networked objects

Design Kit

Get started with exploring, designing and developing prototypes of Networked Objects within minutes.

More information and an electronic version can be found on:

http://networkedobjects.info

This design kit is targeted towards interaction designers who want to get started with Networked Objects, understand their composition, learn to structure a successful design process and build prototypes quickly.

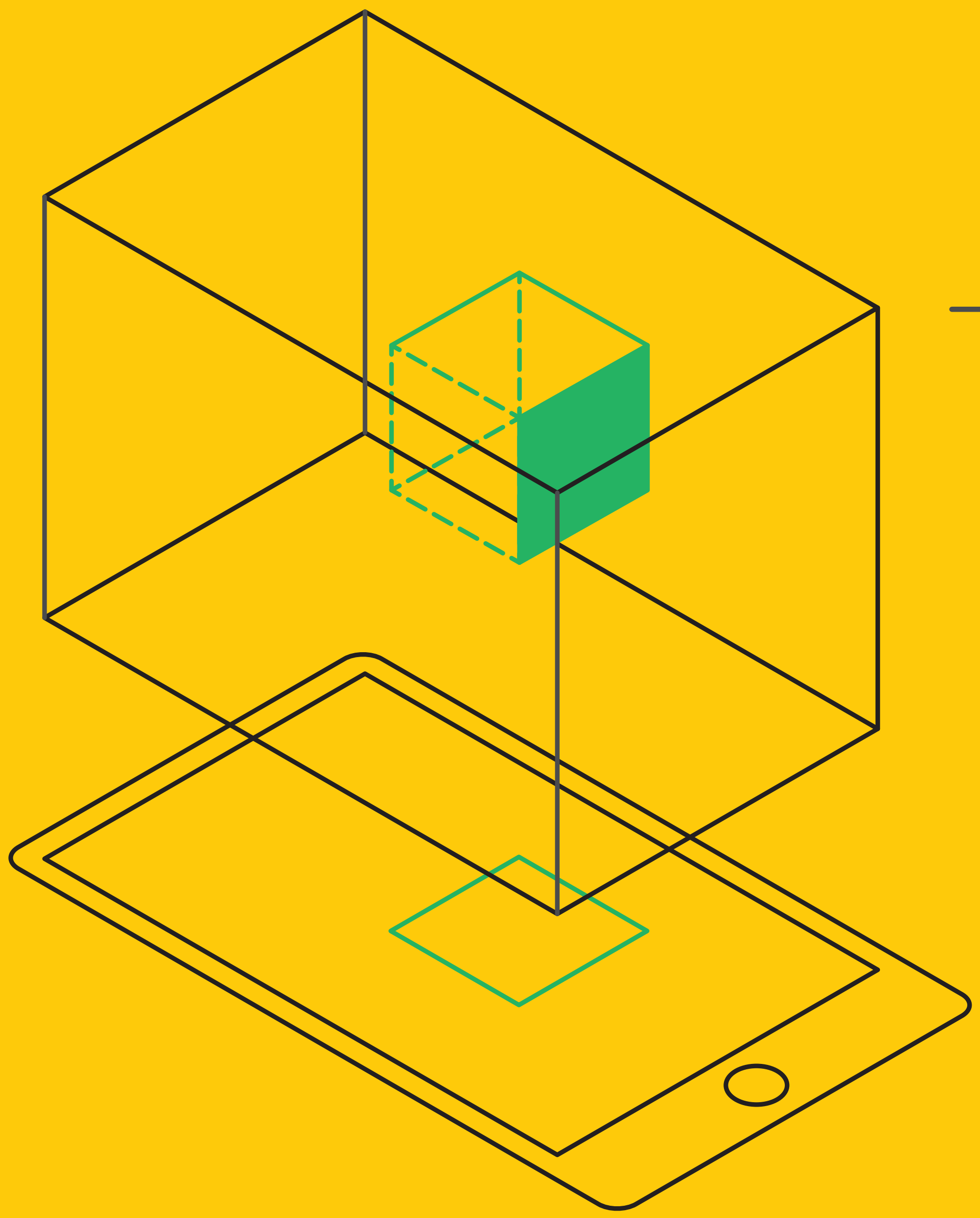
The kit includes a set of event-logic-action cards and two posters with the following content:

An introduction to the concept of Networked Objects and to how interactions will transform in the future.

