

1. Client

a_OOCSI_node 控制網路連接，以及邏輯撰寫，b_light_policy 控制燈光與音訊檔播放，c_audio 控制音訊檔播放底層，d_methods 控制燈光模組底層。

1-1 燈光 pattern 設定

1-1-1 燈光模組宣告

為了使用 I2C 訊號輸出，需要先 import `processing.io.*;` (如下圖)。接著宣告 Methods 物件為 m。

```
main a_OOCSI_node b_light_policy c_audio d_methods ▾
1 import processing.io.*;
2
3 class Policy{
4     ArrayList<ArrayList<Integer>> channels = new ArrayList<ArrayList<Integer>>();
5     ArrayList<ArrayList<Integer>> ports = new ArrayList<ArrayList<Integer>>();
6
7     //m is class where different light behaviours are, behaviours can be combined
8     Methods m = new Methods();
9     Audio audio = new Audio();
```

1-1-2 燈光模組初始化

呼叫 `setChennals()` 函式。

```
10
11 void m_setup(){
12     setChennals();
13     m.alloff();
14 }
```

此初始化過程與後面燈光變化時，會需要模板中幾個必要的宣告，只要記得複製貼上這幾個部分就可以：`chennals` 與 `ports` 變數宣告、`setChennals()` 與 `setPorts()` 函式宣告。

```
main a_OOCSI_node b_light_policy c_audio d_methods ▾
1 import processing.io.*;
2
3 class Policy{
4     ArrayList<ArrayList<Integer>> channels = new ArrayList<ArrayList<Integer>>();
5     ArrayList<ArrayList<Integer>> ports = new ArrayList<ArrayList<Integer>>();
6
7     //m is class where different light behaviours are, behaviours can be combined
8     Methods m = new Methods();
9     Audio audio = new Audio();
```

```

169 void setChennals(){
170     //an arraylist of all lights | channels has been set as the default arraylist, where i
171     //if there are any change in number if lights used, i < 42 will be changed to i < ?, w
172     for (int i=6; i<42; i+=4){
173         ArrayList<Integer> temp = new ArrayList<Integer> ();
174         temp.add(i);
175         temp.add(i+1);
176         temp.add(i+2);
177         temp.add(i+3);
178         channels.add(temp);
179     }
180 }
181 // if not all lights are to be used, this method can be called to create an arraylist of
182 void setPorts(int[] pins){
183     ports.clear();
184     for(int i: pins){
185         ArrayList<Integer> temp = new ArrayList<Integer> ();
186         for(int j=0; j<channels.get(i).size(); j++){
187             temp.add(channels.get(i).get(j));
188         }
189         ports.add(temp);
190     }
191 }

```

1-1-3 燈光 pattern 設計

可自令函式名稱，接著用三個步驟實作每個動作，一是宣告要改變燈光的輸出位置為 p，二是使用 setPorts(p); 設定輸出，三是依照 Methods 提供的函式要求燈光變換 (詳見附錄一)。舉例來說，想要一個會有叉叉閃爍的燈號，使用 blink_more 來設定，多個燈號會重複閃爍 2 次，每次間隔 200 毫秒，最大亮度會達到 4000 (最大值為 4096)。

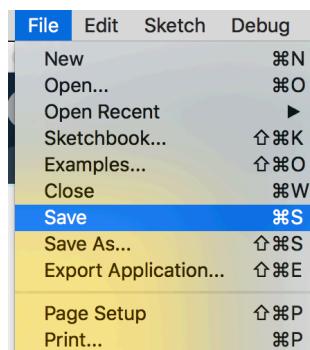
```

102 void CrossBlink(){
103     int[] p = {0,2,4,6,8};
104     setPorts(p);
105     m.blink_more(ports,2,200,4000);
106 }

```

1-1-4 測試範例

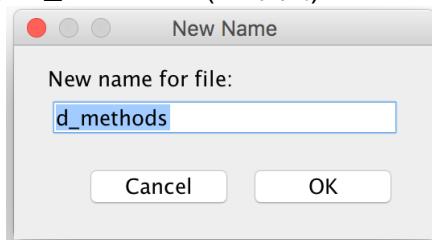
example_light 範例程式中，示範如何設計叉叉閃爍的燈號，每閃兩次叉叉的符號會休息一次。首先，建立一個新的 Processing File，點擊上方工具列的「File」，接著點擊「New」。然後儲存這個新建立的專案，點擊上方工具列的「File」，接著點擊「Save」。



接著在分頁列點選向下的三角形(如下圖)，選擇「New Tab」。



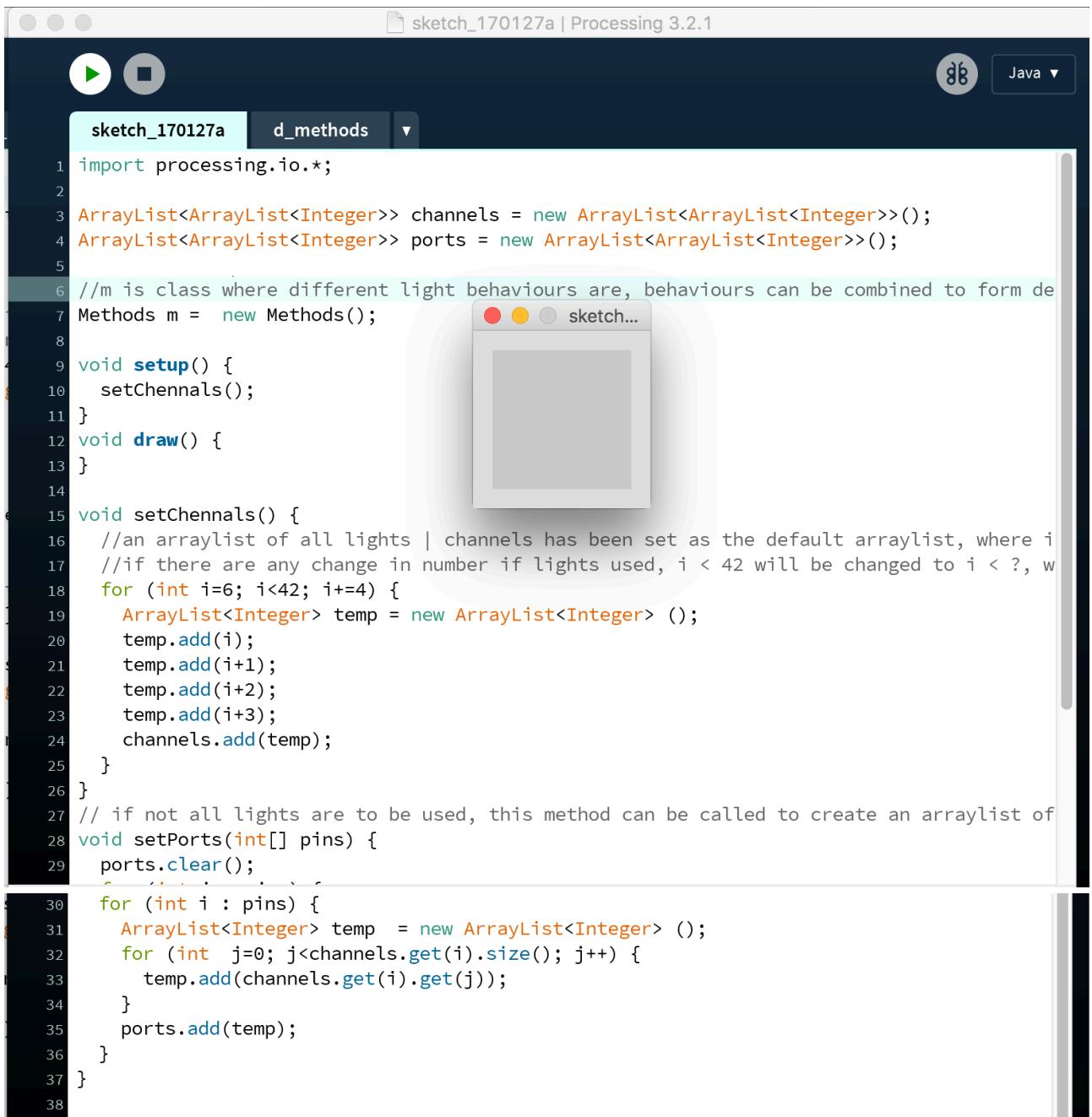
這邊可以任意取名，但是依照習慣，會與裡頭所要裝的物件名稱相同，加上要為物件做子母關係排序，會以字母作為開頭，因此這邊命名要給 Methods 物件的 Tab 為 “d_methods” (如下圖)。



此新分頁中，只需完整複製貼上範例中的 Method 物件。此時會出現 I2C 的錯誤，因為整個專案尚未匯入 processing.io，下一步就是回到原本分頁，加入 import processing.io;。

```
sketch_170127a      d_methods ▾  
1 class Methods {  
2  
3     ArrayList<ArrayList<Integer>> portList = new ArrayList<ArrayList<Integer>>();  
4     ArrayList <Integer> p = new ArrayList<Integer>();  
5  
6     int PWMport = 0x40;  
7     int brightness =0;  
8     int max_brightness = 0;  
9     int paused= 0;  
10    I2C i2c;  
11  
12  
13    Methods() {  
14  
15        //initialise I2C  
16        this.i2c = new I2C(I2C.list()[0]);  
17  
18        //ensure I2C channels are available for writing  
19        i2c.beginTransmission(0x40);  
20        i2c.write(0);  
21        i2c.write(100);  
22        i2c.endTransmission();  
23    }  
24  
25  
26  
27    //switch all lights off  
28    void alloff() {  
29        IntList p = new IntList();
```

