



PROGRAMMING INTERACTION

A STRUCTURED APPROACH TO COMMUNICATING ABOUT YOUR INTERACTIVE CONCEPT AND A WAY TO START BUILDING SKILLS ON DEVELOPING CODE

Technology Skills

Theme: Programming Interaction

1 day of learning individual,

then applying with your partner for your
DPS project

We expect a deliverable due at the
latest Oct 5 at 18:00: a tutorial to build
the prototype you make with your team
(make it on Notion as a project update)

Tutorial Interactive Prototype

Course	Technology Skills
Due	Oct 5, 2020 6:00 PM-6:00 PM
Σ Handed in	0
Project Poster	Empty
Project update	Empty
Prototype Reflecti...	Empty
+ Add a property	

 Add a comment...

✓ 1 resolved comment ✗ 1 backlink

Deliverable

Tutorial for the making of an interactive prototype in the form of a project update, containing: a description of the interactive behaviour, a photo of the prototype, diagram of the electronics, code and any references to libraries used (etc.). Handed in by providing a link to the notion page on Monday Oct. 5 (1800 hrs)

Example

We realize that there is probably not one recipe that fits all cases. We encourage you to research ways of communicating know-how on the reproduction of prototypes.

Here is one that we like:

Smart Thermostat

Connect to this smart thermostat by simply texting it, no need of extra dashboards or platforms. Understanding Telegram Bots Telegram offers a super
https://create.arduino.cc/projecthub/Arduino_Genuino/smart-thermostat-...



Assessment

Contents

- Bridging design and engineering
- Introduction to Acting Machines and code templates
- Assignment 1 - a simple lamp
- Assignment 2 - three state lamp
- Assignment 3 - making things dynamic (until approx. 12:30)
- Afternoon assignment - Pomodoro timer
- Plenary close - show and tell at 16:00

All day watch the
slack channel
#ask-the-doctor for
questions and
updates

Interaction Design

Human Computer Interaction experiences

Designer answers three questions:

how can I make a user do, feel and know?

Use form giving, psychology and engineering to make prototypes.



Engineering for Interaction Design

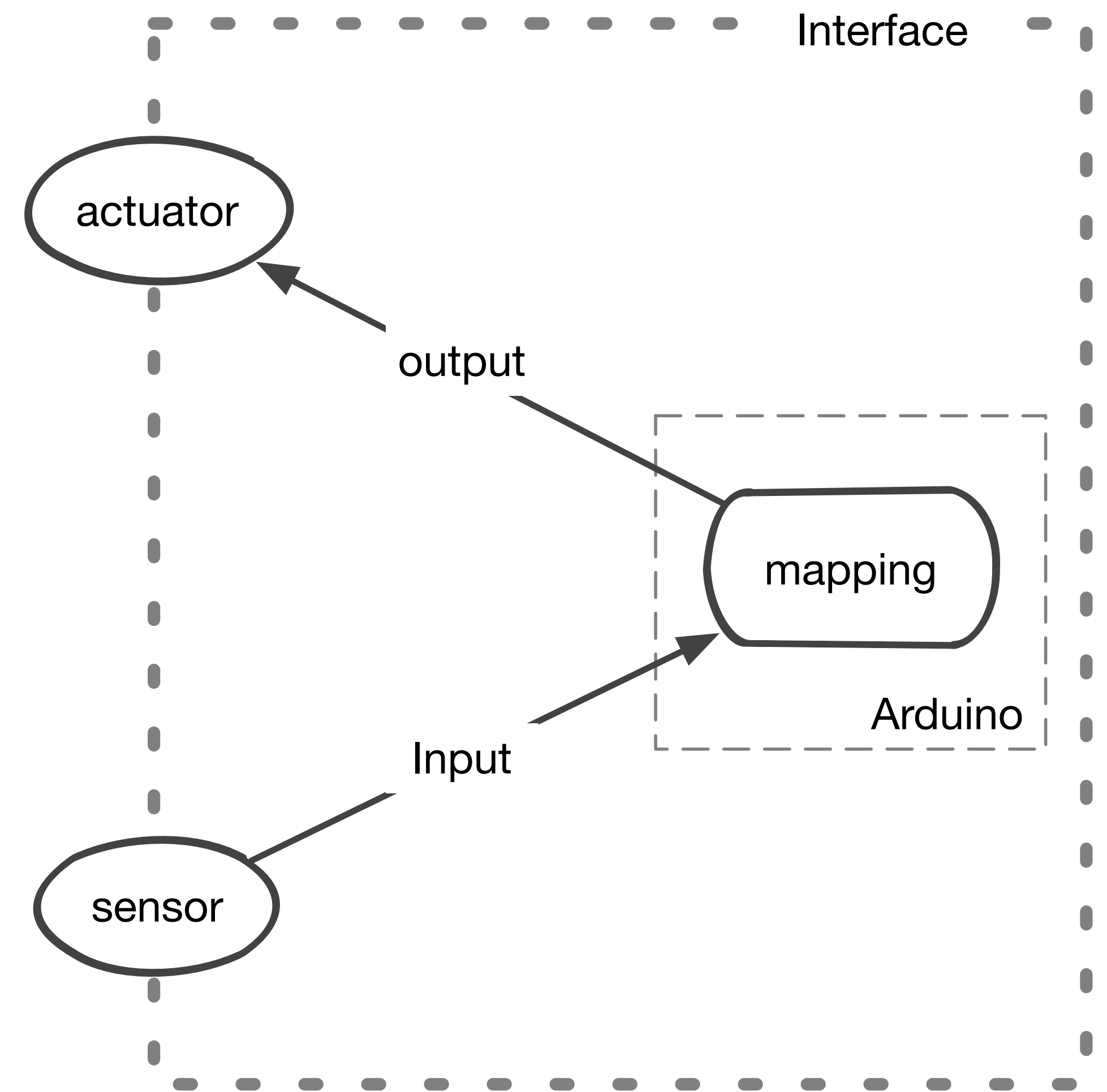
Selecting the appropriate technology

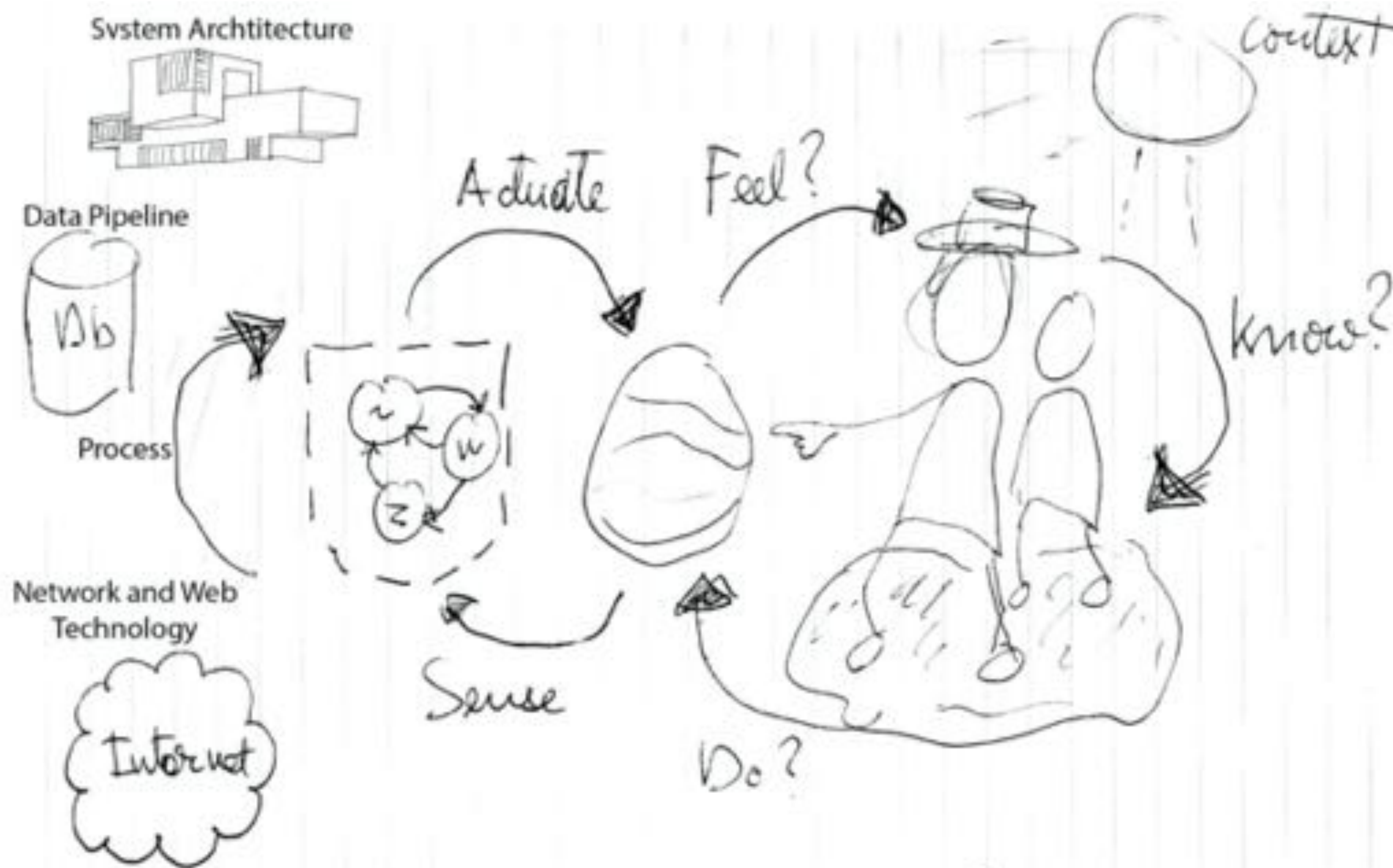
Code the input-mapping-output loop

Input: keeping track of the environment and/or user actions

Mapping: programmatically determine the desired function

Output: activating the behaviour





System Perspective

User Perspective

System Perspective

You place yourself “inside” the Arduino and push and pull all its “levers” to map input to output.

