



f3.js: A Parametric Design Tool for Physical Computing Devices for Both Interaction Designers and End-users

f3.js Edit the project Basic Information Source Code Design Alternatives

Source Code

Provide the source code of a microcontroller or tiny computer in JavaScript. Node.js-based computers are supported. Require f3.js package and use its API to design the device enclosure.

```
1  /** LED blinking app for Intel Edison */
2
3  var n = 2 // number of LEDs [1,4]
4    , width = 130
5    , height = 105
6    , thickness = 45;
7
8  // for building the enclosure layout
9  var f3js = require('f3js')
10 , c = f3js.createContainer()
11 , rect = c.drawJointRectangle(0, 0, width, height)
12 , line = c.drawLine(50, height/2 - 5, width - 30, height/2 - 5);
13
14 // for blinking LEDs
15 var groveDriver = require('jsupm_grove')
16 , leds = [];
17
18 // use this line as the guide path
19 line.layout = { name: 'distribute', rotate: false };
20
21 for (var i = 0; i < n; i++) {
22   var led = new groveDriver.GroveLed(i+2);
23
24   // put an LED module
25   var ledc = c.add(led, line);
26
27   // open a hole for the wire
28   ledc.drawRectangle(- 10, 15, 20, 10);
29
30   // start blinking the LED (details omitted)
31   leds.push(led);
32   setTimeout(function (l) { return function () {
33     l.handler = setInterval(blink(l), 1000);
34     l.l(led).s = 100;
35   }
36 }
37 }
38 }
39 }
```

Layout

Selected: Rectangle
Hovered: -
Page: 1

Customization

number of LEDs (2)

Layout view options

Update Delete

f3.js Get Started IoT Projects Modules

Preview

Page: 1

Customize

number of LEDs (2)

Layout view options

- Show module names
- Show labels near adjacent edges
- Show warnings on module interferences

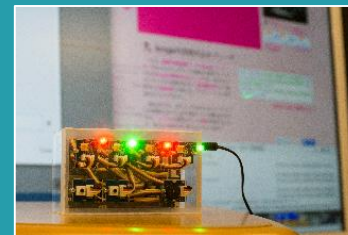
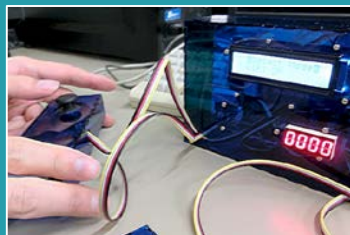
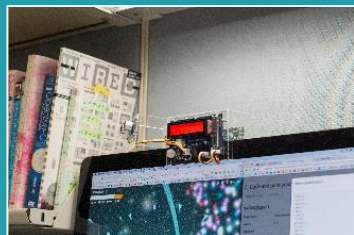
Disabled items are those proposed by somebody with the "O" button and not yet implemented. Got interested?

Building Instructions

- 1 Purchase**
Purchase hardware components
- 2 Print & Assemble**
Print out enclosure components
- 3 Install**
Install and run the program

Modules required for assembly
Layout
Printer type

Program
Archive type



Jun Kato, Masataka Goto

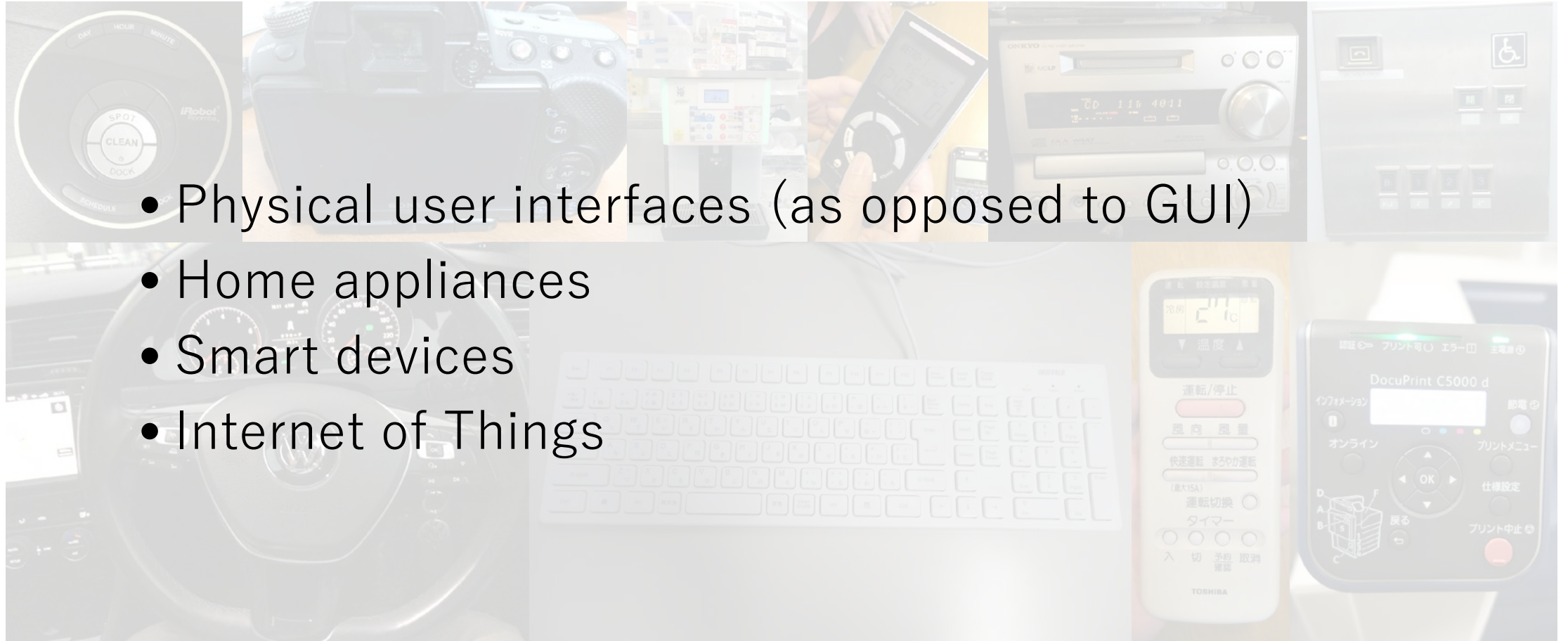
The presented system is publicly available at