主分頁當中依照前面的步驟，宣告並初始化燈光模組，所以會如下所示。
`setup()` 僅在程式開始時會執行一次，因此設定或初始化的函式會放置於此，
`draw()` 函式則會接著不斷循環執行，所以把想要閃爍與休息的指令放置於此。
在左上角點擊向右的三角形，執行 Run，確認可以執行之後，就可以開始設計燈光。



```
sketch_170127a | Processing 3.2.1
Java ▾

sketch_170127a d_methods ▾

1 import processing.io.*;
2
3 ArrayList<ArrayList<Integer>> channels = new ArrayList<ArrayList<Integer>>();
4 ArrayList<ArrayList<Integer>> ports = new ArrayList<ArrayList<Integer>>();
5
6 //m is class where different light behaviours are, behaviours can be combined to form de
7 Methods m = new Methods();
8
9 void setup() {
10     setChennals();
11 }
12 void draw() {
13 }
14
15 void setChennals() {
16     //an arraylist of all lights | channels has been set as the default arraylist, where i
17     //if there are any change in number if lights used, i < 42 will be changed to i < ?, w
18     for (int i=6; i<42; i+=4) {
19         ArrayList<Integer> temp = new ArrayList<Integer> ();
20         temp.add(i);
21         temp.add(i+1);
22         temp.add(i+2);
23         temp.add(i+3);
24         channels.add(temp);
25     }
26 }
27 // if not all lights are to be used, this method can be called to create an arraylist of
28 void setPorts(int[] pins) {
29     ports.clear();
30     for (int i : pins) {
31         ArrayList<Integer> temp = new ArrayList<Integer> ();
32         for (int j=0; j<channels.get(i).size(); j++) {
33             temp.add(channels.get(i).get(j));
34         }
35         ports.add(temp);
36     }
37 }
```

這裡使用 1-1-3 的例子，使用設計好的 CrossBlink()，每閃兩次叉叉的符號，再用 delay(1000) 休息一秒。此外，範例檔在 draw() 裏頭使用 try 和 catch，用以避免程式錯誤中斷執行，遇見錯誤時，只會印出錯誤資訊，不會造成程式中斷。

完整主分頁程式應如下圖所示，同樣執行 Run 確認程式碼正確無誤。

```

example_light d_methods ▾
1 import processing.io.*;
2
3 ArrayList<ArrayList<Integer>> channels = new ArrayList<ArrayList<Integer>>();
4 ArrayList<ArrayList<Integer>> ports = new ArrayList<ArrayList<Integer>>();
5
6 //m is class where different light behaviours are, behaviours can be combined to form de
7 Methods m = new Methods();
8
9 void setup() {
10     setChennals();
11 }
12
13 void draw() {
14     try {
15         CrossBlink(); // Task reject
16         delay(1000);
17     }
18     catch (Exception e) {
19         println("Error in draw(): ", e);
20     }
21 }
22
23 void CrossBlink() {
24     int[] p = {0, 2, 4, 6, 8};
25     setPorts(p);
26     m.blink_more(ports, 2, 200, 4000);
27 }
28
29 void setChennals() {
30     //an arraylist of all lights | channels has been set as the default arraylist, where i
31     //if there are any change in number if lights used, i < 42 will be changed to i < ?, w
32     for (int i=6; i<42; i+=4) {
33         ArrayList<Integer> temp = new ArrayList<Integer> ();
34         temp.add(i);
35         temp.add(i+1);
36         temp.add(i+2);
37         temp.add(i+3);
38         channels.add(temp);
39     }
40 }
41 // if not all lights are to be used, this method can be called to create an arraylist of
42 void setPorts(int[] pins) {
43     ports.clear();
44     for (int i : pins) {
45         ArrayList<Integer> temp = new ArrayList<Integer> ();
46         for (int j=0; j<channels.get(i).size(); j++) {
47             temp.add(channels.get(i).get(j));
48         }
49         ports.add(temp);
50     }
51 }
52

```

1-2 聲音設定

1-2-1 音訊播放器宣告

宣告 Audio 物件。

```
1 class Policy{  
2     ArrayList<ArrayList<Integer>> channels = new ArrayList<ArrayList<Integer>>();  
3     ArrayList <ArrayList<Integer>> ports = new ArrayList<ArrayList<Integer>>();  
4  
5     Methods m;  
6     Audio audio = new Audio();  
7  
8 }  
9
```

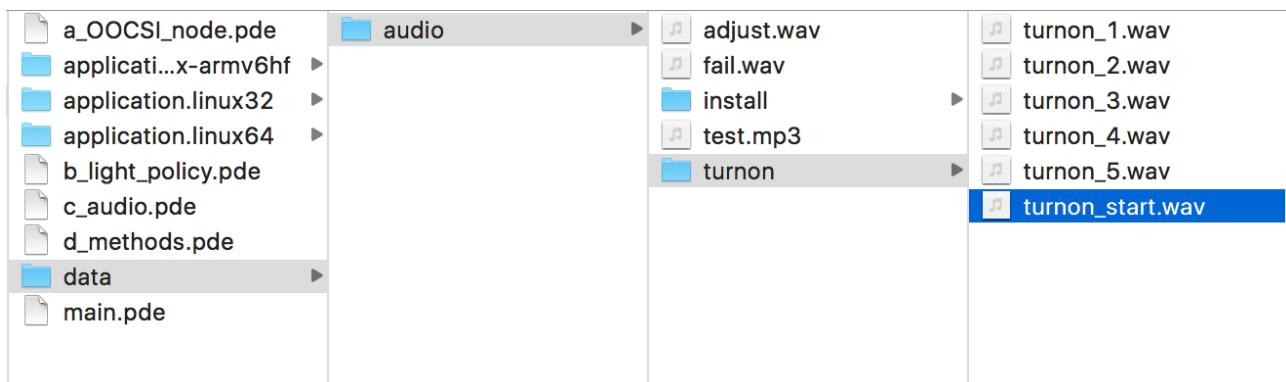
1-2-2 音訊播放

使用 init("FILE_PATH") 播放，後面記得使用 delay 休息，避免多個音訊檔同時播放，彼此干擾。

```
1 Audio audio = new Audio();  
2  
3 void setup() {  
4 }  
5  
6 void draw() {  
7     try {  
8         audio.init("turnon/turnon_start.wav");  
9         delay(1000);  
10    }  
11    catch (Exception e) {  
12        println("Error in draw(): ", e);  
13    }  
14 }  
15 }  
16
```

1-2-3 音訊檔位置

放在 data 資料夾底下的 audio，舉例來說，“turnon/turnon_start.wav”即指 /data/audio/turnon/turnon_start.wav (如下圖)。



1-2-4 藍牙喇叭設定

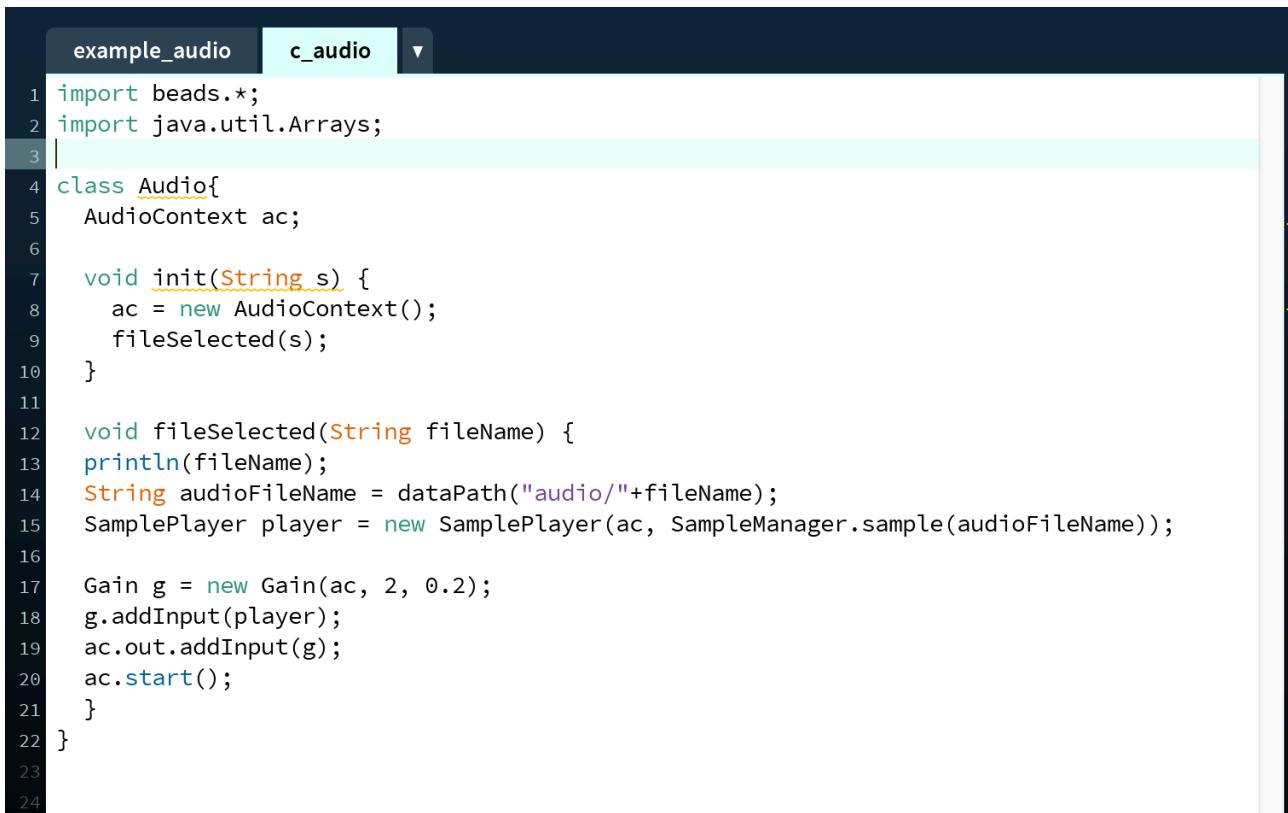
第一次使用，請安裝相關套件，以及設定環境，可參考附件二。

開啟藍牙後，連線至指定喇叭，之後以該音訊輸出裝置播放。

1-2-5 測試範例

example_audio 範例程式中，示範如何播放音訊“turnon_start.wav”。

首先，建立一個新的 Processing File，然後儲存這個新建立的專案。依照類似 1-1-4 的步驟，加入新的 Tab 名為 c_audio，並複製貼上 Audio 物件(如下圖)。確認可以執行 Run 後，回到主分頁準備進行播放器設定與播放測試。



```
1 import beads.*;
2 import java.util.Arrays;
3
4 class Audio{
5     AudioContext ac;
6
7     void init(String s) {
8         ac = new AudioContext();
9         fileSelected(s);
10    }
11
12    void fileSelected(String fileName) {
13        println(fileName);
14        String audioFileName = dataPath("audio/"+fileName);
15        SamplePlayer player = new SamplePlayer(ac, SampleManager.sample(audioFileName));
16
17        Gain g = new Gain(ac, 2, 0.2);
18        g.addInput(player);
19        ac.out.addInput(g);
20        ac.start();
21    }
22}
23
24
```