A look at the individual components of a Networked Object, its feedback loop and design aspects.

An introduction to the design method «Networked Exploration» that helps to explore a design space and build networked prototypes quickly.

A brief elaboration on how prototypes of Networked Objects can be built using several prototyping tools.



What are Networked Objects?

The computerized world of today is heavily based on the availability of mobile devices, like smartphones and tablets, on which the digital experience is given by individual applications that we install on these devices. The screen, which is the central part of the device, is the canvas where the human and the artificial system find common ground and communicate. While the application paints the screen with graphical user interface elements, the human responds by interacting with these virtual elements on the screen. All these interactions happen in the digital space which requires the human to adapt to a limited, artificial system.

In the future, that same experience will be given by the individual networks of objects that reside in our environment. As these objects are located in the physical space, this space becomes the canvas. Rather than downloading applications that each solve a certain problem, we would organize objects in the space and configure them to networks that bring us the desired benefit. The hidden information systems would then paint the space using these objects. In that sense, artificial systems would adapt to us rather than the other way around.

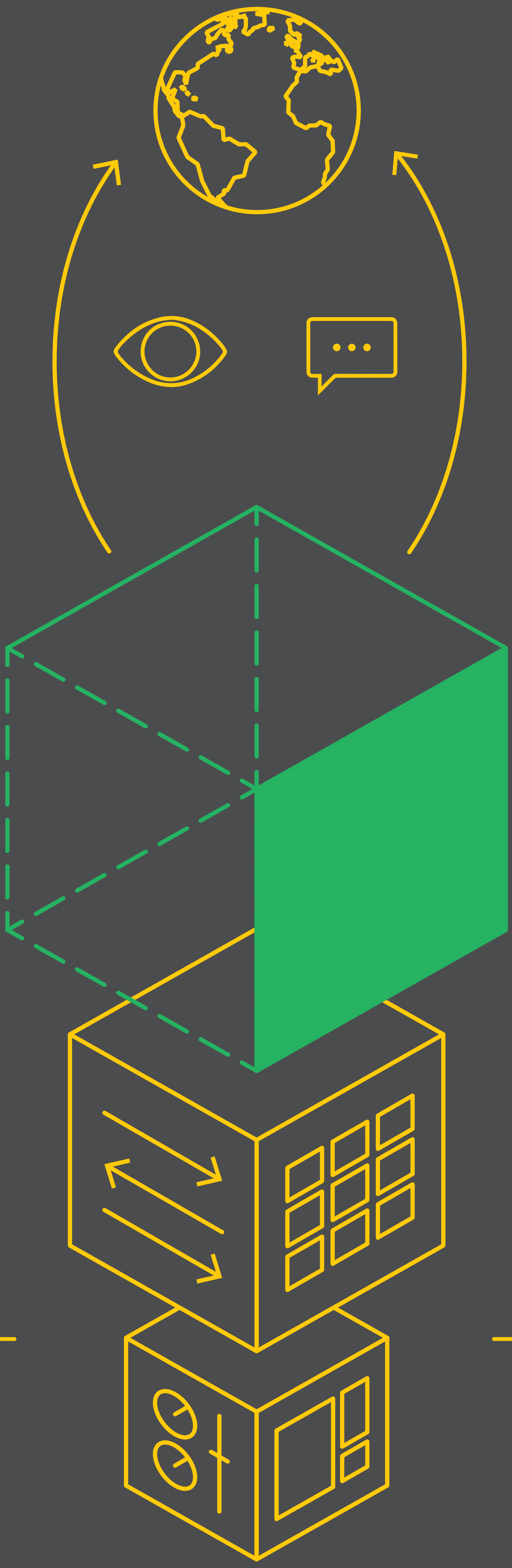
In contrast to the traditional Internet where applications interface with the world through screens and keyboards, the future user will interact with the world using interconnected, physical objects. Wherever we used computers and smartphones to access the digital world in the past, there will be objects that connect the two worlds together and allow us to interact with them directly. These «Networked Objects» will enclose the digital world by providing natural and physical interfaces. Complex computer systems will be hidden and encapsulated in objects that are limited in functionality, specifically located and only accessible in the physical world. Thus, the digital world will be embedded in the physical world, and the combined interface will be more powerful than everything available before.

What is the Anatomy of a Networked Object?