<http://f3js.org>

Physical computing devices, everywhere

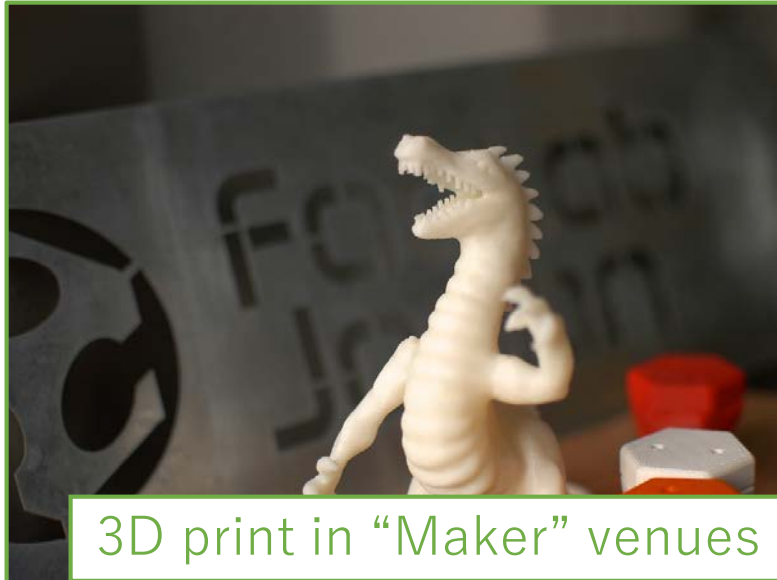


Physical computing devices, everywhere



- Physical user interfaces (as opposed to GUI)
- Home appliances
- Smart devices
- Internet of Things

Personal fabrication made easy



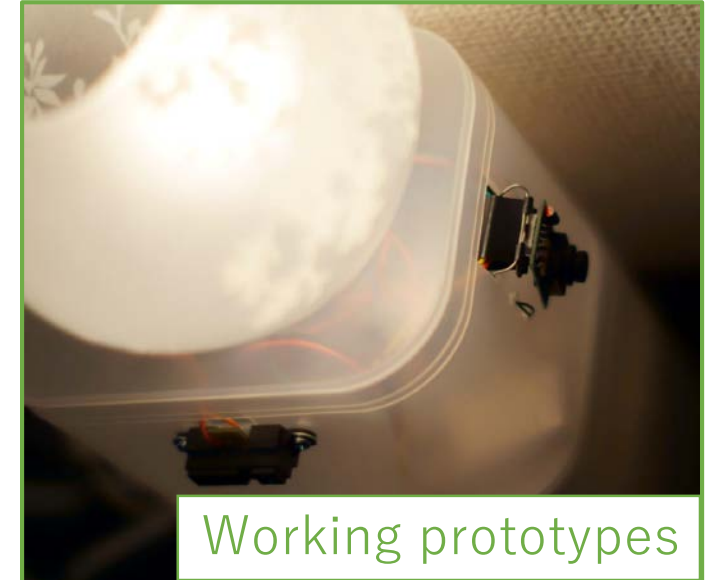
3D print in “Maker” venues

Photo taken by Atsushi Tadokoro (CC BY 2.0)
<https://www.flickr.com/photos/tadokoro/5138646645>