如同 1-2-3 的音訊檔放置說明，創建 data, audio 與 turnon 資料夾，並將“turnon_start.wav”放置在指定位置下以後，主分頁只要如下圖所示的宣告與呼叫，就能順利執行播放！

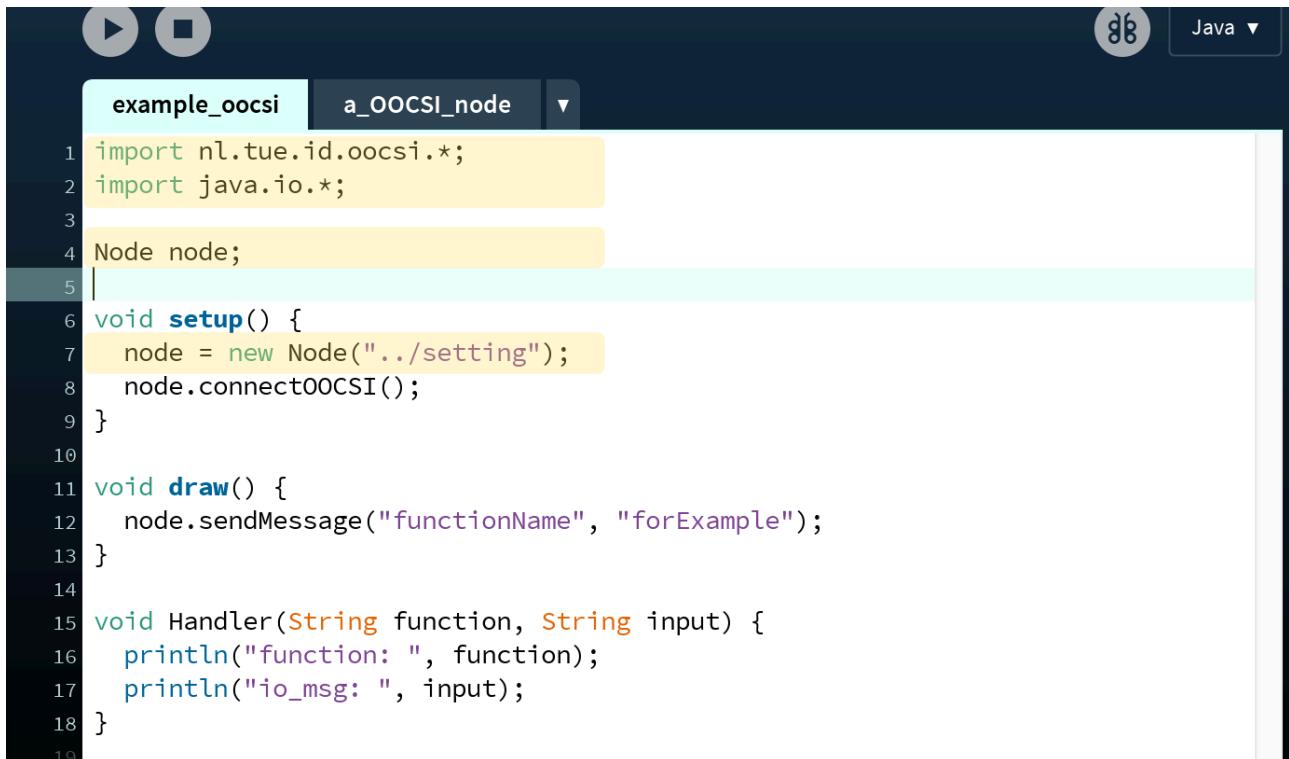


```
1
2 Audio audio = new Audio();
3
4 void setup() {
5 }
6
7 void draw() {
8     try {
9         audio.init("turnon/turnon_start.wav");
10        delay(1000);
11    }
12    catch (Exception e) {
13        println("Error in draw(): ", e);
14    }
15 }
```

1-3 OOCSI 網路設定

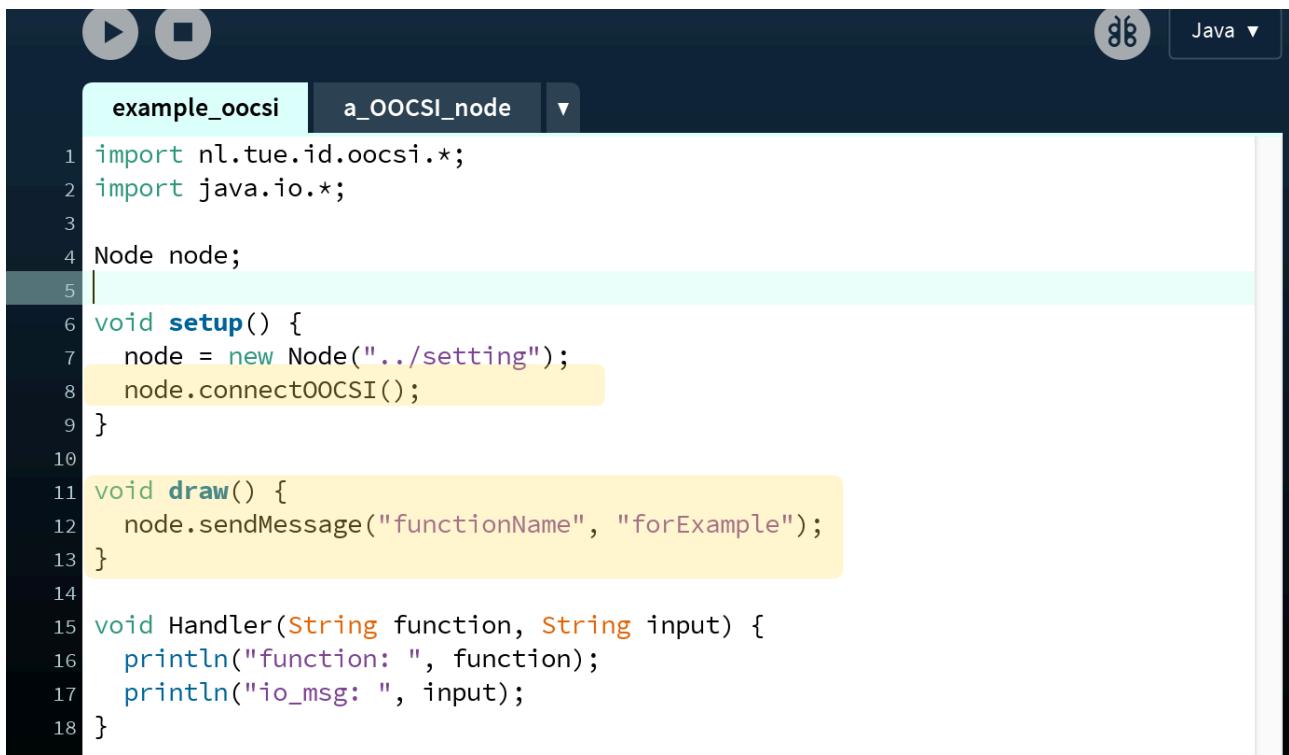
1-3-1 OOCSI_Client 設定

為了使用 OOCSI 套件連線，需要先 import nl.tue.id.oocsi.*; 與 import java.io.*; (如下圖)。接著宣告 Node 物件為 node，後面字串表示 setting 設定檔所在位置，「..../setting」表示上層資料夾中名為「setting」的檔案。



```
example_oocsi a_OOCSI_node ▾
1 import nl.tue.id.oocsi.*;
2 import java.io.*;
3
4 Node node;
5
6 void setup() {
7     node = new Node("../setting");
8     node.connectOOCSI();
9 }
10
11 void draw() {
12     node.sendMessage("functionName", "forExample");
13 }
14
15 void Handler(String function, String input) {
16     println("function: ", function);
17     println("io_msg: ", input);
18 }
```

之後在 setup() 區域進行連線。為了測試，在 draw() 區域循環送出訊息，送出訊息的第一個變數是指令名稱，第二個變數放入指令內容。



```
example_oocsi a_OOCSI_node ▾
1 import nl.tue.id.oocsi.*;
2 import java.io.*;
3
4 Node node;
5
6 void setup() {
7     node = new Node("../setting");
8     node.connectOOCSI();
9 }
10
11 void draw() {
12     node.sendMessage("functionName", "forExample");
13 }
14
15 void Handler(String function, String input) {
16     println("function: ", function);
17     println("io_msg: ", input);
18 }
```

要設計訊息接收後的動作，可以在 Handler 裡面處理，第一個變數同樣是指令名稱，第二個變數是指令內容。

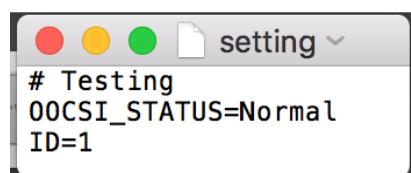


```
1 import nl.tue.id.oocsi.*;
2 import java.io.*;
3
4 Node node;
5
6 void setup() {
7     node = new Node("../setting");
8     node.connectOOCSI();
9 }
10
11 void draw() {
12     node.sendMessage("functionName", "forExample");
13 }
14
15 void Handler(String function, String input) {
16     println("function: ", function);
17     println("io_msg: ", input);
18 }
```

