of etching—can usually be included in the same design file just by marking them in different colours.

HINGES AND JOINTS

Most of the mechanisms you use to construct items with the laser cutter aren't any different from those used in more general woodwork. A few lesser-known techniques, however, are either easier to achieve with the precision of the laser cutter or have found new popularity after being picked up by this new generation of makers.

Lattice (or Living) Hinges

If you're looking to introduce some curves into your design, one of these hinge patterns, reminiscent of the lattice pastry on top of a fruit pie, will do the trick. A series of closely laid-out cuts, perpendicular to the direction of the curve, allows the material to be bent after it has been cut. Varying the number of cuts and their separation affects the resulting flexibility of the hinge

Integrated Elastic Clips

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

Bolted Tenon (or T-Slot) Joints

An alternative to integrated elastic clips, the bolted tenon joint is a modified version of the standard mortise-and-tenon joint which adds a T- or crossshaped slot to the tenon sheet, with the crossbar of the T or cross being just big enough to hold a nut. You can then thread a bolt through a hole in the mortise sheet, down the slot and through the nut.

3D PRINTING

Additive manufacturing, or 3D printing as it's often called, is fast becoming one of the most popular forms in rapid prototyping—largely down to the ever-increasing number of personal 3D printers, available at ever-falling costs. Now a number of desktop models, available for less than L500, produce decent quality results.

The term *additive manufacturing* is used because all the various processes which can be used to produce the output start with nothing and *add* material to build up the resulting model. This is in contrast to *subtractive manufacturing* techniques such as laser cutting and CNC milling, where you start with more material and cut away the parts you don't need.

Various processes are used for building up the physical model, which affect what materials that printer can use, among other things. However, all of them take a three-dimensional computer model as the input. The software slices the computer model into many layers, each a fraction of a millimetre thick, and the physical version is built up layer by layer.

One of the great draws of 3D printing is how it can produce items which wouldn't be possible with traditional techniques. For example, because you can print interlocking rings without any joins, you are able to use the metal 3D printers to print entire sheets of chain-mail which come

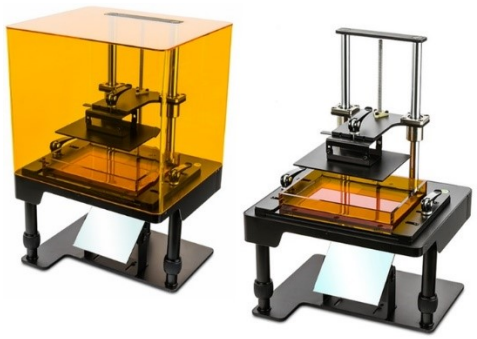
out of the printer already connected together. If only the medieval knights had had access to a metal laser-sintering machine, their armour would have been much easier to produce.

Another common trick with 3D printing is to print pieces which include moving parts: it is possible to print all the parts at the same time and print them ready-assembled. This effect is achieved with the use of what is called “support material”. In some processes, such as the powder-based methods, this is a side effect of the printing technique; while the print is in progress, the raw powder takes up the space for what will become the air-gap.

Afterwards, you can simply shake or blow the loose powder out of your solid print. Other processes, such as the extruded plastic techniques, require you to print a second material, which takes the supporting role. When the print is finished, this support material is either broken off or washed away. (The support material is specifically chosen to dissolve in water or another solution which doesn’t affect the main printing material.)

https://www.youtube.com/watch?v=aPUHraVh-aE

TYPES OF 3D PRINTING



Lots of innovation is still happening in the world of additive manufacturing, but the following are some of the more common methods of 3D printing in use today:

- **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 (discussed in Chapter 9, "Business Models") 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 postprocessing where they are sprayed with a hardening solution. The great advantage of these printers is that when the binder is being applied, it can be mixed with some pigment; therefore, full-colour prints in different colours can be produced in one pass.

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

If you don't need the specialist materials or high resolution of the high-end machines, there's a good chance that your local hackspace or makerspace will have one of the lower-cost desktop machines; the pricing of these machines is such that buying one of your own is also an option. In fact, for most prototyping work, one can argue that the greater access and lower cost of materials in that approach far outweigh the disadvantages.

SOFTWARE

In much the same way as for laser cutting, no definitive software package is recommended for use when generating your 3D designs. If you are already familiar with one 3D design program, see whether it can export files in the correct format for the machine you'll use to print. If you

are using a printing service, it will advise on which program it prefers you to use or what formats it accepts. Or failing that, choose one to suit your budget and that you find easiest to get to grips with.

Working out how to design items in three dimensions through a two dimensional display isn't trivial, so it's more important than usual to work through the tutorials for the software you choose. This gives you the best grounding to manipulating objects, ensuring that the components which make up your design line up correctly to minimize the risk of artefacts in the finished print.

Tinkercad (http://tinkercad.com) and Autodesk's 123D Design Online (http://www.123dapp.com/design) are two options which just run in your web browser. So they let you start designing without having to install any additional software.

Autodesk also has a range of 123D apps available to download and install.

You can find a desktop version of 123D Design and also of 123D Catch, a clever application which takes an array of photos of an object and automatically converts them into a 3D model. This inferred 3D model may then need subsequent refinement—for example, with 123D Design.

SolidWorks (http://www.solidworks.com) and Rhino (http://www.rhino3d.com) are the industry-standard commercial offerings, and SketchUp (http://www.sketchup.com), which was owned by Google for a while but in 2012 was sold to Trimble, is popular with hobbyists.