Soldering with parents

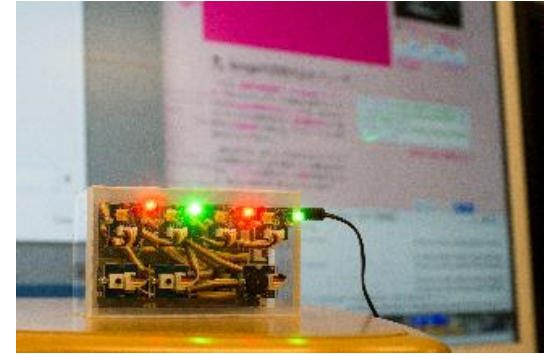
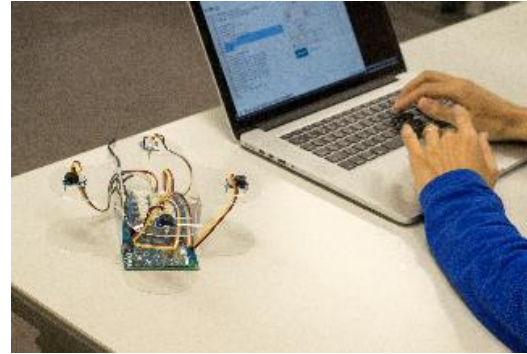
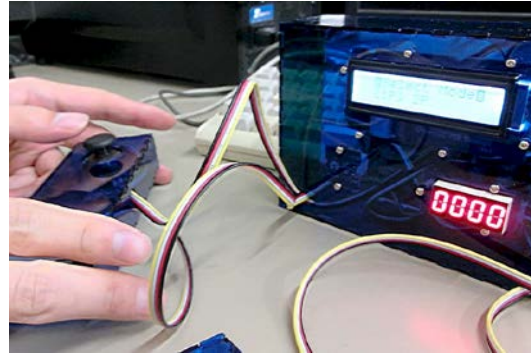
Photo taken by Mitch Altman (CC BY-SA 2.0)
<https://www.flickr.com/photos/maltman23/6954963529>



Working prototypes

How about device programming & assembly?

Research questions regarding physical computing devices



For interaction designers

- How can we support prototyping of the devices?

For end users

- How can we support personal customization of the devices?

Preliminary observations: photos of 200 devices and informal interviews



Last minute additions



エレベータ (4)



オーディオ機器 (28)



カメラ (4)



キーボード (6)



ゲーム機 (2)



コーヒーマシン (5)



プリンタ・プロッタ (11)



リモコン (28)



業務用機材 (18)



空調・照明 (22)



自動車 (3)



洗濯機・洗浄器 (7)



調理器具 (21)



電話機・キオスク (13)



^76DB772295AD
A37E678186C0B
CDA03FAAB95F4
5BE070FB37D7...



20160811_14554
0111_iOS.jpg



20160904_16521
0408_iOS.jpg



20160919_05485
3044_iOS.jpg



20160923_04031
3503_iOS.jpg



WP_20160918_00
_57_19_Pro.jpg

Design patterns in Physical User Interfaces



Line layout

Circle layout

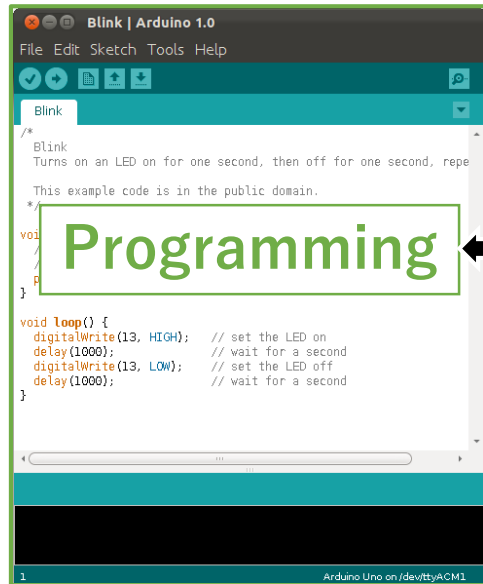
- 187 devices have physical user interfaces on **planar surfaces**
- 139 devices have modules placed along **straight lines**
- 51 devices have modules placed on **circular paths**

Mental gap between software & hardware



- Designers need to imagine hardware while writing code
- “new Button()” does not infer any hardware layout

Difficulties in exploring design alternatives



```
Blink | Arduino 1.0
File Edit Sketch Tools Help
Blink
/*
 * Blink
 * Turns on an LED on for one second, then off for one second, repeatedly.
 * This example code is in the public domain.
 */
const int LED_PIN = 13;

void setup() {
  pinMode(LED_PIN, OUTPUT);
}

void loop() {
  digitalWrite(LED_PIN, HIGH); // set the LED on
  delay(1000);                 // wait for a second
  digitalWrite(LED_PIN, LOW);  // set the LED off
  delay(1000);                 // wait for a second
}

Arduino Uno on /dev/ttyACM1
```