Networked Objects are located at the border between the physical and digital space. They do not only interact with both spaces, but also link them together. Therefore, a Networked Object always includes software development and the building of a physical object. On the one hand, the Networked Object is responsible for translating events that happen in the physical space to input for the information system in the digital space. On the other hand, the Networked Object also translates output from the information system to actions that are executed in the physical space.

Technically, the Networked Object can be divided into two main components the «event-action interface (EAI)» and the «input-output logic (IOL)». The EAI consists of several sensors, which can measure and detect events and actuators that can execute actions. The IOL represents the Networked Object in the digital space. Events emitted by the EAI are managed and further forwarded by the IOL. The separation is straight forward as the EAI is the hardware needed to sense and act upon the environment, while the IOL is the software that deals with the communication with other Networked Objects and information systems.

The observe-notify feedback loop of a Networked Object is the basis for the EIA and IOL which together give a foundation for the four design aspects: interactivity, functionality, configurability and topology.



Observe

Networked Objects observe the environment and the user and detect events. These events then get translated into data which is the input for the information system.

Notify

Output generated by an information system gets translated by the Networked Object to an action. In most cases the actions are a kind of notification that informs the users about changes in the information system.

Input-Output Logic

The digital part of a Networked Object manages the communication with information systems over the Internet. Observed events get forwarded to the information system, processed and stored. In return, the system sends actions that need to be executed by the object.

Event-Action Interface

Every Networked Object has a physical form that includes sensors and actuators that are able to sense and act upon the environment. Sensor readings are translated into events and actions received from the digital side are transformed into instructions for the actuators.

Interactivity

The interaction between a single Networked Object or a system of many and the user should take the form of a conversation in which both sides learn from each other and understand each other's behaviours.

Functionality

Functionality can be distributed among multiple objects, leaving the individual Networked Object simple and comprehensible. The objects can also be seen as building blocks that, when put together, solve a higher task.

Configurability

Networked Objects offer configurability by allowing the user to place, attach and integrate them with the physical space and by including a few physical controls for fine adjustments.

Topology

Tailored packages and sets of Networked Objects give the user examples of meaningful topologies, which can then be extended with additional packages or single products.

How to Design and Prototype Networked Objects

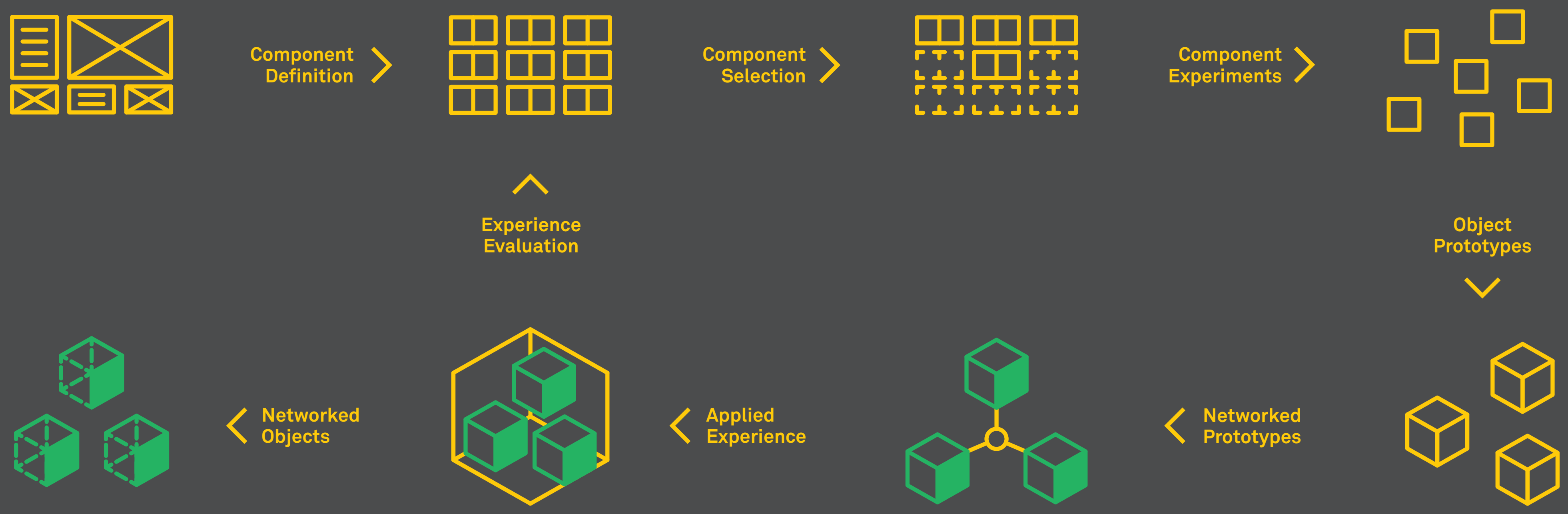
The design method «Networked Exploration» supports the designer in developing his design space through multiple iterations into Networked Objects.