1-3-2 setting 設定檔

此版本的設定檔，僅設計兩個變數控制，若該行開頭有井字號或不是這兩個變數，將會忽略不處理，放置位置則視物件宣告時的第一個變數，由變數所指定的位置而定。

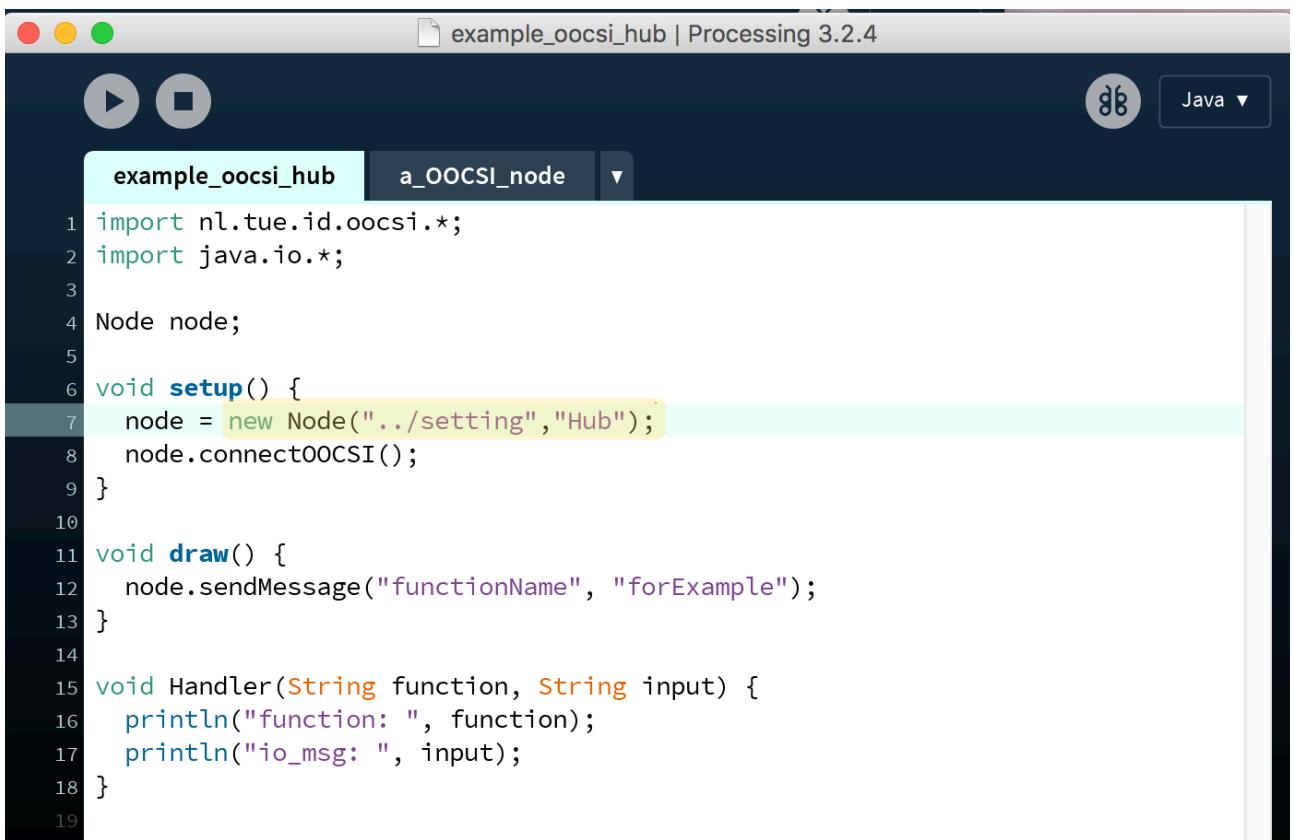
- OOCSI_STATUS: 「Normal」用以模擬正常連線狀況，「Unconnected」則是模擬無法連線的狀況。
- ID: 表示連上物聯網的物品 ID，同一物聯網下不應有任何重複 ID。



1-3-3 Hub 設定

只需要在初始化時，給予第二個變數「Hub」(如下圖)，就可以作為 Hub 設計，所有物聯網下的物品都只能傳訊息給它，也只會處理它所發出的指令。此時第一個字串變數的位置，只由「OOCSI_STATUS」控制這個 Hub 連線與否，「ID」不會影響。

執行時要先讓 Hub 連線，其他的再執行連線，全部開始執行後，再把 Hub 重新啟動，才能順利送出位於 setup() 內的第一個指令給所有物品。



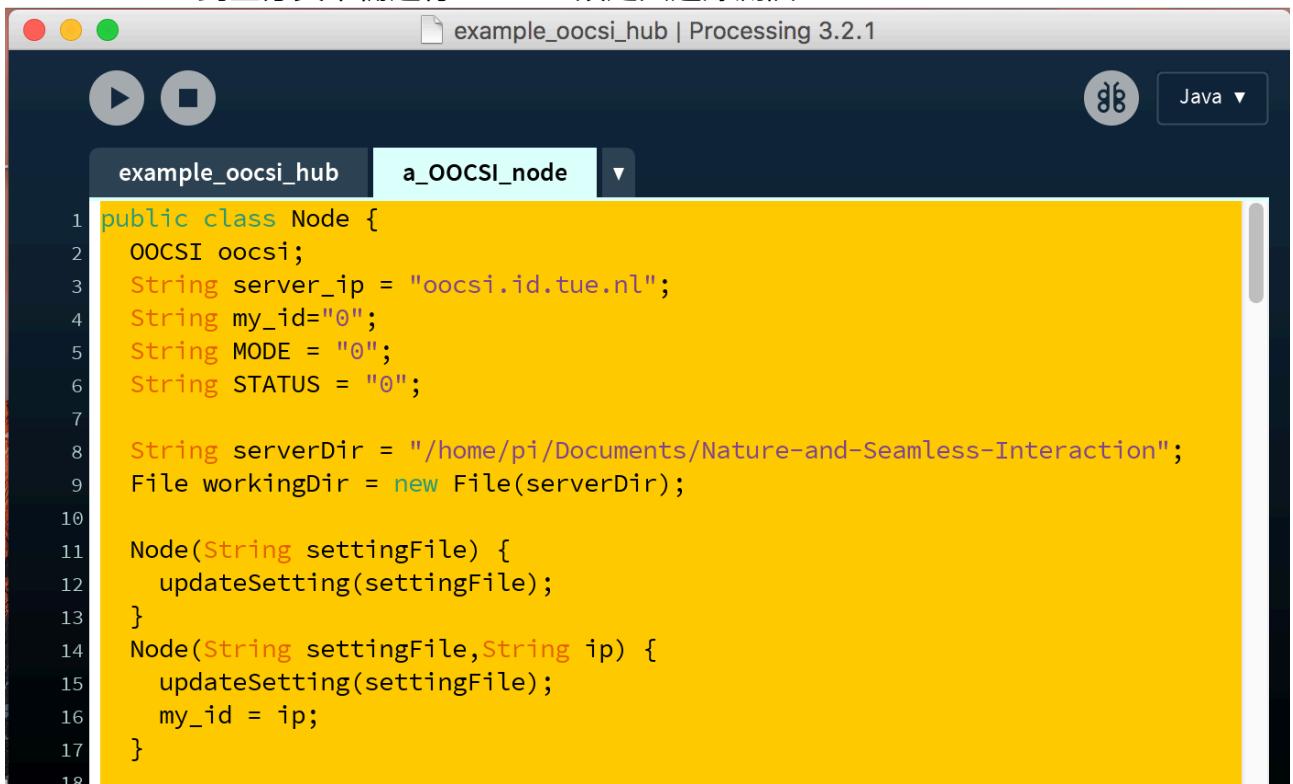
```
example_oocsi_hub | Processing 3.2.4
Java ▾

example_oocsi_hub a_OOCSI_node ▾

1 import nl.tue.id.oocsi.*;
2 import java.io.*;
3
4 Node node;
5
6 void setup() {
7     node = new Node("../setting", "Hub");
8     node.connectOOCSI();
9 }
10
11 void draw() {
12     node.sendMessage("functionName", "forExample");
13 }
14
15 void Handler(String function, String input) {
16     println("function: ", function);
17     println("io_msg: ", input);
18 }
19
```