- **Expensive switching** cost between two activities in two tools
- Prior efforts in either one of these (software **or** hardware)

Preliminary observations (summary)



Last minute additions



エレベータ (4)



オーディオ機器 (28)



カメラ (4)



キーボード (6)



ゲーム機 (2)



コーヒーマシン (5)



プリンタ・プロッタ (11)



リモコン (28)



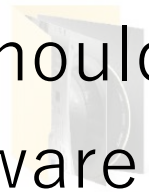
業務用機材 (18)



空調・照明 (22)



自動車 (3)



洗濯機・洗浄器 (7)



調理器具 (21)



電話機・キオスク (13)



^76DB772295AD
#37E678186C0B
@2017F0AB95F4
5BE070FB37D7...



20160811_14554
0111_iOS.jpg



20160904_16521
0408_iOS.jpg



20160919_05485
3044_iOS.jpg



20160923_04031
3503_iOS.jpg

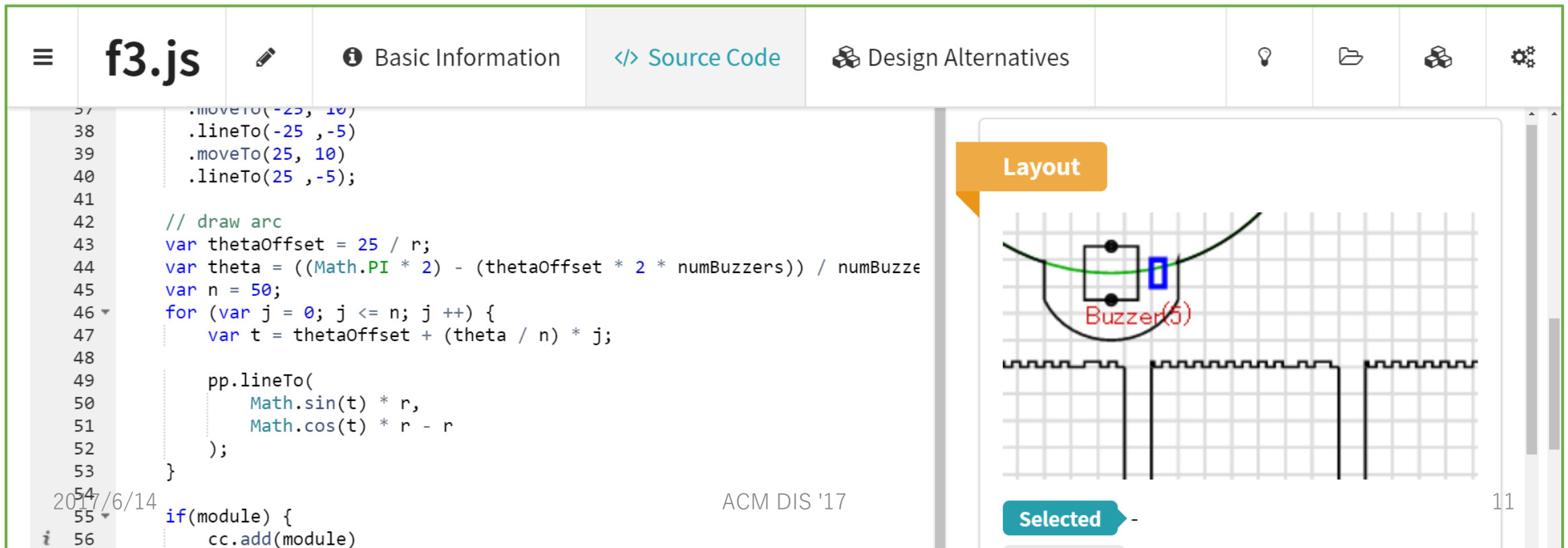


WP_20160918_00
_57_19_Pro.jpg

- Typical **design patterns** should have tool support
- **Mental gap** between software & hardware exists
- **Comparing alternatives** is crucial for good design

f3.js: integrated support for programmers

- **Live Programming with intuitive APIs** of features & layout
- Interactive development of IoT devices in **one environment**