In the open source camp, the main contenders are OpenSCAD (http://www.openscad.org) , which has a rather unorthodox scripting workflow, and FreeCAD (http://free-cad.sourceforge.net). You also can use Blender (http://www.blender.org), but it has a steep learning curve and is better suited to 3D animation than computer-aided design.

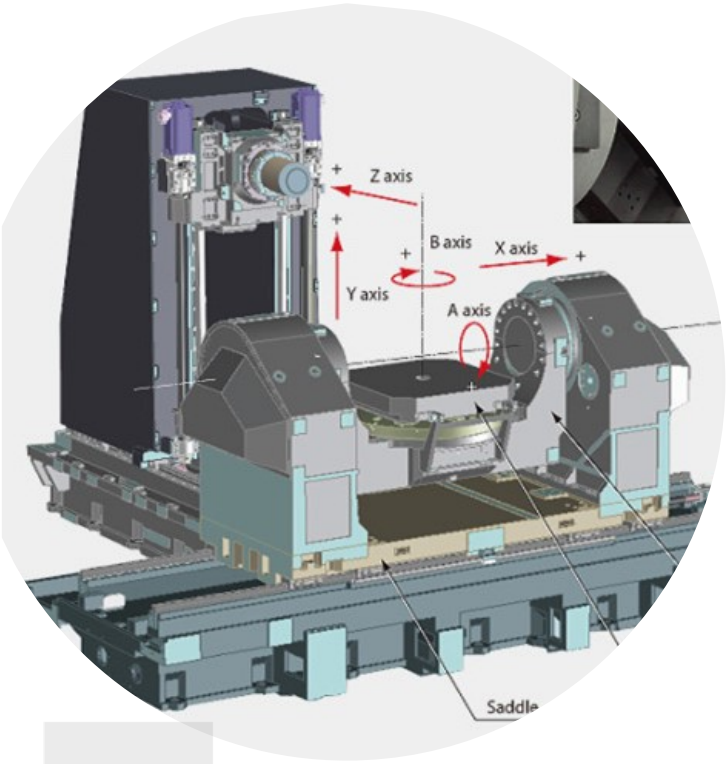
When you have your design ready, you need a further piece of software to convert it into a set of instructions which will be fed to the printer. This is usually known as the *slicing algorithm* because its most important function is to carve the model into a series of layers and work out how to instruct the printer to build up each layer. In most cases the particular slicing software that you use is governed by the specific printer which is building your model, but with the open source designs such as RepRap, you might have a couple of options.

Skeinforge was the first slicing software used by the open source printers, but it has been largely overtaken by the newer and more user-friendly Slic3r. Both will let you tweak all manner of parameters to fine-tune your 3D prints, specifying options like the temperature to which the plastic should be heated, how densely to fill the solid objects, the speed at which the extruder head should move, etc.

Getting those settings right (or right enough) can be daunting for the beginner. With its configuration wizard, Slic3r does a much better job of guiding you through to a usable starting point. Running through some calibration tests will let you tailor the settings to your particular printer and the specific plastic that you're printing.

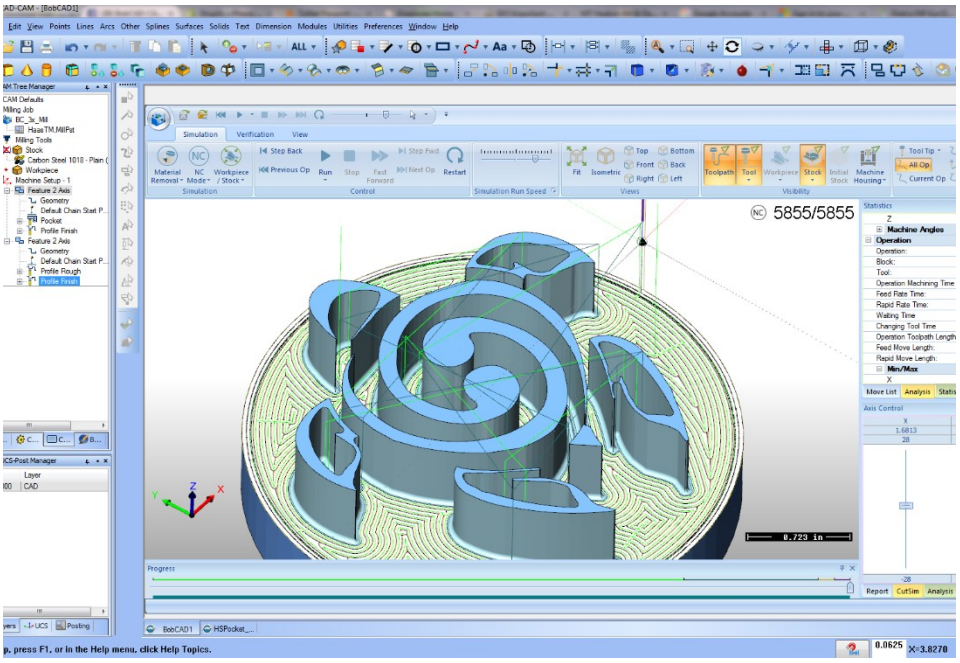
It can feel l ike a bit of a chore to be printing out 20mm cubes when you're itching to set it going with your great design, but taking some time to set things up when the issues are more easily spotted and remedied will pay back in better quality and more successful prints.