If you are unfamiliar with the Arduino world, its the C programming language, Arduino API, electronics, sensors and actuators you might feel overwhelmed and lost.

You need 10.000 hours to become an expert
(4 years full time)

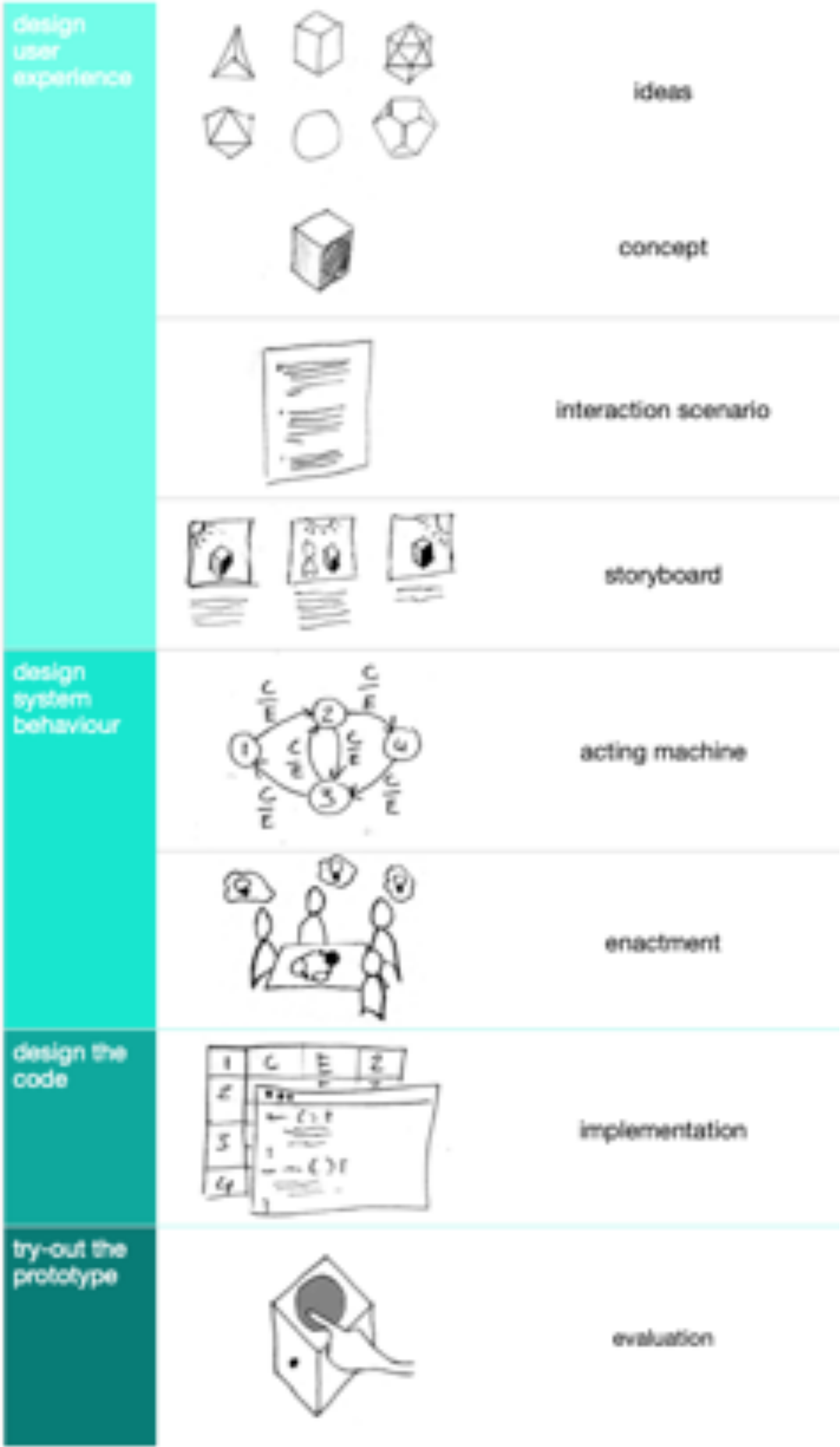


Design process from idea to prototype

Communicate to your team, coach, client, external expert, ...

Communicate about:

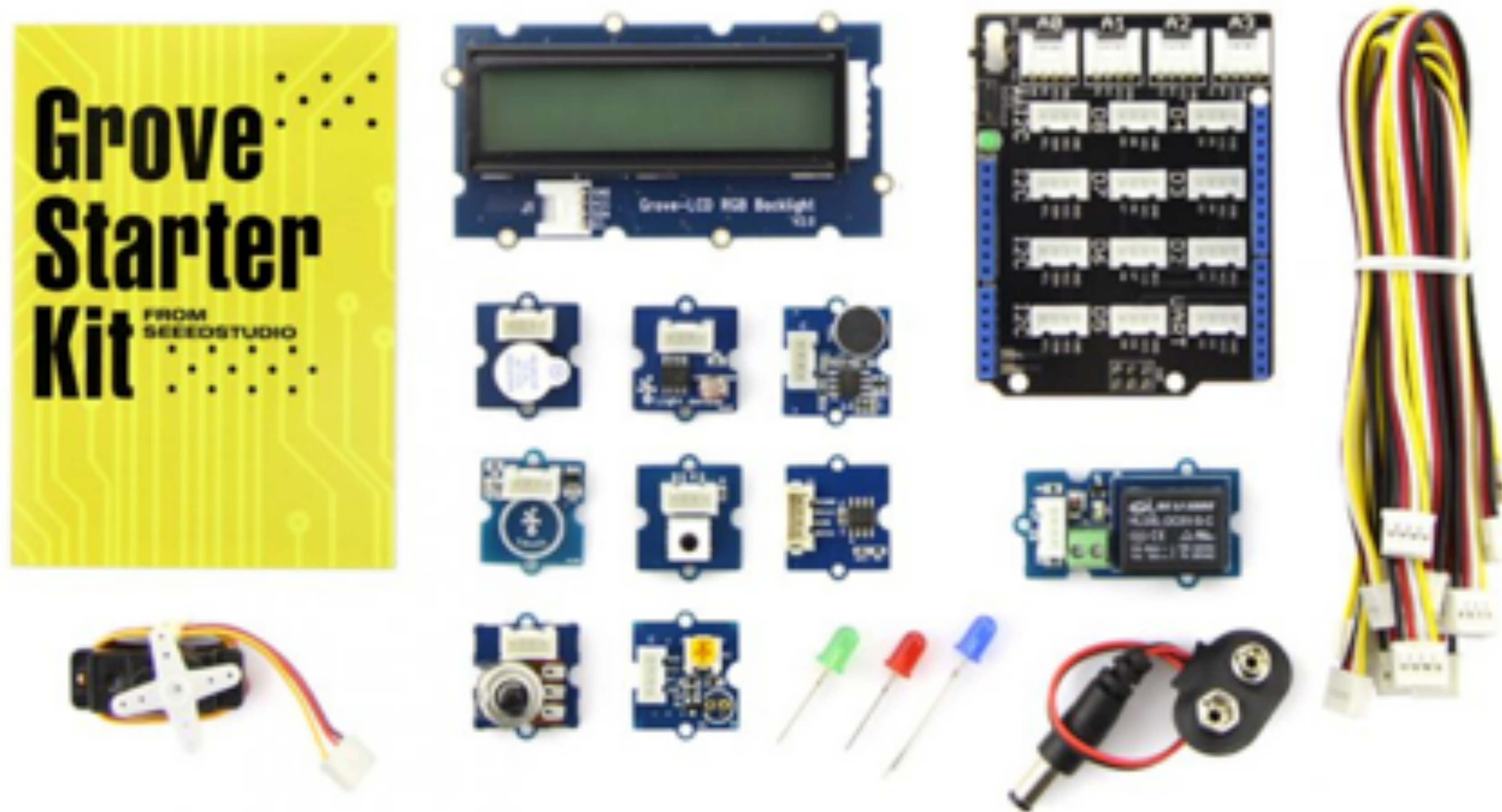
- 1. Design User Experience
- 2. Design System Behaviour
- 3. Design the code
- 4. Learnings from trying-out the prototype



Barriers when coding

1. A simple idea may be very complex to code
2. Determine what library to is the best for you
3. It can be difficult to combine examples/libraries
4. How to use libraries
5. Understanding how prototype behaviour relates back to the code