The screenshot displays the f3.js development environment. The top navigation bar includes a menu icon, the file name 'f3.js', and several tabs: 'Basic Information', 'Source Code' (active), 'Design Alternatives', a lightbulb icon, a folder icon, a cube icon, and a gear icon. The main area is split into two panes. The left pane shows JavaScript source code for drawing a shape with a buzzer. The right pane shows a 'Layout' view with a grid background, a diagram of a buzzer component labeled 'Buzzer(5)', and a waveform plot below it. A 'Selected' indicator is visible at the bottom of the layout pane.

```
37     .moveTo(-25, 10)
38     .lineTo(-25, -5)
39     .moveTo(25, 10)
40     .lineTo(25, -5);
41
42     // draw arc
43     var thetaOffset = 25 / r;
44     var theta = ((Math.PI * 2) - (thetaOffset * 2 * numBuzzers)) / numBuzzers;
45     var n = 50;
46     for (var j = 0; j <= n; j++) {
47         var t = thetaOffset + (theta / n) * j;
48
49         pp.lineTo(
50             Math.sin(t) * r,
51             Math.cos(t) * r - r
52         );
53     }
54
55     if(module) {
56         cc.add(module)
```

2017/6/14

ACM DIS '17

11

Module repository for hardware metrics

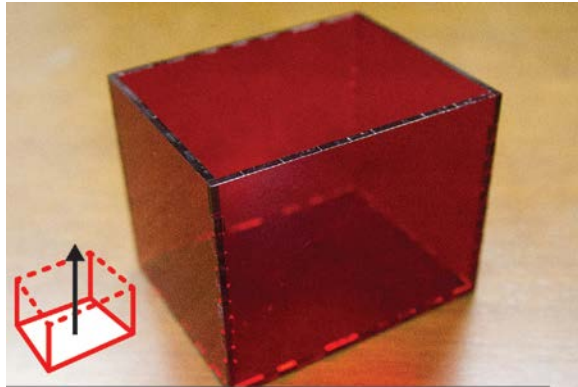
The screenshot displays the f3.js module repository interface. The top navigation bar includes 'f3.js', 'Get Started', 'IoT Projects', and 'Modules'. The main content area shows a grid of hardware modules, each with an image, title, description, and 'Intel Edison' compatibility tag. The modules shown are:

- Grove Buzzer**: '16/9/18 8:50' - Intel Edison
- Grove Temperature Sensor**: '16/9/18 8:49' - Intel Edison
- Grove relay switch**: '16/9/18 8:46' - Intel Edison
- Grove LCD RGB Backlight**: '16/9/18 8:42' - Intel Edison

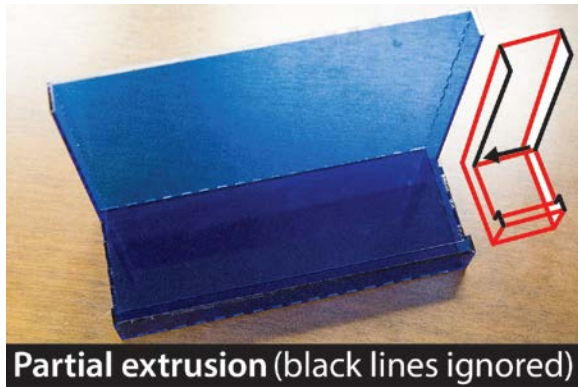
Two detailed views of the 'Grove Buzzer' module are shown on the right:

- Basic Information**: Shows the module name 'Grove Buzzer', driver 'pjsm, buzzer.Buzzer', author 'Jan Kato', and update date '16/9/18 8:50'.
- Layout information**: Provides the source code for the module layout in JavaScript and a diagram of the module's physical layout with pins labeled 'Buzzer'.

APIs for 3D extrusion and 2D layout

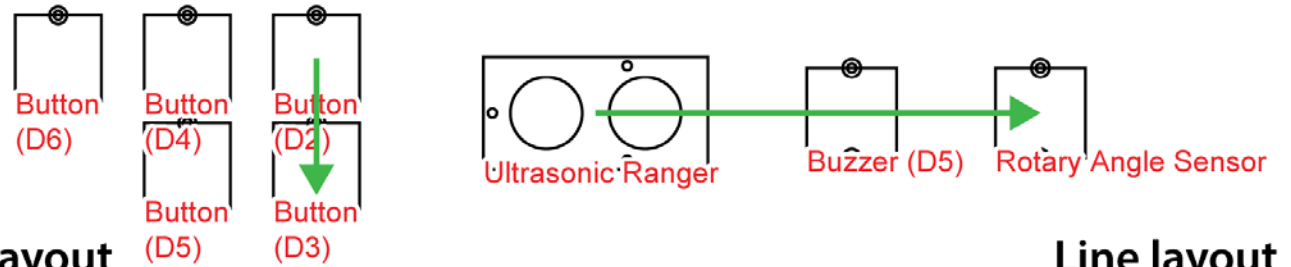


Extrusion (w/o opposite plane)



