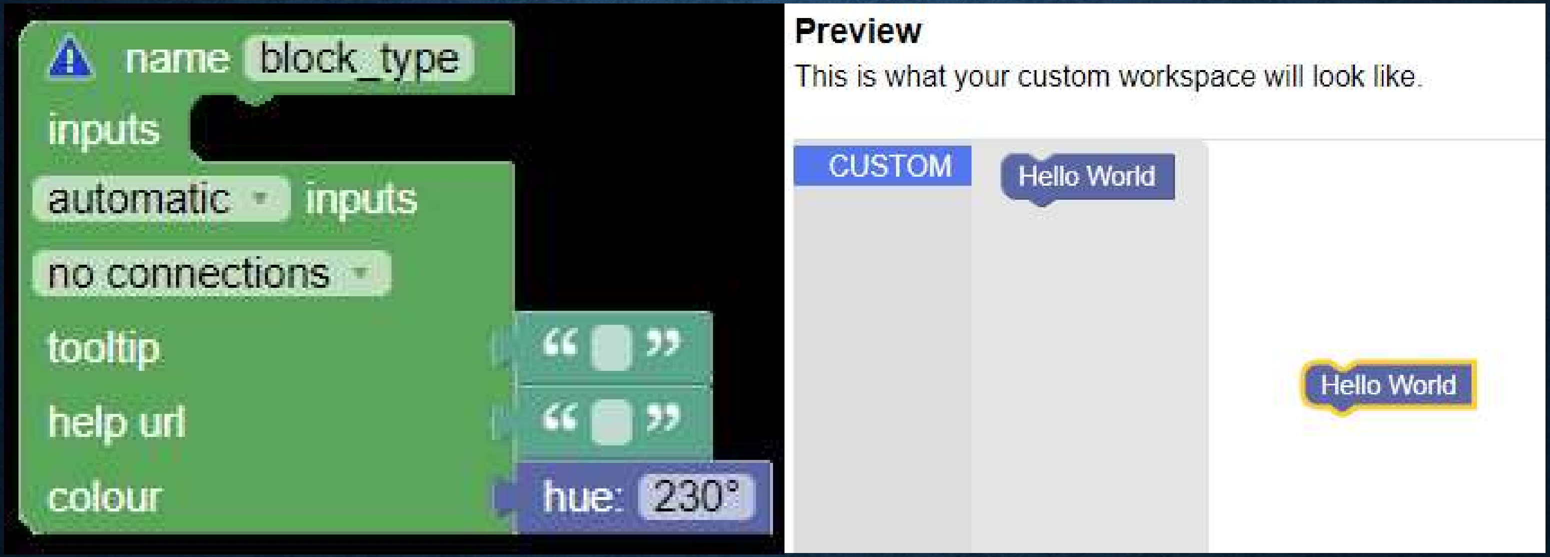
GOOGLE BLOCKLY
自訂積木撰寫與影像專題實作



講師：鳳山科技中心 傅仲儀主任

2021/10/30

課程表

09:00-09:30	Blockly技術文件與基礎架構介紹
09:30-12:00	Blockly Developer Tools
	Circus EZ Start Kit+全套積木實作
	Blocklyduino F1自訂積木環境設定
13:00-13:30	ESP32-CAM常見應用介紹
13:30-16:00	影像模組積木實作專題

研習檔案與教學資源下載

研習檔案下載　教學影片

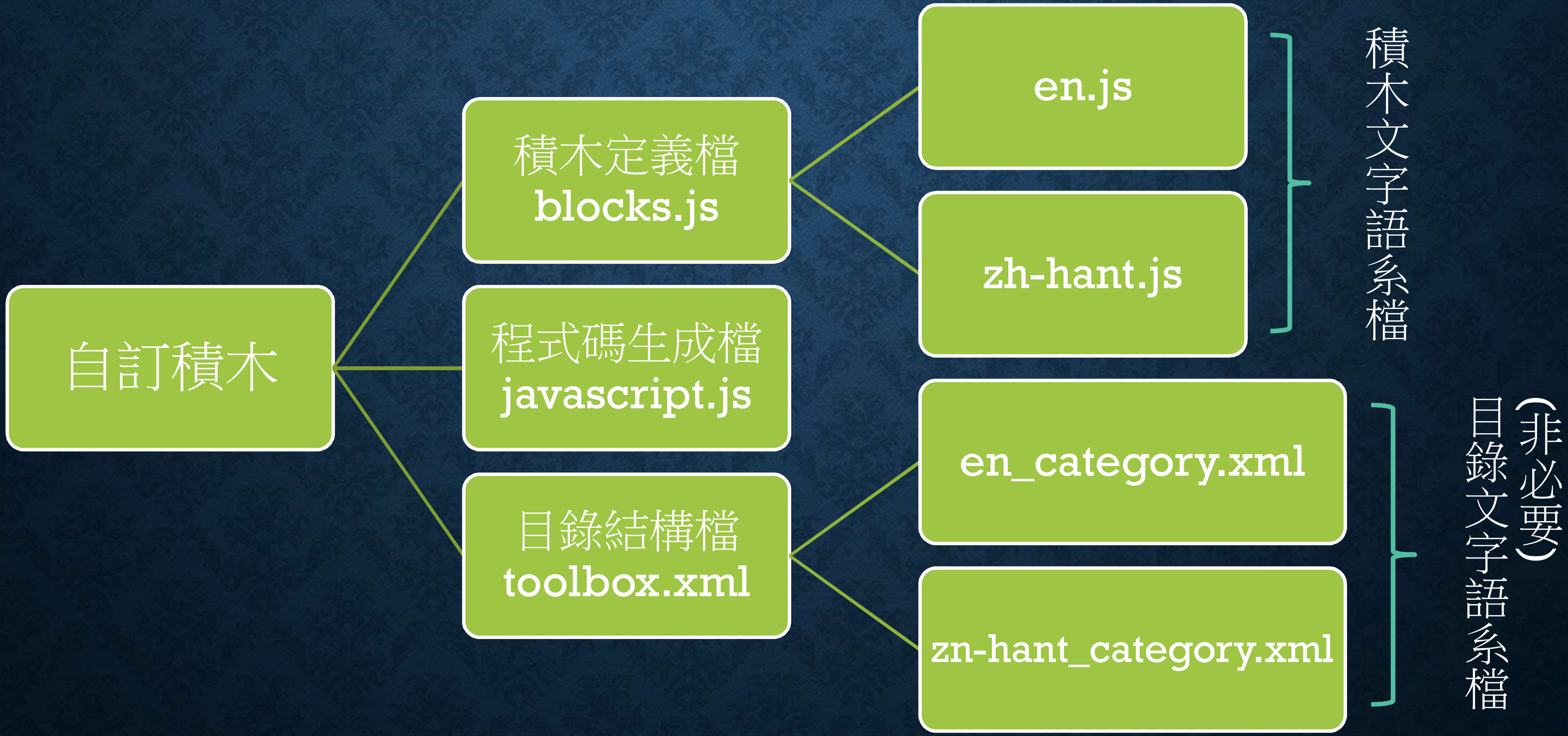
開發環境建置

1. Blocklyduino V3 下載
2. Blocklyduino F1升級包 下載
3. Notepad++程式編輯軟體 下載
4. Arduino IDE (PORTABLE) 連結
5. Blockly developer tool 連結
6. SpBlockly developer tool 連結
7. 吉哥積木 連結

網路學習資源

1. Blockly首頁 連結
2. Blockly自訂積木說明 連結
3. Blockly developer tool教學 連結
4. Blockly核心檔 連結
5. Blockly社群 連結
6. Blockly Colelabs 連結
7. NW.js 連結
8. Chrome API 連結
9. JavaScript 教學 連結
10. Arduino函式 連結

BlocklyDuino自訂積木檔案架構



如何設計自訂積木？

1. 撰寫自訂積木想做到的功能？
2. 自訂積木適用的開發板？
3. 尋找相關功能程式碼範例與函式說明：Arduino官網、函式庫提供的範例、書籍、Github、Google搜尋關鍵字…
4. 決定要使用的函式庫與程式碼範例
5. 分析範例程式碼結構：定義區(definition)、初始化區(setup)或隨積木擺放位置而產出程式碼等。
6. 分析範例程式碼內容：隨積木使用自動產生於特定區域程式碼、使用者輸入變數資料與指令程式碼串接成動態程式碼等。
7. 依使用者想執行的功能、變數資料、函式庫參數設定等決定積木種類與數目。
8. 依程式碼結構與使用者輸入內容決定積木定義

自訂積木開發步驟

1. 指令範例程式碼分析
   區分：DEFINITION區、SETUP區、LOOP區、FUNCTION區、變數輸入等

2. 規劃積木種類、組成元素、樣式等

3. Blockly developer tool設計積木
   產出：積木定義函式、程式碼產出函式、目錄結構

4. 程式碼產出內容
   串接字串： `var code = 'Serial.println(' +str+');\n';`
   取代字串： `var code = 'Serial.println(%1);\n'.replace('%1',str);`

5. 程式碼產出特定區域內容

```
Blockly.Arduino.definitions_['NAME'] = '#include <WiFi.h>\n#include <WiFiClientSecure.h>; ';
Blockly.Arduino.setups_['NAME'] = 'Serial.begin(115200);';
Blockly.Arduino.functions_['NAME'] = 'String say() {\n  return "Hello, World!";\n}';
```

6. 編輯積木定義，將文字改為變數設定可切換多國語言(非必要)

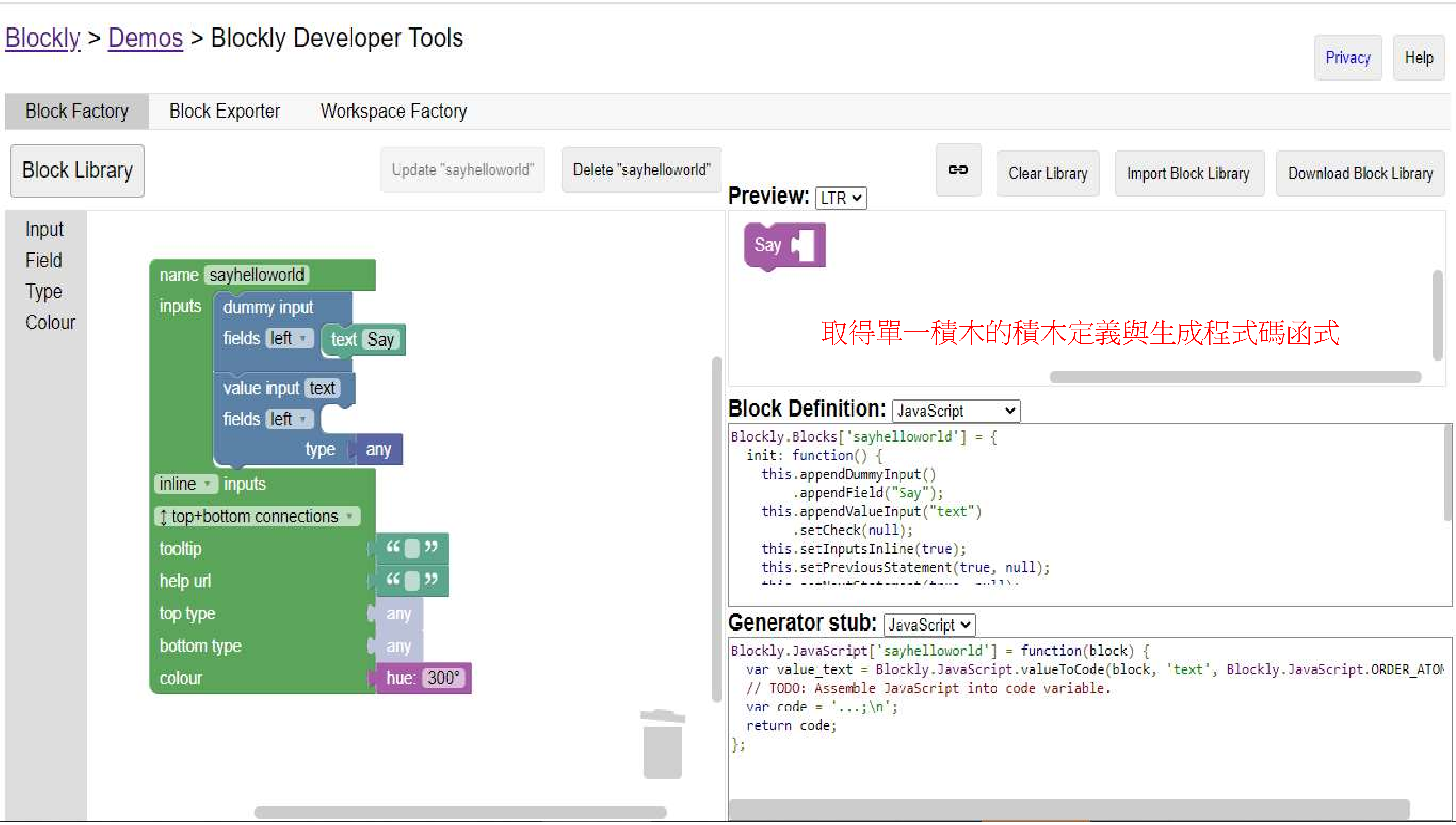
```
.appendField(Blockly.Msg["CATLOGIC"]);
```

7. 編輯目錄結構，將文字或顏色改為變數設定可切換多國語言(非必要)

```
<category id="category_logic" name="%{BKY_CATLOGIC}" colour="%{BKY_LOGIC_HUE}">
    <block type="helloworld"></block>
</category>

Blockly.Msg["CATLOGIC"] = "LOGIC";
Blockly.Msg["LOGIC_HUE"] = "100";
```

