Chapter 1
User Experience Design

Ms.T.Kavitha
Assistant Professor
CSE,KITS

What's Different About User Experience Design for the Internet of Things?

- User experience (UX) design and human–computer interaction (HCI) emerged in a world of desktop computers
- Many of our interactions now take place on mobile phones, tablets, ereaders, and smart TVs

- We’re still figuring out the best ways to design for new devices and experiences.
- Interactions can happen in a wide variety of contexts, especially for mobile devices.
- They can happen on a variety of scales from tiny wrist-tops, to smartphones, to TV user interfaces (UIs) viewed from 10 feet away.

What's Different About User Experience Design for the Internet of Things?

- The "Internet of Things" (IoT) refers to the growing range of everyday objects acquiring connectivity, sensing abilities, and increased computing power.
  - Connected home technology (such as thermostats, lighting, and energy monitoring)
  - Wearables (such as activity/fitness trackers and "smart" watches)

What's Different About User Experience Design for the Internet of Things?

- Medical/wellness devices (such as bathroom scales and blood pressure monitors)
- Connected cars (which may provide access to smartphone apps via dashboard controls, engine diagnostics, and automatic alerting of authorities in case of a crash)
- Urban systems (such as air quality sensors, city rental bikes, and parking meters/sensors)

How Is UX for IoT Different?

- Design of the Internet of Things (IoT) raises all the challenges of cross-platform design, and more.
- UX is a holistic term referring to a wide range of design disciplines involved in creating systems that are useful, usable, and pleasurable to use.
- The user interface is not the (design) solution, but instead is the medium through which users interact with the solution.

FIGURE 1-2
The Lockitron connected door lock is one of a huge number of connected devices with no screen (image: Lockitron)

Issues

- The challenges of distributing functionality across multiple devices
- How the focus of the UX is increasingly in the service
- Whether we are ready for the real world to start behaving like the Internet
- How the ways devices connect to the network affects the UX
- How multiple devices create more complexity for the user to understand
- How controlling distributed devices is similar to programming
- How what seem like simple systems can rapidly become complex
- The problems of having many different technical standards
- How data is at the core of many IoT services
- The layers of UX thinking required to create a successful IoT product: from UI and interaction design all the way down to the platform

How Is UX for IoT Different?

- Designing for IoT comes with a bunch of challenges that will be new to designers accustomed to pure digital services.
- How tricky these challenges prove will depend on:
  - The maturity of the technology you're working with
  - The context of use or expectations your users have of the system
  - The complexity of your service (e.g., how many devices the user has to interact with)

FUNCTIONALITY CAN BE DISTRIBUTED ACROSS MULTIPLE DEVICES WITH DIFFERENT CAPABILITIES

- IoT devices come in a wide variety of form factors with varying input and output capabilities.
- Some may have screens, such as heating controllers or washing machines. Some may have other ways of communicating with us, such as flashing LEDs or sounds.

FIGURE 1-4
The GlowCaps connected pill bottle lid uses light and sound notifications to remind the user to take medication (image: GlowCaps)

FIGURE 1-3
The Honeywell evohome connected radiator valve has a basic LCD screen (image: Honeywell Environmental Controls)

FUNCTIONALITY CAN BE DISTRIBUTED ACROSS MULTIPLE DEVICES WITH DIFFERENT CAPABILITIES

- Some may have no input or output capabilities at all and are unable to tell us directly what they are doing.
- Despite the differences in form factors, users need to feel as if they are using a coherent service rather than a set of disjointed UIs.
- It's important to consider not just the usability of individual UIs but interusability: distributed user experience across multiple devices

FIGURE 1-5
The Nest Learning Thermostat can be controlled by the on-device UI, a smartphone app, or a web app (image: Nest)

THE FOCUS OF THE USER EXPERIENCE MAY BE IN THE SERVICE

- When we talk about IoT, we tend to focus on the devices, particularly those with striking or novel forms.
- But the behavior of the device might be generated by a program that lives on another device on the network.
- This means that the service around a connected device is often just as critical in delivering the user experience, if not more so, than the device itself.