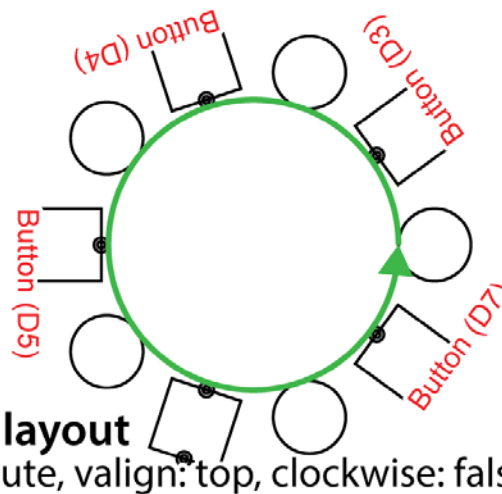
Partial extrusion (black lines ignored)

3D extrusion

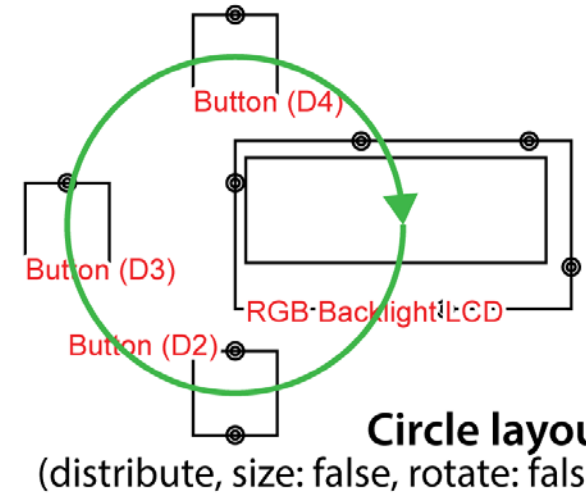


Line layout
(align, padding: 25mm, wrap: true)

Line layout
(distribute, rotate: false)



Circle layout
(distribute, valign: top, clockwise: false)



Circle layout
(distribute, size: false, rotate: false)

2D layout

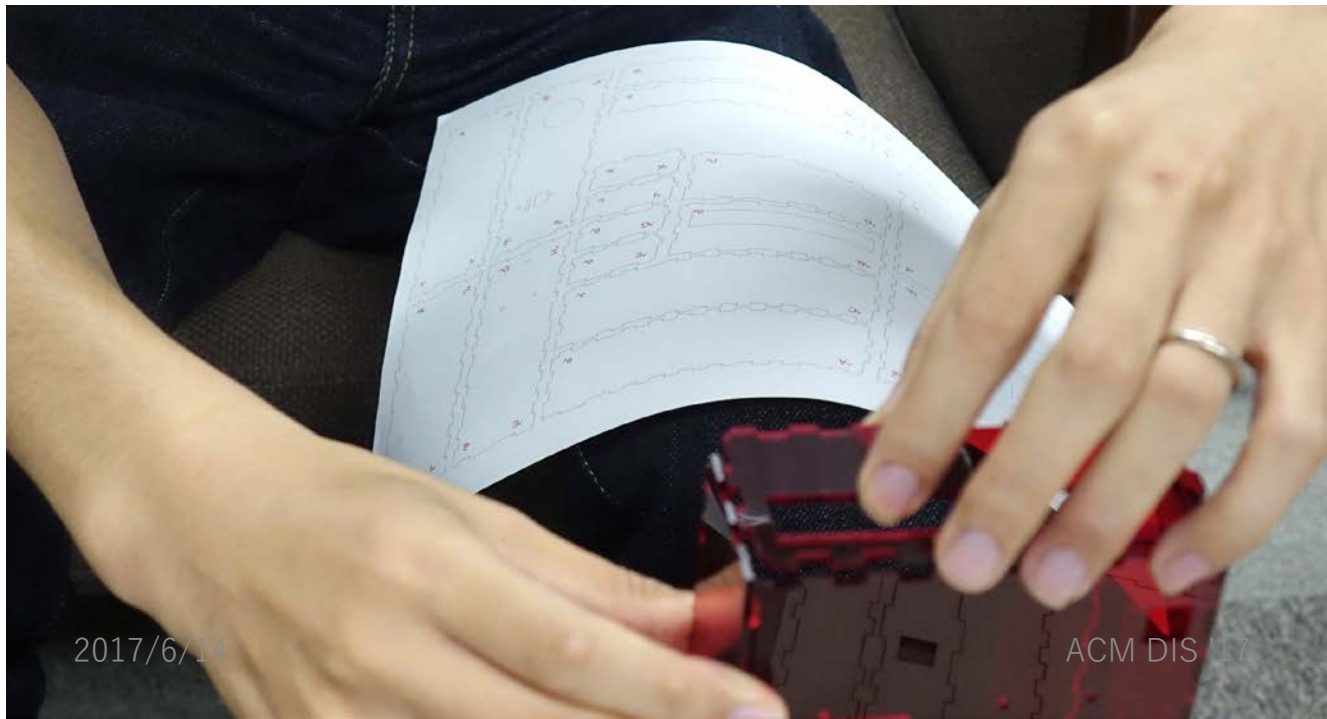
f3.js for parametric design of physical computing devices



- Typical **design patterns** should have tool support supported
- **Mental gap** between software & hardware exists addressed
- **Comparing alternatives** is crucial for good design supported

f3.js: customizing support for end-users

- **Interactive UIs** for customization
- Automatic generation of **device building instructions**



f3.js Get Started IoT Projects Modules

Simple blinking LED project

This page shows the project information created within f3.js (Simple blinking LED project).