1-3-3 單物品測試範例

example_oocsi 與 example_oocsi_hub 範例程式中，示範 Hub 如何與其他物品溝通。首先，建立一個新的 Processing File，然後儲存這個新建立的專案為 example_oocsi_hub。依照類似 1-1-4 的步驟，加入新的 Tab 名為 a_OOCSI_node，並複製貼上 Node 物件(如下圖)。確認可以執行 Run 後，回到主分頁準備進行 OOCSCI 設定與連線測試。



```
example_oocsi_hub | Processing 3.2.1
Java ▾

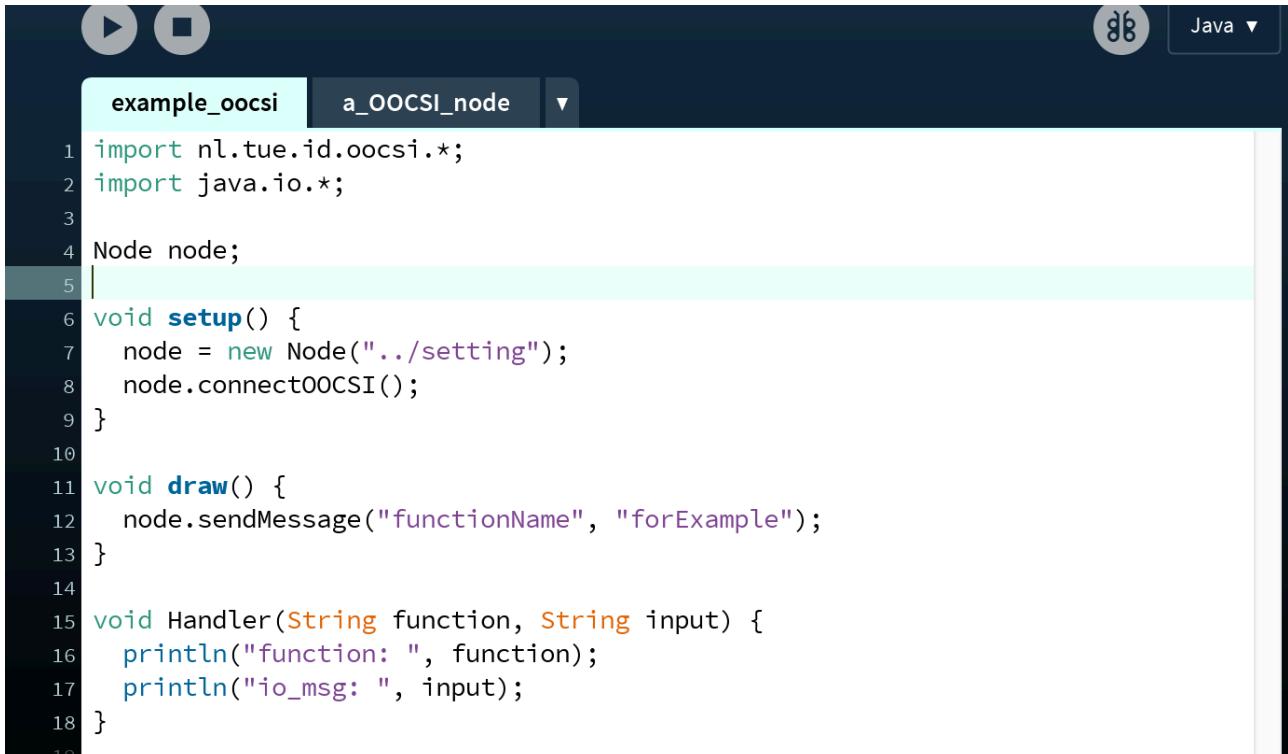
example_oocsi_hub a_OOCSI_node ▾

public class Node {
    OOCSI oocsi;
    String server_ip = "oocsi.id.tue.nl";
    String my_id="0";
    String MODE = "0";
    String STATUS = "0";

    String serverDir = "/home/pi/Documents/Nature-and-Seamless-Interaction";
    File workingDir = new File(serverDir);

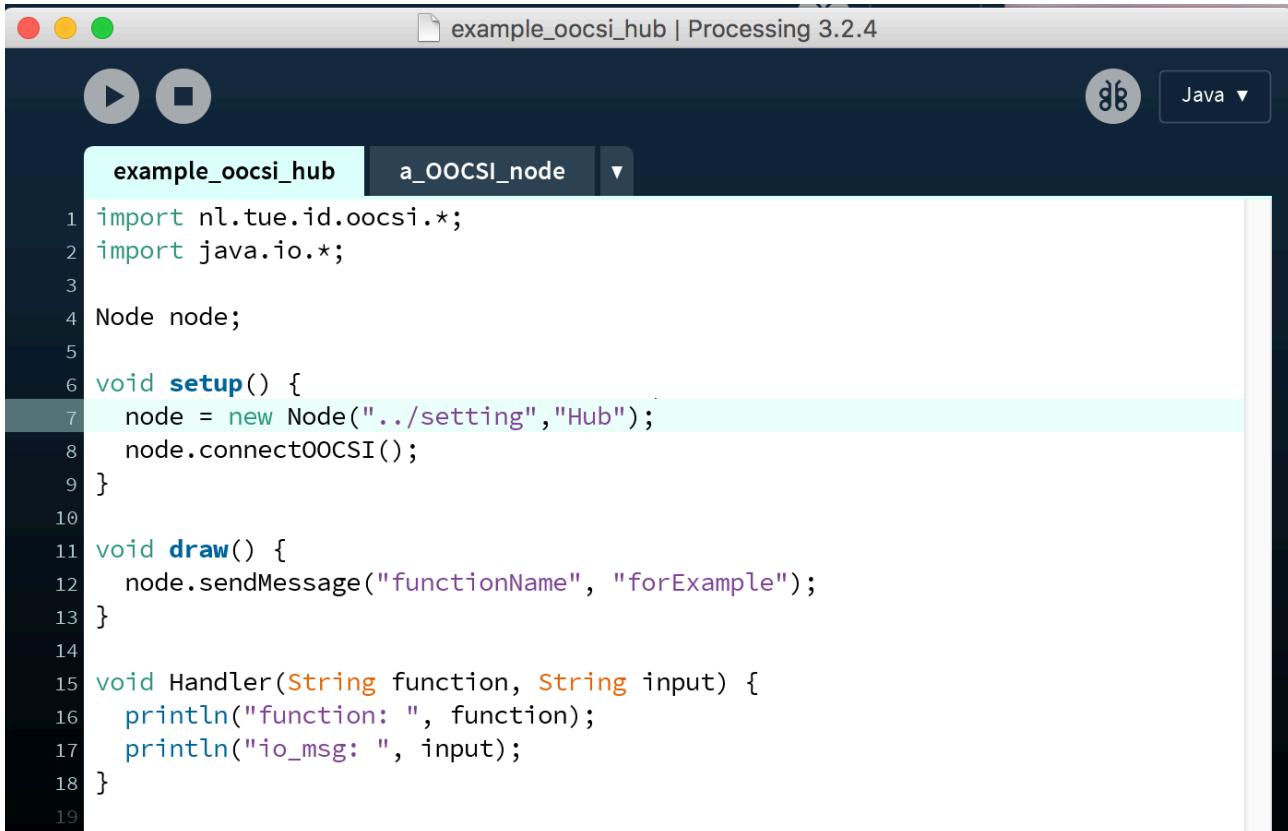
    Node(String settingFile) {
        updateSetting(settingFile);
    }
    Node(String settingFile, String ip) {
        updateSetting(settingFile);
        my_id = ip;
    }
}
```

example_oocsi 使用以下程式。



```
example_oocsi    a_OOCSI_node ▾
1 import nl.tue.id.oocsi.*;
2 import java.io.*;
3
4 Node node;
5
6 void setup() {
7     node = new Node("../setting");
8     node.connectOOCSI();
9 }
10
11 void draw() {
12     node.sendMessage("functionName", "forExample");
13 }
14
15 void Handler(String function, String input) {
16     println("function: ", function);
17     println("io_msg: ", input);
18 }
```

example_oocsi_hub 使用以下程式。



```
example_oocsi_hub    a_OOCSI_node ▾
1 import nl.tue.id.oocsi.*;
2 import java.io.*;
3
4 Node node;
5
6 void setup() {
7     node = new Node("../setting", "Hub");
8     node.connectOOCSI();
9 }
10
11 void draw() {
12     node.sendMessage("functionName", "forExample");
13 }
14
15 void Handler(String function, String input) {
16     println("function: ", function);
17     println("io_msg: ", input);
18 }
```

最後，先執行 example_oocsi_hub，再執行 example_oocsi，應該可以看到下面的輸出。

1-3-4 多物品測試範例

從 1-3-3 的 example_oocsi 修改並另存新檔為 example_multi_oocsi，只要在 Handler 使用 switch...case 判斷 node.my_id 就可以為多個物品設計好不同動作，卻使用同一份程式碼就夠，此處 node.my_id 指 setting 內會出現的 ID。

```
14 void Handler(String function, String input) {  
15     println("function: ", function);  
16     switch(node.my_id) {  
17         case "1":  
18             println("First item get input: ", input);  
19             break;  
20         case "2":  
21             println("Second item get input: ", input);  
22             break;  
23         case "3":  
24             println("Third item get input: ", input);  
25             break;  
26         default:  
27             println("No id: ", node.my_id);  
28     }  
29 }
```

Hub 使用和 1-3-3 相同的 example_oocsi_hub 即可，不同物品可以看到 Console 區塊，印出不一樣的回報。

1-4 邏輯撰寫

1-4-1 單物品 OOCSI 與燈號

example_oocsi_with_light 結合前面的測試範例，新增並貼上 a_OOCSI_node 與 d_methods 兩個分頁，分別包含 Node 與 Methods 物件。之後在主頁宣告並設定使用這兩個物件會需要的部分，如下圖所示，注意 draw() 已經沒有任何東西，因為所有動作都會由 hub 傳入的指令控制。

```
example_oocsi_with_light | Processing 3.2.4
Java ▾

example_oocsi_with_light a_OOCSI_node d_methods ▾

1 import nl.tue.id.oocsi.*;
2 import java.io.*;
3 import processing.io.*;

4
5 ArrayList<ArrayList<Integer>> channels = new ArrayList<ArrayList<Integer>>();
6 ArrayList<ArrayList<Integer>> ports = new ArrayList<ArrayList<Integer>>();

7 //m is class where different light behaviours are, behaviours can be combined
8 Methods m = new Methods();

9 Node node;

10
11 void setup() {
12     setChennals();
13     node = new Node("../setting");
14     node.connectOOCSI();
15 }

16
17 }

18 void draw() {
19 }
20

21
22 void setChennals() {
23     //an arraylist of all lights | channels has been set as the default arraylis
24     //if there are any change in number if lights used, i < 42 will be changed t
25     for (int i=6; i<42; i+=4) {
26         ArrayList<Integer> temp = new ArrayList<Integer> ();
27         temp.add(i);
28         temp.add(i+1);
29         temp.add(i+2);
30         temp.add(i+3);
31         channels.add(temp);
32     }
33 }
34 // if not all lights are to be used, this method can be called to create an ar
35 void setPorts(int[] pins) {
36     ports.clear();
37     for (int i : pins) {
38         ArrayList<Integer> temp = new ArrayList<Integer> ();
39         for (int j=0; j<channels.get(i).size(); j++) {
40             temp.add(channels.get(i).get(j));
41         }
42         ports.add(temp);
43     }
44 }

45 void Handler(String function, String input) {}
```

下一步是實作 Handler()，使用 switch...case 來判斷 function name，如果傳入第一個變數是“execute”，就接著判斷後面跟進來的 input 值，也就是傳入的指令內容。指令一執行中央一個燈的漸亮，然後向 hub 回報任務完成；指令二則執行中央燈亮到外圈，然後向 hub 回報任務完成；指令三執行外圈燈的閃爍，然後向 hub 回報任務完成。

```
46 void Handler(String function, String input) {  
47     println("function: ", function);  
48     println("io_msg: ", input);  
49     switch(function) {  
50         case "execute":  
51             switch(input) {  
52                 case "1":  
53                     oneFadeIn();  
54                     finishReport();  
55                     break;  
56                 case "2":  
57                     insideOut();  
58                     finishReport();  
59                     break;  
60                 case "3":  
61                     CircleBlink();  
62                     finishReport();  
63                     break;  
64             default:  
65                 println("No input: ", input);  
66             }  
67             break;  
68         default:  
69             println("No function: ", function);  
70     }  
71 }  
72 }
```

燈號的實作已經在 1-1-3 解釋，在此不做贅述。最後實作任務回報的部分，在 finishReport() 裏頭，送出回報專用的 function name 跟自己的 id。

```
72 |  
73 void finishReport() {  
74     node.sendMessage("finishReport", node.my_id);  
75 }
```