FIGURE 1-6

Hong Kong's Octopus payment service can be used with an NFC phone as well as a smart card (Octopus reader on KMB bus by Ka890, CC licence via Wikimedia Commons; Octopus app: Octopus Cards Ltd.)

WE DON'T EXPECT INTERNET-LIKE GLITCHES FROM THE REAL WORLD

- It's frustrating when a web page is slow to download or a Skype call fails. But we accept that these irritations are just part of using the Internet.
- By contrast, real-world objects respond to us immediately and reliably.
- When we interact with a physical device over the Internet, that interaction is subject to the same latency and reliability issues as any other Internet communication.
- So there's the potential for delays in response and for our requests and commands to go missing altogether. This could make the real world start to feel very broken.
- Imagine if you turned your lights on and they took two minutes to respond, or failed to come on at all.

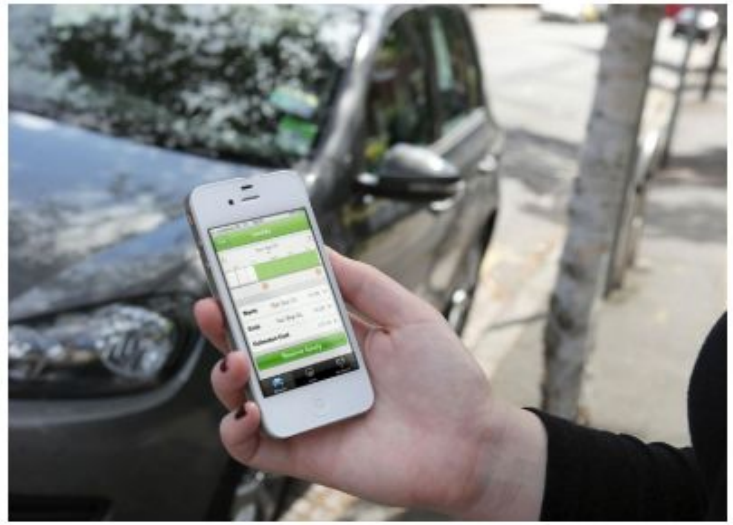


FIGURE 1-7

When you book a Zipcar online, the service sends details of the reservation to the car; swiping a smart card authenticates you as the person who made the booking (image: Zipcar)

IOT IS LARGELY ASYNCHRONOUS

- When we design for desktops, mobiles, and tablets, we tend to assume that they will have constant connectivity.
- Well-designed mobile apps handle network outages gracefully, but tend to treat them as exceptions to normal functioning.
- We assume that the flow of interactions will be reasonably smooth, even across devices.
- If we make a change on one device (such as deleting an email), it will quickly propagate across any other devices we use with the same service.