BLOCKL DEVELOPER TOOL

使用說明



Preview: LTR

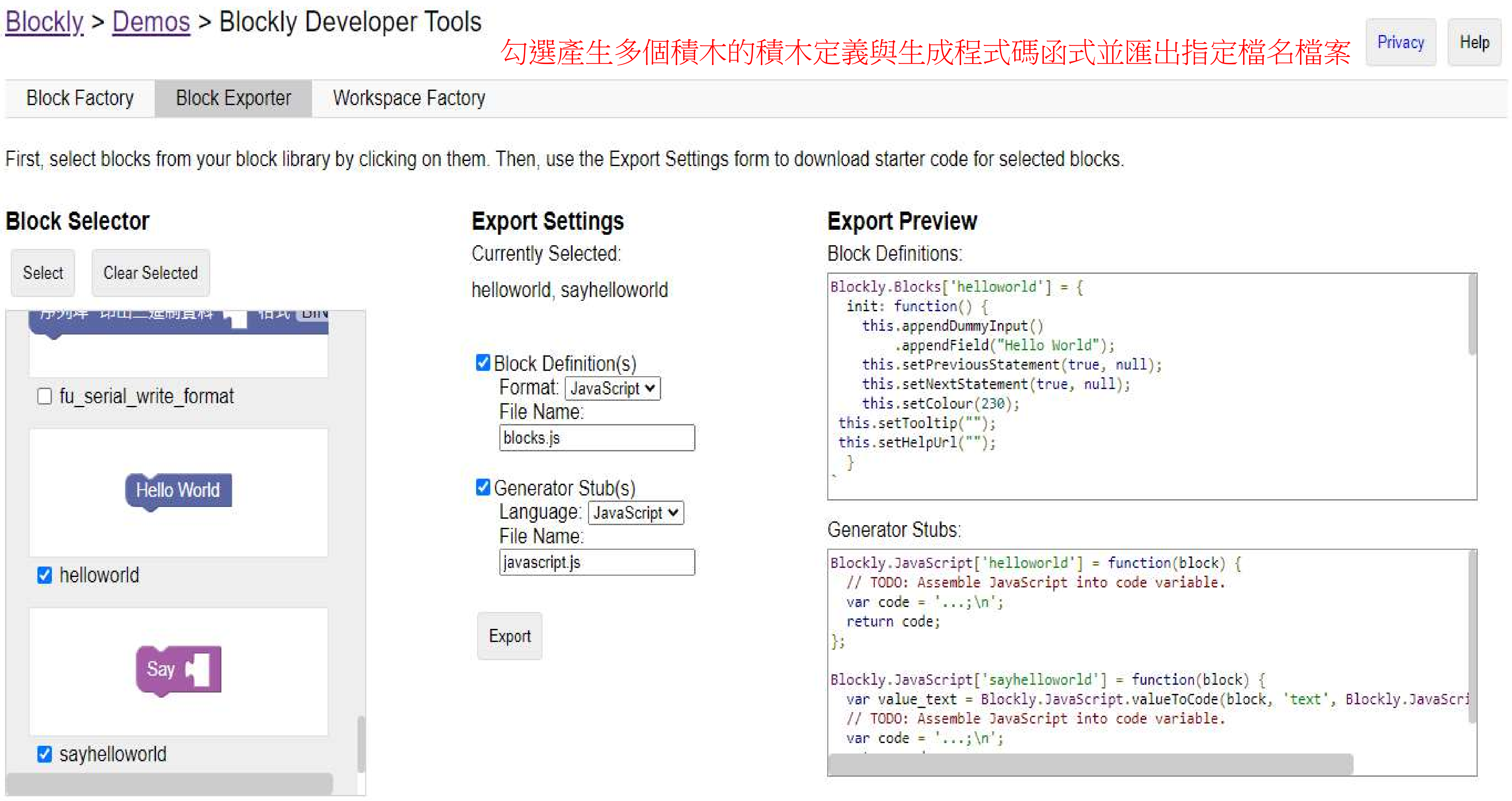
取得單一積木的積木定義與生成程式碼函式

Block Definition: JavaScript

```
Blockly.Blocks['sayhelloworld'] = {
  init: function() {
    this.appendDummyInput()
        .appendField("Say");
    this.appendValueInput("text")
        .setCheck(null);
    this.setInputsInline(true);
    this.setPreviousStatement(true, null);
```

Generator stub: JavaScript

```
Blockly.JavaScript['sayhelloworld'] = function(block) {
  var value_text = Blockly.JavaScript.valueToCode(block, 'text', Blockly.JavaScript.ORDER_ATO
  // TODO: Assemble JavaScript into code variable.
  var code = '...;\n';
  return code;
};
```



勾選產生多個積木的積木定義與生成程式碼函式並匯出指定檔名檔案

Blockly > Demos > Blockly Developer Tools

First, select blocks from your block library by clicking on them. Then, use the Export Settings form to download starter code for selected blocks.

Block Selector

☐ fu_serial_write_format

☑ helloworld

☑ sayhelloworld

Export Settings

Currently Selected:

helloworld, sayhelloworld

☑ Block Definition(s)
Format: JavaScript
File Name: blocks.js

☑ Generator Stub(s)
Language: JavaScript
File Name: javascript.js

Export Preview

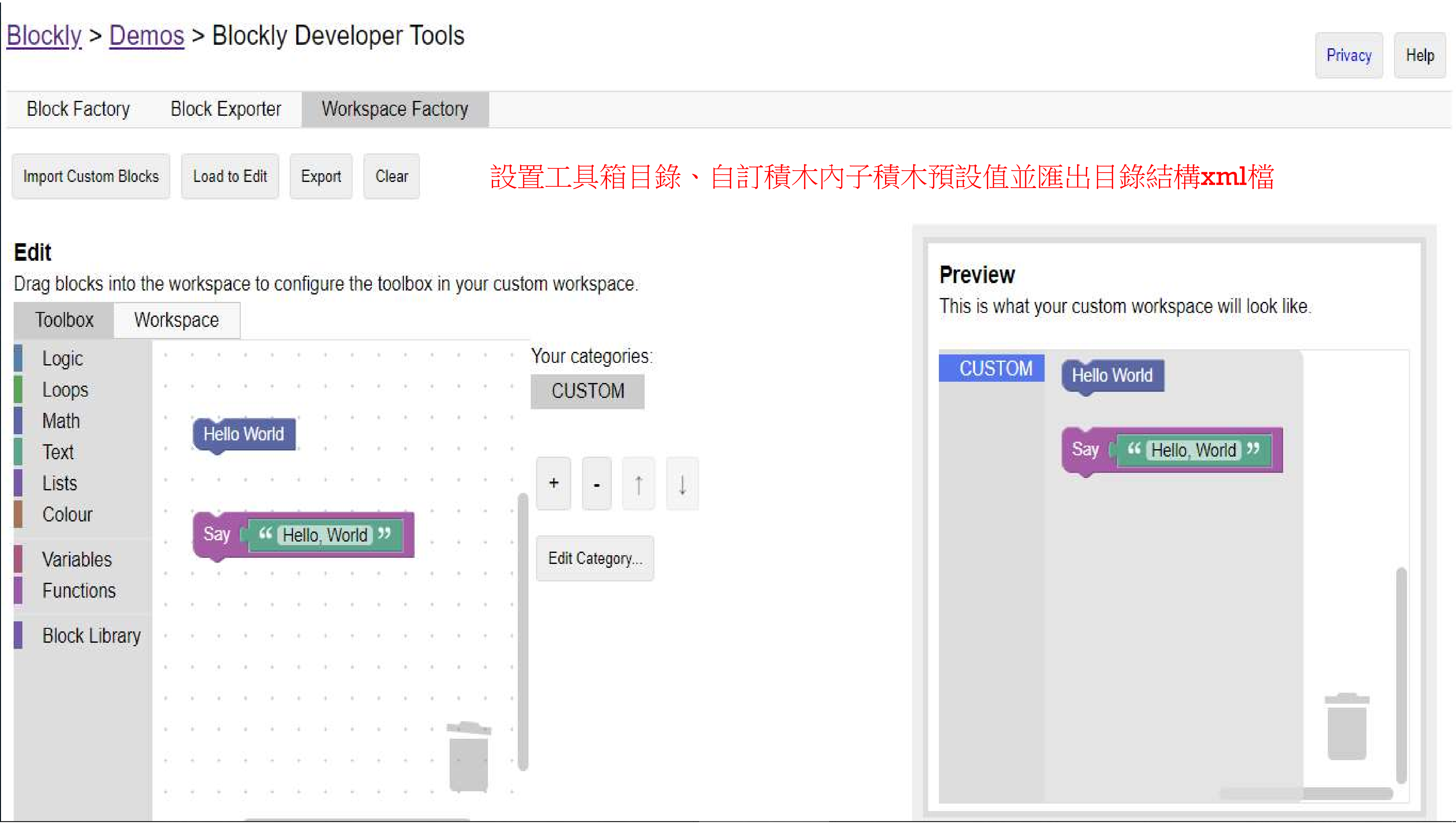
Block Definitions:

```
Blockly.Blocks['helloworld'] = {
  init: function() {
    this.appendDummyInput()
        .appendField("Hello World");
    this.setPreviousStatement(true, null);
    this.setNextStatement(true, null);
    this.setColour(230);
 this.setTooltip("");
 this.setHelpUrl("");
  }
```

Generator Stubs:

```
Blockly.JavaScript['helloworld'] = function(block) {
  // TODO: Assemble JavaScript into code variable.
  var code = '...;\n';
  return code;
};

Blockly.JavaScript['sayhelloworld'] = function(block) {
  var value_text = Blockly.JavaScript.valueToCode(block, 'text', Blockly.JavaScri
  // TODO: Assemble JavaScript into code variable.
  var code = '...;\n';
```

設置工具箱目錄、自訂積木內子積木預設值並匯出目錄結構**xml**檔


Blockly > Demos > Blockly Developer Tools

Edit

Drag blocks into the workspace to configure the toolbox in your custom workspace.

Preview

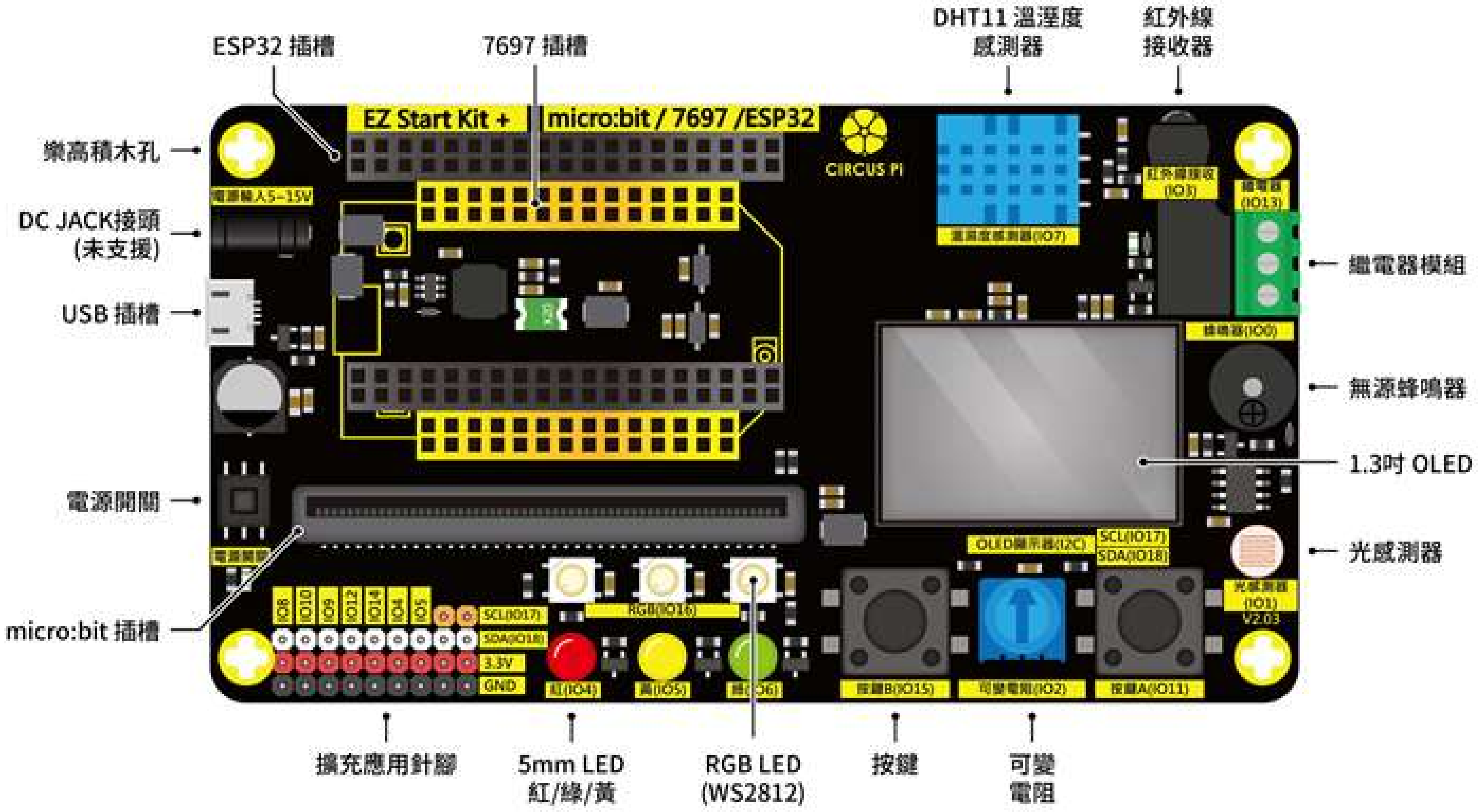
This is what your custom workspace will look like.

CIRCUS EZ START KIT+ FOR ESP32 自訂積木實作

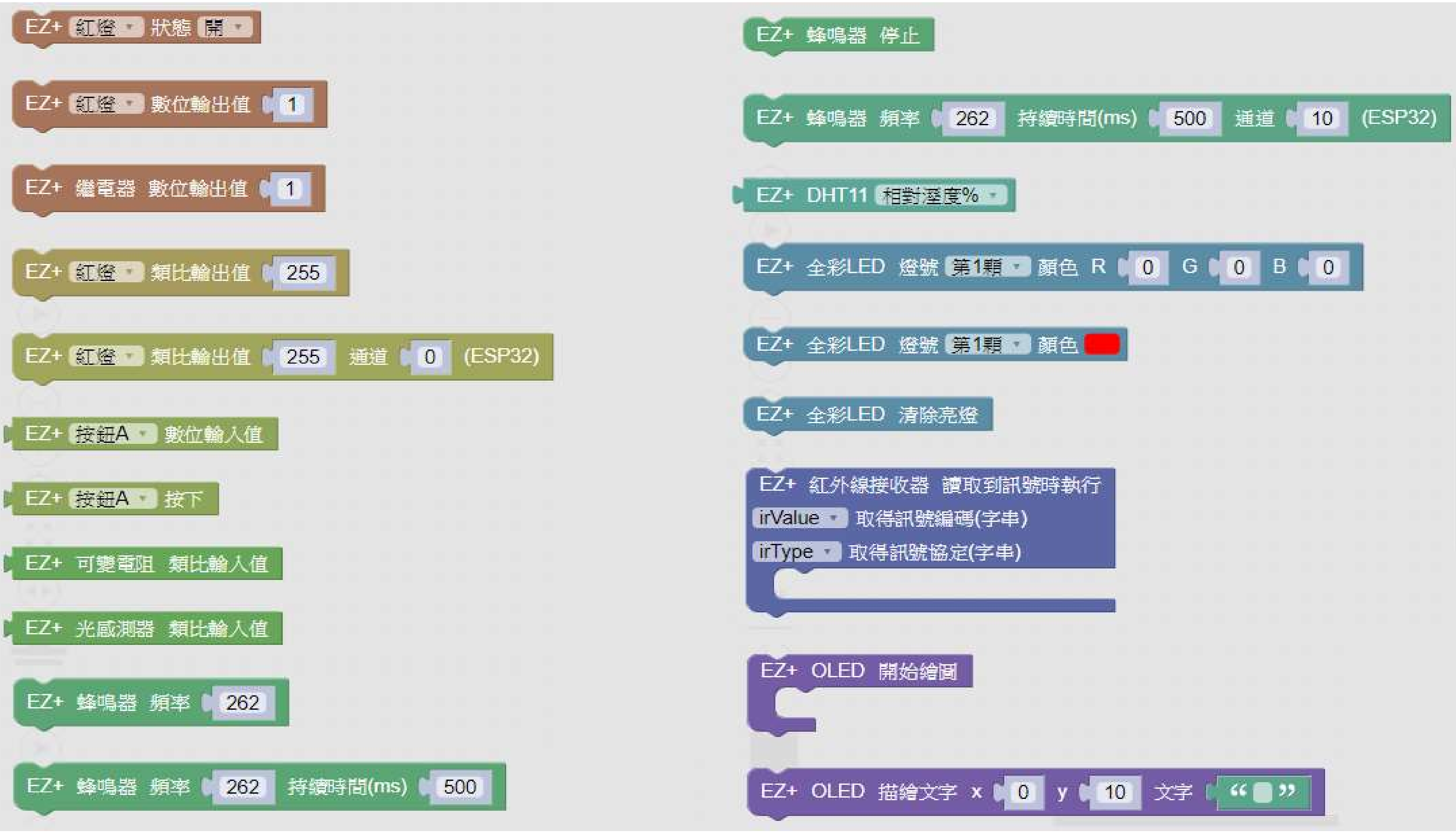
基礎教學影片

課堂練習

請跟著影片說明一步一步製作出第一個自訂積木
輸出程式碼 "Hello, World"