Hub 的部分則是改變三個地方，一是宣告邏輯判斷會需要的全域變數，以及初始化，numItems 表示這個網路下的物品總數，邏輯撰寫上會確認所有物品都完成指令後，才發出下一道命令，mode 表示現在的指令模式，會在 setup() 初始化為 1，並發送指令，numFinish 紀錄已經完成指令的物品個數；二是在 draw() 內容增加邏輯，判斷所有物品都完成指令後，要接著送出甚麼指令，以及下一個要進入的指令模式，如果沒有指定，將會停留在同樣的指令模式下；三是在 Handler() 裏頭增加收到回覆後的動作，這邊範例使用 numFinish 變數記錄完成的個數，所以增加一來表示多一個完成的物品。

主頁完整程式碼如下頁所示。

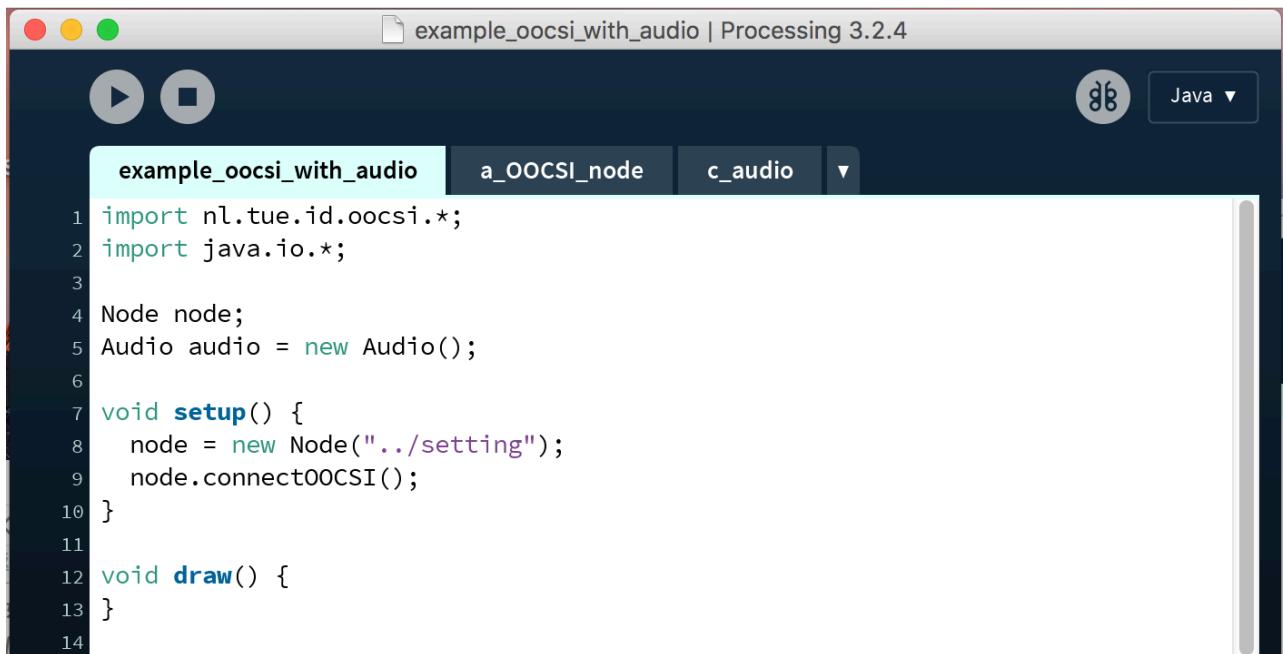
The screenshot shows the Processing IDE interface with the title bar "example_oocsi_with_light_hub | Processing 3.2.4". The code editor displays Java pseudocode for a Processing sketch. The code defines a Node object and handles setup and draw cycles, including sending messages and handling responses.

```
example_oocsi_with_light_hub
a_OOCSI_node ▾

1 import nl.tue.id.oocsi.*;
2 import java.io.*;
3
4 Node node;
5
6 int numItems = 1;
7 int mode, numFinish;
8
9 void setup() {
10    node = new Node("../setting", "Hub");
11    node.connectOOCSI();
12    mode = 1;
13    node.sendMessage("execute", "1");
14 }
15
16 void draw() {
17    if (numFinish==numItems) {
18       switch(mode) {
19          case 1:
20             node.sendMessage("execute", "2");
21             mode = 2;
22             break;
23          case 2:
24             node.sendMessage("execute", "3");
25             mode = 0;
26             break;
27          default:
28             println("Run into wrong mode: ", mode);
29             mode = 0;
30       }
31       // Reset the number of Items finished the command
32       numFinish = 0;
33    }
34 }
35
36 void Handler(String function, String input) {
37    switch(function) {
38       case "finishReport":
39          numFinish += 1;
40          println("finishReport: ", input);
41          break;
42       default:
43          println("No function: ", function);
44    }
45 }
46
47 }
```

1-4-2 單物品 OOCSI 與音訊檔

example_oocsi_with_audio 結合 1-2-5 與 1-3-3 的測試範例，新增並貼上 a_OOCSI_node 與 _methods 兩個分頁，分別包含 Node 與 Audio 物件。之後在主頁宣告並設定使用這兩個物件會需要的部分，如下圖所示，注意 draw() 已經沒有任何東西，因為所有動作都會由 hub 傳入的指令控制。



```
example_oocsi_with_audio | Processing 3.2.4
Java ▾

example_oocsi_with_audio [selected] a_OOCSI_node c_audio ▾

1 import nl.tue.id.oocsi.*;
2 import java.io.*;
3
4 Node node;
5 Audio audio = new Audio();
6
7 void setup() {
8     node = new Node("../setting");
9     node.connectOCSI();
10 }
11
12 void draw() {
13 }
14
```

下一步是實作 Handler()，與 1-4-1 極其相似，使用 switch...case 來判斷 function name，如果傳入第一個變數是“execute”，就接著判斷後面跟進來的 input 值，也就是傳入的指令內容。因此唯一的不同只有在判斷後要執行的地方 oneFadeIn()、insideOut() 和 CircleBlink() 被替換成 audio.init()。指令一播放 turnon_1.wav，然後向 hub 回報任務完成；指令二則播放 turnon_2.wav，然後向 hub 回報任務完成；指令三播放 turnon_3.wav，然後向 hub 回報任務完成。唯一的不同之處是 try...catch，為了防止無法播放而整個程式終止執行，使用 try...catch 來避免，當遇到播放錯誤，只會進入 catch() 印出錯誤資訊，而程式繼續執行。

```

15 void Handler(String function, String input) {
16     println("function: ", function);
17     println("io_msg: ", input);
18     switch(function) {
19         case "execute":
20             try {
21                 switch(input) {
22                     case "1":
23                         audio.init("turnon/turnon_1.wav");
24                         finishReport();
25                         break;
26                     case "2":
27                         audio.init("turnon/turnon_2.wav");
28                         finishReport();
29                         break;
30                     case "3":
31                         audio.init("turnon/turnon_3.wav");
32                         finishReport();
33                         break;
34                     default:
35                         println("No input: ", input);
36                     }
37                     break;
38                 }
39             catch (Exception e) {
40                 println("Error in Handler(): ", e);
41             }
42         }
43     }
44 }
45 void finishReport() {
46     node.sendMessage("finishReport", node.my_id);
47 }
48 }
```

Hub 的部分則是使用跟 1-4-1 一模一樣的程式碼，執行即可。

1-4-3 單物品綜合測試範例

example_oocsi_with_light_audio 綜合上述兩例，使用 1-4-1 的完成品為基底，修改下圖黃框提示處：一是增加 Audio 物件，二是在 Handler() 中，把要播放音效的地方加入 audio.init()。

```

5 ArrayList<ArrayList<Integer>> channels = new ArrayList<ArrayList<Integer>>();
6 ArrayList<ArrayList<Integer>> ports = new ArrayList<ArrayList<Integer>>();
7
8 //m is class where different light behaviours are, behaviours can be combined
9 Methods m = new Methods();
10 Node node;
11 Audio audio = new Audio();
12
13 void setup() {
```

```
void Handler(String function, String input) {
    println("function: ", function);
    println("io_msg: ", input);
    switch(function) {
        case "execute":
            switch(input) {
                case "1":
                    audio.init("turnon/turnon_1.wav");
                    oneFadeIn();
                    finishReport();
                    break;
                case "2":
                    audio.init("turnon/turnon_2.wav");
                    insideOut();
                    finishReport();
                    break;
                case "3":
                    audio.init("turnon/turnon_3.wav");
                    CircleBlink();
                    finishReport();
                    break;
                default:
                    println("No input: ", input);
            }
            break;
        default:
            println("No function: ", function);
    }
}
```

Hub 的部分修改 1-4-1 一模一樣的程式碼，將 numItems 初始化為 2 (如下圖)，之後在其他物品執行前先執行一次即可。

```
7 int numItems = 2;
8 int mode, numFinish;
```

1-4-4 多物品綜合測試範例

example_multi_oocsi_with_light_audio 修改自 1-4-3，僅在 Handler 把該階段要分開執行的燈光變化或音效移動到 node.my_id 的 if 判斷中(如下圖)。Hub 的部分同樣使用跟 1-4-1 一模一樣的程式碼，在其他物品執行前先執行一次即可。

```

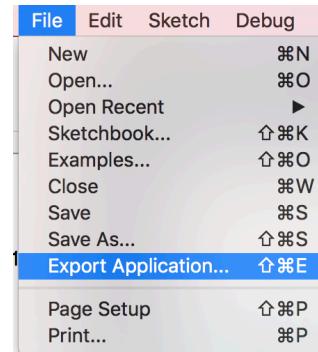
46 void Handler(String function, String input) {
47   println("function: ", function);
48   println("io_msg: ", input);
49   switch(function) {
50     case "execute":
51       switch(input) {
52         case "1":
53           if (node.my_id=="1") {
54             audio.init("turnon/turnon_1.wav");
55             oneFadeIn();
56           }
57           finishReport();
58           break;
59         case "2":
60           if (node.my_id=="2") {
61             audio.init("turnon/turnon_2.wav");
62             oneFadeIn();
63           }
64           finishReport();
65           break;
66         case "3":
67           if (node.my_id=="1") audio.init("turnon/turnon_1.wav");
68           else if (node.my_id=="2") audio.init("turnon/turnon_2.wav");
69           oneFadeIn();
70           finishReport();
71           break;
72         default:
73           println("No input: ", input);
74       }