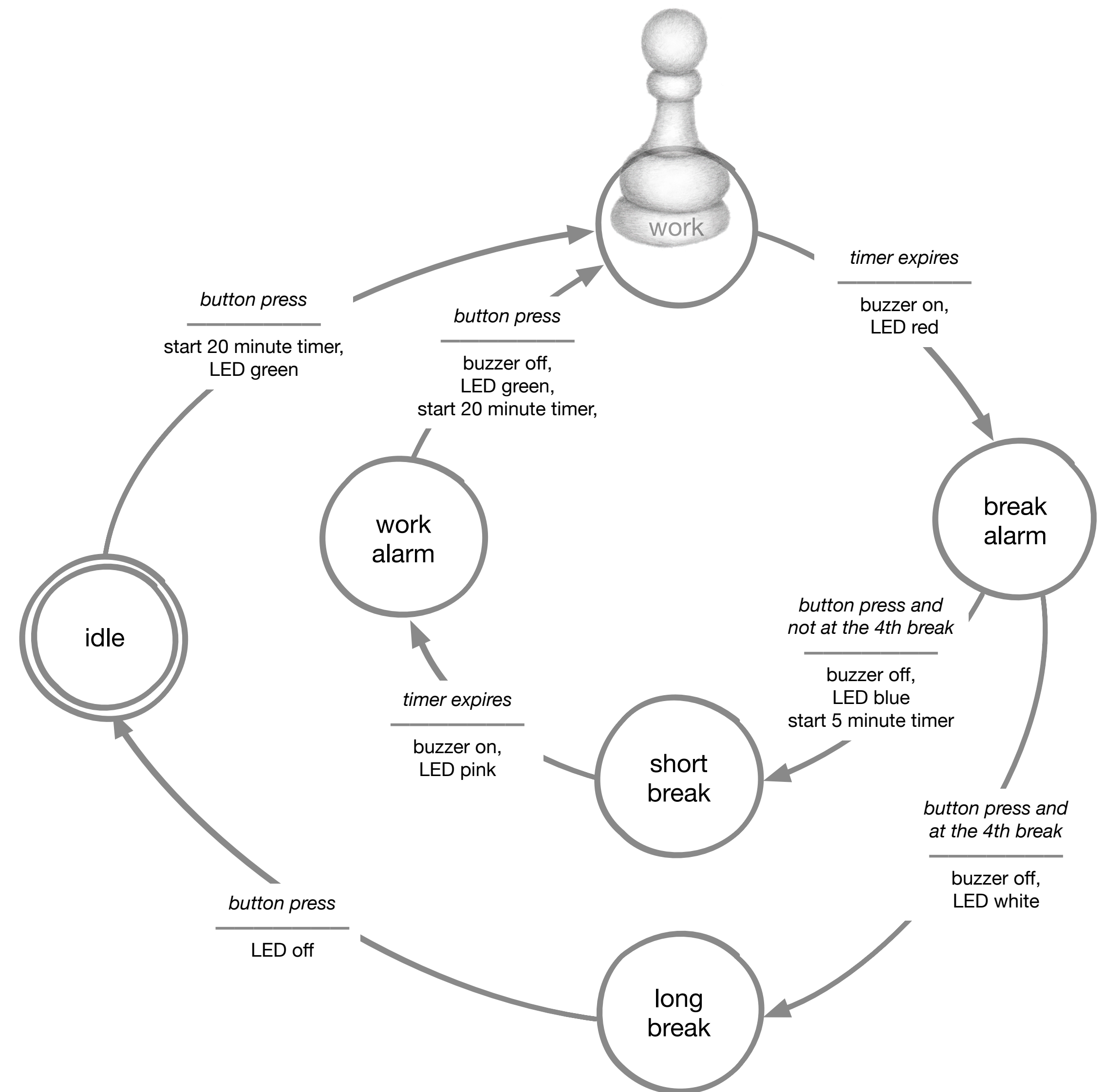




**Reducing the Electronics-skill level during prototyping by using modular sensor/actuator components
(expensive but worth it in the explorative phase of design)**


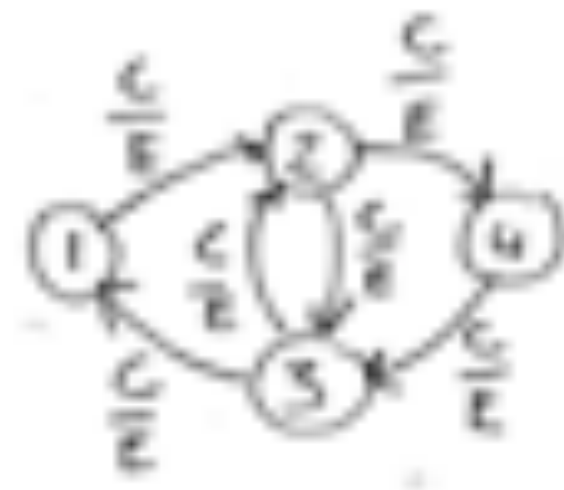

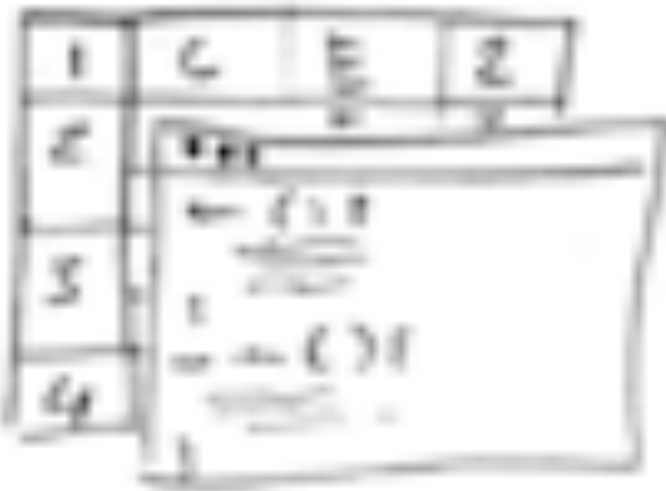

Acting Machines to the Rescue

- Embedded in the design process
- Useful paradigm for HCI applications
- Based on State Transition Diagrams
- Making code from a diagram is easy
- Support shared understanding



From storyboard to working code

- Draw an Acting Machine Diagram
- Use role-playing to validate/improve it
- Use the code template to transform the diagram into a working program

design user experience		storyboard
design system behaviour		acting machine
		enactment
design the code		implementation
try-out the prototype		evaluation

Recipe for making an Acting Machine Diagram for a lamp



User is reading a book,
it is slowly getting
darker



User presses on the
button of the lamp to
turn it on



The light of the lamp
makes it possible to
read more in the book

1. Changes in behaviour (**state**): 1) lamp is off 2) or on
2. Order of the states (**transition**): First lamp is off then it is on (and off again)
3. For each transition define behaviour change (**effect**): light turns on and off
4. For each transition define what activates it (**cause**): when user presses button

Acting Machine Diagram explained

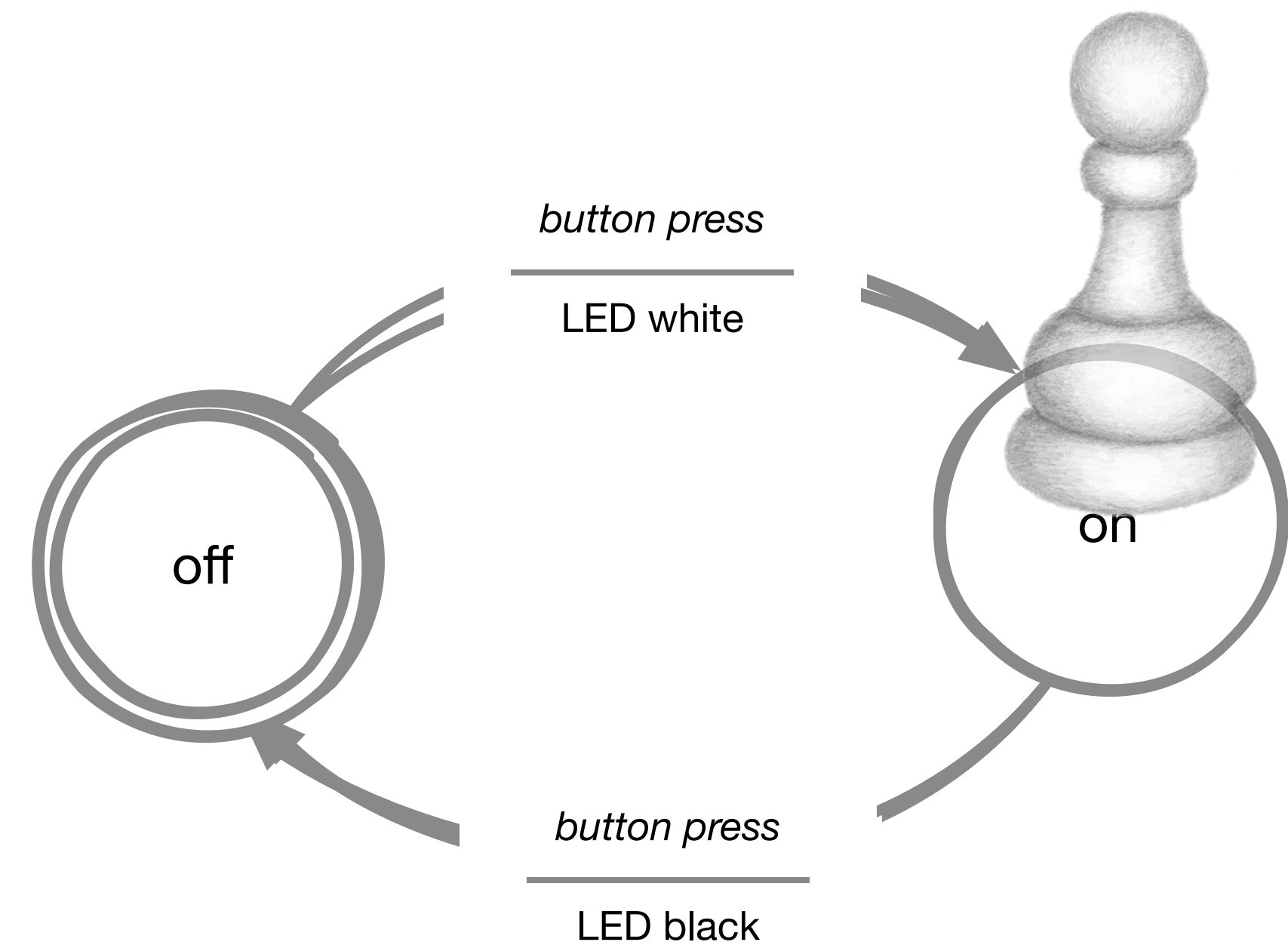
States: circles - nothing happens in the states