ESP32 插槽
7697 插槽
DHT11 溫溼度感測器
紅外線接收器
樂高積木孔
DC JACK接頭(未支援)
USB 插槽
電源開關
micro:bit 插槽
繼電器模組
無源蜂鳴器
1.3吋 OLED
光感測器
擴充應用針腳
5mm LED 紅/綠/黃
RGB LED (WS2812)
按鍵
可變電阻
EZ Start Kit +
micro:bit / 7697 /ESP32
CIRCUS PI
電源輸入5~15V
溫溼度感測器(IO7)
紅外線接收(IO3)
繼電器(IO13)
蜂鳴器(IO0)
OLED顯示器(I2C)
SCL(IO17)
SDA(IO18)
光感測器(IO1)
V2.03
按鍵B(IO15)
可變電阻(IO2)
按鍵A(IO11)
電源開關
IO8
IO10
IO9
IO12
IO14
IO4
IO5
SCL(IO17)
SDA(IO18)
3.3V
GND
RGB(IO16)
紅(IO4)
黃(IO5)
綠(IO6)

EZ Start Kit +	IO	Micro:Bit	LinkIt 7697	ESP32S
無源蜂鳴器	0	0	14	14
光感測器	1	1	15	39
可變電阻	2	2	16	34
紅外線接收器	3	8	17	33
5mm 圓頭 LED(紅)	4	13	13	16
5mm 圓頭 LED(黃)	5	14	12	12
5mm 圓頭 LED(綠)	6	15	11	13
DHT 11 溫溼度感測器	7	16	10	15
按鍵(A)	11	5	0	5
繼電器模組	13	9	5	25
按鍵(B)	15	11	7	36
全彩 RGB LED	16	12	4	26



EZ+ 紅燈 狀態 開

EZ+ 紅燈 數位輸出值 1

EZ+ 繼電器 數位輸出值 1

EZ+ 紅燈 類比輸出值 255

EZ+ 紅燈 類比輸出值 255 通道 0 (ESP32)

EZ+ 按鈕A 數位輸入值

EZ+ 按鈕A 按下

EZ+ 可變電阻 類比輸入值

EZ+ 光感測器 類比輸入值

EZ+ 蜂鳴器 頻率 262

EZ+ 蜂鳴器 頻率 262 持續時間(ms) 500

EZ+ 蜂鳴器 停止

EZ+ 蜂鳴器 頻率 262 持續時間(ms) 500 通道 10 (ESP32)

EZ+ DHT11 相對溼度%

EZ+ 全彩LED 燈號 第1顆 顏色 R 0 G 0 B 0

EZ+ 全彩LED 燈號 第1顆 顏色 🟥

EZ+ 全彩LED 清除亮燈

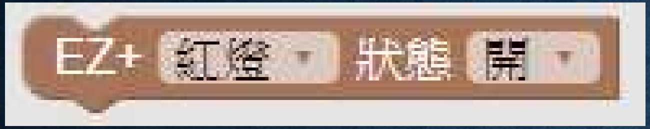
EZ+ 紅外線接收器 讀取到訊號時執行
irValue 取得訊號編碼(字串)
irType 取得訊號協定(字串)

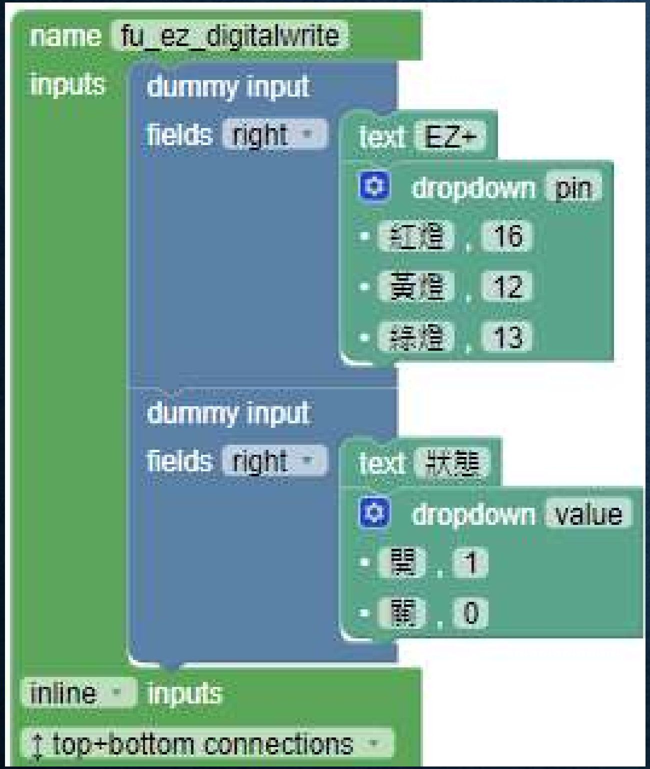
EZ+ OLED 開始繪圖

EZ+ OLED 描繪文字 x 0 y 10 文字 “ ”

紅黃綠LED燈(數位輸出)

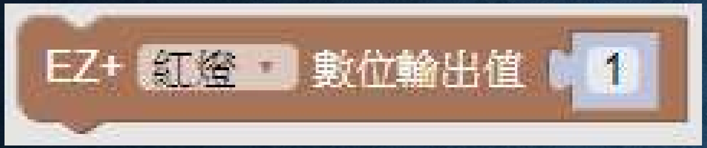
```
void setup()
{
  pinMode(16, OUTPUT);    //可自動輸出於setup區
}

void loop()
{
  digitalWrite(16, 1);    //隨積木移動輸出程式碼
}
```

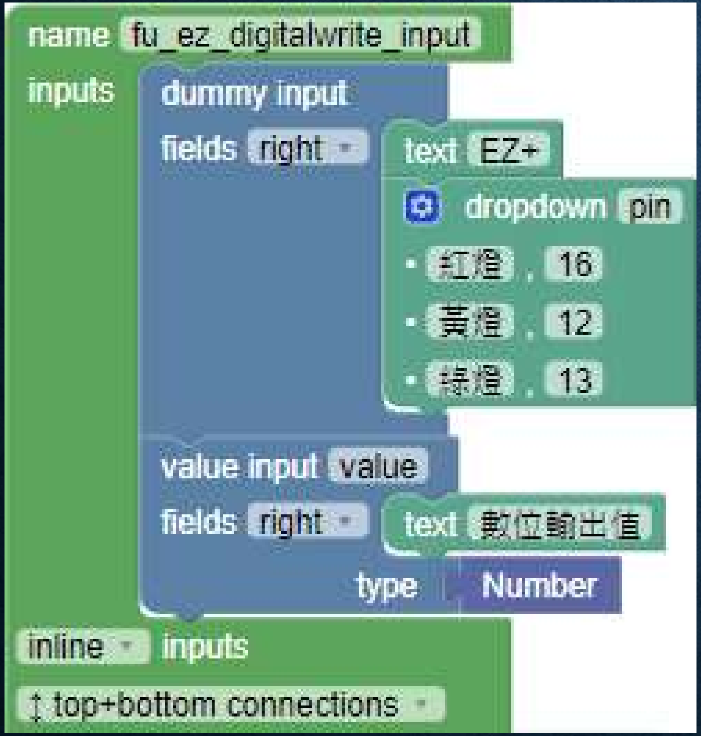




```
Blockly.Blocks['fu_ez_digitalwrite'] = {
 init: function() {
  this.appendDummyInput()
     .setAlign(Blockly.ALIGN_RIGHT)
     .appendField("EZ+")
     .appendField(new Blockly.FieldDropdown([
        ["紅燈","16"],
        ["黃燈","12"],
        ["綠燈","13"]
      ]), "pin");
  this.appendDummyInput()
     .setAlign(Blockly.ALIGN_RIGHT)
     .appendField("狀態")
     .appendField(new Blockly.FieldDropdown([
        ["開","1"],
        ["關","0"]
      ]), "value");
  this.setInputsInline(true);
  this.setPreviousStatement(true, null);
  this.setNextStatement(true, null);
  this.setColour(195);
  this.setTooltip("");
  this.setHelpUrl("");
 }
};
```

```
Blockly.Arduino['fu_ez_digitalwrite'] = function(block) {
  var dropdown_pin = block.getFieldValue('pin');
  var dropdown_value = block.getFieldValue('value');

  //新增pinMode程式碼於setup區，NAME值須有固定格式綁定pin值。
  Blockly.Arduino.setups_['pinmode_'+ dropdown_pin] = 'pinMode('+ dropdown_pin +', OUTPUT);';

  var code = 'digitalWrite(%1, %2);\n';
  code = code.replace("%1", dropdown_pin );
  code = code.replace("%2", dropdown_value );
  return code;
};
```



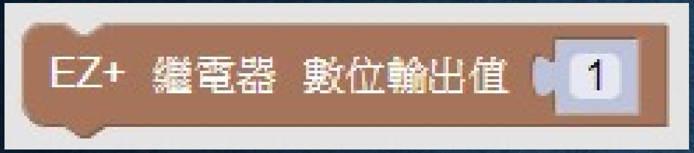

```
name fu_ez_digitalwrite_input
inputs  dummy input
        fields right    text EZ+
                        dropdown pin
                        • 紅燈 , 16
                        • 黃燈 , 12
                        • 綠燈 , 13
        value input value
        fields right    text 數位輸出值
                  type  Number
inline inputs
↕ top+bottom connections
```

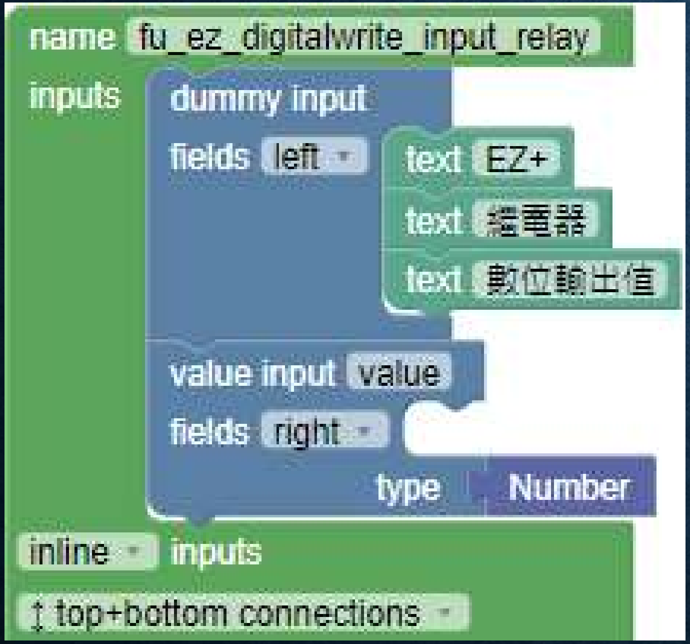
```
Blockly.Blocks['fu_ez_digitalwrite_input'] = {
  init: function() {
    this.appendDummyInput()
        .setAlign(Blockly.ALIGN_RIGHT)
        .appendField("EZ+")
        .appendField(new Blockly.FieldDropdown([
          ["紅燈","16"],
          ["黃燈","12"],
          ["綠燈","13"]
         ]), "pin");
    this.appendValueInput("value")
        .setCheck("Number")
        .setAlign(Blockly.ALIGN_RIGHT)
        .appendField("數位輸出值");
    this.setInputsInline(true);
    this.setPreviousStatement(true, null);
    this.setNextStatement(true, null);
    this.setColour(195);
    this.setTooltip("");
    this.setHelpUrl("");
  }
};
```

```
Blockly.Arduino['fu_ez_digitalwrite_input'] = function(block) {
  var dropdown_pin = block.getFieldValue('pin');
  var value_value = Blockly.Arduino.valueToCode(block, 'value', Blockly.Arduino.ORDER_ATOMIC);

  Blockly.Arduino.setups_['pinmode_'+ dropdown_pin] = 'pinMode('+ dropdown_pin +', OUTPUT);';

  var code = 'digitalWrite('+ dropdown_pin+', '+ value_value +');\n';
  return code;
};
```

繼電器(數位輸出)

```
void setup()
{
  pinMode(25, OUTPUT);  //可自動輸出於setup區
}

void loop()
{
  digitalWrite(25, 1);  //隨積木移動輸出程式碼
}
```



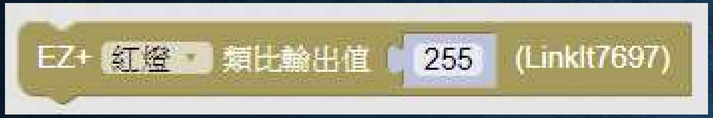


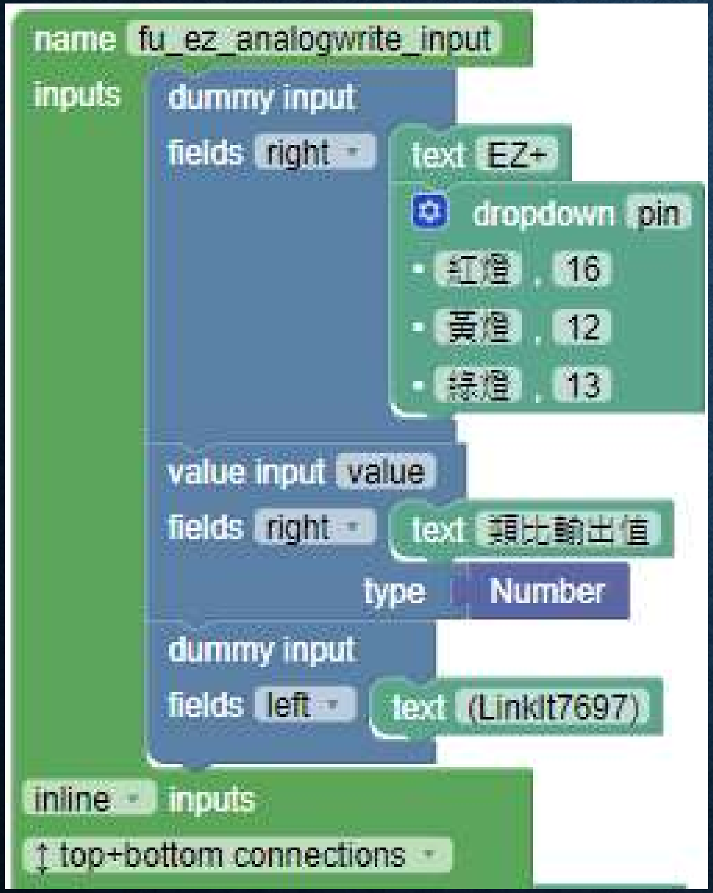
```
Blockly.Blocks['fu_ez_digitalwrite_input_relay'] = {
 init: function() {
   this.appendDummyInput()
       .appendField("EZ+")
       .appendField("繼電器")
       .appendField("數位輸出值");
  this.appendValueInput("value")
       .setCheck("Number")
       .setAlign(Blockly.ALIGN_RIGHT);
   this.setInputsInline(true);
   this.setPreviousStatement(true, null);
   this.setNextStatement(true, null);
   this.setColour(195);
   this.setTooltip("");
   this.setHelpUrl("");
 }
};
```

```
Blockly.Arduino['fu_ez_digitalwrite_input_relay'] = function(block) {
  var pin = 25;
  var value_value = Blockly.Arduino.valueToCode(block, 'value', Blockly.Arduino.ORDER_ATOMIC);

  Blockly.Arduino.setups_['pinmode_'+ pin] = 'pinMode('+ pin +', OUTPUT);';

  var code = 'digitalWrite('+ pin +',  '+ value_value +');\n';
  return code;
};
```