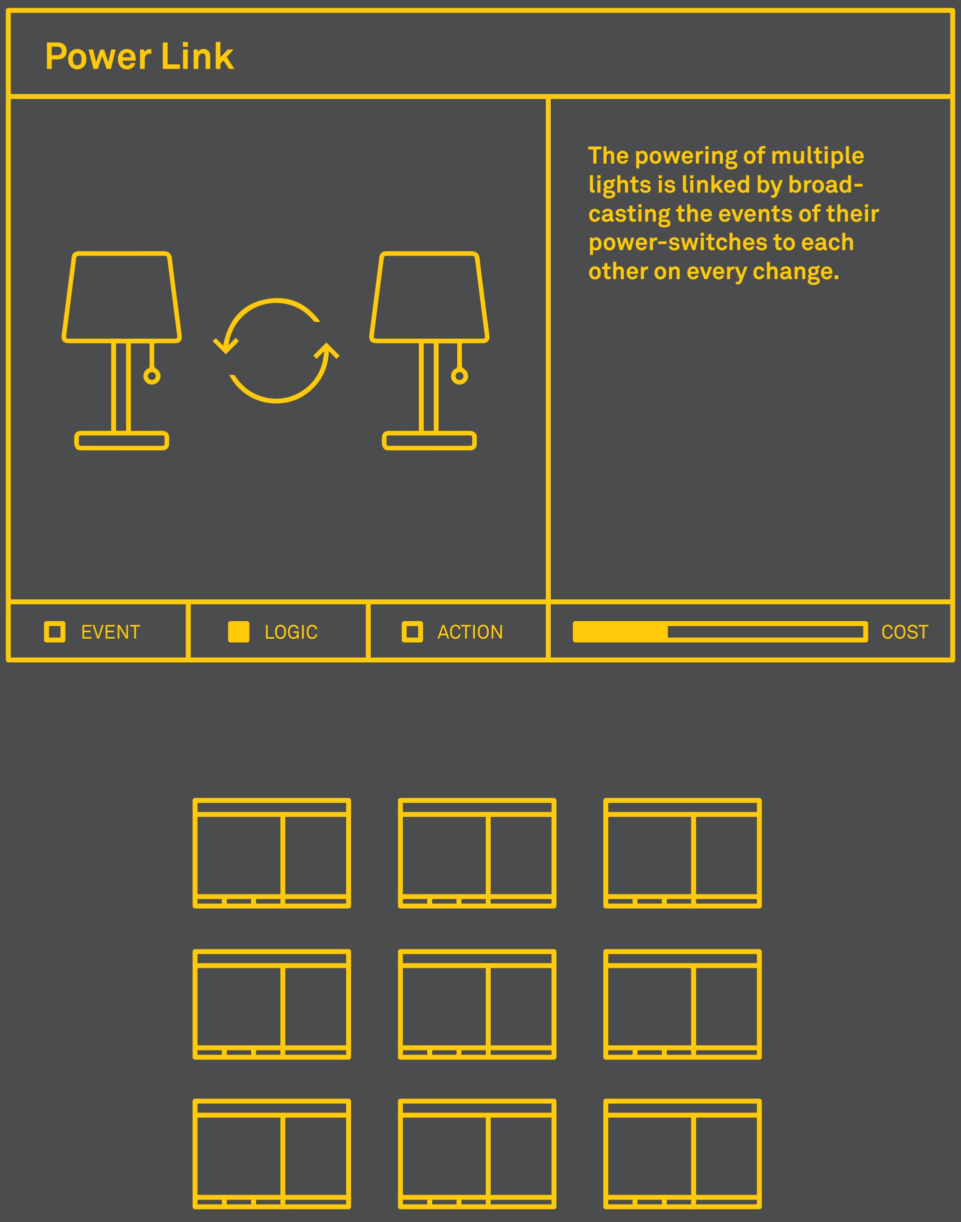
To help uncover the necessary components and work units, the design space is first organized using the «Event-Logic-Action Board (ELAB)». The ELAB is inspired by the project management technique Kanban which helps organizing a team and structuring a development process. Just like Kanban, the ELAB also uses cards that represent a unit of work, which in our case would be an event, logic or action component.