```

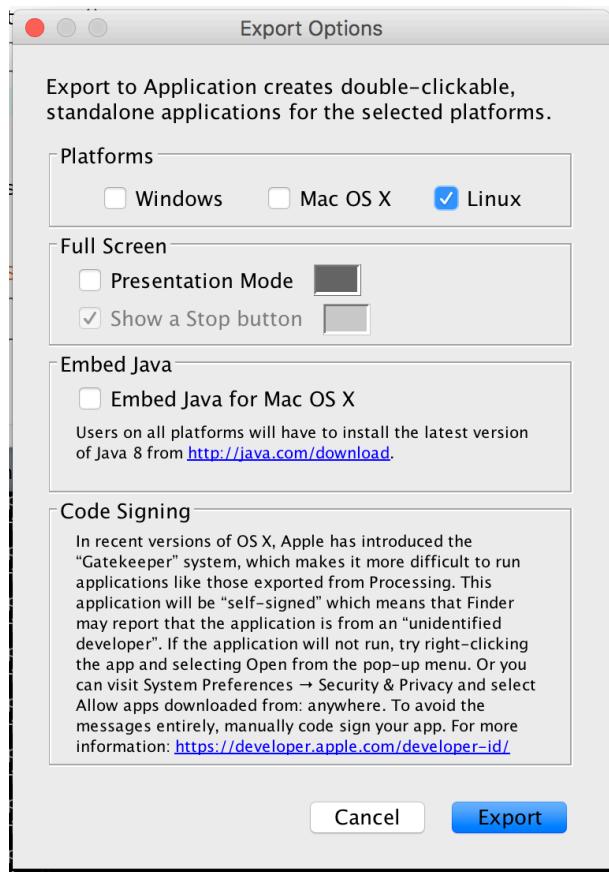
1-5 執行檔輸出與自動執行

1-5-1 執行檔輸出

首先點擊上方工具列的「File」，接著點擊「Export Application...」(如下圖)。



接著勾選「Platforms」內的「Linux」，並點選視窗右下角「Export」確認輸出(如下圖)，即可看到專案資料夾底下新增加「application.linux-armv6hf」的資料夾，內含 Raspberry Pi 的執行檔。



1-5-2 自動執行設定

在終端機執行「`vim ~/.config/lxsession/LXDE-pi/autostart`」，以 vim 修改開機自動執行檔。

2. OOCSI_Server (optional)

OOCSI 在連線時，會需要設定 Server ip，除了使用已經有運作的主機 (`oocsi.id.tue.nl`)，我們也可以在區域網路中，自行搭建擁有固定 ip 的 Server，建立自己的物聯網，避免其他物聯網互相干擾，也避免遭到竊聽。

如下圖，先下載 OOCSI 專案放在 GitHub 上的檔案，點擊「OOCSI_server.jar」進行下載。網址：<https://github.com/iddi/oocsi/wiki/Running-an-OOCSI-server/>

Running an OOCSI server

Mathias Funk edited this page on 16 Mar 2016 · 4 revisions

Download

The server comes as a pre-compiled .jar file: [OOCSI_server.jar](#) (~250kB)

Running / Configuration

It can be run simply by double-clicking on the JAR file in most operating systems. There are, however, command line parameters that are explained in the following:

Switch on logging to a file with:

```
java -jar OOCSI_server.jar -logging
```

▼ Pages 4

- [Home](#)
- [OOCSI Protocol](#)
- [Running an OOCSI server](#)
- [Using the Java OOCSI client](#)

Clone this wiki locally

<https://github.com/iddi/oocsi>

[Clone in Desktop](#)

下載之後，以終端機(command line)進入該檔案(OOCSI_server.jar)所處資料夾，執行下列指令「`java -jar OOCSI_server.jar`」，此後修改 1-3-3 連線 Server ip，便能夠連線至該主機的 ip。

```
終端機 — java -jar OOCSI_server.jar -logging — 80x24
yirudeMacBook-Pro% java -jar OOCSI_server.jar -logging
Wed Feb 01 03:16:12 CST 2017 Started OOCSI server v1.6 for max. 30 parallel clients and activated logging.
Wed Feb 01 03:16:12 CST 2017 [TCP socket server]: Started TCP service @ local address 'yirudeMBP' on port 4444 for TCP
```

3. Demo 設計

4-1 喚醒物品

基本上就是依 1-4-4 所寫，主程式修改處如下圖黃色提示框所示。完整程式會執行物件 ID=3 的音效與中央燈漸亮，接著其他四個同時發出音效與同樣的漸亮，完整程式碼如夾帶檔案 example_demo_1 所示。

```
46 void Handler(String function, String input) {  
47     println("function: ", function);  
48     println("io_msg: ", input);  
49     switch(function) {  
50         case "execute":  
51             switch(input) {  
52                 case "1":  
53                     if (node.my_id.equals("3")) {  
54                         audio.init("turnon/turnon_start.wav");  
55                         oneFadeIn();  
56                     }  
57                     finishReport();  
58                     break;  
59                 case "2":  
60                     audio.init("turnon/turnon_"+node.my_id+".wav");  
61                     oneFadeIn();  
62                     finishReport();  
63                     break;  
64                 default:  
65                     println("No input: ", input);  
66                 }  
67                 break;  
68             default:  
69                 println("No function: ", function);  
70             }  
71 }
```

Hub 只要將 numItems 物品數量增加為五(如下圖)，並多加額外的判斷即可。
另外，這邊將上面範例程式再修改，draw() 內程式更通用，方便後面 demo 其他狀況使用，但其實是做相同的動作，在於發送指令。

```
6 int numItems = 5;  
7 int mode, numFinish;  
8  
9 void setup() {  
10    node = new Node("../setting", "Hub");  
11    node.connectOOCISI();  
12    mode = 1;  
13    node.sendMessage("execute", "1");  
14 }  
15  
16 void draw() {  
17    if (numFinish==numItems) {  
18        switch(mode) {  
19            case 1:  
20                mode += 1;  
21                node.sendMessage("execute", str(mode));  
22                break;  
23            default:  
24                println("Run into wrong mode: ", mode);  
25                mode = 0;  
26            }  
27            // Reset the number of Items finished the command  
28            numFinish = 0;  
29        }  
30    }
```

4-2 物品加入物聯網

基本上就是依 4-1 修改，修改處如下圖黃色提示框所示。完整程式會依次執行 ID 一到五的物件各自的音效與中央燈漸亮，接著五個同時發出音效與同樣的漸亮，完整程式碼如夾帶檔案 example_demo_2 所示。

```
46 void Handler(String function, String input) {  
47     println("function: ", function);  
48     println("io_msg: ", input);  
49     switch(function) {  
50         case "execute":  
51             switch(input) {  
52                 case "1":  
53                 case "2":  
54                 case "3":  
55                 case "4":  
56                 case "5":  
57                     if (node.my_id.equals(input)) {  
58                         audio.init("install/install_"+input+".wav");  
59                         oneFadeIn();  
60                     }  
61                     finishReport();  
62                     break;  
63                 case "6":  
64                     audio.init("install/install_finish.wav");  
65                     oneFadeIn();  
66                     finishReport();  
67                     break;  
68             default:  
69                 println("No input: ", input);  
70             }  
71             break;  
72         default:  
73             println("No function: ", function);  
74     }
```

4-3 任務失敗 (微調)

由 4-1 修改而來，修改處如下圖黃色提示框所示。完整程式會一個發出訊息燈號，接著其他呈現接收訊息燈號。最後 ID=1 的物品拒絕，呈現 X 閃爍燈號；ID=2 和 4 的兩個物品同意，呈現 O 閃爍燈號；ID=5 的物品是不確定，呈現 U 型搖擺燈號，完整程式碼如夾帶檔案 example_demo_3 所示。

```
46 void Handler(String function, String input) {  
47     println("function: ", function);  
48     println("io_msg: ", input);  
49     switch(function) {  
50         case "execute":  
51             switch(input) {  
52                 case "1":  
53                     finishReport();  
54                     if (node.my_id.equals("3")) insideOut();  
55                     break;  
56                 case "2":  
57                     if (!node.my_id.equals("3")) OutsideIn();  
58                     finishReport();  
59                     break;  
60                 case "3":  
61                     if (node.my_id.equals("1")) CrossBlink();  
62                     else if (node.my_id.equals("2")) CircleBlink();  
63                     else if (node.my_id.equals("4")) CircleBlink();  
64                     else if (node.my_id.equals("5")) Swing();  
65                     finishReport();  
66                     break;  
67                 default:  
68                     println("No input: ", input);  
69                 }  
70                 break;  
71             default:  
72                 println("No function: ", function);  
73             }  
74 }
```

4-4 同步成功

幾乎和 4-3 相同，在此模擬所有物件都同意任務後，開始同步的狀況，修改處如下圖黃色提示框所示。完整程式會一個發出訊息燈號，接著其他呈現接收訊息燈號，最後出現同步的燈號，並顯示圓形閃爍，表示同步完成，完整程式碼如夾帶檔案 example_demo_4 所示。

```
46 void Handler(String function, String input) {  
47     println("function: ", function);  
48     println("io_msg: ", input);  
49     switch(function) {  
50         case "execute":  
51             switch(input) {  
52                 case "1":  
53                     if (node.my_id.equals("3")) insideOut();  
54                     finishReport();  
55                     break;  
56                 case "2":  
57                     if (!node.my_id.equals("3")) OutsideIn();  
58                     finishReport();  
59                     break;  
60                 case "3":  
61                     Sync();  
62                     finishReport();  
63                     break;  
64                 case "4":  
65                     CircleBlink();  
66                     finishReport();  
67                     break;  
68             default:  
69                 println("No input: ", input);  
70             }  
71             break;  
72     default:  
73         println("No function: ", function);  
74     }
```

4-5 物品退出系統

幾乎和 4-3 相同，在此模擬 ID 為 3 的物件退出系統，修改處如下圖黃色提示框所示。完整程式會有所有物件同意的 O 出現，接著 ID 為 3 的物件呈現退出燈號，其他呈現全亮，完整程式碼如夾帶檔案 example_demo_5 所示。

```
example_demo_5 a_OOCSI_node c_audio d_methods ▾
void Handler(String function, String input) {
    println("function: ", function);
    println("io_msg: ", input);
    switch(function) {
        case "execute":
            switch(input) {
                case "1":
                    CircleOn();
                    finishReport();
                    break;
                case "2":
                    if (node.my_id.equals("3")) Quit();
                    else m.allon(4000);
                    finishReport();
                    break;
                default:
                    println("No input: ", input);
            }
            break;
        default:
            println("No function: ", function);
    }
}
```