LED燈(類比輸出)

```
void setup()
{
  pinMode(16, OUTPUT); //可自動輸出於setup區
}

void loop()
{
  analogWrite(16, 255); //隨積木移動輸出程式碼
}
```


EZ+ 紅燈 類比輸出值 255 (LinkIt7697)

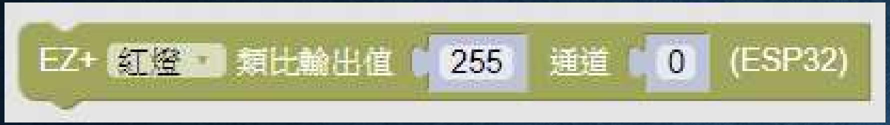

name fu_ez_analogwrite_input
inputs dummy input
fields right text EZ+
dropdown pin
紅燈 16
黃燈 12
綠燈 13
value input value
fields right text 類比輸出值
type Number
dummy input
fields left text (LinkIt7697)
inline inputs
↕ top+bottom connections

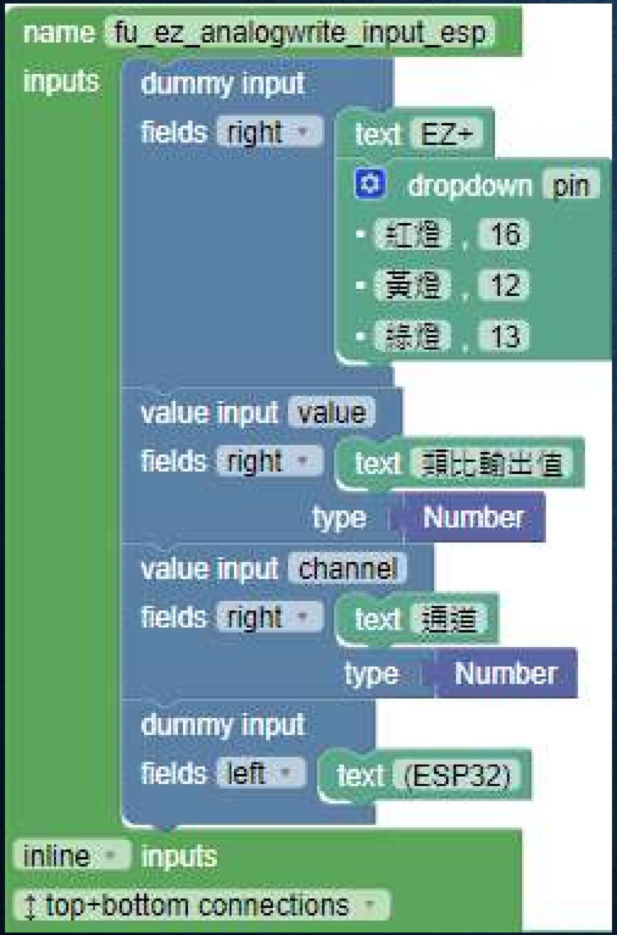
```
Blockly.Blocks['fu_ez_analogwrite_input'] = {
 init: function() {
  this.appendDummyInput()
     .setAlign(Blockly.ALIGN_RIGHT)
     .appendField("EZ+")
     .appendField(new Blockly.FieldDropdown([
       ["紅燈","16"],
       ["黃燈","12"],
       ["綠燈","13"]
     ]), "pin");
  this.appendValueInput("value")
     .setCheck("Number")
     .setAlign(Blockly.ALIGN_RIGHT)
     .appendField("類比輸出值");
  this.setInputsInline(true);
  this.setPreviousStatement(true, null);
  this.setNextStatement(true, null);
  this.setColour(195);
  this.setTooltip("");
  this.setHelpUrl("");
 }
};
```

```
Blockly.Arduino['fu_ez_analogwrite_input'] = function(block) {
  var dropdown_pin = block.getFieldValue('pin');
  var value_value = Blockly.Arduino.valueToCode(block, 'value', Blockly.Arduino.ORDER_ATOMIC);

  Blockly.Arduino.setups_['pinmode_'+ dropdown_pin] = 'pinMode('+ dropdown_pin +', OUTPUT);';

  var code = 'analogWrite('+ dropdown_pin+',  '+ value_value +');\n';
  return code;
};
```

LED燈(ESP32類比輸出)

```
void setup()
{
  ledcAttachPin(16,0);   //可自動輸出於setup區
  ledcSetup(0,5000,8); //可自動輸出於setup區
}

void loop()
{
  ledcWrite(0, 255); //隨積木移動輸出程式碼
}
```



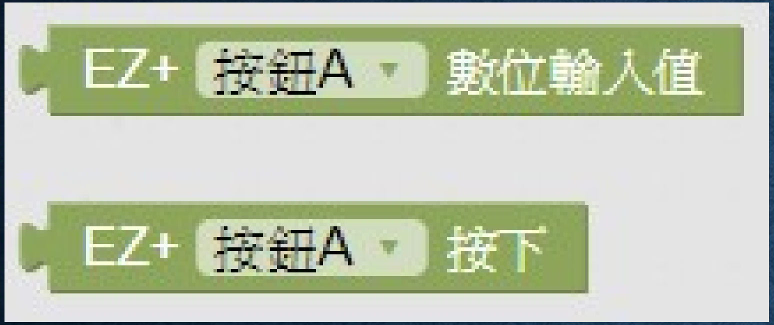

name fu_ez_analogwrite_input_esp
inputs dummy input
fields right
text EZ+
dropdown pin
紅燈 , 16
黃燈 , 12
綠燈 , 13
value input value
fields right
text 類比輸出值
type Number
value input channel
fields right
text 通道
type Number
dummy input
fields left
text (ESP32)
inline inputs
↕ top+bottom connections

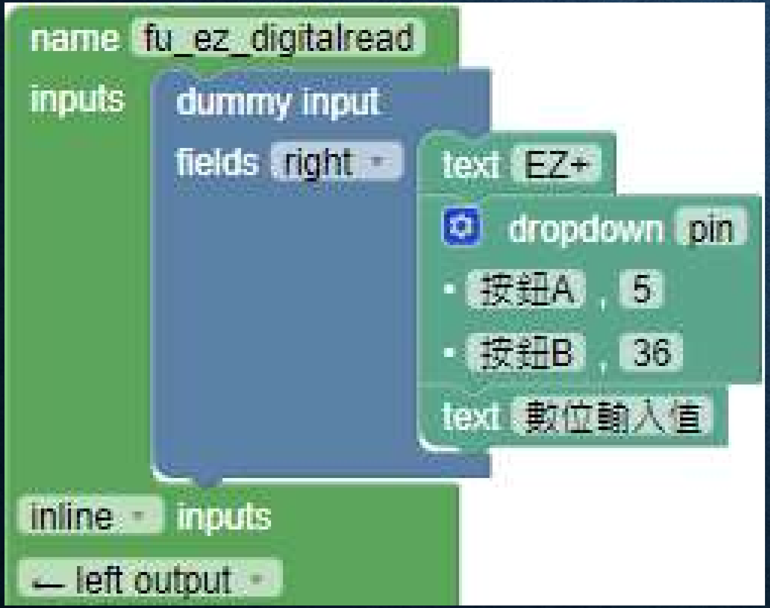
```
Blockly.Blocks['fu_ez_analogwrite_input_esp'] = {
  init: function() {
    this.appendDummyInput()
      .setAlign(Blockly.ALIGN_RIGHT)
      .appendField("EZ+")
      .appendField(new Blockly.FieldDropdown([
        ["紅燈","16"],
        ["黃燈","12"],
        ["綠燈","13"]
      ]), "pin");
    this.appendValueInput("value")
      .setCheck("Number")
      .setAlign(Blockly.ALIGN_RIGHT)
      .appendField("類比輸出值");
    this.appendValueInput("channel")
      .setCheck("Number")
      .setAlign(Blockly.ALIGN_RIGHT)
      .appendField("通道");
    this.setInputsInline(true);
    this.setPreviousStatement(true, null);
    this.setNextStatement(true, null);
    this.setColour(195);
    this.setTooltip("");
    this.setHelpUrl("");
  }
};
```

```
Blockly.Arduino['fu_ez_analogwrite_input_esp'] = function(block) {
  var dropdown_pin = block.getFieldValue('pin');
  var value_value = Blockly.Arduino.valueToCode(block, 'value', Blockly.Arduino.ORDER_ATOMIC);
  var value_channel = Blockly.Arduino.valueToCode(block, 'channel', Blockly.Arduino.ORDER_ATOMIC);

  Blockly.Arduino.setups_['ledc_'+ dropdown_pin] = ''+
              'ledcAttachPin('+ dropdown_pin+','+ value_channel+');\n  ledcSetup('+ value_channel+',5000,8);';

  var code = 'ledcWrite('+value_channel+', '+value_value+');\n';
  return code;
};
```

按鈕(數位輸入)

```
void setup()
{
    pinMode(5, INPUT_PULLUP);  //可自動產生於setup區
}

void loop()
{
  digitalRead(5)   //隨積木移動輸出程式碼
}
```



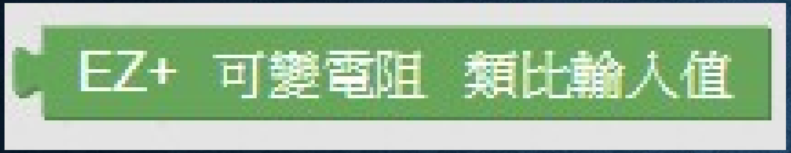


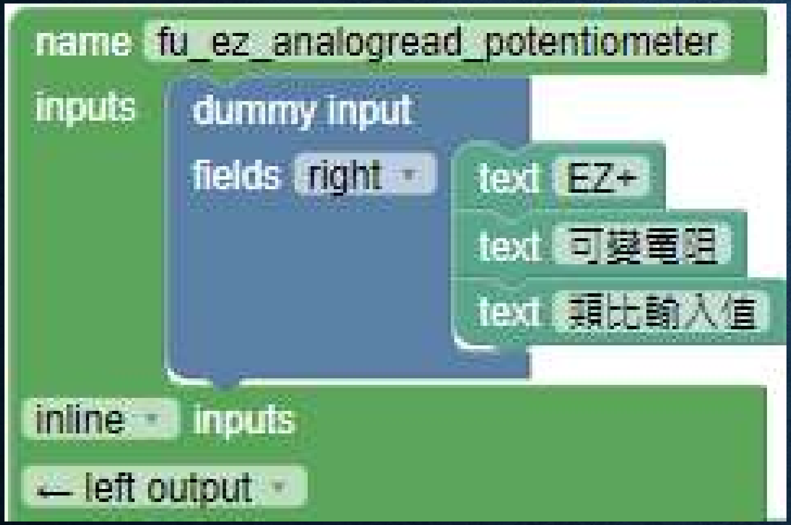
```
Blockly.Blocks['fu_ez_digitalread'] = {
  init: function() {
    this.appendDummyInput()
        .setAlign(Blockly.ALIGN_RIGHT)
        .appendField("EZ+")
        .appendField(new Blockly.FieldDropdown([
            ["按鈕A","5"],
            ["按鈕B","36"]
        ]), "pin")
         .appendField("數位輸入值");
    this.setInputsInline(true);
    this.setOutput(true, "Number");
    this.setColour(195);
    this.setTooltip("");
    this.setHelpUrl("");
  }
};
```

```
Blockly.Arduino['fu_ez_digitalread'] = function(block) {
  var pin = block.getFieldValue('pin');

  Blockly.Arduino.setups_['pinmode_' + pin] = 'pinMode('+ pin +', INPUT_PULLUP);';

  var code = 'digitalRead('+ pin +')' ;
  return [code, Blockly.Arduino.ORDER_NONE];
};
```

可變電阻(類比輸入)

```
void setup()
{
}

void loop()
{
  analogRead(34)   //隨積木移動輸出程式碼
}
```



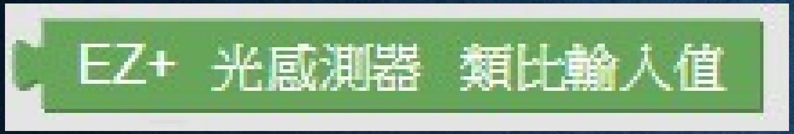


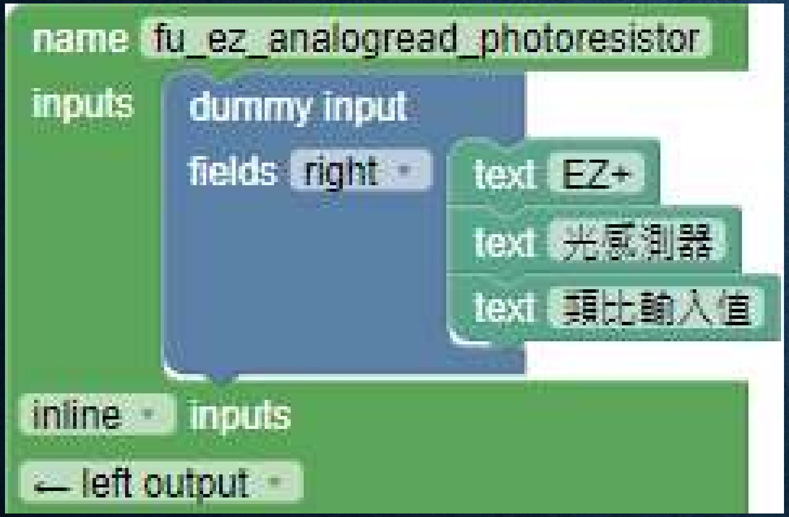
```
Blockly.Blocks['fu_ez_analogread_potentiometer'] = {
  init: function() {
    this.appendDummyInput()
        .setAlign(Blockly.ALIGN_RIGHT)
        .appendField("EZ+")
        .appendField("可變電阻")
        .appendField("類比輸入值");
    this.setInputsInline(true);
    this.setOutput(true, "Number");
    this.setColour(195);
    this.setTooltip("");
    this.setHelpUrl("");
  }
};
```

```
Blockly.Arduino['fu_ez_analogread_potentiometer'] = function(block) {
  var pin = 34;
  Blockly.Arduino.setups_['pinmode_'+ pin] = 'pinMode('+ pin +', INPUT);';

  var code = 'analogRead('+ pin+')';
  return [code, Blockly.Arduino.ORDER_NONE];
};
```