CNC Milling



- Computer Numerically Controlled (CNC) milling is similar to 3D printing but is a *subtractive* manufacturing process rather than *additive*.
- The CNC part just means that a computer controls the movement of the milling head, much like it does the extruder in an FDM 3D printer.
- However, rather than building up the desired model layer by layer from nothing, it starts with a block of material larger than the finished piece and cuts away the parts which aren't needed.
- Because cutting away material is easier, CNC mills can work with a much greater range of materials than 3D printers can.
- CNC mills can also be used for more specialised tasks, such as creating custom printed circuit boards.
- A wide range of CNC mills is available, depending on the features you need and your budget.
    - Sizes range from small mills which will fit onto your desktop through to much larger machines with a bed size measured in metres.
    - Bigger is not always better, though; the challenges of accurately moving the carriage around increase with their size, so smaller mills are usually able to machine to higher tolerances. That said, the difference in resolution is only from high to extremely high.
- Beyond size and accuracy, the other main attribute that varies among CNC mills is the number of axes of movement they have:

    - **2.5 axis:** Whilst this type has three axes of movement—X, Y, and Z—it can move only any two at one time.
    - **3 axis:** Like the 2.5-axis machine, this machine has a bed which can move in the X and Y axes, and a milling head that can move in the Z. However, it can move all three at the same time.
    - **4 axis:** This machine adds a rotary axis to the 3-axis mill to allow the piece being milled to be rotated around an extra axis, usually the X (this is known as the *A axis*). An indexed axis just allows the piece to be rotated to set points to allow a further milling pass to then be made, for example, to flip it over to mill the underside; and a fully controllable rotating axis allows the rotation to happen as part of the cutting instructions.
    - **5 axis:** This machine adds a second rotary axis—normally around the Y—which is known as the *B axis*.
    - **6 axis:** A third rotary axis—known as the *C axis* if it rotates around Z—completes the range of movement in this machine.
- As with 3D printing, the software you use for CNC milling is split into two types:
    - **CAD** (Computer-Aided Design) software lets you design the model.
    - **CAM** (Computer-Aided Manufacture) software turns that into a suitable toolpath—a list of co-ordinates for the CNC machine to follow which will result in the model being revealed from the block of material.
- The toolpaths are usually expressed in a quasi-standard called *G-code*.
- Whilst most of the movement instructions are common across machines, a wide variety exists in the codes for things such as initialising the machine.
- That said, a number of third-party CAM packages are available, so with luck you will have a choice of which to use.



Repurposing / Recycling

• One reason to reuse mechanisms or components would be to piggyback onto someone else's economies of scale.

– If sections or entire subassemblies that you need are available in an existing product, buying those items can often be cheaper than making them in-house.

– That's definitely the case for your prototypes but may extend to production runs, too, depending on the volumes you'll be manufacturing.

• We've drifted away from the idea of prototyping as a way to explore and develop your idea, but that is probably the most common case where reuse and repurposing of existing items are useful.

• Given that the prototyping phase is all about rapid iteration through ideas, anything that helps speed up the construction period and gets you to where you can test your theories is useful.

- If the final design requires processes with massive up-front costs (such as tooling up for injection moulding the plastics) or the skills of a designer that you don't have the funds to hire right now, maybe a product already exists that is near enough to work as a proxy.
- That lets you get on with taking the project forwards, ending up at a point, one hopes, where making the bigger investment makes sense.
- And, of course, it doesn't have to be a finished item that you reuse.
    - The website Thingiverse is a repository of all manner of designs, most of which are targeted at 3D printing or laser cutting, and all available under creative commons licenses which allow you to use the design as is or often allow you to amend or extend it to better suit your own needs.

Getting started with API

- The most important part of a web service, with regards to an Internet of Things device, is the Application Programming Interface, or API.
- An API is a way of accessing a service that is targeted at machines rather than people.
- If you think about your experience of accessing an Internet service, you might follow a number of steps. For example, to look at a friend's photo on Flickr, you might do the following:

  - Launch Chrome, Safari, or Internet Explorer.
  - Search for the Flickr website in Google and click on the link.
  - Type in your username and password and click “Login”.
  - Look at the page and click on the “Contacts” link.
  - Click on a few more links to page through the list of contacts till you see the one you want.
  - Scroll down the page, looking for the photo you want, and then click on it.
- Although these actions are simple for a human, they involve a lot of looking, thinking, typing, and clicking. A computer can’t look and think in the same way.
- The tricky and lengthy process of following a sequence of actions and responding to each page is likely to fail the moment that Flickr slightly changes its user interface.
  - For example, if Flickr rewords “Login” to “Sign in”, or “Contacts” to “Friends”, a human being would very likely not even notice, but a typical computer program would completely fail.
- Mashing Up API’s
  - Perhaps the data you want is already available on the Internet but in a form that doesn’t work for you?
  - The idea of “mashing up” multiple APIs to get a result has taken off and can be used to powerful effect.
  - For example:
    - Using a mapping API to plot properties to rent or buy—for example, Google Maps to visualise properties to rent via 99acres.
    - Showing Twitter trends on a global map or in a timeline or a charting API.
    - Fetching Flickr images that are related to the top headlines retrieved from *The Guardian* newspaper’s API.
  - Some of the more visible and easy-to-use APIs want to embed your data within them—for example, the Google Maps API.
  - This means that they are ideal to use within a web browser, but you aren’t in control of the final product, and there might be limited scope for accessing them from a microcontroller.
- Scraping
  - In many cases, companies or institutions have access to fantastic data but don’t want to or don’t have the resources or knowledge to make them available as an API.
  - While you saw in the Flickr example above that getting a computer to pretend to be a browser and navigate it by looking for UI elements was fragile, that doesn’t mean that doing so is impossible.