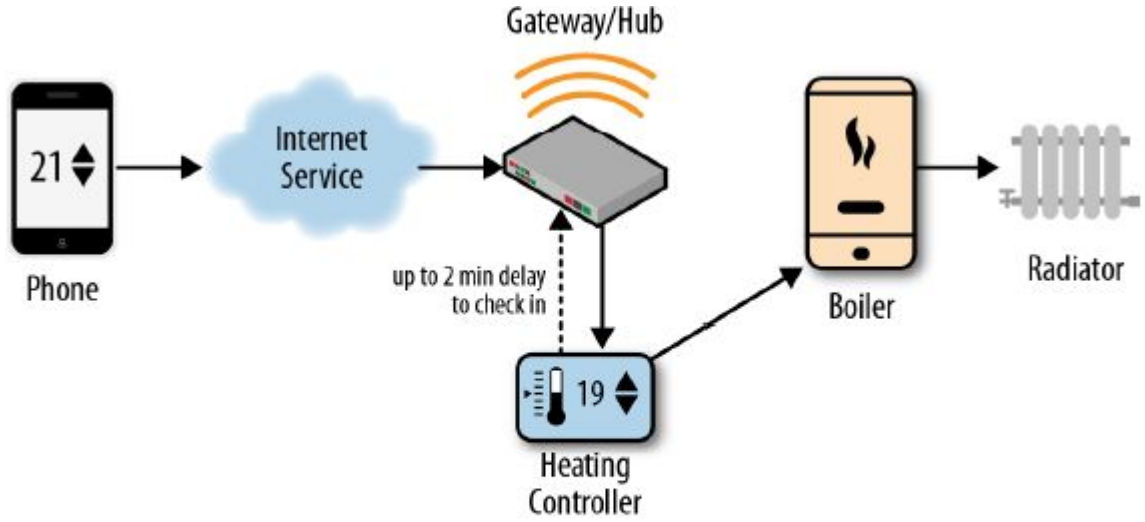


FIGURE 1-8

Schematic of heating system with app and controller giving different status information

- Many connected devices run on batteries, and need to conserve electricity.
- Maintaining network connections uses a lot of power, so they only connect intermittently.
- This means that parts of the system can be out of sync with one another, creating discontinuities in the user experience.
- These discontinuities won't always be noticed: sometimes the delays will be very short, and sometimes users won't be around to notice them—for example, when they are turning on a remote device.
- So the UX may feel synchronous, even if the service technically isn't. But when we do notice, the UX may feel quite disjointed.

CODE CAN RUN IN MANY MORE PLACES

- The configuration of devices and code that makes a system work is called the system model.
- We don't need to understand the technical architecture of a conventional Internet service, like Amazon, in order to use it successfully.
- Instead, we form a conceptual model of what Amazon does and how it works that's good enough to help us understand what to do.
- But as a consumer of an IoT service right now, you can't always get away from some of this technical detail.

IoT Service

- A typical IoT service is composed of:
  - One or more *embedded* devices
  - An Internet service
  - Perhaps a *gateway* device (a separate device needed to connect some embedded devices to the Internet)
  - One or more mobile or web apps for the user to interact with the service via a mobile, tablet, or PC
- Compared to a conventional web service, there are more places where code can run. There are more parts of the system that can, at any point, be offline. Depending on what code is running on which device, some functionality may at any point be unavailable.

- For example, imagine you have a connected lighting system in your home. It has controllable bulbs or fittings, perhaps a gateway that these connect to, an Internet service, and a smartphone app to control them all.
- You have an automated rule set up to turn on some of your lights at dusk if there's no one home. If your home Internet connection goes down, does that rule still work? If the rule runs on the Internet service or your smartphone, it won't. If it runs on the gateway, it will.
- As a user, you want to know whether your security lights are running or not. You need to understand a little about the system model to understand which devices are responsible for which functionality, and how the system may fail.
- It would be nice if we could guarantee no devices would ever lose connectivity, but that's not realistic.
- System designers have to ensure that important functions (such as home security alarms) continue to work as well as possible when parts go offline and make it as easy as possible for users to understand what's happening, and recover from any problems.

FIGURE 1-9
The Philips Hue system consists of connected bulbs, a gateway, an Internet service, and a smartphone app (image: Philips)

DEVICES ARE DISTRIBUTED IN THE REAL WORLD

- The shift from desktop to mobile computing means that we now use computers in a wide variety of situations. Hence, mobile design requires a far greater emphasis on understanding the user's needs in a particular context of use.
- IoT pushes this even further: computing power and networking is embedded in more and more of the objects and environments around us.
- For example, a connected security system can track not just whether the home is occupied, but who is inside, and potentially video record them. Hence, the social and physical contexts in which connected devices and services can be used are even more complex and varied.