Basic Information

Update

One or more LEDs blink at a certain interval.

Enclosure layout

Preview Page: 1

Customize

number of LEDs (2)

Layout view options

Show module names

Building Instructions

- 1 Purchase**
Purchase hardware components
- 2 Print & Assemble**
Print out enclosure components
- 3 Install**
Install and run the program

Modules

2 Grove LED

Layout

Printer type
Laser cutter (Trotec R...)

Labels near adjacent edges

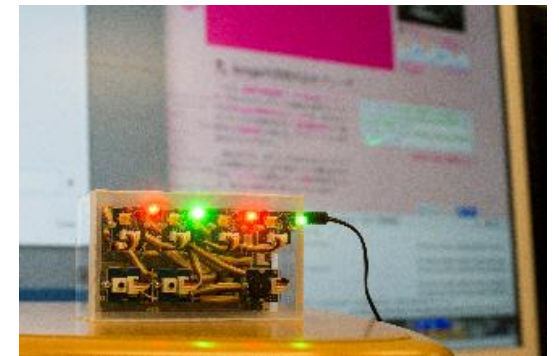
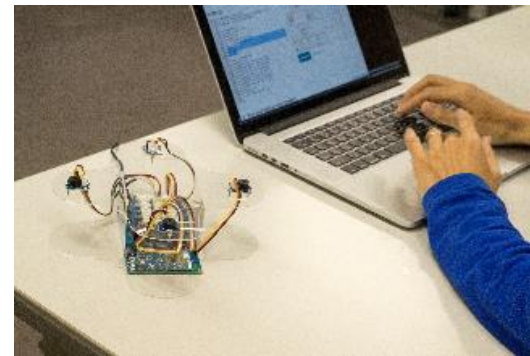
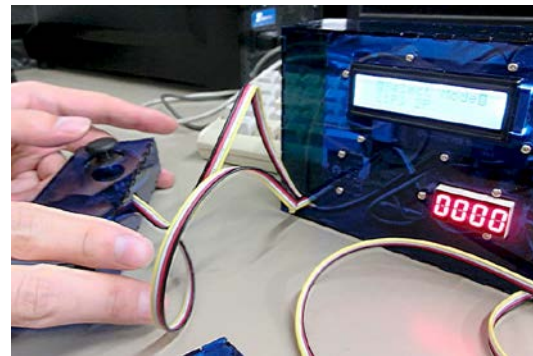
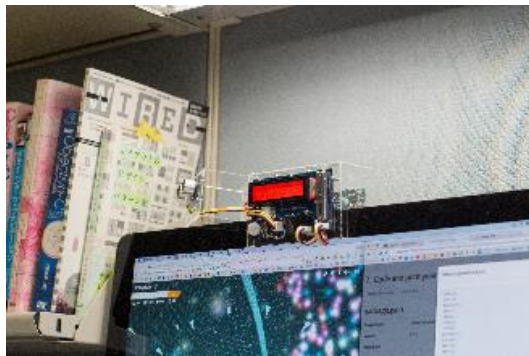
Program

Archive type
TAR

Program

User studies

- 14 teams to create physical computing devices with f3.js
 - 5 interaction designers and 16 university students
 - Intel Edison and Grove modules, acrylic panels and screws provided
- 3 interaction designers and 3 end-users with revised f3.js
 - 3 interaction designers asked to create parametric designs
 - 3 end-users asked to customize and assemble devices



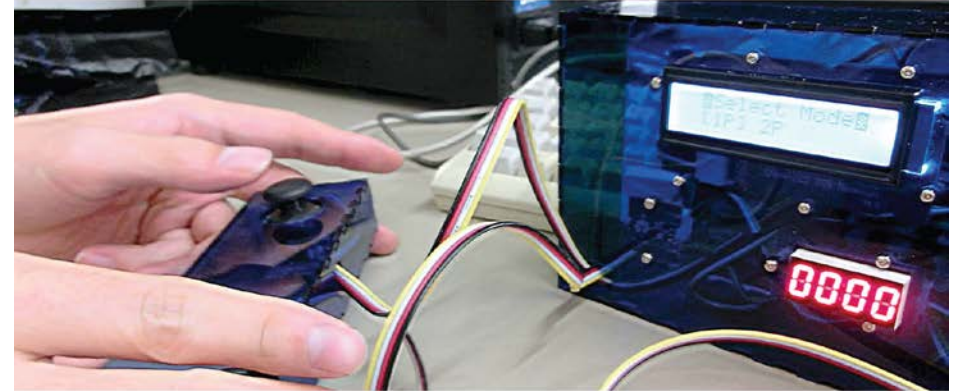
User studies: results & discussions

- Creativity support environments, not tools
- 3D vs 2D layout managers
- Interface builders are important
- Code-centric tool complements to 3D modeling tools
- Domain-specific language support (like HTML)