  - In general, we refer to this, perhaps a little pejoratively, as “screen-scraping”.

- Legalities
  - Screen-scraping may break the terms and conditions of a website. For example, Google doesn’t allow you to screen-scrape its search pages but does provide an API.
  - Other data is protected by copyright or, for example, database rights. For example, one project that would be a scraper that read football fixtures and moved a “compass” to point to the relative direction that your team was playing in. However, certainly in the UK, fixtures lists are copyrighted, and the English and Scottish football leagues have sued various operators for not paying them a licensing fee for that data.
  - Alternative sources of information often are available. For example, you could use OpenStreetMap instead of Google Maps.

Writing a new API

- Assuming the data you want to play with isn’t available or can’t be easily mashed up or scraped using other existing tools and sources, perhaps you want to create an entirely new source of information or services.
- Perhaps you plan to assemble the data from free or licensed material you have and process it. Or perhaps your Internet-connected device can populate this data!
- To take you through the process of building your own API, we use an example project, Clockodillo. This is an Internet of Things device uses the Pomodoro time management technique.
- With the Pomodoro system you split your tasks into 25-minute chunks and use a kitchen-timer to help track how long the task takes you, and to encourage you to block out distractions during each 25-minute block.
- Clockodillo explores how the Internet of Things might help with that: connecting the kitchen-timer to the Internet to make the tracking easier while keeping the simplicity of the physical twist-to-set timer for starting the clock and showing progress as it ticks down.
- Although the process of designing a web application to be used on a browser can mix up the actions that users will accomplish with the flows they will take to navigate through the application, writing the back-end API makes you think much more in terms of the data that you want to process.
- When you know what data you have, what actions can be taken on it, and what data will be returned, the flows of your application become simple.
- This is a great opportunity to think about programming without worrying about the user interface or interactions.
- Although this might sound very different from writing a web application, it is actually an ideal way to start: by separating the business problem from the front end, you decouple the model (core data structure) from the view (HTML/JavaScript) and controller (widgets, form interaction, and so on).

- The best news is, if you start designing an API in this way, you can easily add a website afterwards.
- Clockodillo
    - Clockodillo is an Internet-connected task timer.
    - The user can set a dial to a number of minutes, and the timer ticks down until completed.
    - It also sends messages to an API server to let it know that a task has been started, completed, or cancelled.
    - A number of API interactions deal precisely with those features of the physical device:
        - Start a new timer
        - Change the duration of an existing timer
        - Mark a timer completed
        - Cancel a timer
    - Some interactions with a timer data structure are too complicated to be displayed on a device consisting mostly of a dial—for example, anything that might require a display or a keyboard! Those could be done through an app on your computer or phone instead.
        - View and edit the timer's name/description
    - And, naturally, the user may want to be able to see historical data:
        - Previous timers, in a list
        - Their name/description
        - Their total time and whether they were cancelled
    - We assume that each device will send some identifying token, such as a MAC address. So the user will somehow identify himself with the server, after which all the preceding actions will relate just to a given user ID.

The human-facing Clockodillo login page.

- Security
  - How important security is depends a lot on how sensitive the information being passed is and whether it's in anyone's interest to compromise it.
  - For Clockodillo, perhaps a boss might want to double-check that employees are using the timer.
  - Or a competitor might want to check the descriptions of tasks to spy what your company is working on. Or a competitor might want to disrupt and discredit the service by entering fake data.
  - If the service deals with health or financial information, it may be an even more attractive target.
  - Location information is also sensitive; burglars might find it convenient to know when you are out of the house.
  - Security is a really important concern, so you need to bear it in mind while designing the API!
  - The request has to pass details to identify the user, which is the problem of identity; that is, the application needs to know for which user to create the timer so that the user can retrieve information about it later.
  - But the application should also authenticate that request.
    - A password is "good enough" authentication for something that isn't hypersensitive.
  - Tasks 1–4 could be requested by the physical timer device.

    - To pass a description, display details about a list of timers, or get information about them would require more input and output capability than the timer will have!

– But for tasks 1–4, how will the timer pass on the username and password?

    - The user could configure them with a computer, via USB. But doing so is potentially complex and means that the device will need some persistent storage.

One technique that is commonly used for microcontrollers is that they can send a physical ID, commonly their MAC address. As this is unique, it can be tied to a user.

– Also, you have to consider the risks in sending the identification or authentication data over the Internet — whether that's a MAC address or username and password.

– If the username and password are in "clear text", they can be read by anyone who is packet sniffing.

– The two main cases here are as follows:

    - **Someone who is targeting a specific user and has access to that person's wired or (unencrypted) wireless network.**
    - **Someone who has access to one of the intermediate nodes.**

– One obvious solution to the problem of sending cleartext passwords would be to encrypt the whole request, including the authentication details.

    - For a web API, you can simply do this by targeting https:// instead of http://.

– Resolving this problem may be harder for the device. Encryption requires solving very large equations and takes CPU and memory. The current Arduino platform doesn't have an HTTPS library, for example.

– If you are defining your own API, there are cryptography libraries for Arduino, so there's scope for using them for a custom form of secure communications.

