



Form Follows Function()

形状と機能を単一コードで開発できる統合開発環境

WISS' 15 (3-R07)

加藤 淳, 後藤 真孝
産業技術総合研究所

研究のポイント

- JavaScriptを書けば、マイコンの振る舞いだけでなく筐体設計まで完成する**統合開発環境**
- GUI アプリ開発のように、筐体表面にセンサやアクチュエータを容易に配置できる**API**
- エンドユーザでも、用途に合わせて機能と形状をカスタマイズできる**GUIウィジェット**

統合開発環境

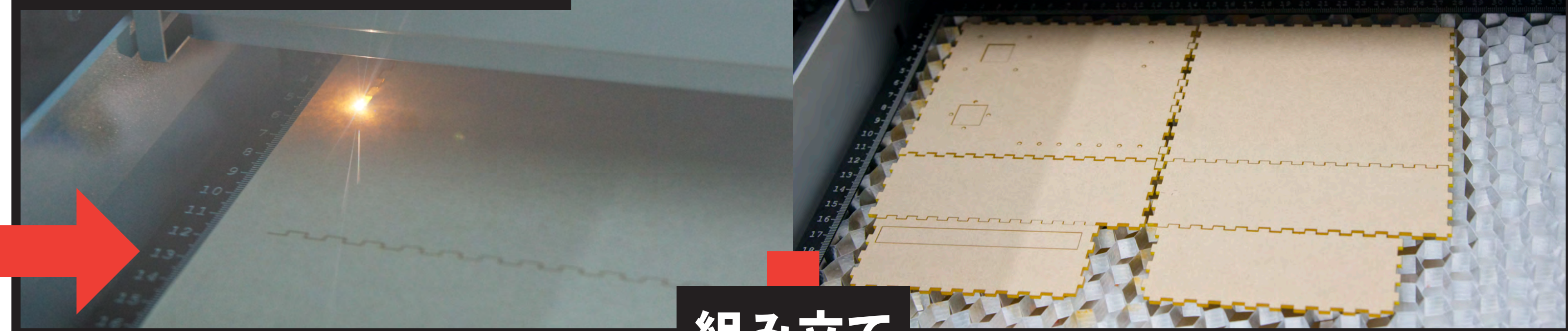
index.js

```
// instantiate sensors and actuators
11 var groveSensor = require('jsupm_grove');
12 var gcl = require('jsupm_grovecircularled');
13 var circle = new gcl.GroveCircularLED(5, 4);
14
15 // set layout parameters
16 var fff = require('fff-print')
17 , x = 10
18 , y = 10
19 , width = 130
20 , height = 105
21 , thickness = 42
22 , dw = 6 /* Dovetail width [0, 10] */
23 , dh = 2 /* Dovetail height [0, 10] */
24 , useCountdown = true /* Use countdown */;
25
26 // put base board
27 fff.drawDovetailRectangle(x, y, width, height, dw, dh, true);
28 fff.drawDovetailRectangle(x + width + dh, y, width, height, dw, dh, true);
29
30 // put sensors and actuators
31 var leftMargin = 28 // Left margin [0, 100]
32 , topMargin = 25;
33 fff.add(camera, x + leftMargin, y + topMargin);
34 fff.add(circle, x + leftMargin, y + 75);
35 var button = null;
36 if (useCountdown) {
37   button = new groveSensor.GroveButton(3);
38   fff.add(button, x + width - 20, y + topMargin);
39   fff.drawRectangle(x + width - 20 - 6, y + topMargin - 18, 12, 3);
40 }
41
42 // draw circles...
43 for (var i = 0; i < 7; i++) {
44   fff.drawCircle(
45     x + width - (i + 1) * 10
46     , y + height - 10
47     , 1); // additional holes
48 }
49
50 // put side boards
51 var margin = dh;
52 fff.drawDovetailRectangle(
53   x, y + height + margin - dh,
54   width, thickness, dw, dh,
55   true, false);
56 fff.drawDovetailRectangle(
57   x + width + dh, y + height + margin - dh,
58   width, thickness, dw, dh,
59   false, true);
```

プレビュー

マイコン用ファームウェアとしてそのまま転送

レーザーカッターで切断



組み立て



完成!!!

既存手法との比較

- | | | |
|------------------------------------------------|---|--------------------|
| • ソースコードと配線可視化の双方向編集 (.NET Gadgeteer) | ↔ | 配線設計に留まらず |
| • GUI で配線・振る舞いシミュレーション (Autodesk 123D Circuit) | ↔ | エンドユーザが触るUI 設計まで支援 |
| • 配線を手描きしてモジュールを配置できる開発環境 (PaperPulse) | ↔ | プログラミングを用いることで |
| • センサやアクチュエータから筐体を自動生成できる CAD (Enclosed) | ↔ | 詳細に再利用性の高い設計が可能 |

API・GUI ウィジェット

- ・ センサやアクチュエータのネジ穴などのレイアウト情報を提供（ユーザが拡張可能）

```
var button = new gs.GroveButton(2); // もともと button.getValue() などが使える  
fff.add(button, 50, 40); // これで座標 (50, 40) にボタンを配置できるようネジ穴が開く
```

- ・ 立体形状を定義すれば、複数の切断面を、繋ぎ目のギザギザも含め自動的に生成

```
var rect = fff.drawRectangle(5, 5, 60, 20); var planes = rect.extrude(50);  
// 矩形を 50mm 押し出した立体形状を作り、6 面を生成
```

- ・ 変数にコメントをつけると、機能と筐体をカスタマイズできるGUIウィジェットが出現

```
var useCountdown = true; // カウントダウン機能を使う  
if (useCountdown) { /* 変数値次第で機能と筐体レイアウト両方を差し替え可能 */ }
```



<http://f3js.org> で一般公開予定！（デモ動画など情報配信中）

仕様

※灰色は今後実装予定・検討中の機能

対応マイコン：Intel Edison, Raspberry Pi, Tessel.io, Arduino Yun

対応モジュール：Grove システム, ユーザ向け GitHub リポジトリ公開予定

OS: Windows 7/8/8.1/10, Mac OS X, Linux