REMOTE CONTROL AND AUTOMATION ARE PROGRAMMING-LIKE ACTIVITIES

- User interfaces based on direct manipulation "depend on visual representation of the objects and actions of interest, physical actions or pointing instead of complex syntax, and rapid incremental reversible operations whose effect on the object of interest is immediately visible.
- This strategy can lead to user interfaces that are comprehensible, predictable and controllable."
- Direct manipulation is successful because interface actions are aligned with the user's understanding of the task. They receive immediate feedback on the consequences of their actions, which can be undone

FIGURE 1-10
An image by the designer Susan Kare, created in MacPaint 1.0: an early example of a popular direct manipulation interface (image: Wikipedia)

Example

- For example, you might set up a home automation rule to turn on a video camera and raise the alarm when the house is unoccupied and a motion sensor is disturbed. Or you might unlock your porch door from your work computer to allow a courier to drop off a parcel.
- Unlocking the door remotely is an easier action to comprehend. But we are distanced from the consequences of our actions, and this poses other challenges. Can we be sure the door was locked again once the parcel had been left? A good system should send a confirmation, but if our smartphone (or the lock) lost connectivity, we might not receive this.

- Both of these break the principles of direct manipulation. To control things that happen in the future, you must anticipate your future needs and abstract the desired behaviour into a set of logical conditions.
- It is a much harder cognitive task than a simple, direct interaction. That's not necessarily a bad thing, but it may not be appropriate for all users or all situations. It impacts usability and accessibility.

COMPLEX SERVICES CAN HAVE MANY USERS, MULTIPLE UIS, MANY DEVICES, MANY RULES AND APPLICATIONS

- A simple IoT service might serve only one or two devices (e.g., a couple of connected lights). You could control these with a very simple app.
- But as you add more devices, there are more ways for them to coordinate with one another. If you add a security system with motion sensors and a camera, you may wish to turn on one of your lights when the alarm goes off. So the light effectively belongs to two functions or services: security and lighting. Then add in a connected heating system that uses information from the security system to know when the house is empty.
- And assume that there are several people in the house with slightly different access privileges to each system.

- For example, some can change the heating schedule, some can only adjust the current temperature. Some have admin rights to the security system, some can only set and unset the alarm. What started out as a straightforward system has become a complex web of interrelationships
- For a user, understanding how this system works will become more challenging as more devices and services are added. It will also become more time consuming to manage

MANY DIFFERING TECHNICAL STANDARDS MAKE INTEROPERABILITY HARD

- Getting devices talking to one another is a big enough challenge, as there are many different network standards.
- Lack of common technology standards is causing headaches.
- There are also some emerging platforms that seek to aggregate devices from a number of manufacturers and enable them to interoperate.
- For the near future, the onus will be largely on the consumer to research which devices work with their existing devices before purchasing them.
- The service that promises to unify all your connected devices may not support some of their more advanced or unique functions: you might be able to turn all the lights on and off but only dim some of them.

Examples

- Devices within the same manufacturer's ecosystem, such as Withings, will work together. But this is the only given. In the case of Withings, this means that devices share data with a common Internet service, which the user accesses via a smartphone app.
- Apple's Airplay is an example of a proprietary ecosystem in which devices talk directly to one another.
- The connected home platform SmartThings supports a range of network types and devices from manufacturers such as
  - Schlage and Kwikset (door locks);
  - GE and Honeywell (lighting and power sockets);
  - Sonos (home audio); and
  - Philips Hue, Belkin, and Withings (connected home products)

FIGURE 1-11
The Withings ecosystem of devices (images: Withings)

IoT IS ALL ABOUT DATA

- Networked, embedded devices allow us to capture data from the world that we didn't have before, and use it to deliver better services to users.
- For example, drivers looking for parking spaces cause an estimated 30% of traffic congestion in US cities. Smart parking applications such as Streetline's Parker use sensors in parking spaces to track where spaces are free, for drivers to find via a mobile app.
- Networked devices with onboard computation are also able to use data, and in some cases act on it autonomously. For example, a smart energy meter can easily detect when electrical activity is being used above baseload. This is a good indicator that someone is in the house and up and about. This data could be used by a heating system to adjust the temperature or schedule timing.