– The OAuth 1.0 protocol — used by services such as Twitter to allow third party applications to access your account without requiring your password — is a good example of providing strong authentication without using HTTPS.

– Here's a revised table; we also added some requests to add and check the MAC address for a user and categorised the previous requests by the type of resource they affect.

– As you can see, the first four options can be set by the device, but more methods relate to the clients (website or native) than the device itself.

– This is often the case: the device provides a number of functions that its inputs and outputs are particularly well suited for, but a whole host of other functions to support the data, control authentication, and present and edit it may require a richer input device.

- Implementing the API
  - 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.
  - Here are a few of the most common standards that you should consider:
    - **Representational State Transfer (REST):** Access a set of web URLs like http://timer.roomofthings.com/timers/ or http:://timer.roomofthings.com/timers/1234 using HTTP methods such as GET and POST, but also PUT and DELETE. The result is often XML or JSON but can often depend on the HTTP content-type negotiation mechanisms.
    - **JSON-RPC:** Access a single web URL like http://timer.roomofthings.com/api/, passing a JSON string such as {‘method’:’update’, ‘params’: [{‘timer-id’:1234, ‘description’:’Writing API chapter for book’}], ‘id’:12}. The return value would also be in JSON, like {‘result’:’OK’, ‘error’:null, ‘id’:12}.
    - **XML-RPC:** This standard is just like JSON-RPC but uses XML instead of JSON.
    - **Simple Object Access Protocol (SOAP):** This standard uses XML for transport like XML-RPC but provides additional layers of functionality, which may be useful for very complicated systems.
  - For this chapter, we use a REST API because it is popular, well supported, and simple to interact with for a limited microcontroller.
  - REST has some disadvantages, too. For example, some of the HTTP methods aren’t well supported by every client or, indeed, server. In particular, web browsers only natively support GET and POST, which can complicate things when interacting with REST from a web page.
  - In REST, you attach each resource to a URL and act on that.
    - For example, to interact with a timer, you might speak to /timers/1234 for a timer with ID 1234. To create an entirely new timer, you would talk to /timers.
  - As you can see, you can use different “methods” depending on whether you want to GET a resource, POST it onto the server in the first place, PUT an update to it, or DELETE it from the server.

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

- Using CURL to test
  - While you're developing the API, and afterwards, to test it and show it off, you need to have a way to interact with it.
  - You could create the client to interface with it at the same time (either an application on the web or computer, or the code to make your Internet of Things project connect to it).
  - In this case, while we were developing Clockodillo, the API was ready long before the physical device.
  - Luckily, many tools can interact with APIs, and one very useful one is curl, a command-line tool for transferring all kinds of data, including HTTP.
  - You can easily issue GET requests by simply calling curl http://timer.roomofthings.com/timers.json, for example.
  - curl simply makes an HTTP request and prints out the result to a terminal.
  - Because the command line requests JSON, the result comes back in that format, with a dictionary of status and id values.
- Going Further
  - The preceding sketch is missing a few tweaks before it can become a production-ready API.
    - The timer duration changing is rudimentary.
    - The code doesn't handle the case in which the timer should already have expired by the time the user tries to change it.

    - Perhaps the data structure should also be expanded to store more history about a single timer.

- This example also has a number of architectural features that we didn't examine at all.

- **API Rate Limiting**

    - If the service becomes popular, managing the number of connections to the site becomes critical.

    - Setting a maximum number of calls per day or per hour or per minute might be useful. You could do this by setting a counter or each period that you want to limit.

    - Then the authentication process could simply increment these counters and fail if the count is above a defined threshold. The counters could be reset to zero in a scheduled cron job.

    - While a software application can easily warn users that their usage limit has been exceeded and they should try later, if a physical device suddenly fails, users might assume that it is broken! Solutions to this problem might include simply not applying the limit to calls made by a device.

- **OAuth for Authenticating with Other Services**

    - While OAuth may not (currently) be the best solution for connecting with a microcontroller, there is no reason why the back-end service should not accept OAuth to allow hooks to services like Twitter, music discovery site last.fm, or the web automation of If This Then That.

- **Interaction via HTML**

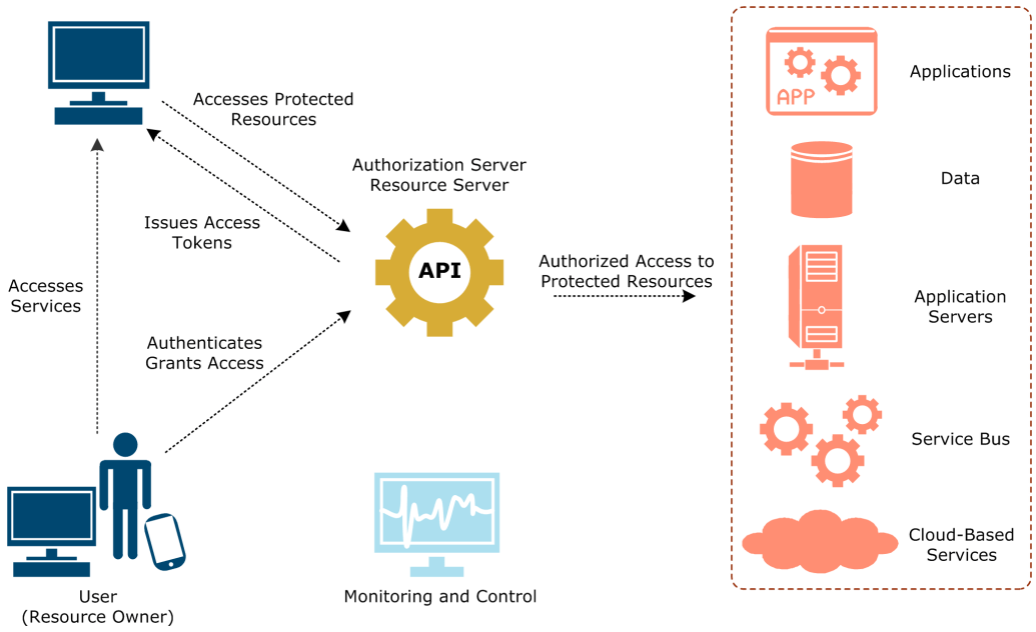
    - The API currently serialises the output only in JSON, XML, and Text formats. You might also want to connect from a web browser.