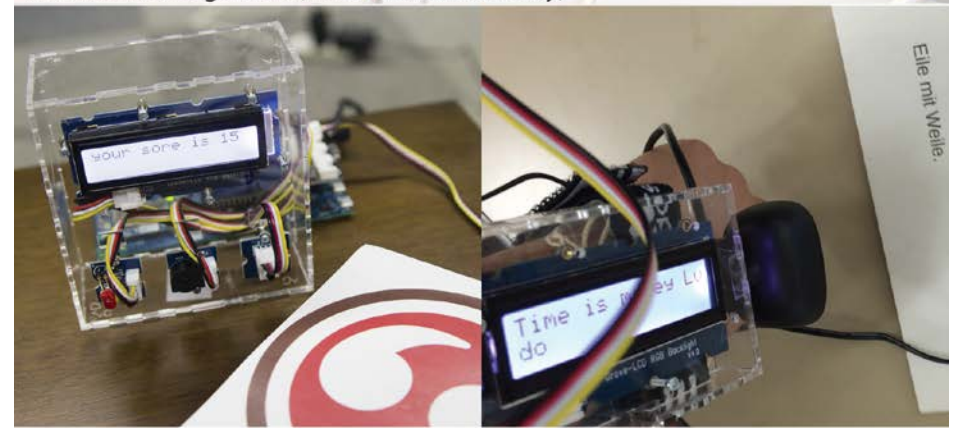


(Device assembly with instructions)

B. Music Kiosk (from the first study)



C. Card Matching Game (from the second study)



D. Rhythm Game (from the third study)

D. Translator (from the second study)



f3.js: A Parametric Design Tool for Physical Computing Devices for Both Interaction Designers and End-users

Source Code

Provide the source code of a microcontroller or tiny computer in JavaScript. Node.js-based computers are supported. Require f3.js package and use its API to design the device enclosure.

```
1  /** LED blinking app for Intel Edison */
2
3  var n = 2 // number of LEDs [1,4]
4    , width = 130
5    , height = 105
6    , thickness = 45;
7
8  // for building the enclosure layout
9  var f3js = require('f3js')
10 , c = f3js.createContainer()
11 , rect = c.drawJointRectangle(0, 0, width, height)
12 , line = c.drawLine(50, height/2 - 5, width - 30, height/2 - 5);
13
14 // for blinking LEDs
15 var groveDriver = require('jsupm_grove')
16 , leds = [];
17
18 // use this line as the guide path
19 line.layout = { name: 'distribute', rotate: false };
20
21 for (var i = 0; i < n; i++) {
22   var led = new groveDriver.GroveLed(i+2);
23
24   // put an LED module
25   var ledc = c.add(led, line);
26
27   // open a hole for the wire
28   ledc.drawRectangle(- 10, 15, 20, 10);
29
30   // start blinking the LED (details omitted)
31   leds.push(led);
32   setTimeout(function () { return function () {
33     1.handler = setInterval(blink(1), 1000);
34     1(led).i = 100;
35   }
36 }
37 }
38 }
```

Layout

Selected: Rectangle
Hovered: -
Page: 1

Customization

number of LEDs (2)

Layout view options

Update Delete

Preview

Page: 1

Customize

number of LEDs (2)

Layout view options

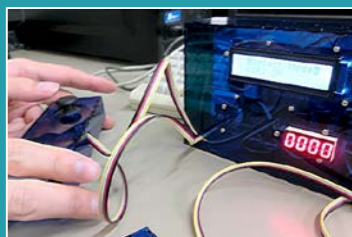
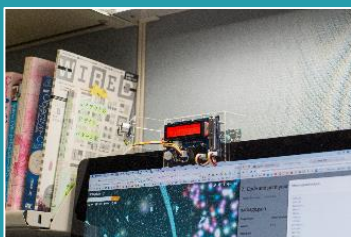
Show module names
Show labels near adjacent edges
Show warnings on module interferences

Disabled items are those proposed by somebody with the "O" button and not yet implemented. Got interested?

Building Instructions

- 1 Purchase**
Purchase hardware components
- 2 Print & Assemble**
Print out enclosure components
- 3 Install**
Install and run the program

Modules required for assembly
Layout
Program



Jun Kato, Masataka Goto

The presented system is publicly available at

<http://f3js.org>