FIGURE 1-13
Streetline's Parker app (image: Streetline)

A Design Model for IoT

The two most visible and tangible forms of design for IoT are:

- The UI/visual design
- The industrial design of the physical hardware

The UI/visual design

- For example, the screen layout and look and feel of the web or mobile apps, or devices themselves.
- UIs don’t have to be visual; they can use audio, haptics, and other channels. But it’s rare for a service to have no screen-based UI at all

FIGURE 1-15

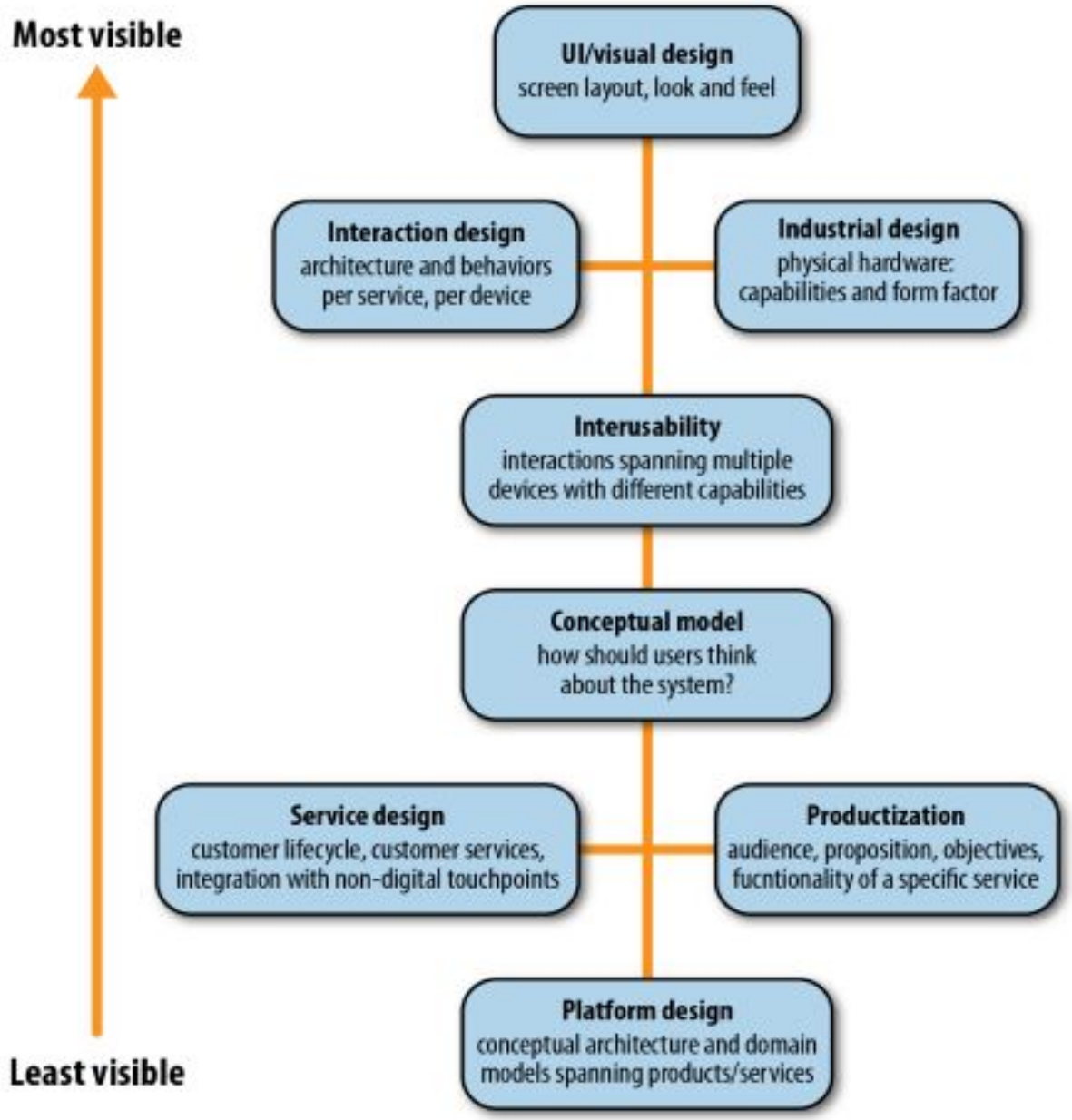
The Philips Hue UI allows users to change the color of light emitted by an LED bulb (image: Philips Communications)