Transitions: arrows - determine control flow

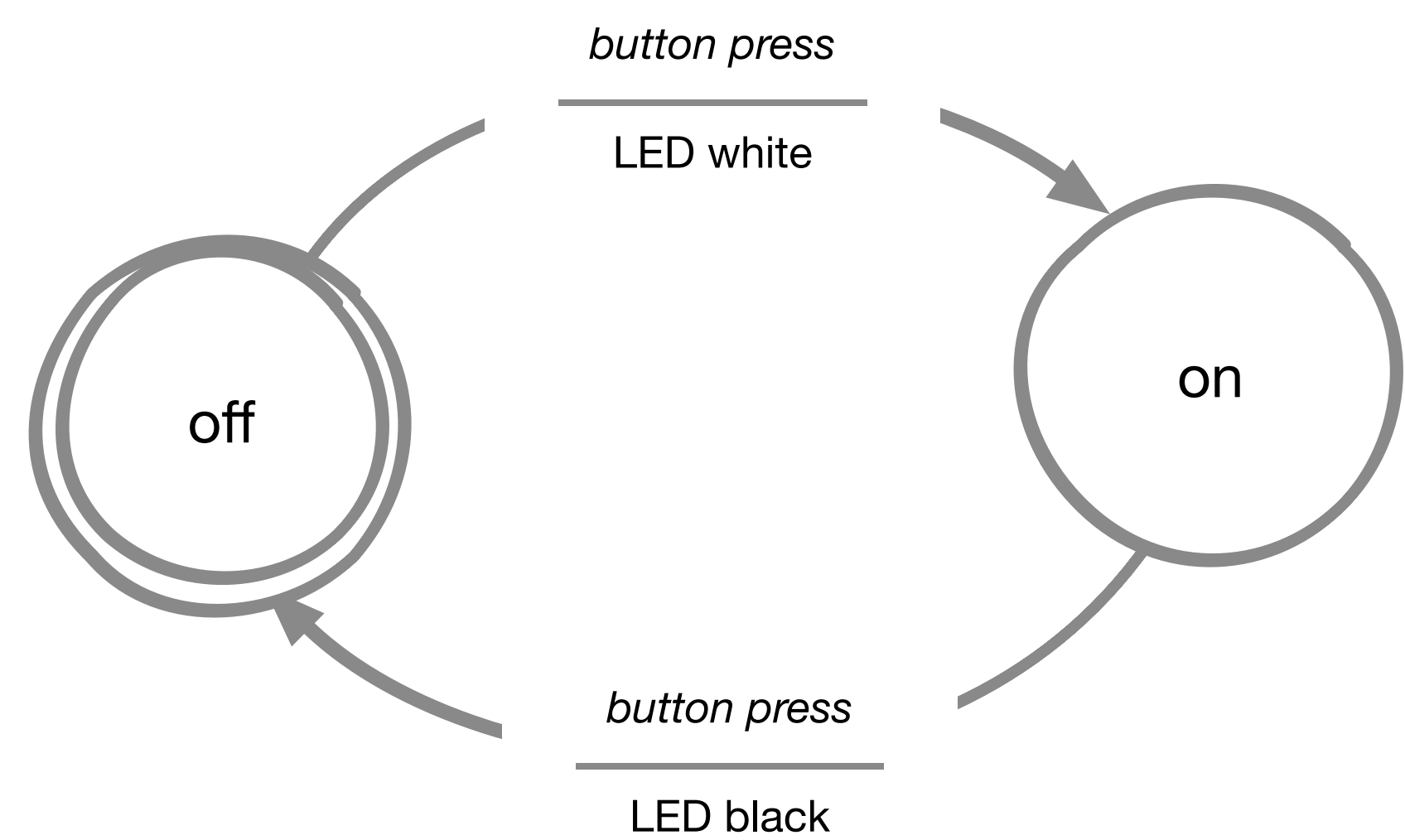
Effects: on a transition (below line) - controlling actuators

Causes: on a transition (above line) - input from sensors or a timer event

Current state: the pawn - determines what transitions are evaluated



From Acting Machine Diagram to code - Acting Machine Table



from state	if this happens	then do this	go to state
state-off	button is pressed	turn lamp on	state-on
state-on	button is pressed	turn lamp off	state-off

Acting Machine Table to code

variables for the Acting Machine

hand control over to the Acting Machine

code for effects - turn lamp on and off

code for causes - determine if a transition is activated

Acting Machine control flow

state-off	button is pressed	turn lamp on	state-on
state-on	button is pressed	turn lamp off	state-off

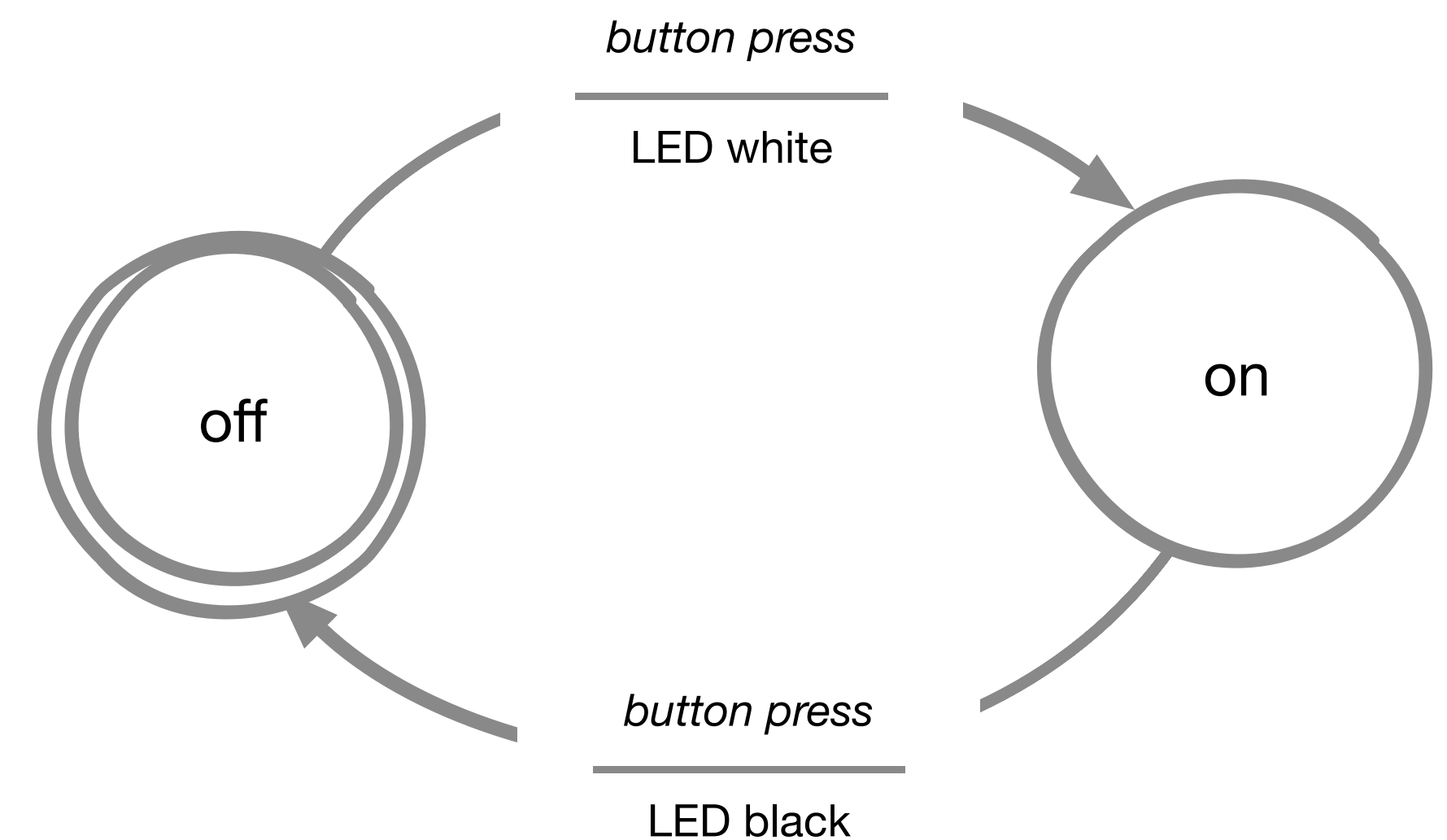
```
1 // Assignment1
2 // ----- Libraries
3
4 // ----- Variables
5 const int led_pin = 13;      // pin of internal LED
6 const int button_pin = 2;    // connect a button to D2
7 const int analog_pin = A0;   // connect a analog sensor to A0
8
9 // Acting Machine Variables
10 const int state_off = 0;
11 const int state_on = 1;
12 int current_state = state_off; // we start with lamp off
13
14 // ----- Setup and loop
15 void setup() {
16   pinMode(led_pin, OUTPUT); // setup LED pin properly
17 }
18
19 void loop() {
20   updateActingMachine(); // hand control to the Acting Machine
21 }
22
23 // ----- Acting Machine effect functions
24 void switchLamp(boolean state) {
25   // Use the Arduino API calls to control the internal LED on pin led_pin
26 }
27
28 // ----- Acting Machine cause functions
29 boolean buttonPress() {
30   // Use the Arduino API calls to check user input on pin button_pin or analog_pin
31 }
32
33 // ----- Acting Machine control flow
34 void updateActingMachine() {
35   switch (current_state) {
36     case state_off:
37       if (buttonPress() == true) {
38         switchLamp(true);
39         current_state = state_on;
40       }
41       break;
42     case state_on :
43       if (buttonPress() == true) {
44         switchLamp(false);
45         current_state = state_off;
46       }
47       break;
48   }
49 }
```