Every designer starts with a defined design space that includes background research, mood-boards, videos and notes to a certain topic. From this material he defines multiple, different event, logic and action components that his Networked Objects could be made of. By noting down each individual component onto a card, he creates the ELAB. After selecting a first set of events, logics and actions, the designer tests the concepts in individual experiments. After that, he will combine the experiments to at best two or more object prototypes. In the next step, the designer networks the prototypes together, thus creating a working system that can be experienced by a user.

The system can now be applied to the target environment and evaluated. The gained insights and learnings are merged with the design space and used to refine the concept and update the ELAB. Then, a new loop begins in which the designer repeats the steps mentioned before. The overall idea is to go through the whole loop as many times as possible and to master each stage as fast as possible. In later iterations more attention can be given to details in order to reach a higher quality.

If the result is satisfying the process can be completed by finalizing the prototypes into self-contained Networked Objects.





Power Link
The powering of multiple lights is linked by broad-casting the events of their power-switches to each other on every change.
EVENT
LOGIC
ACTION
COST

Sketch

A rough sketch should visualize figuratively how the aspired method of observing, processing or executing is implemented and applied.

Description

Additionally to the sketch, a short description points out all important information and needed resources in the form of an instruction.

Category

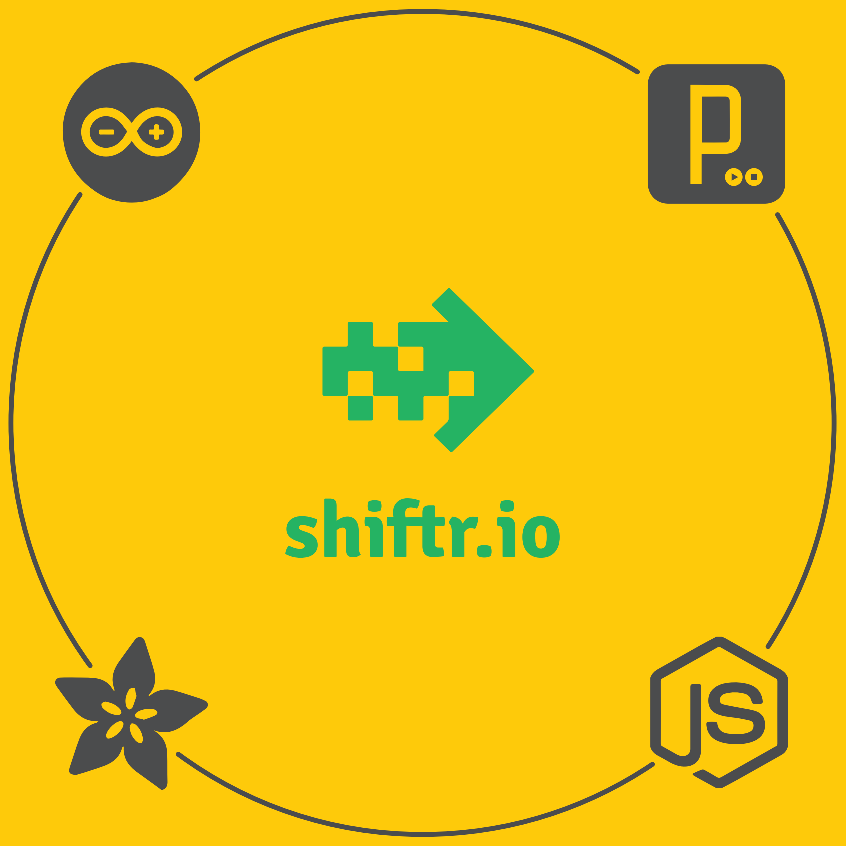
While events symbolize sources of change that are tracked by the object, actions are the object's capability to change the environment. Both are connected by logic components that process and correlate multiple events and decide on actions. For simplification the logic component includes both the input and output logic.