The industrial design of the physical hardware

- The form factor, styling, and capabilities of the connected devices themselves.
- UI and industrial design are important but not the whole picture. The UX is not just shaped by what the user can see or encounter directly. To create a valuable, appealing, usable, and coherent IoT service, we have to consider design on many different layers.

FIGURE 1-16
The Nest Learning Thermostat has a striking industrial design (image: Nest)

- There are many different facets of design involved in delivering a good UX for an IoT service.
- UI, interaction design, and interusability need to be thought about together. UX design at the platform layer will emerge as a need once you start adding multiple devices to a service.
- A good overall product requires integrated thinking across all these layers. A stunning UI means nothing if your product concept makes no sense. A beautiful industrial design may sell products in the short term but can't mask terrible service.
- Depending on the type and complexity of your service, layers will require more or less of your time.



UI/VISUAL DESIGN

- UI/visual design refers to screen layout, visual styling, and look and feel on a device. This is the form that a device interface takes.
- Not all UIs are visual, of course: for a gestural or audio interface, the equivalent function might be defining the aesthetics of the gestures or voice.

FIGURE 1-18

Style tiles (visual concepts) and UI modules for a redesign of the IDA Institute's website (*idainstitute.com*; images: Halei Liu and Hans Pelle Jart for *Kontrapunkt.com*)

INTERACTION DESIGN

- Interaction design is the design of device behaviors.
- Interaction designers shape the sequences of actions between the user and the device needed to achieve particular goals or activity.
- They also determine how to organize the user-facing functions of the device.
- For example, a heating controller might have several modes, such as schedule on/off or frost protection, and some hierarchical functions, such as schedule setting. The organization of these functions defines how easy or otherwise it may be for users to find their way around them.

- Interaction design is closely aligned to UI design in the sense that the two are usually done in tandem and often by the same people. But interaction design is primarily concerned with behaviours and actions, whereas UI/visual design is concerned with layout and aesthetics.
- Typical outputs for interaction design might include user flows, low-medium fidelity interactive prototypes, and for a visual UI, screen wireframes.

FIGURE 1-19
Device and app interaction design concepts for Hackaball, a programmable ball for children (*www.hackaball.com*; images: Map and Made by Many)

INTERUSABILITY

- Interusability is a relatively new term. It refers to the additional considerations of designing interactions that span multiple devices.
- The goal is to make the overall experience feel like a coherent service, even when the devices involved may have quite different form factors and input/output capabilities.
- Interusability isn’t a separate set of design activities.
- It’s an extra set of considerations to be addressed in tandem with interaction and UI design.

INTERUSABILITY

The key differences to a single device UX design process would typically be:

- Specifying which functionality belongs on each device
- Creating design guidelines that span multiple device types
- Designing cross-device user flows for key interactions
- Designing multiple device UIs in parallel

INDUSTRIAL DESIGN

- Industrial design refers to the aesthetic and functional design of the physical hardware in the service: the choice of form, materials, and capabilities it may have.
- Connected devices contain electronic circuitry and radio antennae, which impose particular requirements on the industrial design.
- Devices can also have input and output capabilities, which require collaboration between industrial designers and UI/interaction design/interusability.