Assignment 1 - 30 minutes

- Open assignments/simple_lamp.ino
- Implement the buttonPress cause function
- Implement the switchLamp effect function

Try this if it was easy

- Connect the Grove Chainable LED to D7
- Rewrite your code to make that work



Demonstrate

Chainable LED code

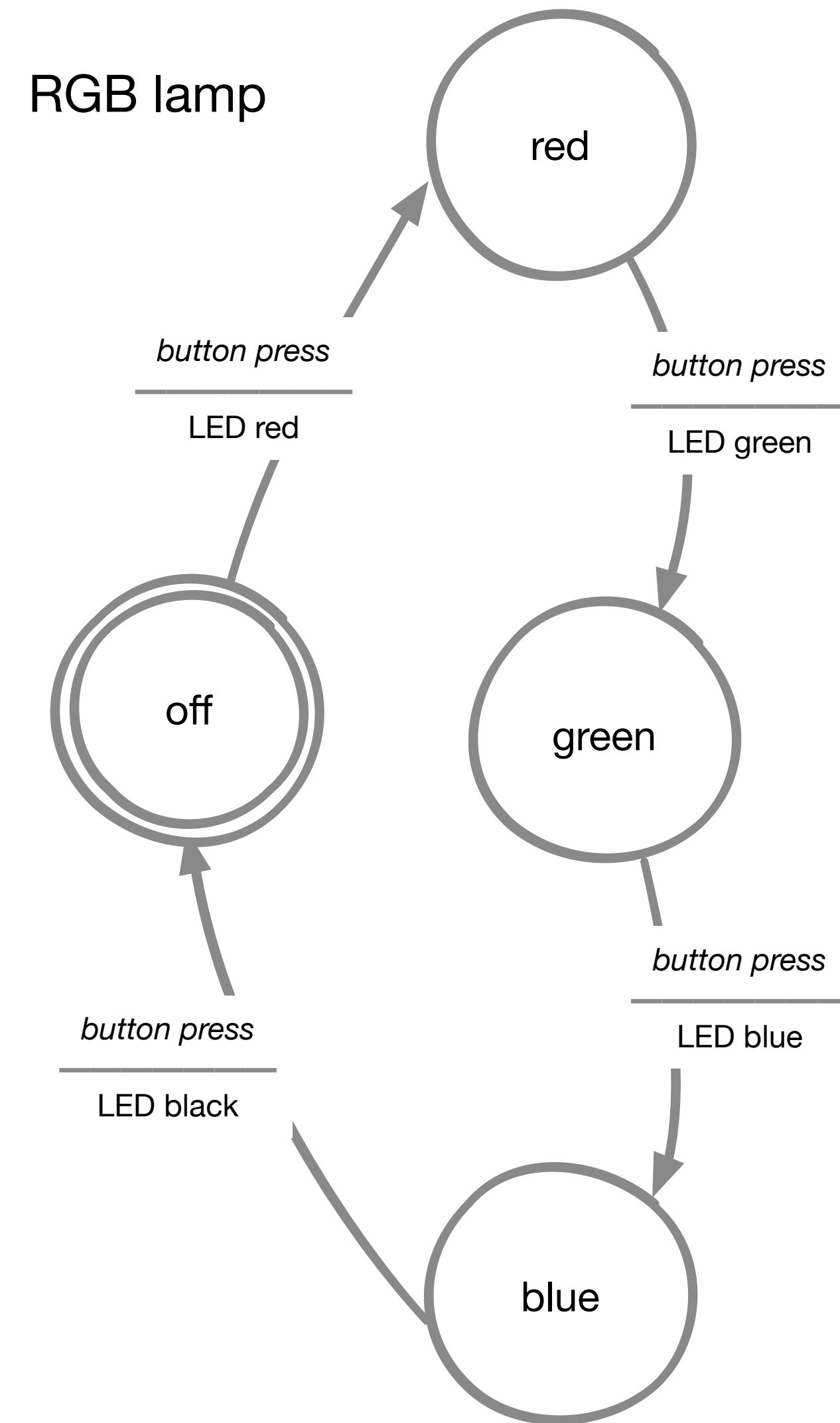
Better buttonPress

Assignment 2 - 30 minutes

- Open assignments/rgb-lamp
- Extend the Acting Machine code to achieve the functionality in the diagram on the right

If you want to challenge yourself

- Start with an empty Arduino program
- Implement the code template for Acting Machines



Demonstrate

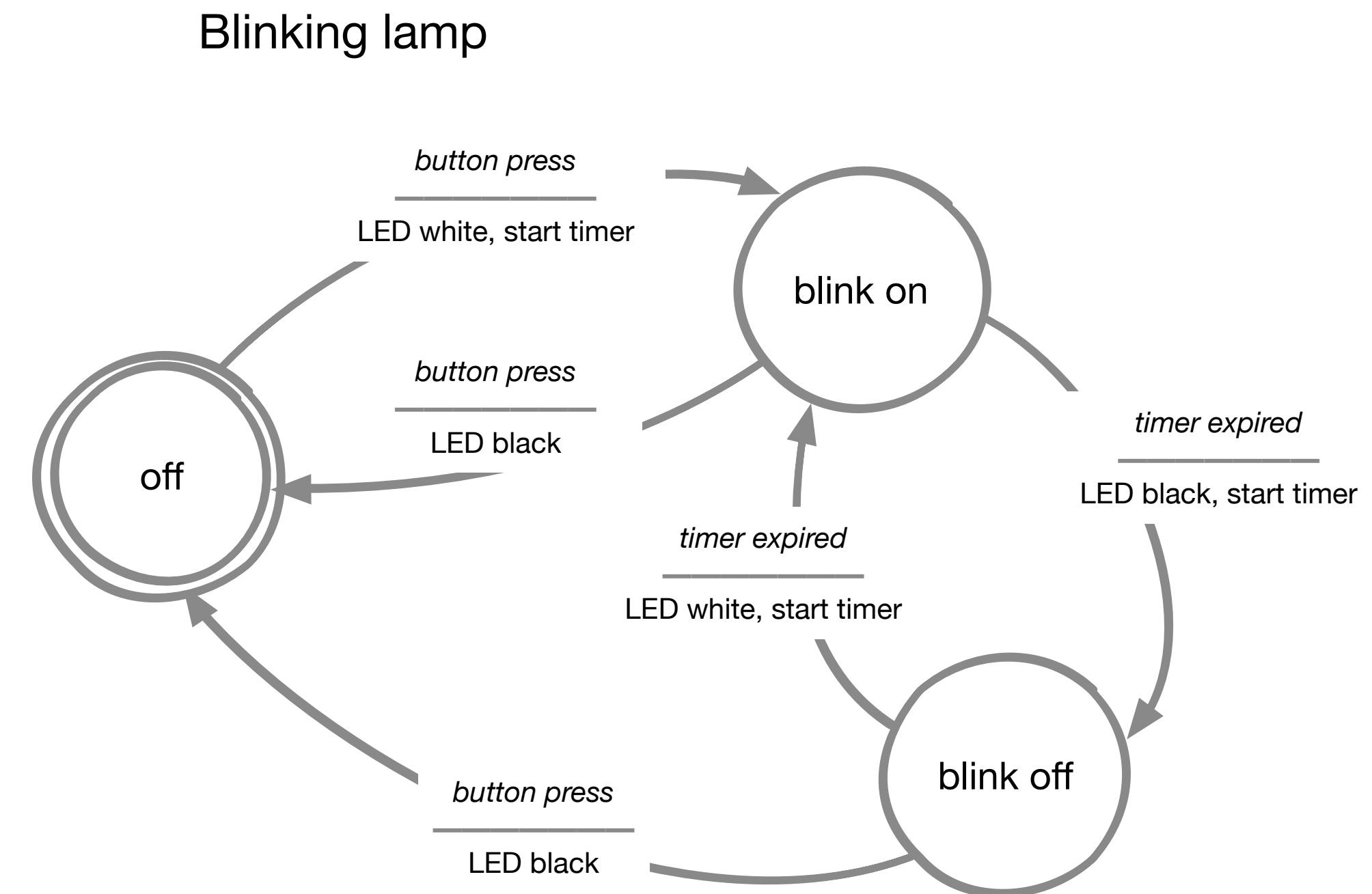
Timer code

Assignment 3 - 45 minutes

- Implement the Blinking Lamp
- Make a copy of simple-lamp and extend the Acting Machine code for the blinking lamp