光感測器(類比輸入)

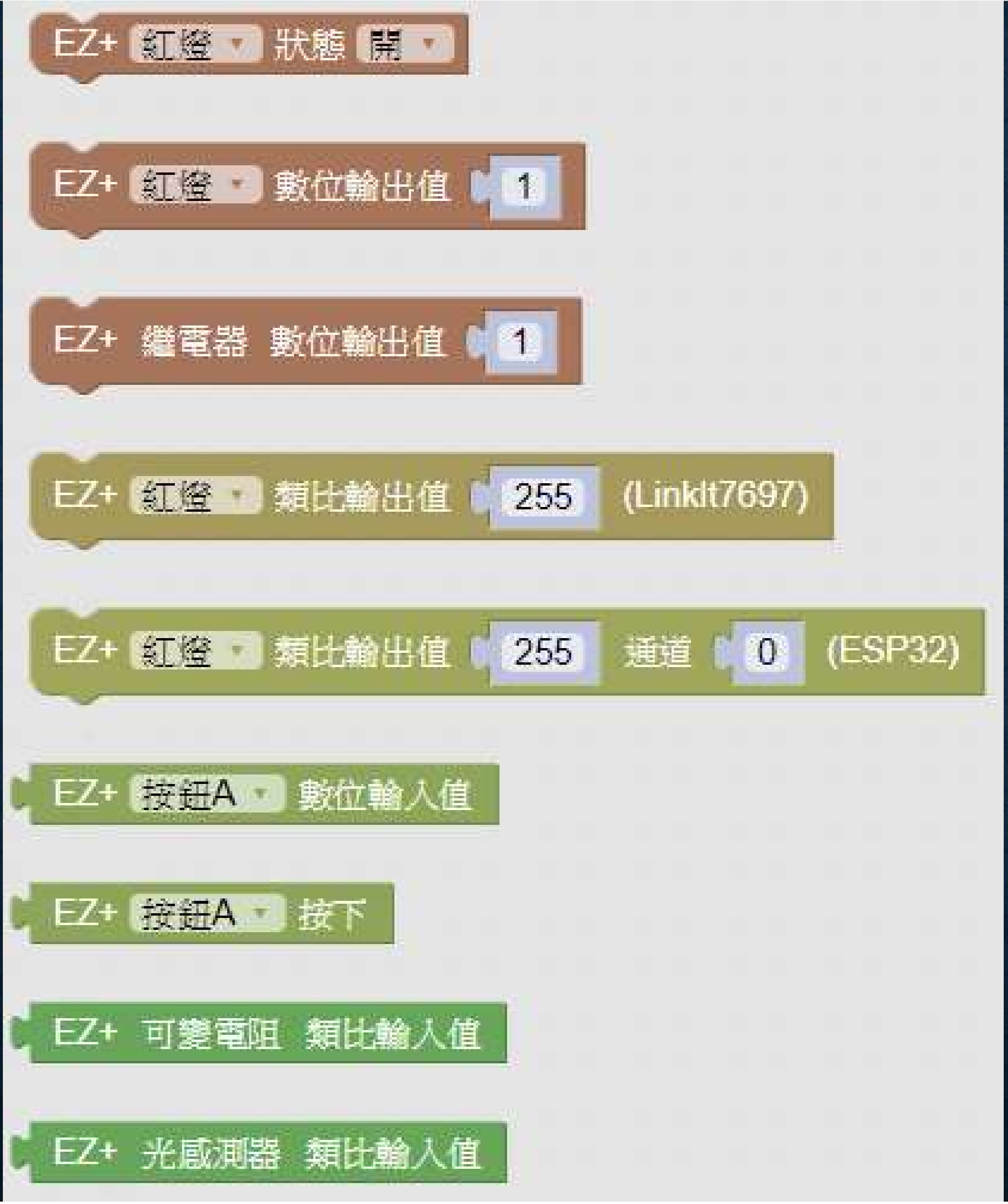
```
void setup()
{
}

void loop()
{
    analogRead(39)   //隨積木移動輸出程式碼
}
```





```
Blockly.Blocks['fu_ez_analogread_photoresistor'] = {
  init: function() {
    this.appendDummyInput()
        .setAlign(Blockly.ALIGN_RIGHT)
        .appendField("EZ+")
        .appendField("光感測器")
        .appendField("類比輸入值");
    this.setInputsInline(true);
    this.setOutput(true, "Number");
    this.setColour(195);
    this.setTooltip("");
    this.setHelpUrl("");
  }
};
```

```
Blockly.Arduino['fu_ez_analogread_photoresistor'] = function(block) {
  var pin = 39;
  Blockly.Arduino.setups_['pinmode_'+ pin] = 'pinMode('+ pin +', INPUT);';

  var code = 'analogRead('+ pin+')';
  return [code, Blockly.Arduino.ORDER_NONE];
};
```


EZ+ 紅燈 狀態 開

EZ+ 紅燈 數位輸出值 1

EZ+ 繼電器 數位輸出值 1

EZ+ 紅燈 類比輸出值 255 (LinkIt7697)

EZ+ 紅燈 類比輸出值 255 通道 0 (ESP32)

EZ+ 按鈕A 數位輸入值

EZ+ 按鈕A 按下

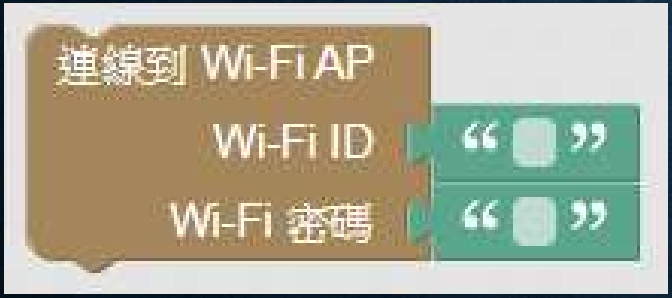
EZ+ 可變電阻 類比輸入值

EZ+ 光感測器 類比輸入值

```xml
<category id="ez" name="EZ+" colour="100">
  <block type="fu_ez_digitalwrite">
    <field name="pin">16</field>
    <field name="value">HIGH</field>
  </block>
  <block type="fu_ez_digitalwrite_input">
    <field name="pin">16</field>
    <value name="value">
      <shadow type="math_number">
        <field name="NUM">1</field>
      </shadow>
    </value>
  </block>
  <block type="fu_ez_digitalwrite_input_relay">
   <value name="value">
      <shadow type="math_number">
        <field name="NUM">1</field>
      </shadow>
    </value>
  </block>
  <block type="fu_ez_analogwrite_input">
    <field name="pin">16</field>
    <value name="value">
      <shadow type="math_number">
        <field name="NUM">255</field>
      </shadow>
    </value>
  </block>
  <block type="fu_ez_analogwrite_input_esp">
    <field name="pin">16</field>
    <value name="value">
      <shadow type="math_number">
        <field name="NUM">255</field>
      </shadow>
    </value>
    <value name="channel">
      <shadow type="math_number">
        <field name="NUM">0</field>
      </shadow>
    </value>
  </block>
  <block type="fu_ez_digitalread">
    <field name="pin">5</field>
  </block>
  <block type="fu_ez_analogread_potentiometer"></block>
  <block type="fu_ez_analogread_photoresistor"></block>
</category>
```

課堂練習

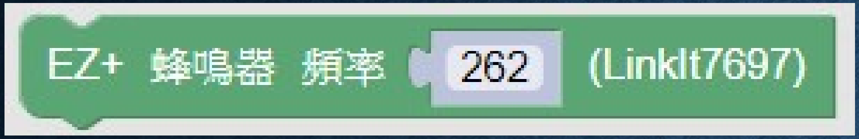
請試做**WiFi**連線的積木

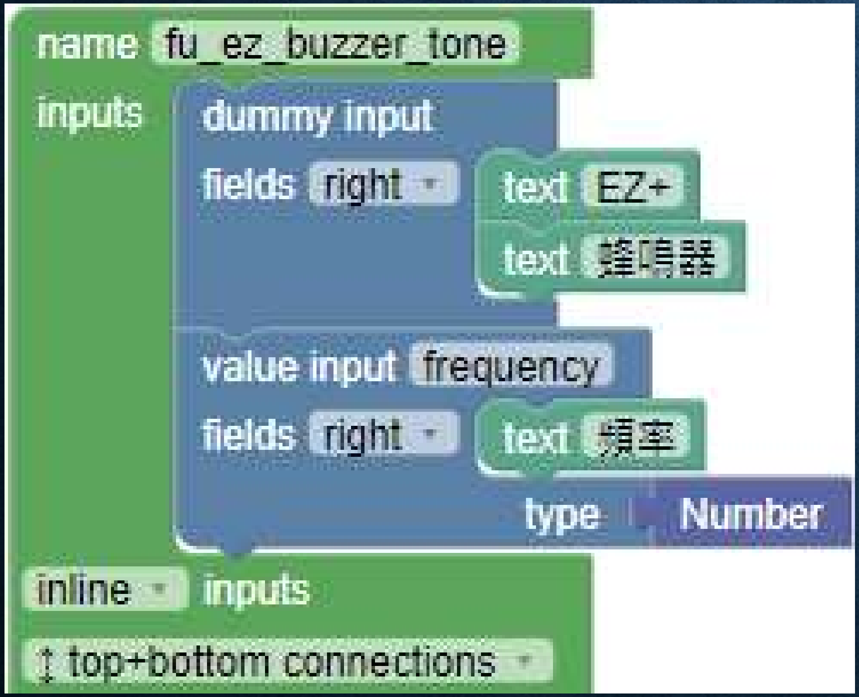


```
#include <WiFi.h>
char wifi_ssid[] = "helloworld";
char wifi_pass[] = "12345678";

void setup()
{
  while (WiFi.begin(wifi_ssid, wifi_pass) != WL_CONNECTED){

  }
}

void loop()
{
}
```

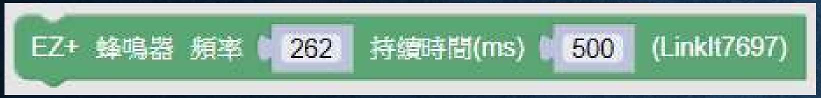
蜂鳴器

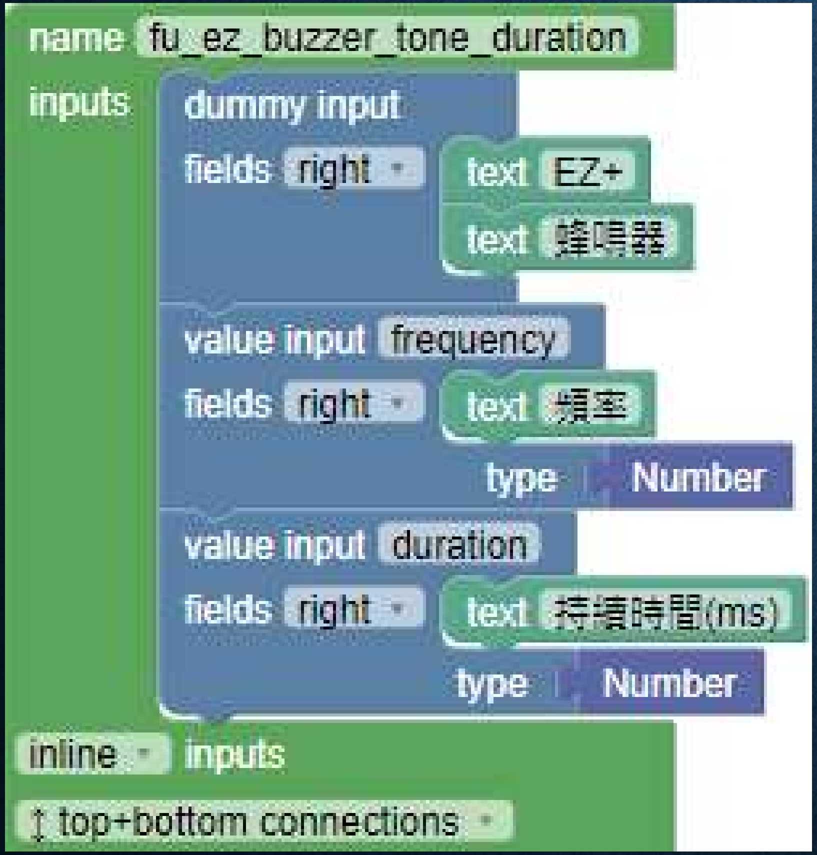
```
void setup()
{
   pinMode(14, OUTPUT);  //可自動產生於setup區
}

void loop()
{
   tone(14, 262);              //積木1
   delay(1000);
   noTone(14);                 //積木2
   tone(14, 262, 1000);        //積木3
}
```


EZ+ 蜂鳴器 頻率 262 (LinkIt7697)

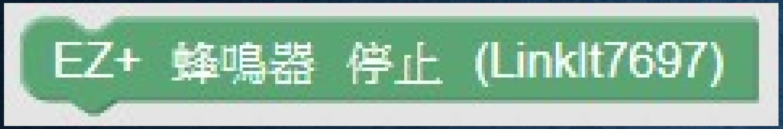

name fu_ez_buzzer_tone
inputs dummy input
fields right text EZ+
text 蜂鳴器
value input frequency
fields right text 頻率
type Number
inline inputs
top+bottom connections

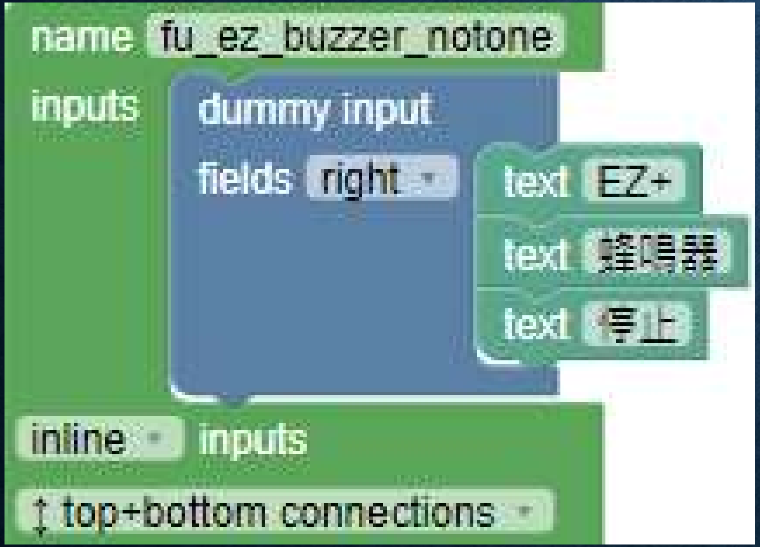
```
Blockly.Blocks['fu_ez_buzzer_tone'] = {
 init: function() {
  this.appendDummyInput()
    .setAlign(Blockly.ALIGN_RIGHT)
    .appendField("EZ+")
    .appendField("蜂鳴器");
  this.appendValueInput("frequency")
    .setCheck("Number")
    .setAlign(Blockly.ALIGN_RIGHT)
    .appendField("頻率");
  this.setInputsInline(true);
  this.setPreviousStatement(true, null);
  this.setNextStatement(true, null);
  this.setColour(195);
  this.setTooltip("");
  this.setHelpUrl("");
 }
};
```