    - When we first looked at the API design, we split up tasks into those that the device could do and then the rest. The latter could easily be done in a browser-based application.

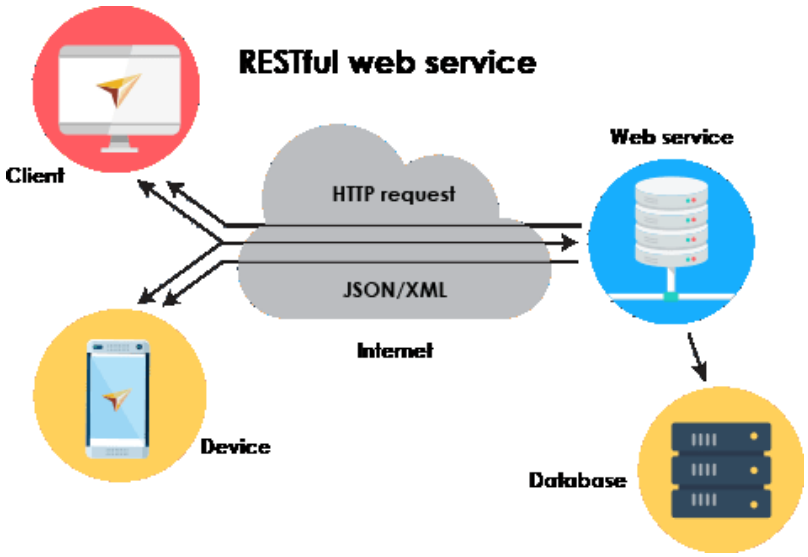
    - Of course, the users won't want to make raw API calls, and the flows taken to carry out an action may well be slightly different, but the basic data being manipulated is the same: The calls we've looked at would form the heart of the web application, just as they do the experience with the physical device.

    - Note that not every Internet of Things product needs a browser application. Perhaps the API is all you need, with maybe a static home page containing some documentation about how to call the API.

    - In the case of Clockodillo, however, we do want to have a set of web pages to interact with: Users should be able to look at their timers, assign descriptions, and so on.



- Drawbacks
  - Although web browsers do speak HTTP, they don't commonly communicate in all the methods.
  - In particular, they tend to support the following:
    - GET: Used to open pages and click on links to other pages.
    - POST: Used when submitting a form or to upload files.
  - But what about the lovingly crafted PUT and DELETE methods? Web browsers don't commonly support those.
    - One option is to make these calls in JavaScript, which can indeed support them.
    - Another is to "tunnel" the requests through a POST.
  - Alternatively, you could write the web application in an entirely different code base and interact with the main service through the API.
  - This can be a winning combination because it forces the human-facing code to use the same API that the device uses, increasing the amount of testing it receives and preventing the device-facing code from being neglected.
  - Whether you decide to follow that path would depend very much on your team's skill set.



- Designing a Web Application for Humans
  - Along with the text-based API you can also have an elegant and well-designed application for humans to interact with.
  - For example, the following figure shows a static login page.
    - This page is entirely for the convenience of a human. All the labels like "Your email address" and the help text like "Remember your password is case sensitive" are purely there to guide the user.
  - That's a simple example, but the following figure shows an even more extreme change.
    - The list of timers is highly formatted. The dates are formatted for humans. The duration of the timer and the status (in progress, completed, abandoned) are visualised with colours, progress bars, and a duration "badge". The page also links to other actions: The "Edit" button opens a page that allows access to the actions that change description, and so on. The menu bar at the top links to other functions to help users flow through the tasks they want to carry out.

The human-facing list of timers.