4-6 綜合控制

綜合 4-1 到 4-5 把 switch(function) 的 case 從各自的一個 “execute” ，合併成 “execute_1”, “execute_2” 一直到 “execute_5” 即可。完整程式碼如夾帶檔案 example_demo_6 所示。

附錄一、Methods 提供的函式

如果有附錄中無法實作的變換，可參考更底層的 type() 函式，自行設定。

alloff()

//switch all lights off

allon(int brightness)

//switch all lights on

fade1(int times, ArrayList portList, int max_brightness)

//method: fade in AND out (breathing lights), the lights fade one by one

//times: how many times of breathing cycle

//1 cycle = fade in and fade out

//portlist= the lights involved

//max_brightness = how bright the light should be ; can range from 0 to 4095, with 0 being no light and 4095 being the brightest

fadein1(ArrayList portList, int waiting, int max_brightness)

//method: lights fade in one by one

//waiting: duration of pause in between two lights fading in, if any

//portlist= the lights involved

//max_brightness = how bright the light should be ; can range from 0 to 4095, with 0 being no light and 4095 being the brightest

fadeout1(ArrayList portList, int waiting, int max_brightness)

//method: lights fade out one by one

//waiting: duration of pause in between two lights fading out, if any

//portlist= the lights involved

//max_brightness = how bright the light should be ; can range from 0 to 4095, with 0 being no light and 4095 being the brightest

fade(int times, ArrayList portList, int max_brightness)

//method: fade in AND out (breathing lights), the lights fade together

//times: how many times of breathing cycle

//1 cycle = fade in and fade out

//portlist= the lights involved

//max_brightness = how bright the light should be ; can range from 0 to 4095, with 0 being no light and 4095 being the brightest

fadein(ArrayList portList, int max_brightness)

//method: lights fade in together

//portlist= the lights involved

//max_brightness = how bright the light should be ; can range from 0 to 4095, with 0 being no light and 4095 being the brightest

fadeout(ArrayList portList, int max_brightness)

//method: lights fade out together

//portlist= the lights involved

//max_brightness = how bright the light should be ; can range from 0 to 4095, with 0 being no light and 4095 being the brightest

```

off(ArrayList portList, int paused)
//method: switch off selected lights
//paused: duration of wait between two lights switching off, if any

on(ArrayList portList, int paused, int brightness)
//method: switch on selected lights
//paused: duration of wait between two lights switching on, if any
//brightness: how bright the light is when turned on, with 0 being no light and 4095 being
the brightest

blink(ArrayList portList, int blink, int waiting, int brightness)
// method: lights blink one by one
//waiting: duration of pause in between two lights blinking, if any
//portlist= the lights involved
//blink = number of times of blinking
//max_brightness = how bright the light should be ; can range from 0 to 4095, with 0
being no light and 4095 being the brightest

blink_more(ArrayList portList, int blink, int waiting, int brightness)
// method: lights blink all at once
//waiting: duration of pause in between two lights blinking, if any
//portlist= the lights involved
//blink = number of times of blinking
//max_brightness = how bright the light should be ; can range from 0 to 4095, with 0
being no light and 4095 being the brightest

```

出現在範例程式的燈號實作：

```

void oneFadeIn(){
    int[] p = {0};
    setPorts(p);
    m.fadein(ports,4000);
}

void allFade(){
    int[] p = {0,1,2,3,4,5,6,7,8};
    setPorts(p);
    m.fade(3,ports,4000);
}

void insideOut(){
    int[] p_1 = {0};
    setPorts(p_1);
    m.fade(1,ports,1800);
    int[] p_2 = {1,2,3,4,5,6,7,8};
    setPorts(p_2);
    m.fade(1,ports,1200);
}

```

```

void OutsideIn(){
    int[] p_1 = {1,2,3,4,5,6,7,8};
    int[] p_2 = {0};
    setPorts(p_1);
    m.fade(1,ports,1200);
    setPorts(p_2);
    m.fade(1,ports,1800);
}

void CrossBlink(){
    int[] p = {0,2,4,6,8};
    setPorts(p);
    m.blink_more(ports,2,200,4000);
}

void CircleBlink(){
    int[] p = {1,2,3,4,5,6,7,8};
    setPorts(p);
    m.blink_more(ports,2,200,4000);
}

void Swing(){
    int start = 3;
    int[][] p={{start+2,start+1,start,start+4,start+3},
               {start+3,start+2,start+1,start,start+4},{start+4,start+3,start+2,start+1,start},
               {start+3,start+4,start+2,start+1,start},{start+2,start+3,start+4,start+1,start},
               {start+1,start+2,start+3,start,start+4},
               {start,start+1,start+2,start+4,start+3},{start+1,start,start+2,start+4,start+3},
               {start+2,start+1,start,start+4,start+3}};
    for(int i=0;i<3;i++){ // Come and go twice
        for(int j=0;j<p.length;j++){
            setPorts(p[j]);
            m.switchon(ports,3,4000);
            delay(200);
        }
    }
    m.alloff();
}

void Quit(){
    int[][] p = {{5},{4,5,6},{0,3,4,5,6,7},
                 {0,2,3,4,5,6,7,8},{0,1,2,3,4,5,6,7,8}};
    for(int i=0;i<p.length;i++){
        setPorts(p[i]);
        m.on(ports,4000);
        delay(400);
    }
    m.alloff();
}

```

```
void CircleOn(){
    int[] p_1 = {8};
    setPorts(p_1);
    m.fadein(ports,2000);
    int[] p = {8,7,6,5,4,3,2,1};
    for(int i=0;i<=p.length;i++){
        setPorts(p);
        m.switchon(ports,3,4000);
        delay(200);
        for(int j=0;j<p.length;j++){
            p[j] = p[j]%p.length+1;
        }
    }
    m.alloff();
}
```

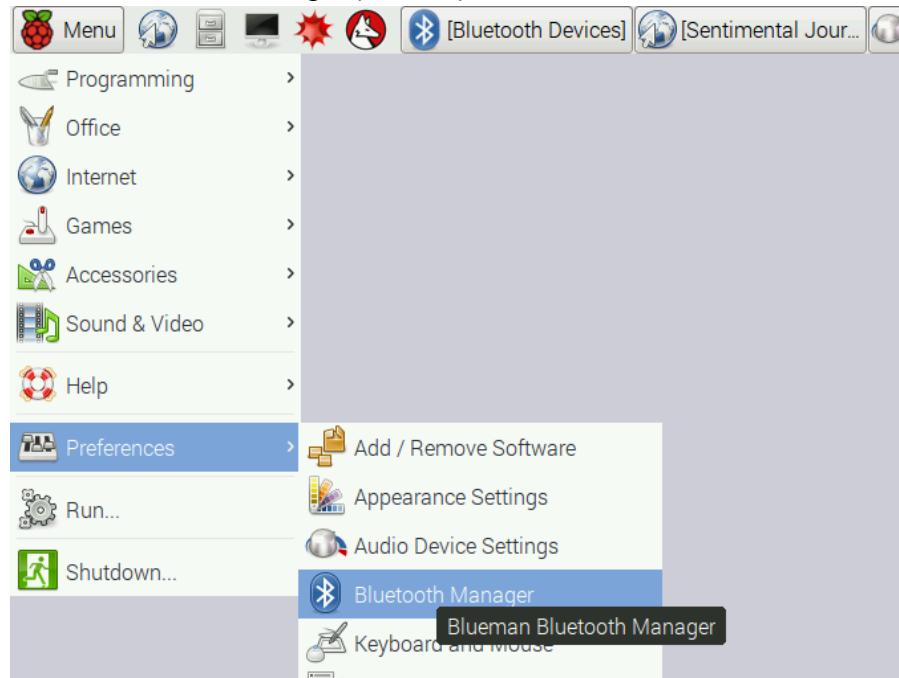
```
void Sync(){
    int[] p_1 = {1,3,5,7};
    int[] p_2 = {2,4,6,8};
    setPorts(p_1);
    m.alloff();
    m.on(ports,4000);
    delay(300);
    setPorts(p_2);
    m.alloff();
    m.on(ports,4000);
    delay(300);
    setPorts(p_1);
    m.alloff();
    m.on(ports,4000);
    delay(300);
    m.alloff();
}
```

附件二、藍牙播放器環境設定

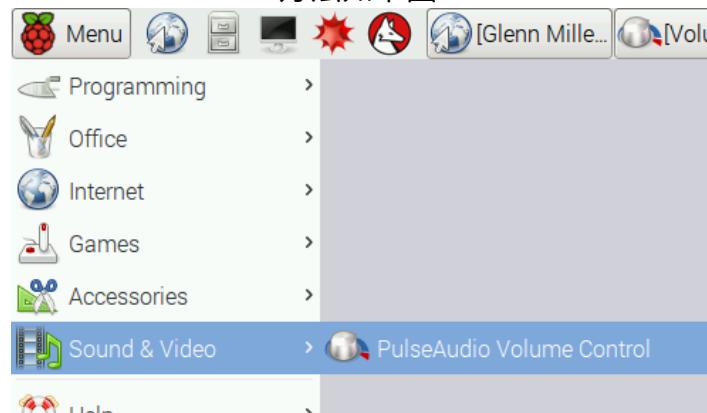
終端機依序執行下列指令：

```
sudo apt-get update  
sudo apt-get upgrade  
sudo apt-get install pi-bluetooth  
sudo apt-get install blueman pulseaudio pavucontrol pulseaudio-module-bluetooth  
pulseaudio -D
```

安裝完成後，打開 Bluetooth Manager(如下圖)，進行 Search 與 Connect。



之後打開 PulseAudio Volume Control，方法如下圖。



接著把 Config 中 bcm2835 ALSA 的 Profile 改為 Off。設定完成，可開啟網路瀏覽器(Menu > Internet > Firefox Web Browser)，連限到 Youtube 網頁，播放音樂作確認。

- 參考：<http://plugable.com/2016/03/14/listening-to-bluetooth-audio-on-your-raspberry-pi-3-pi-2-or-pi-zero/>