FIGURE 1-20
Hardware prototype and color/material/form explorations for Hackaball, a programmable ball for children (*www.hackaball.com*; images: Map and Made by Many)

SERVICE DESIGN

- Service design is an emerging discipline that addresses this holistic view of user experience. It looks at the whole lifespan of a user's experience with a service, provides a view of all the components of the user experience, and specifies how these function together as a coherent whole. The service experience is critical to the success of an IoT product,

In addition to device interactions, it might include:

- Customer support interactions
- Instructional guides
- Marketing or sales materials
- In-store experiences
- Email communications and notifications
- The UX of software updates and rolling out new functionality

CONCEPTUAL MODEL

- The conceptual model is the understanding and expectations you want the user to have of the system.
- What components does it have, how does it work, and how can they interact with it? It’s the mental scaffolding that enables users to figure out how to interact with your service.
- Regardless of whether a conceptual model is designed or not, users will form one. If they get it wrong, they’ll struggle to use the system.
- IoT services are often inherently complex systems. You can create a clear conceptual model through careful system and interaction design and supporting documentation.
- You want users to feel in control from the start—they should feel confident that they will be able to use the system, even if they don’t understand all the details yet.

PRODUCTIZATION

- Productization is the activity of defining a compelling product proposition. It addresses the audience, proposition, objectives, and overall functionality of service (and often its business model).
- Does your product solve a real problem for a real audience? Is it presented so that they understand that? Does it appeal to them? This isn’t always the domain of the UX designer on a project, but it’s the underpinnings of good UX.
- All the frontend design in the world won’t make a killer product unless it does something of value for people, in a way that appeals to them and makes sense.

PLATFORM DESIGN

- A platform is a software framework.
- It takes care of low-level details to help developers build applications more easily.
- At their most basic, IoT platforms make it easy to put data from connected devices onto the Internet.
- Slightly more advanced platforms may provide frameworks to enable different types of devices to interoperate.
- A software platform will aim to solve many technical issues, many of which may not directly have an impact on the UX.
- But some more advanced platform functionality is very much to do with UX.

Example

- For example, a platform like Hue or Withings may provide standard ways to:
- Discover new devices and applications
- Add devices and applications onto the system
- Manage devices and users
- Manage how devices share data

These are basic building blocks for the UX.

- A more complex platform might also provide ways of organizing and coordinating multiple devices.
- For example, if a user adds a light to an existing home system, they might reasonably expect the system to know that it should control it along with their other lights and/or offer it as part of the security system.
- It’s not important to make it talk to the toaster. That may be common sense to a human.
- But the system won’t know that unless this kind of logic is encoded in the platform.
- But once your system has multiple, interconnected devices, there will be design challenges that will require platform logic to solve. Designers should get involved in shaping platforms to ensure they support good higher-level UX.

Summary

- UX for connected devices is not just about UI and interaction design.
- It also requires designers to think about interusability, industrial design, service design, conceptual models, productization, and platform design.

Embedded devices often save power by connecting only intermittently	... which means parts of the system can be out of sync, creating discontinuities in UX
Latency on the Internet is out of your control (and reliability is not 100%)	... which means although we expect physical things to respond immediately and reliably, this might not happen
Code can run in many more places	... which means users have to engage with the system model to predict how it will work if parts are offline
Devices are distributed in the real world	... which means social and physical context of use is complex and varied
Functionality can be distributed across multiple UIs	... which means designers need to consider not just usability but interusability
Much of the information processing happens in the Internet service	... which means the service experience is often equally or more important than the single device UX
Remote control and automation are programming-like activities	... which means IoT breaks direct manipulation—the basis of most successful consumer UXes
Many differing technical standards	... which means getting things to work together is hard
Complex services can have many users, many UIs, many devices, many rules and applications	... which means understanding and managing how they all interrelate can be extremely difficult. Users will turn off if admin becomes too onerous.
IoT enables us to capture and act on data we didn't have before	... which means designers need to understand how to use information as a design material