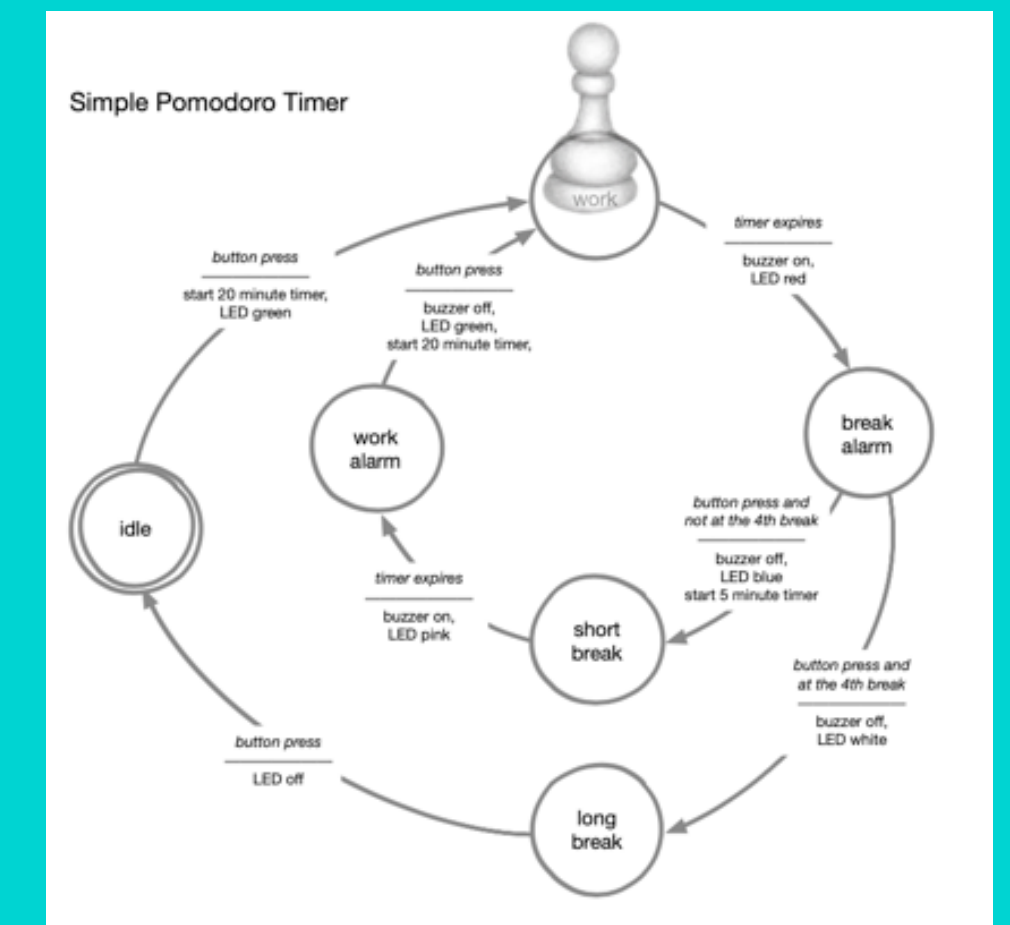
Try this if it was easy

- Instead of blinking do fading



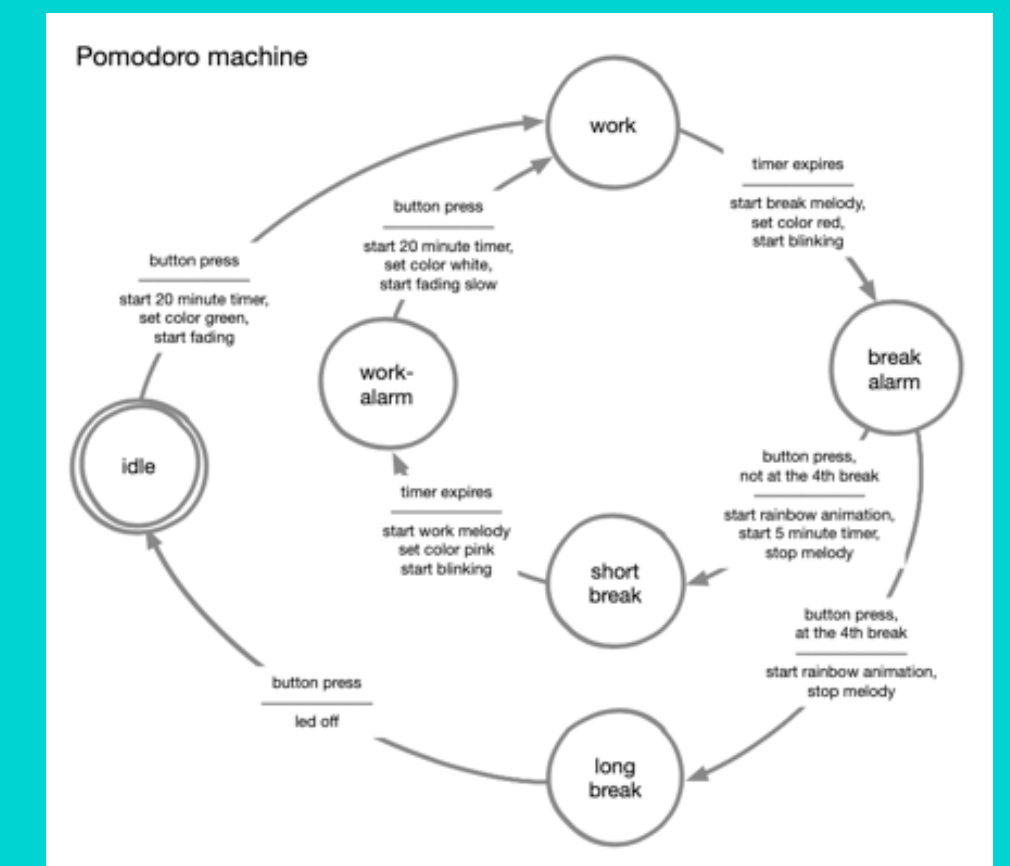
Afternoon Assignment

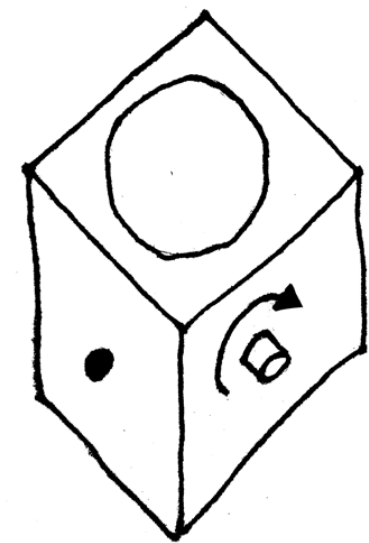
- Start with the sweet-n-simple Pomodoro timer code
- Implement part the supplied Acting Machine Diagram
- For sweet-n-simple you work with simple Arduino API calls



Or if you what to challenge yourself

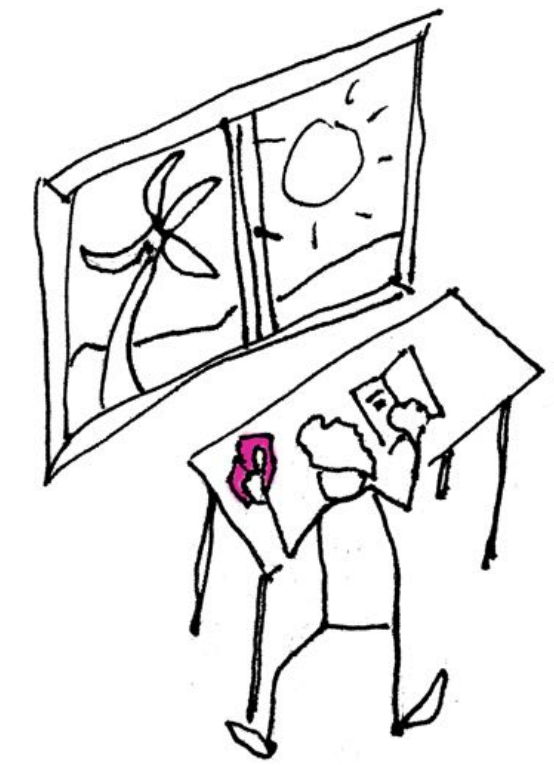
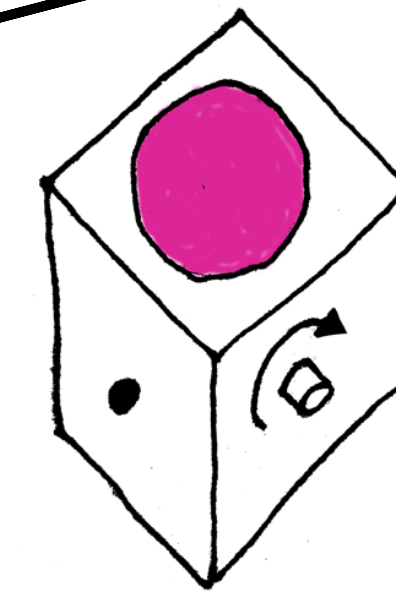
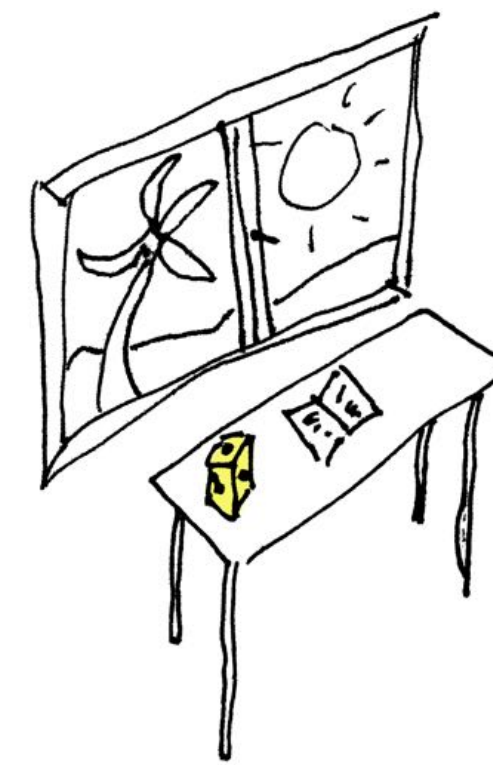
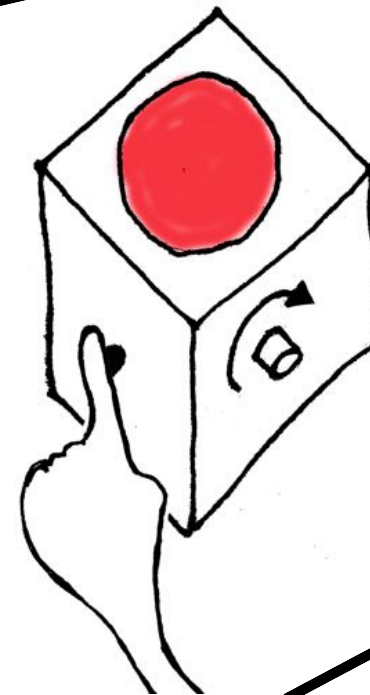
- Start from storyboard, construct an Acting Machine Diagram
- Work with the fancy-fancy Pomodoro timer code
- Implement as much as possible of the Acting Machine Diagram
- Dynamic feedback with the supplied sound, led, or servo machine