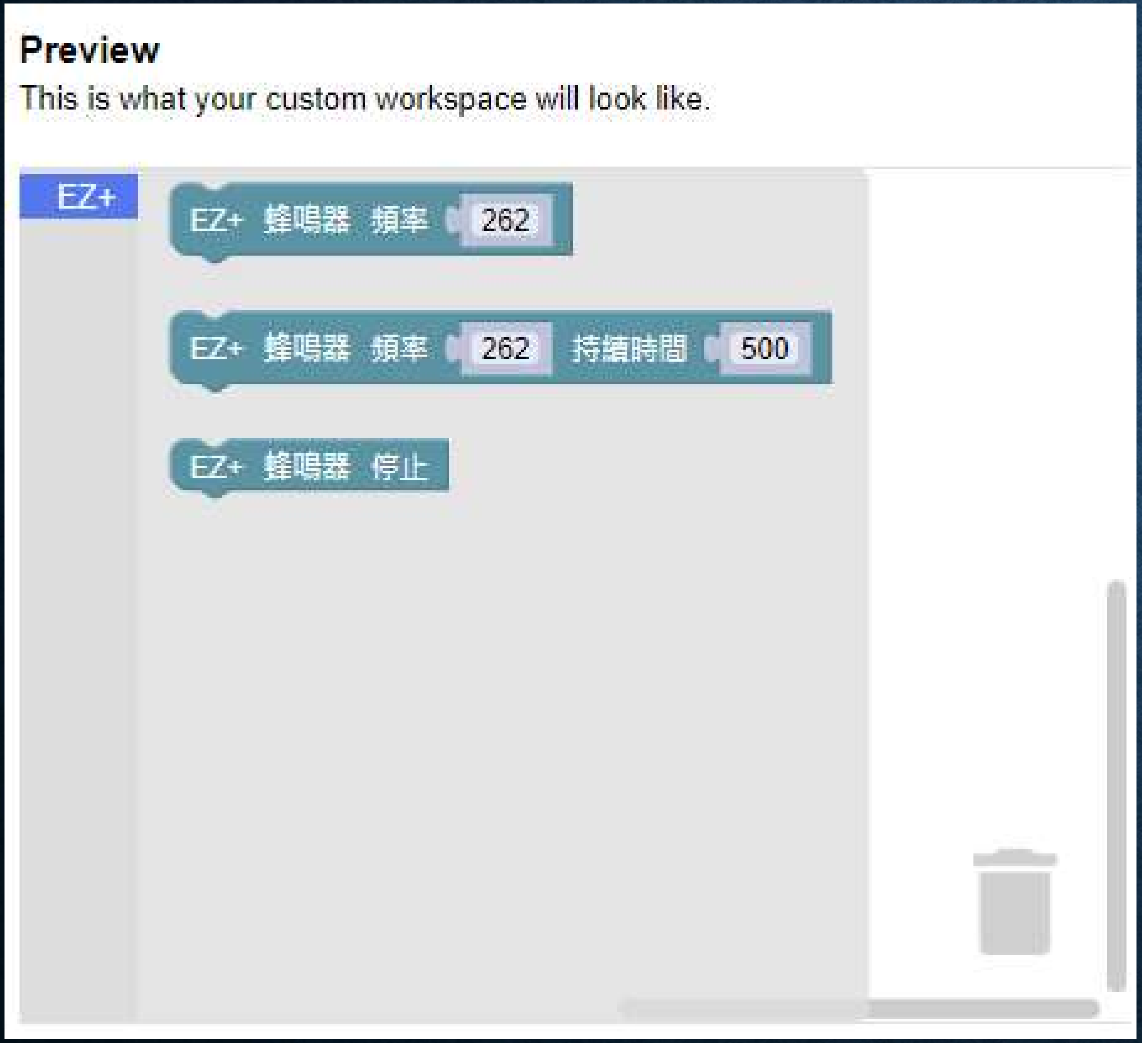
```
Blockly.Blocks['fu_ez_buzzer_tone_duration'] = {
  init: function() {
    this.appendDummyInput()
        .setAlign(Blockly.ALIGN_RIGHT)
        .appendField("EZ+")
        .appendField("蜂鳴器");
    this.appendValueInput("frequency")
        .setCheck("Number")
        .setAlign(Blockly.ALIGN_RIGHT)
        .appendField("頻率");
    this.appendValueInput("duration")
        .setCheck("Number")
        .setAlign(Blockly.ALIGN_RIGHT)
        .appendField("持續時間");
    this.setInputsInline(true);
    this.setPreviousStatement(true, null);
    this.setNextStatement(true, null);
    this.setColour(195);
    this.setTooltip("");
    this.setHelpUrl("");
  }
};
```


EZ+ 蜂鳴器 停止 (LinkIt7697)


name fu_ez_buzzer_notone
inputs dummy input
fields right
text EZ+
text 蜂鳴器
text 停止
inline inputs
top+bottom connections

```
Blockly.Blocks['fu_ez_buzzer_notone'] = {
  init: function() {
    this.appendDummyInput()
        .setAlign(Blockly.ALIGN_RIGHT)
        .appendField("EZ+")
        .appendField("蜂鳴器")
        .appendField("停止");
    this.setInputsInline(true);
    this.setPreviousStatement(true, null);
    this.setNextStatement(true, null);
    this.setColour(195);
 this.setTooltip("");
 this.setHelpUrl("");
  }
};
```

```
Blockly.Arduino['fu_ez_buzzer_tone'] = function(block) {
 var pin = 14;
 var value_frequency = Blockly.Arduino.valueToCode(block, 'frequency', Blockly.Arduino.ORDER_ATOMIC);

 Blockly.Arduino.setups_['pinmode_'+ pin] = 'pinMode('+ pin +', OUTPUT);';

 var code = 'tone('+ pin +', '+ value_frequency +');\n';
 return code;
};


Blockly.Arduino['fu_ez_buzzer_tone_duration'] = function(block) {
 var pin = 14;
 var value_frequency = Blockly.Arduino.valueToCode(block, 'frequency', Blockly.Arduino.ORDER_ATOMIC);
 var value_duration = Blockly.Arduino.valueToCode(block, 'duration', Blockly.Arduino.ORDER_ATOMIC);

 Blockly.Arduino.setups_['pinmode_' + pin] = 'pinMode('+ pin +', OUTPUT);';

 var code = 'tone('+ pin +', '+ value_frequency +', '+ value_duration +');\n';
 return code;
};


Blockly.Arduino['fu_ez_buzzer_notone'] = function(block) {
 var pin = 14;

 Blockly.Arduino.setups_['pinmode_' + pin] = 'pinMode('+ pin +', OUTPUT);';
 var code = 'noTone('+ pin +');\n';
 return code;
};
```



Preview

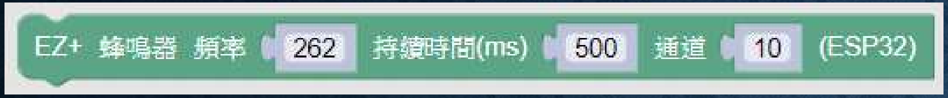
This is what your custom workspace will look like.

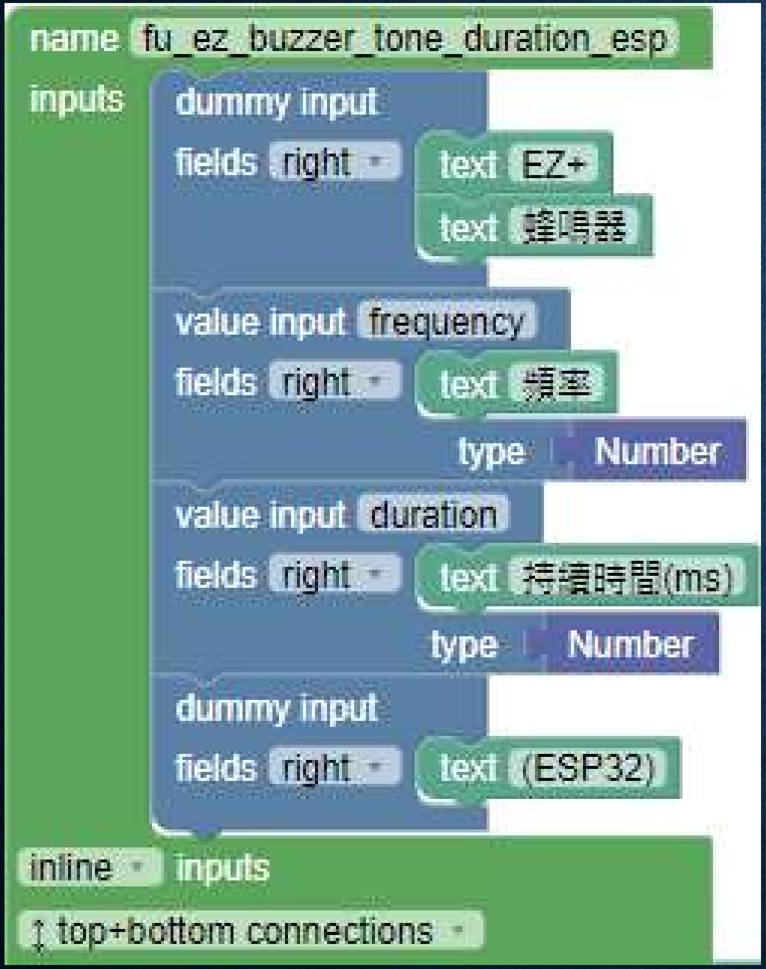
```
<category id="ez" name="EZ+" colour="100">
    <block type="fu_ez_buzzer_tone">
    <value name="frequency">
      <shadow type="math_number">
        <field name="NUM">262</field>
      </shadow>
    </value>
  </block>
  <block type="fu_ez_buzzer_tone_duration">
    <value name="frequency">
      <shadow type="math_number">
        <field name="NUM">262</field>
      </shadow>
    </value>
    <value name="duration">
      <shadow type="math_number">
        <field name="NUM">500</field>
      </shadow>
    </value>
  </block>
  <block type="fu_ez_buzzer_notone"></block>
</category>
```

ESP32蜂鳴器

```
void setup()
{
  //可自動產生於setup區
  ledcSetup(10 , 2000, 8);
  ledcAttachPin(14, 10);
}

void loop()
{
  ledcWriteTone(10, 262);    //積木
  delay(500);
  ledcWriteTone(10, 0);
}
```

```
void setup()
{
 //可自動產生於setup區
  ledcSetup(10 , 2000, 8);
  ledcAttachPin(14, 10);
}

void loop()
{
 tone(10,262,500);  //積木
}

//自動產生於definition區
void tone(int channel,int frequency, int delaytime) {
  ledcWriteTone(channel, frequency);
  delay(delaytime);
  ledcWriteTone(channel, 0);
}
```




```
name fu_ez_buzzer_tone_duration_esp
inputs
  dummy input
  fields right   text EZ+
                 text 蜂鳴器
  value input frequency
  fields right   text 頻率
                 type Number
  value input duration
  fields right   text 持續時間(ms)
                 type Number
  dummy input
  fields right   text (ESP32)
inline inputs
top+bottom connections
```

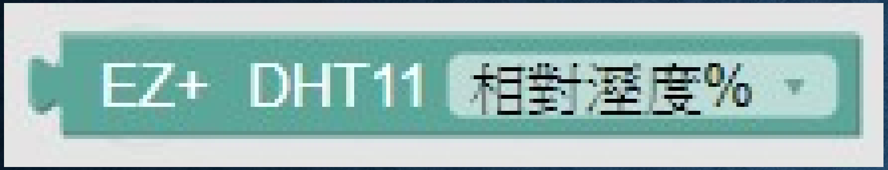
```
Blockly.Blocks['fu_ez_buzzer_tone_duration_esp'] = {
 init: function() {
  this.appendDummyInput()
    .setAlign(Blockly.ALIGN_RIGHT)
    .appendField("EZ+")
    .appendField("蜂鳴器");
  this.appendValueInput("frequency")
    .setCheck("Number")
    .setAlign(Blockly.ALIGN_RIGHT)
    .appendField("頻率");
  this.appendValueInput("duration")
    .setCheck("Number")
    .setAlign(Blockly.ALIGN_RIGHT)
    .appendField("持續時間(ms)");
  this.appendDummyInput()
    .setAlign(Blockly.ALIGN_RIGHT)
    .appendField("(ESP32)");
  this.setInputsInline(true);
  this.setPreviousStatement(true, null);
  this.setNextStatement(true, null);
  this.setColour(195);
  this.setTooltip("");
  this.setHelpUrl("");
 }
};
```

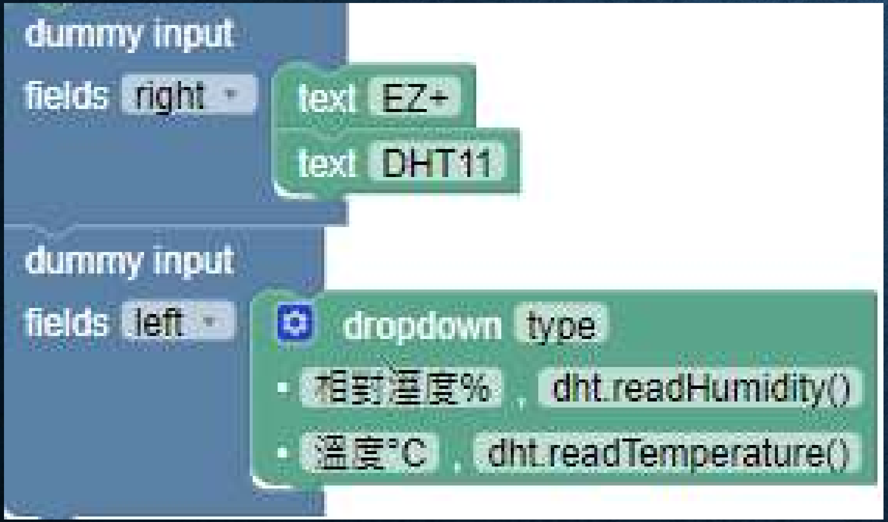
```
Blockly.Arduino['fu_ez_buzzer_tone_duration_esp'] = function(block) {
  var pin = 14;
  var value_frequency = Blockly.Arduino.valueToCode(block, 'frequency', Blockly.Arduino.ORDER_ATOMIC);
  var value_duration = Blockly.Arduino.valueToCode(block, 'duration', Blockly.Arduino.ORDER_ATOMIC);
  var value_channel = Blockly.JavaScript.valueToCode(block, 'channel', Blockly.JavaScript.ORDER_ATOMIC);

  Blockly.Arduino.setups_['ledc_'+ pin] = 'ledcSetup('+ value_channel +' , 2000, 8);\n'+
                                          'ledcAttachPin('+ pin +', '+ value_channel +');\n'+

  Blockly.Arduino.definitions_['tone'] = 'void tone(int channel, int frequency, int delaytime) {\n'+
                                             '  ledcWriteTone(channel, frequency);\n'+
                                             '  delay(delaytime);\n'+
                                             '  ledcWriteTone(channel, 0);\n'+
                                             '}';

  var code = 'tone('+ value_channel +', '+ value_frequency +', '+ value_duration +');\n' ;
  return code;
};
```