Cost

Every component needs a certain amount of time and money to be built. The cost bar should reflect the need for those resources to help with planning bigger projects or managing time constraints.

How to Network Hardware and Software Together

Building the EAI is the easiest when using an electronics prototyping platform like «Arduino». Numerous components and libraries allow a designer to quickly sketch an electronic circuit and give him the possibility to interact with the user or environment. If the project is simple enough, the IOL can be programmed directly on the Arduino. If the project gains in complexity, the data can be forwarded to a computer where it gets processed by a higher order application written for example in «Processing». Processing is a very good tool when it comes to developing a small, central information system that manages data coming in from multiple objects. In order to build bigger systems, platforms like «node.js», that have a rich ecosystem of libraries and contributors are recommended. Finally, the code that is distributed throughout these machines needs to be connected using an intermediary service like «shiftr.io».



Arduino

«Arduino is an open-source electronics platform based on easy-to-use hardware and software. It's intended for anyone making interactive projects.»

https://arduino.cc

Processing

«Processing is a programming language, development environment, and online community. Since 2001, Processing has promoted software literacy within the visual arts and visual literacy within technology.»

https://processing.org

Adafruit

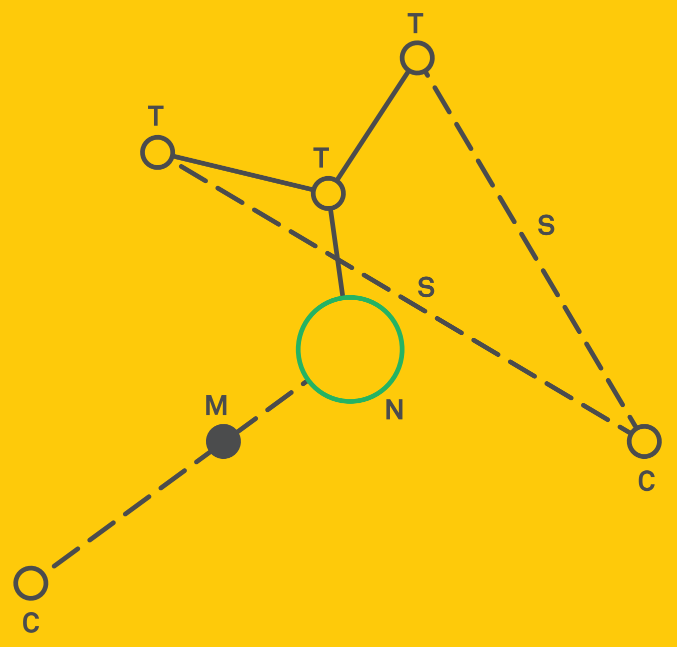
«Adafruit was founded in 2005 by MIT engineer, Limor «Ladyada» Fried. Her goal was to create the best place online for learning electronics and making the best designed products for makers of all ages and skill levels.»

https://adafruit.com

node.js

«Node.js is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.»

https://nodejs.org



shiftr.io

«With shiftr.io the process of interconnecting objects, devices and apps becomes more accessible and less complex. Regardless of wether you are building an interactive installation, prototyping the next connected product or simply playing around with new technologies, shiftr.io lets you add connectivity to your project in an early stage.»

https://shiftr.io

MQTT

The shiftr.io platform is built on top of the MQTT protocol, which is a publish-subscribe protocol. Clients (C) always connect to a central broker, which in the case of shiftr.io is called a «Namespace» (N). A namespace is made of topics (T) that are similar to twitter hashtags. Clients can publish messages (M) that get received by other clients if they have made a matching subscription (S).

```
// connect the first light to the shiftr.io namespace
light1.connect("mqtt://key:sec@connect.shiftr.io");

// subscribe to published messages matching the pattern
light1.subscribe("lights/*", function(message) {
  // handle message
});
```

```
// connect the second light to the shiftr.io namespace
light2.connect("mqtt://key:sec@connect.shiftr.io");

// publish a new message containing a change
light2.publish("lights/2", "on");
// publish another message some time later
light2.publish("lights/2", "off");
```

Example code that shows how two lights could be interconnected with node.js using shiftr.io.
More detailed explanations and examples can be found online.

https://docs.shiftr.io

TITLE	
SKETCH	NOTES

☐ EVENT	☐ LOGIC	☐ ACTION	COST

networked objects