Pomodoro timer
ala Romagnoli

<https://id-studiolab.github.io/pomodoro/>



Lorenzo is an actor, he has to memorise a text for a play he is performing in. He is using the Pomodoro technique at home to structure his study time in a productive and healthy manner.

He starts by pressing a button on the side of the device. A green light appears on the top of the timer and the passage of time is made visible by a small clock-hand.

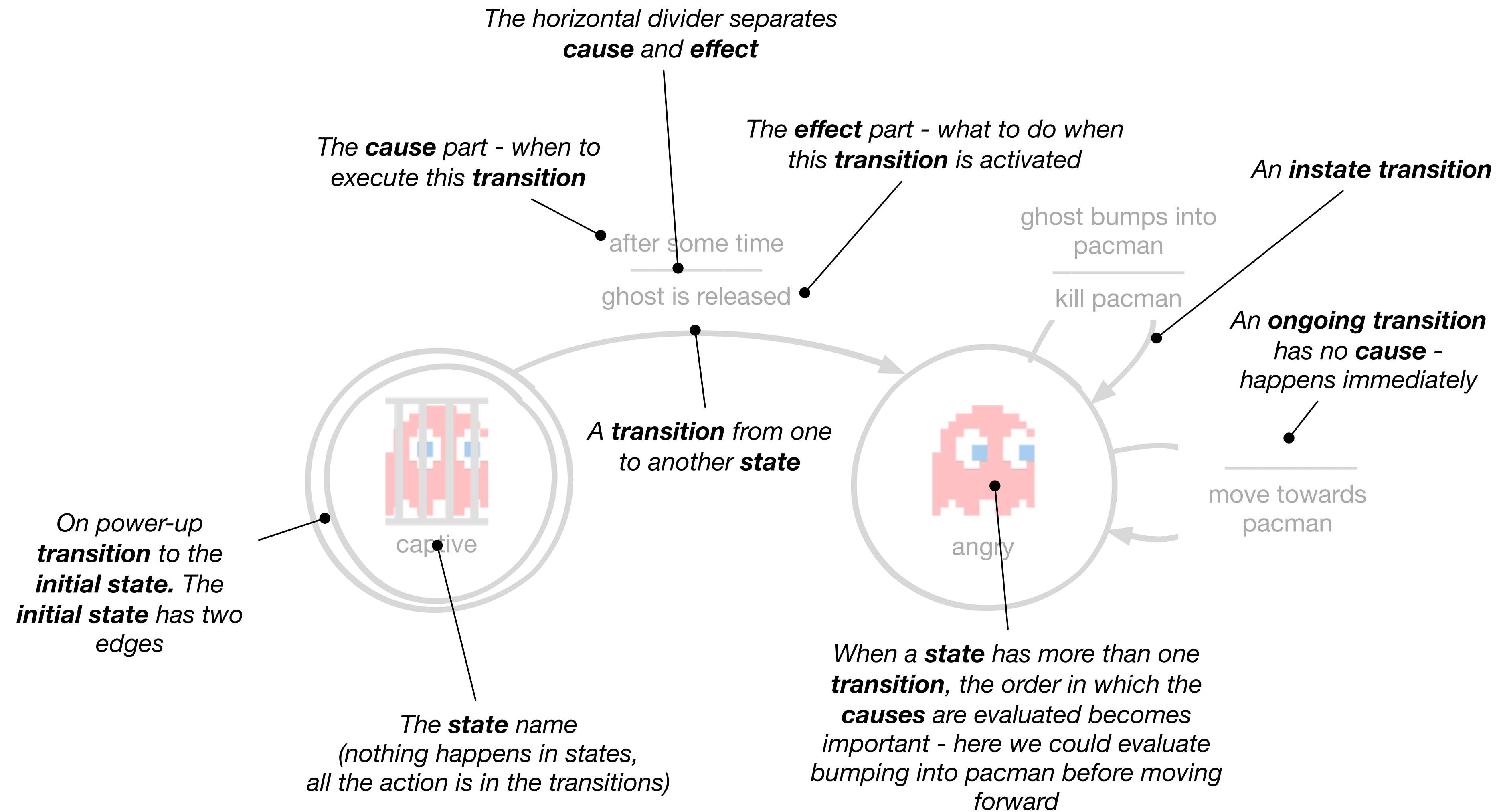
After 20 minutes an audible alarm sounds with an accompanying light animation, indicating it is time to have a short break. Lorenzo stops the alarm by pressing the button on the device.

Lorenzo walked to the kitchen and is making a cup of coffee.

After 5 minutes another alarm is played signalling it is now time to start another work period.

Lorenzo returns and presses the button. The alarm then stops and another 20 minutes work period is started. The timer repeats this process 4 times and then it switches itself off. In Pomodoro terms it is now time for a longer break.