DHT11溫溼度感測器

函式庫說明

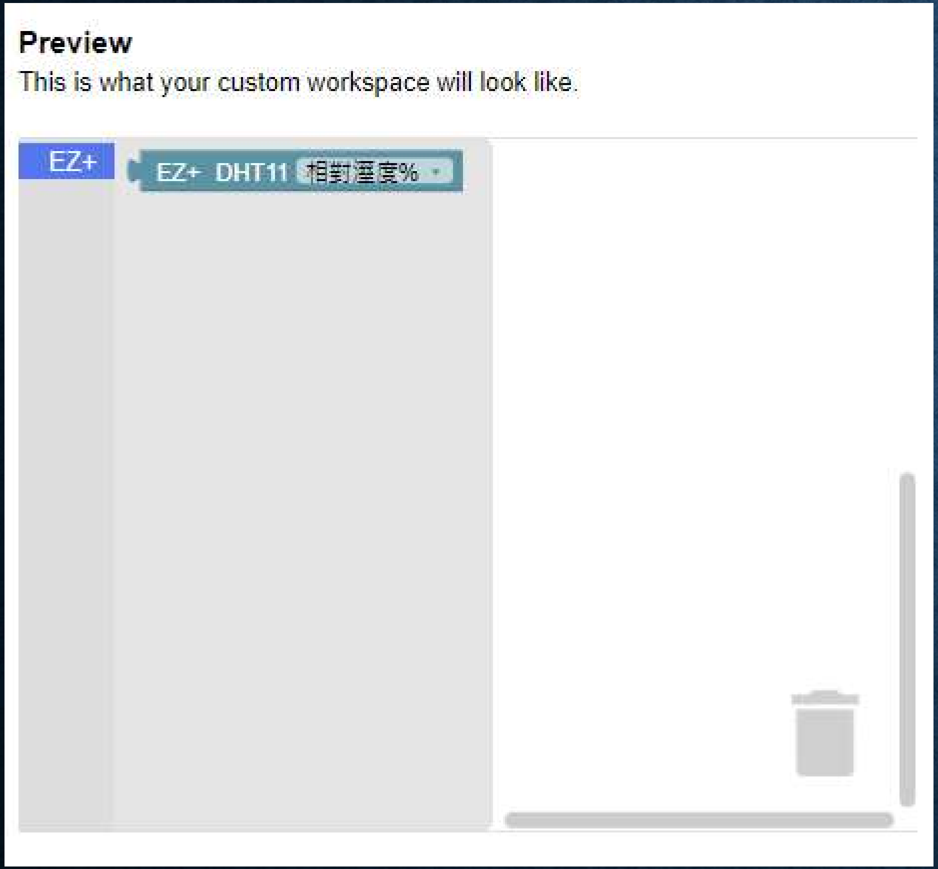
```
#include <DHT.h>   //自動輸出於definition區
DHT dht (15, DHT11);    //自動輸出於definition區

void setup()
{
  dht.begin();  //自動輸出於setup區
}

void loop()
{
  dht.readTemperature()   //隨積木移動輸出程式碼積木
  dht.readHumidity()   //隨積木移動輸出程式碼
}
```





```
Blockly.Blocks['fu_ez_dht11'] = {
 init: function() {
  this.appendDummyInput()
     .setAlign(Blockly.ALIGN_RIGHT)
     .appendField("EZ+")
     .appendField("DHT11");
  this.appendDummyInput()
     .appendField(new Blockly.FieldDropdown([
      ["相對溼度%","dht.readHumidity()"],
      ["溫度°C","dht.readTemperature()"]
      ]), "type");
  this.setInputsInline(true);
  this.setOutput(true, null);
  this.setColour(195);
 this.setTooltip("");
 this.setHelpUrl("");
  }
};
```

```
Blockly.Arduino['fu_ez_dht11'] = function(block) {
  var pin = 15;
  var dropdown_type = block.getFieldValue('type');

  Blockly.Arduino.definitions_['dht11_library'] = '#include <DHT.h>';
  Blockly.Arduino.definitions_['dht11_' + pin] = 'DHT dht ('+pin+', DHT11);';
  Blockly.Arduino.setups_['dht11_begin'] = 'dht.begin();';

  var code = dropdown_type;
  return [code, Blockly.Arduino.ORDER_NONE];
};
```



Preview

This is what your custom workspace will look like.

```
<category id="ez" name="EZ+" colour="100">
   <block type="fu_ez_dht11">
     <field name="type">dht.readHumidity()</field>
   </block>
</category>
```

多國語系積木製作

積木定義 格式一

```
Blockly.Blocks['fu_ez_dht11'] = {
 init: function() {
   this.appendDummyInput()
       .setAlign(Blockly.ALIGN_RIGHT)
       .appendField("EZ+")
       .appendField("DHT11");
   this.appendDummyInput()
       .appendField(new Blockly.FieldDropdown([
        ["相對溼度%","dht.readHumidity()"],
        ["溫度°C","dht.readTemperature()"]
        ]), "type");
   this.setInputsInline(true);
   this.setOutput(true, null);
   this.setColour(195);
 this.setTooltip("");
 this.setHelpUrl("");
  }
};
```

```
Blockly.Blocks['fu_ez_dht11'] = {
 init: function() {
   this.appendDummyInput()
       .setAlign(Blockly.ALIGN_RIGHT)
       .appendField("%{BKY_EZ_TITLE}")
       .appendField("%{BKY_EZ_DHT11_TITLE}");
   this.appendDummyInput()
       .appendField(new Blockly.FieldDropdown([
        ["%{BKY_EZ_DHT11_HUMIDITY}","dht.readHumidity()"],
        ["%{BKY_EZ_DHT11_TEMPERATURE}","dht.readTemperature()"]
        ]), "type");
   this.setInputsInline(true);
   this.setOutput(true, null);
   this.setColour(195);
 this.setTooltip("");
 this.setHelpUrl("");
  }
};
```

積木定義 格式二

```
Blockly.Blocks['fu_ez_dht11'] = {
 init: function() {
  this.appendDummyInput()
    .setAlign(Blockly.ALIGN_RIGHT)
    .appendField("EZ+")
    .appendField("DHT11");
  this.appendDummyInput()
    .appendField(new Blockly.FieldDropdown([
     ["相對溼度%","dht.readHumidity()"],
     ["溫度°C","dht.readTemperature()"]
     ]), "type");
  this.setInputsInline(true);
  this.setOutput(true, null);
  this.setColour(195);
 this.setTooltip("");
 this.setHelpUrl("");
 }
};
```

```
Blockly.Blocks['fu_ez_dht11'] = {
 init: function() {
  this.appendDummyInput()
    .setAlign(Blockly.ALIGN_RIGHT)
    .appendField(Blockly.Msg["EZ_TITLE"])
    .appendField(Blockly.Msg["EZ_DHT11_TITLE"]);
  this.appendDummyInput()
    .appendField(new Blockly.FieldDropdown([
     [Blockly.Msg["EZ_DHT11_HUMIDITY"],"dht.readHumidity()"],
     [Blockly.Msg["EZ_DHT11_TEMPERATURE"],"dht.readTemperature()"]
     ]), "type");
  this.setInputsInline(true);
  this.setOutput(true, null);
  this.setColour(195);
 this.setTooltip("");
 this.setHelpUrl("");
 }
};
```

積木目錄

```
<category id="ez" name="EZ+" colour="100">
  <block type="fu_ez_dht11">
    <field name="type">dht.readHumidity()</field>
  </block>
</category>

<category id="ez" name="%{BKY_EZ_CATEGORY}" colour="%{BKY_EZ_CATEGORY_HUE}">
  <block type="fu_ez_dht11">
    <field name="type">dht.readHumidity()</field>
  </block>
</category>
```

語系變數

en.js

```
Blockly.Msg["EZ_TITLE"] = "EZ+";
Blockly.Msg["EZ_DHT11_TITLE"] = "DHT11";
Blockly.Msg["EZ_DHT11_HUMIDITY"] = "Humidity %";
Blockly.Msg["EZ_DHT11_TEMPERATURE"] = "Temperature ˚C";
Blockly.Msg["EZ_CATEGORY"] = "EZ Start Kit +";
Blockly.Msg["EZ_CATEGORY_HUE"] = "200";
```

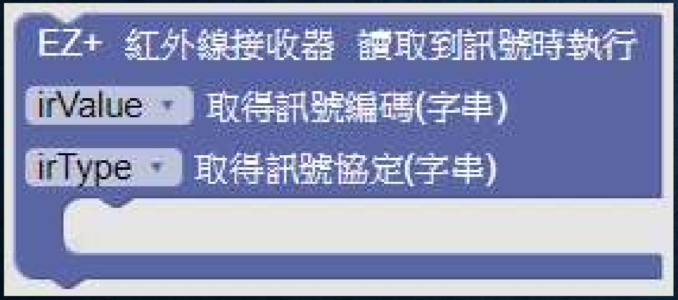
Zh-hant.js

```
Blockly.Msg["EZ_TITLE"] = "EZ+";
Blockly.Msg["EZ_DHT11_TITLE"] = "DHT11溫溼度感測器";
Blockly.Msg["EZ_DHT11_HUMIDITY"] = "相對濕度 %";
Blockly.Msg["EZ_DHT11_TEMPERATURE"] = "溫度 ˚C";
Blockly.Msg["EZ_CATEGORY"] = "EZ Start Kit +";
Blockly.Msg["EZ_CATEGORY_HUE"] = "200";
```

課堂練習

請依照範例程式碼設計紅外線接收器積木。

函式庫說明


EZ+ 紅外線接收器 讀取到訊號時執行
irValue 取得訊號編碼(字串)
irType 取得訊號協定(字串)

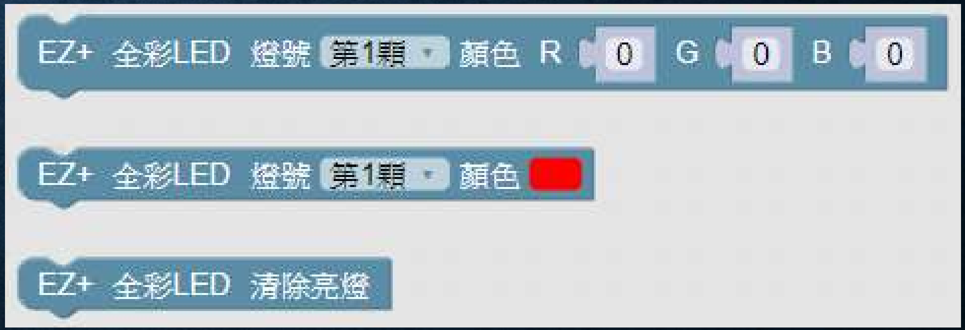
```
#include <IRremote.h>    //定義區自動輸出
IRrecv irrecv(33);
decode_results results;

void setup() {
  irrecv.enableIRIn();    //setup區自動輸出
}

void loop() {
  receiveData()  //隨積木移動輸出程式碼
}

String receiveData() {    //定義區自動輸出
  if (irrecv.decode(&results)) {
    String data = String(results.value, HEX);
    irrecv.resume();
    return data;
  }
  else
    return "";
  delay(300);
}
```

課堂練習

請依照範例程式碼設計控制ws2812燈條的積木。

函式庫說明

```
#include <Adafruit_NeoPixel.h>    //定義區自動輸出
Adafruit_NeoPixel pixels(3, 26, NEO_GRB + NEO_KHZ800);

void setup() {
  pixels.begin();  //setup區自動輸出
  pixels.clear();

 pixels.setPixelColor(0, pixels.Color(255, 255, 255));  //隨積木移動輸出程式碼
  pixels.show();
}

void loop() {
}
```



EZ+ 全彩LED 燈號 第1顆 顏色 R 0 G 0 B 0

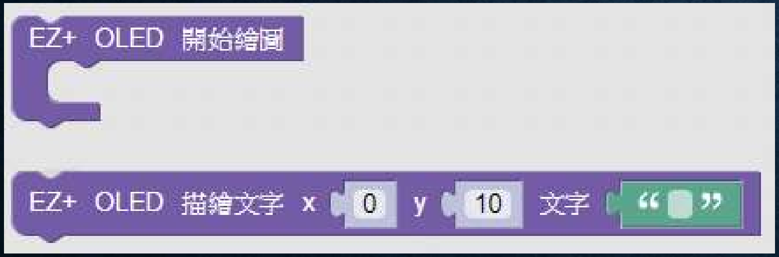
EZ+ 全彩LED 燈號 第1顆 顏色 🟥

EZ+ 全彩LED 清除亮燈

課堂練習

請依照範例程式碼設計控制OLED顯示器的積木。

函式庫說明



```
#include <U8g2lib.h>   //定義區自動輸出
#include <Wire.h>
U8G2_SSD1306_128X64_NONAME_F_HW_I2C u8g2(U8G2_R0,/* reset=*/ U8X8_PIN_NONE);

void setup(void) {
 u8g2.begin();  //setup區自動輸出
 u8g2.setFont(u8g2_font_ncenB08_tr);

 u8g2.clearBuffer();   //積木1

 u8g2.drawStr(0,10,"Hello World!");       //積木2
 u8g2.drawStr(0,20,"Are you ready?");

 u8g2.sendBuffer();   //積木1
}

void loop(void) {
}
```