Real time Reactions

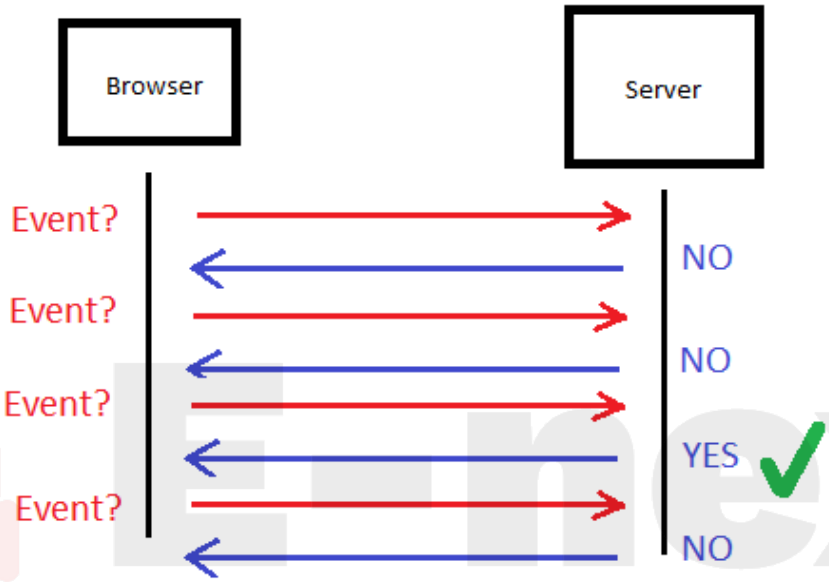
- We've looked at a traditional sort of API, where you make an HTTP request to the server and receive a response. This method has some disadvantages if you want a very responsive system.
  - To establish an HTTP request requires several round-trips to the server. There is the TCP "three-step handshake" consisting of a SYN (synchronise) request from the client, a SYN-ACK from the server to "acknowledge" the request, and finally an ACK from the client. Although this process can be near instantaneous, it could also take a noticeable amount of time.
  - The time taken to establish the connection may or may not matter. Any of the most powerful boards is able to run the connection in the background and respond to it when it's completed.
  - For a bare-bones board such as the Arduino, the current Ethernet/HTTP shields and libraries tend to block during the connection, which means that during that time, the microcontroller can't easily do any other processing.
  - If you want to perform an action the instant that something happens on your board, you may have to factor in the connection time. If the server has to perform an action immediately, that "immediately" could be nearly a minute later, depending on the connection time. For example, with the task timer example, you might want to register the exact start time from when the user released the dial, but you would actually register that time plus the time of connection.
- We look at two options here: polling and the so-called "Comet" technologies.
- And then, in the section on non-HTTP protocols, MQTT, XMPP, and CoAP offer alternative solutions.

- 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.

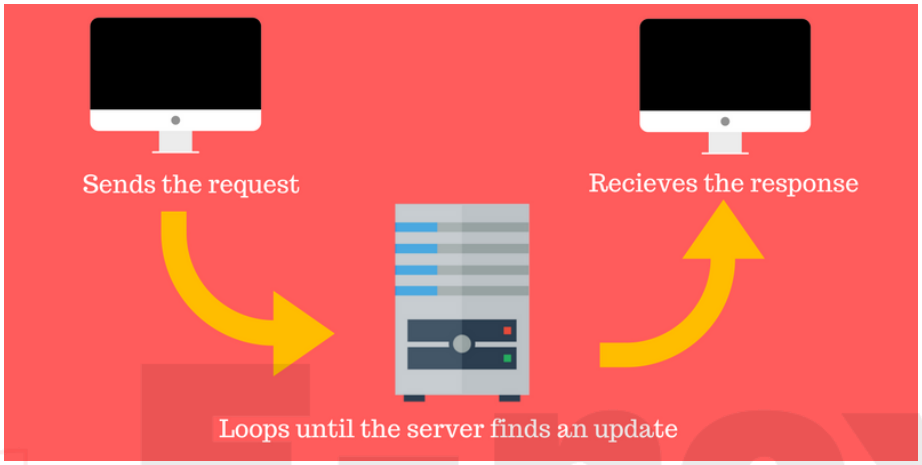
– As with many technologies, many of them were developed before the "brand" of Comet was invented; however, having a name to express the ideas is useful to help discuss and exchange ideas and push the technology forward.

– Long Polling (Unidirectional)

- The first important development was "long polling", which starts off with the client making a polling request as usual. However, unlike a normal poll request, in which the server immediately responds with an answer, even if that answer is "nothing to report", the long poll waits until there is something to say.
- This means that the server must regularly send a keep-alive to the client to prevent the Internet of Things device or web page from concluding that the server has simply timed out.
- Long polling would be ideal for the case of WhereDial: the dial requests to know when the next change of a user's location will be. As

soon as WhereDial receives the request, it moves the dial and issues a new long poll request.

- Of course, if the connection drops (for example, if the server stops sending keep-alive messages), the client can also make a new request.
- However, it isn't ideal for the task timer, with which you may want to send messages from the timer quickly, as well as receive them from the server.
- Although you can send a message, you have to establish a connection to do so. Hence, you can think of long polling as unidirectional.



– Multipart XMLHttpRequest (MXHR) (Unidirectional)