Acting Machine Diagrams - explained

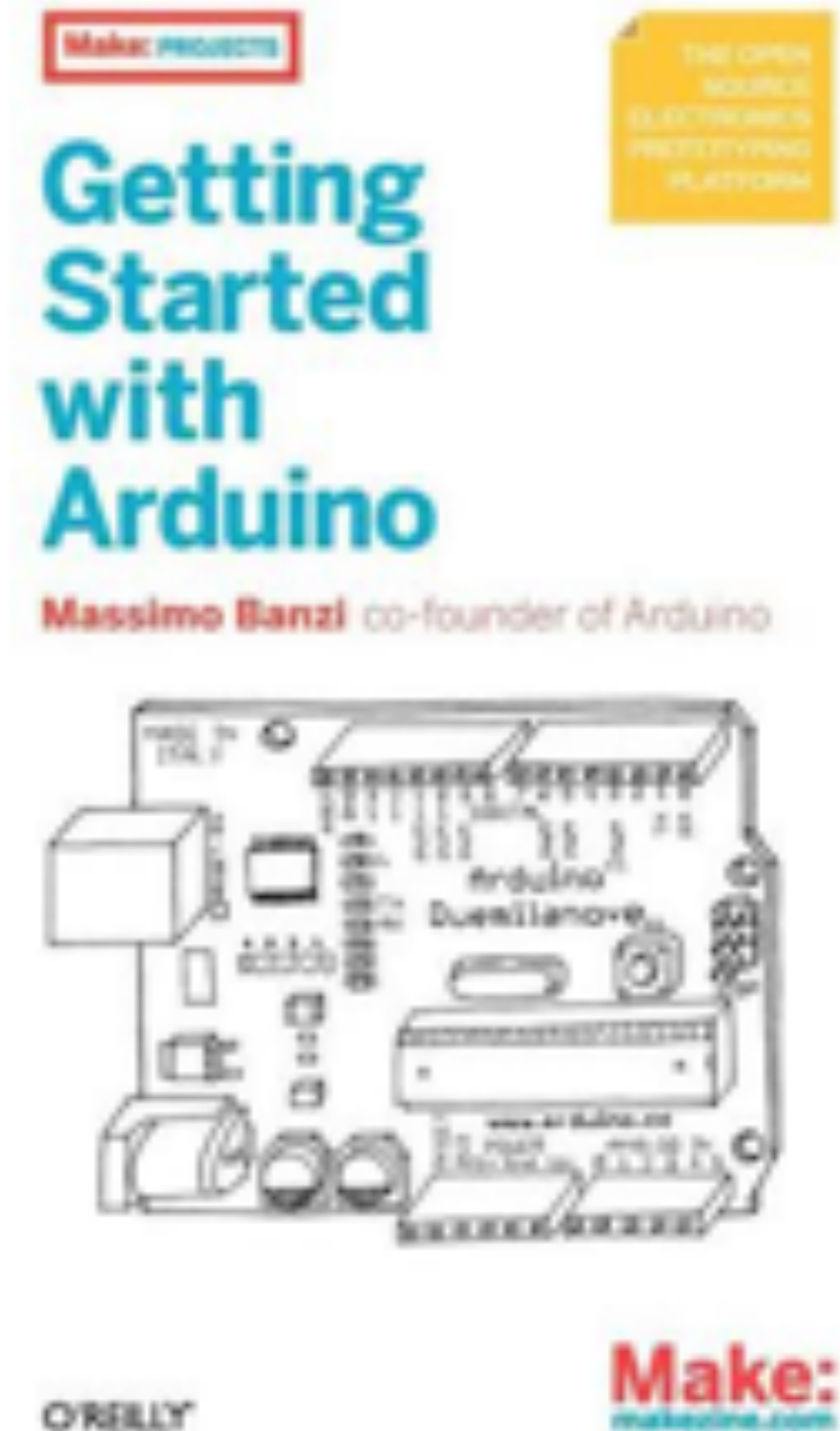


Coding conventions

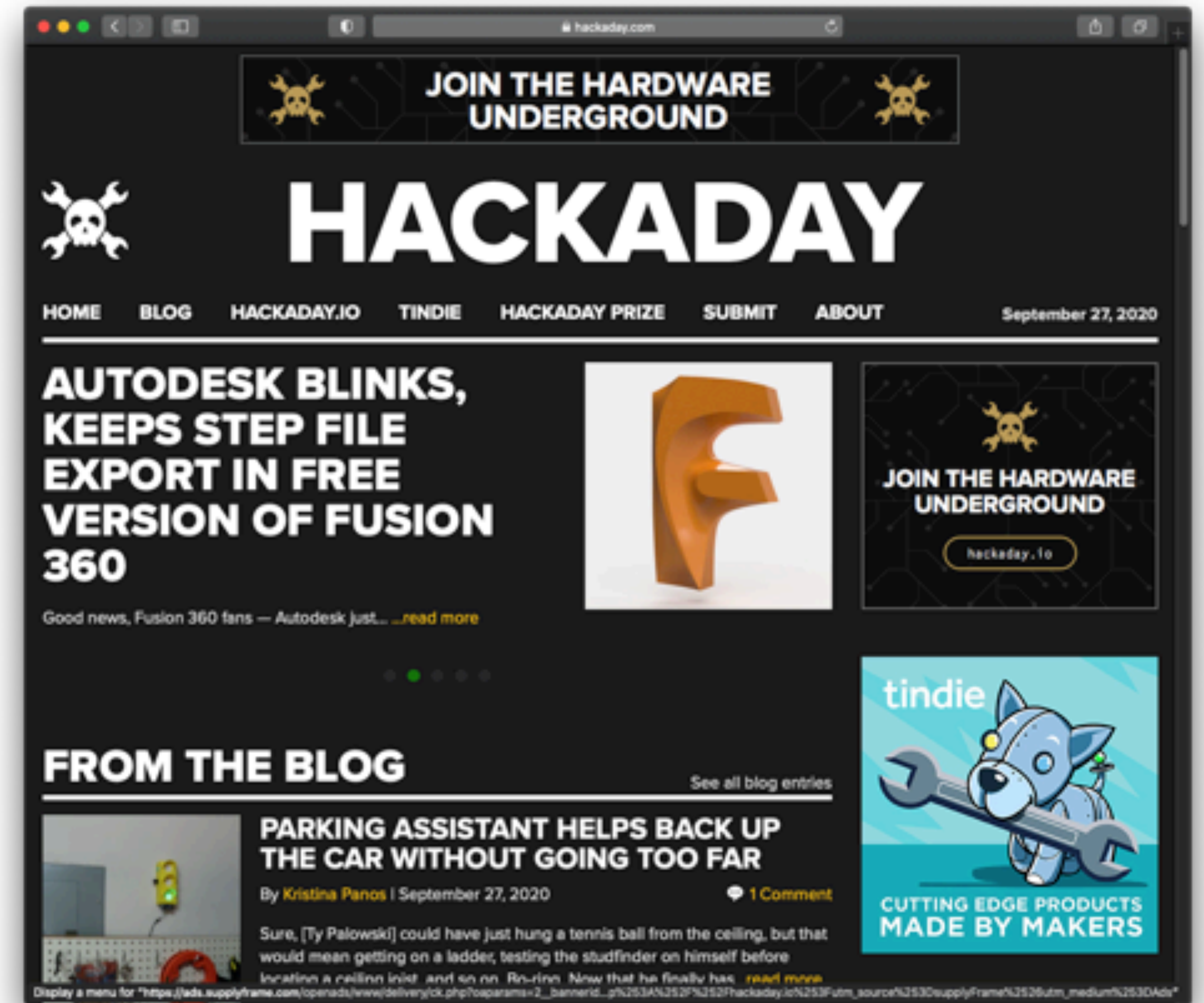
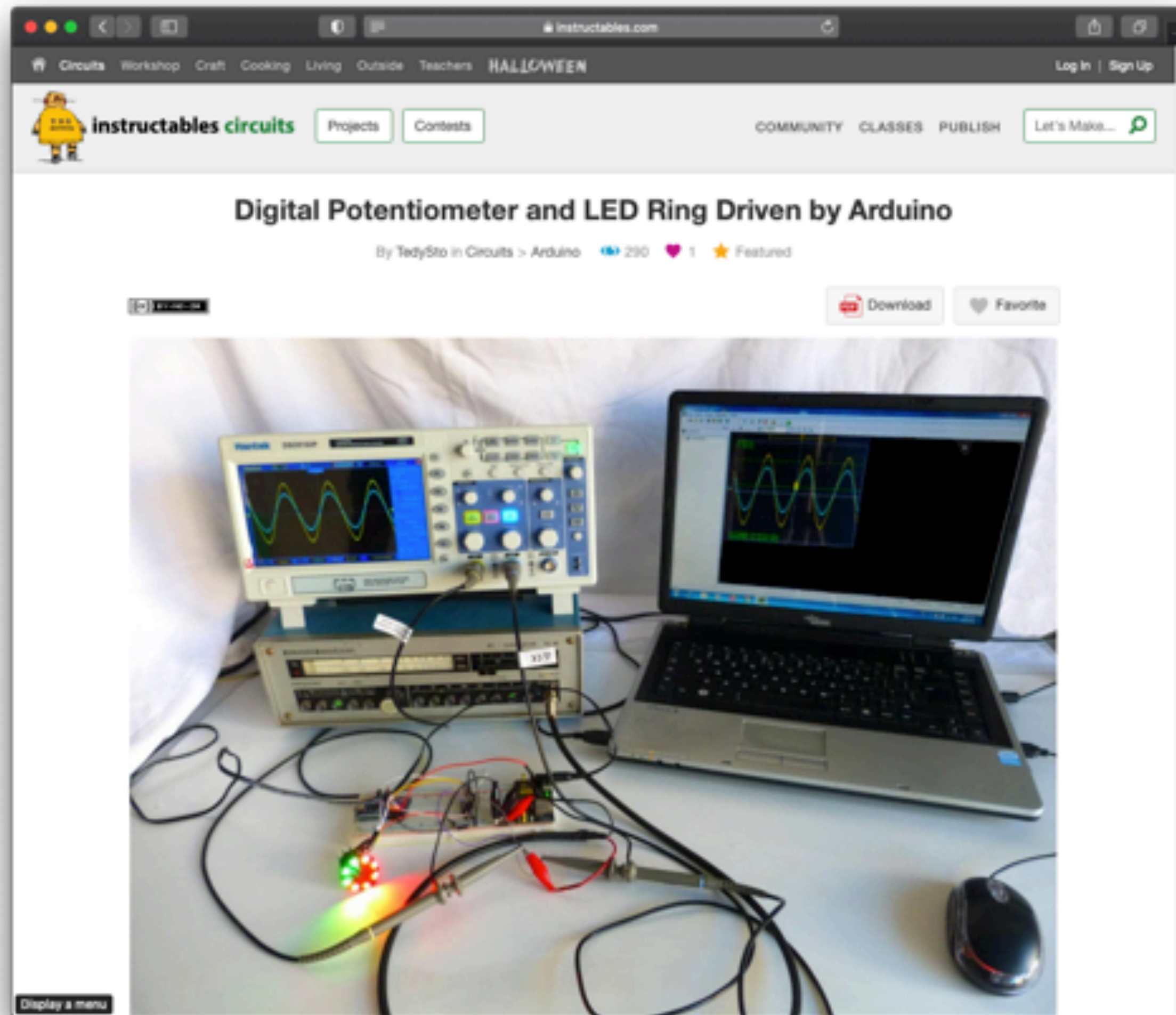
Variables: all lowercase, if multiple words separate them with underscore: state_off

Function names: capitalise words except the first one: updateActingMachine()

Order of program: (from top to bottom) libraries, variables, setup/loop, effects, causes, acting machine control flow



Some great books



The web has more - is often more up to date
(perhaps less depth than books)