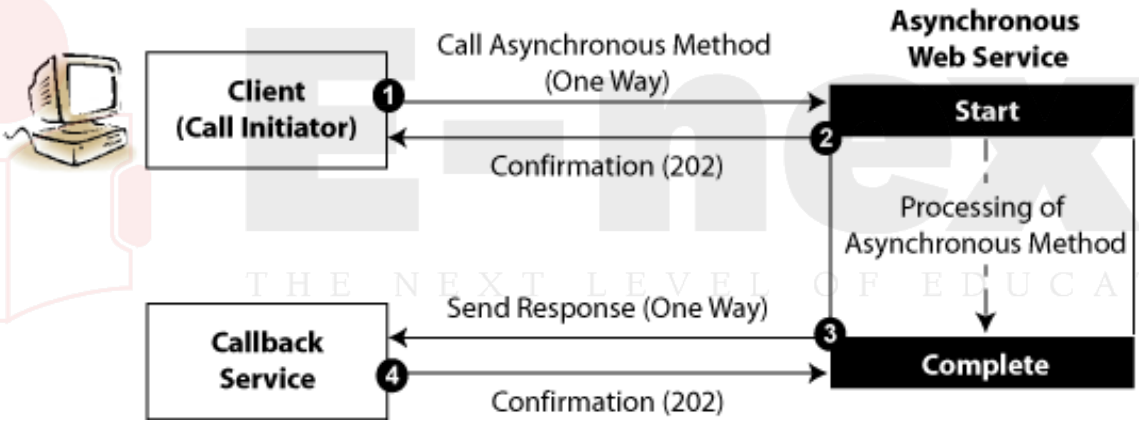
- When building web applications, it is common to use a JavaScript API called XMLHttpRequest to communicate with the web server without requiring a full new page load.
- From the web server's point of view, these requests are no different from any other HTTP request, but because the intended recipient is some client-side code, conventions and support libraries have developed to address this method of interaction specifically.
- Many browsers support a multipart/x-mixed-replace content type, which allows the server to send subsequent versions of a document via XHR.
- Note that XMLHttpRequest is a misnomer because there's no requirement to actually use XML at all.
- Using this content type is perhaps more sophisticated if you want to be able to receive multiple messages from the server.
- It is perfectly possible to simply long poll and create a new request on breaking the old one, but this means that you might miss a message while you're establishing the connection.
- In the example of WhereDial, this is unlikely; you're unlikely to change location first to Home and then to Work in quick succession.

- However, for an Internet of Things device such as Adrian's Xively meter, which tries to show the state of a Xively feed in real time, being able to respond to changes from the server almost immediately is the essential purpose of the device.

– Comet [contd…]
  – Implementations
    - The options described in the preceding section seemed to us to have most traction currently; however, as a fast-changing area with no absolute consensus as yet, the actual details of transports and limitations are bound to change.
    - It is worth paying attention to these transports as they develop. The Wikipedia page on Comet is a useful starting point for tracking the current state of play.
    - Let's look at support for these techniques on the three main platforms that you may need to consider for an Internet of Things application: the browser web app (if applicable), the microcontroller itself, and the server application.
    - On the browser side, it is often possible to abstract the actual transport using a library which chooses which method to connect to the server.
      – For example, it might use WebSockets if available; otherwise, it will fall back to MXHR or long polling. This capability is useful because each web browser currently has varying levels of support for the different techniques. There are well-known Comet libraries for jQuery and for Dojo.
      – In addition, many web servers have abstractions to support Comet techniques. Web::Hippie::Pipe provides a unified bidirectional abstraction for Perl web servers such as Twiggy, again using WebSockets if available, otherwise MXHR or long polling. You can find similar abstractions for node.js (JavaScript), Thin (Rails), jetty (Java), and so on.
      – There are also libraries for the microcontroller; however, they tend to support only one scheme. For example, several dedicated WebSockets libraries are available for Arduino. In fact, the fallback to different methods of interchanging data aren't really needed on the Arduino. Unlike the case of a desktop web app, with Arduino you don't have to worry about the users having different browsers because you'll be providing the firmware for the device.
  – Scaling
    - An important consideration is that all these Comet techniques require the client to have a long-term connection with the server.
    - For a single client, this is trivial. But if there are many clients, the server has to maintain a connection with each of them. If you run a

server with multiple threads or processes, you effectively have an instance of the server for each client.

- As each thread or process will consume system resources, such as memory, this doesn't scale to many clients.
- Instead, you might want to use an asynchronous web server, which looks at each client connection in turn and services it when there is new input or output.
- If the server can service each client quickly, this approach can scale up to tens of thousands of clients easily.
- There is a problem that each process on a typical Unix server has a maximum number of sockets, so you are restricted to that number of simultaneous clients.
- This, of course, is a good problem to have! When you hit that wall, you can look at load-balancing and other techniques that a good systems team will be able to apply to scale up the load. You also might be able to let your front-end proxy (Varnish or similar) do some of the juggling of persistent client connections.



Other Protocols

- Although HTTP is an extremely popular protocol on the Internet, it isn't ideally suited to all situations.
- Rather than work around its limitations with one of the preceding solutions, another option—if you have control of both ends of the connection—is to use a different protocol completely.
- There are plenty of protocols to choose from, but we will give a brief rundown of some of the options better suited to Internet of Things applications.
- There are plenty of protocols to choose from, but we will give a brief rundown of some of the options better suited to Internet of Things applications.
- EXTENSIBLE MESSAGING AND PRESENCE PROTOCOL
    - Another messaging solution is the Extensible Messaging and Presence Protocol, or XMPP.

- XMPP grew from the Jabber instant messaging system and so has broad support as a general protocol on the Internet.
- This is both a blessing and a curse: it is well understood and widely deployed, but because it wasn't designed explicitly for use in embedded applications, it uses XML to format the messages.
- This choice of XML makes the messaging relatively verbose, which could preclude it as an option for RAM-constrained microcontrollers.

Questions :

1. Write a short note on non-digital methods for prototyping the physical design.
2. Explain the different types of hinges and joints used in laser cutting.
3. Explain the different types of 3D printers.
4. Compare laser cutting, 3D printing and CNC milling in terms of working mechanism, choosing the device, and software.
5. What is scraping and what legalities are involved in scraping?
6. What standards should be considered when implementing an API?
7. How can CURL be used to test the API? Explain with suitable examples.
8. Which other protocols can be used for IoT besides HTTP? Explain.