BLOCKLYDUINO F1

開發環境介紹

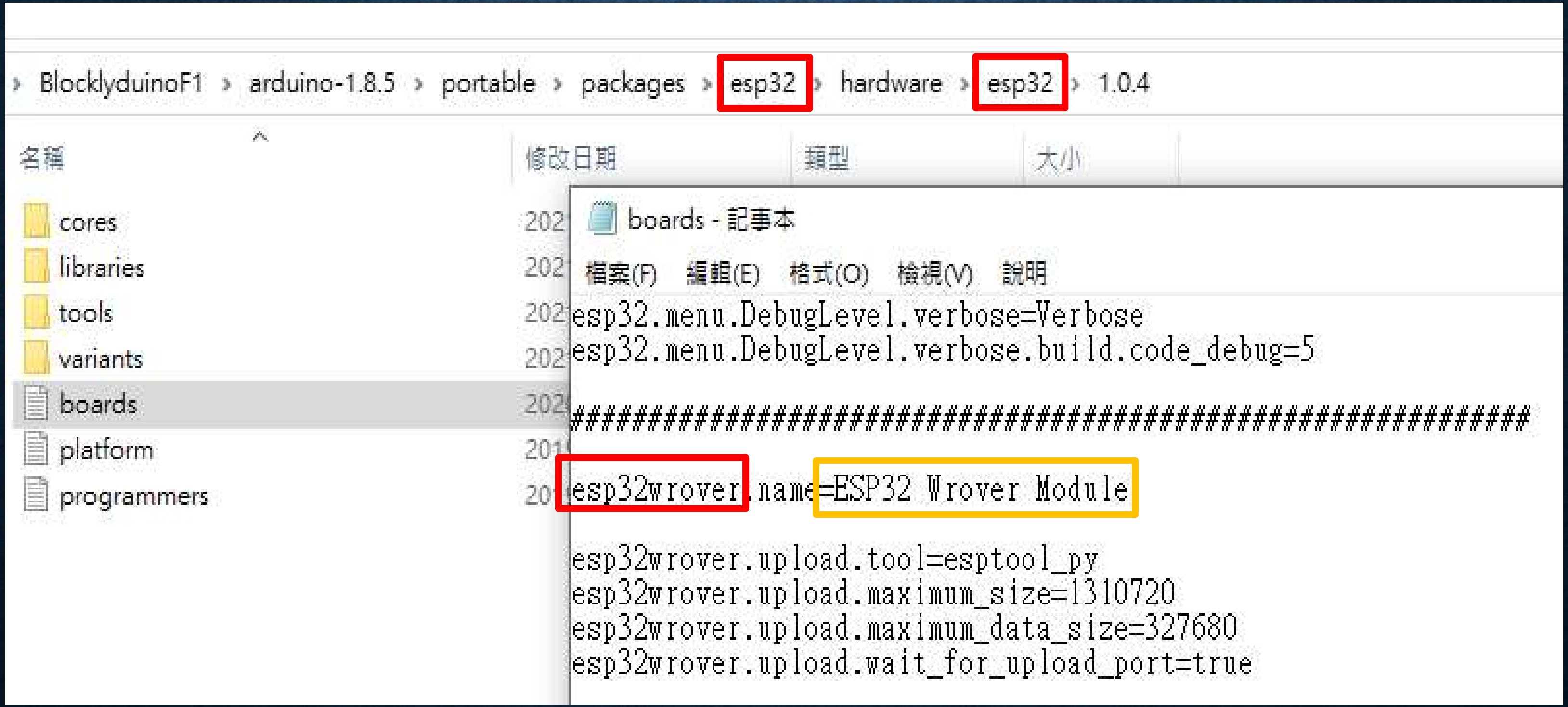
NW.JS簡介

1. 使用HTML5、CSS3和WebGL等web技術，編寫原生應用的新途徑
2. 全面支持所有瀏覽器特性
3. 全面支持Node.js的API及所有第三方模組
4. 可以從DOM和Web Worker層面，調用Node.js的模組
5. JavaScript原始碼保護
6. 一次編寫，就可以運行在多平台上，包括：Linux、Mac OS X和Windows
7. 可以支持Chrome API，執行啟動本機應用程式、序列埠通訊、儲存資料等。NW.js建議使用版本：v0.41.3。

Blocklyduino F1開發板設定

開發板選單：BlocklyduinoF1\package.nw\board_F1.xml

```
<board upload="esp32:esp32:esp32wrover" name="ESP32 Wrover Module"></board>
```



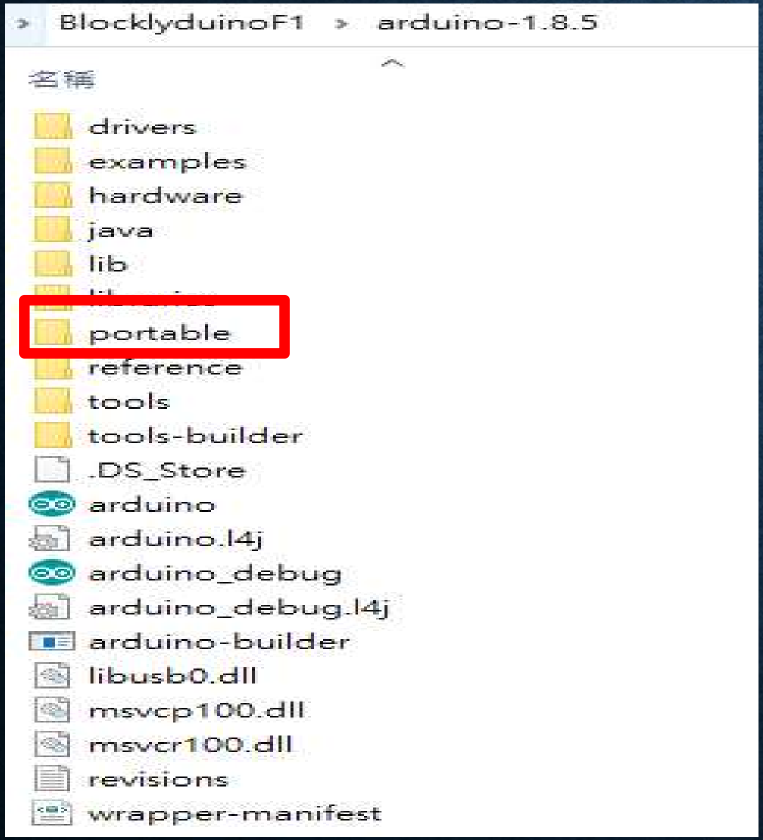
BlocklyduinoF1 › arduino-1.8.5 › portable › packages › esp32 › hardware › esp32 › 1.0.4
名稱
修改日期
類型
大小
cores
libraries
tools
variants
boards
platform
programmers
boards - 記事本
檔案(F) 編輯(E) 格式(O) 檢視(V) 說明

```
esp32.menu.DebugLevel.verbose=Verbose
esp32.menu.DebugLevel.verbose.build.code_debug=5

##############################################################

esp32wrover.name=ESP32 Wrover Module

esp32wrover.upload.tool=esptool_py
esp32wrover.upload.maximum_size=1310720
esp32wrover.upload.maximum_data_size=327680
esp32wrover.upload.wait_for_upload_port=true
```

初次使用新增的開發板，開啟內建**Arduino IDE**執行此開發板燒錄一次，以更新**boards.txt**預設燒錄設定。

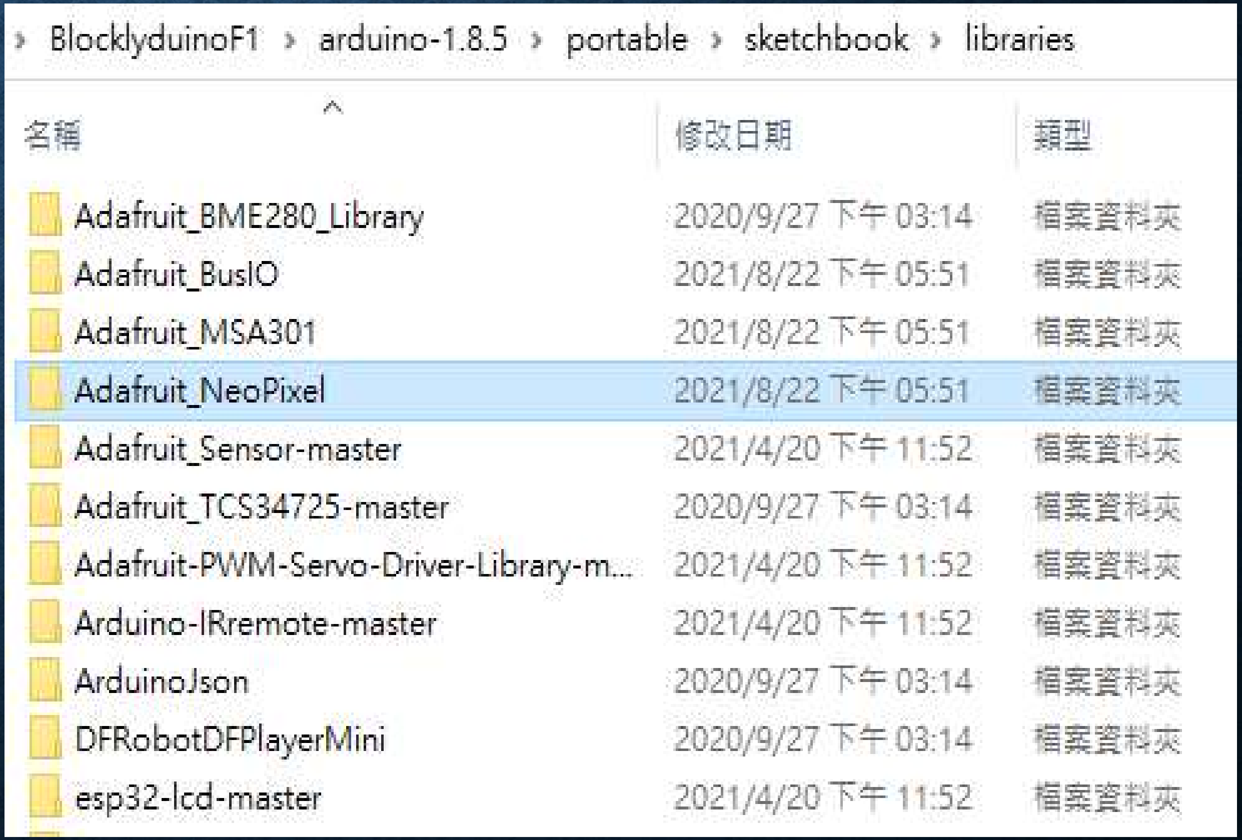
Arduino IDE可攜版設定

在Arduino IDE手動新增資料夾命名為portable，並執行Arduino IDE應用程式自動產生可攜版檔案環境，移機只需要複製portable資料夾。官方說明



› BlocklyduinoF1 › arduino-1.8.5

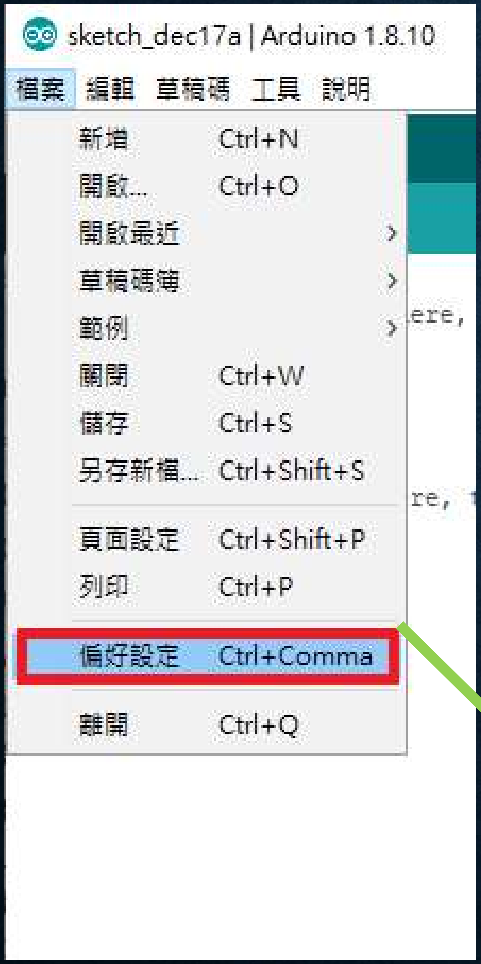
名稱
- drivers
- examples
- hardware
- java
- lib
- li[illegible]
- portable
- reference
- tools
- tools-builder
- .DS_Store
- arduino
- arduino.l4j
- arduino_debug
- arduino_debug.l4j
- arduino-builder
- libusb0.dll
- msvcp100.dll
- msvcr100.dll
- revisions
- wrapper-manifest

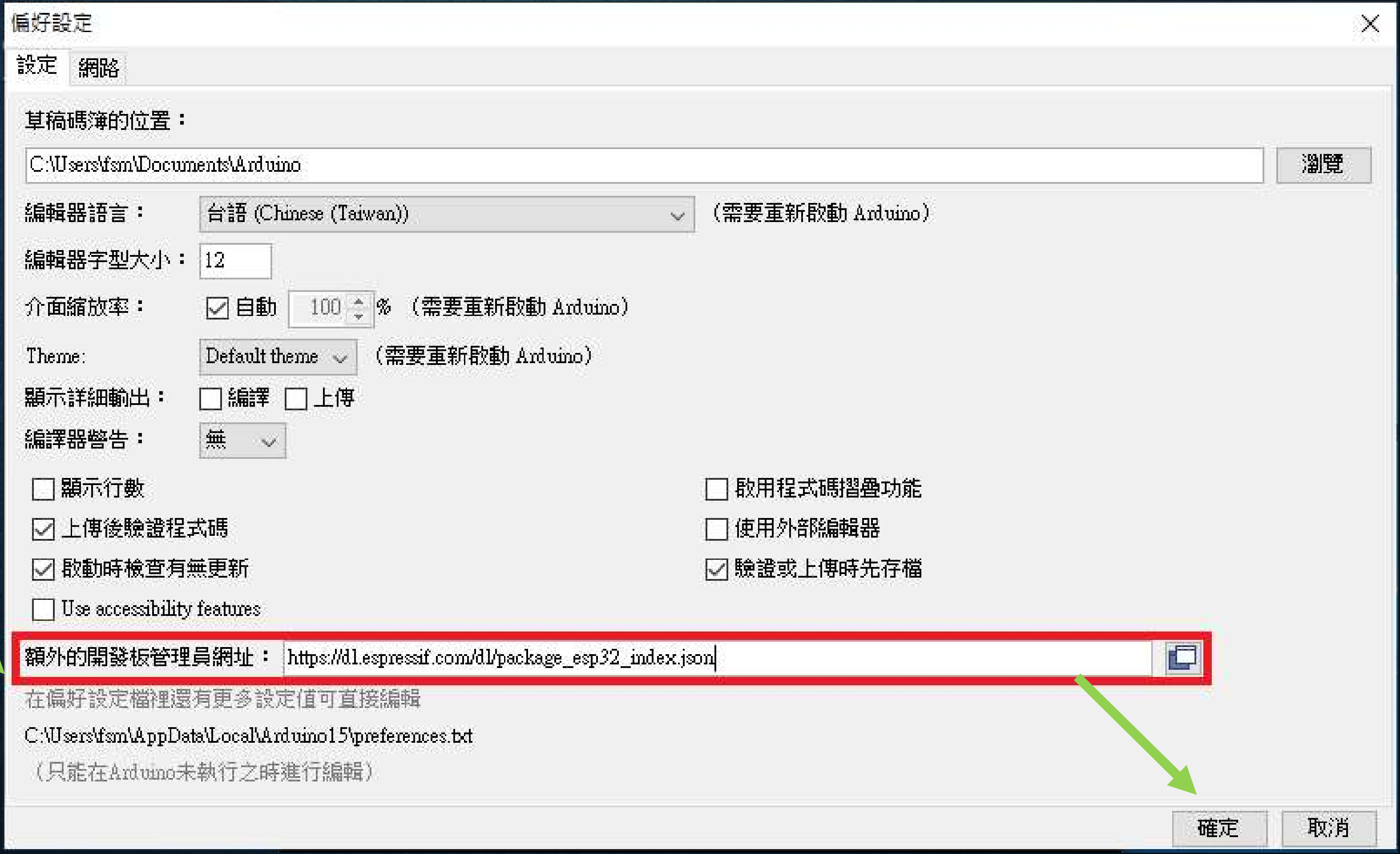


› BlocklyduinoF1 › arduino-1.8.5 › portable › sketchbook › libraries

名稱	修改日期	類型
Adafruit_BME280_Library	2020/9/27 下午 03:14	檔案資料夾
Adafruit_BusIO	2021/8/22 下午 05:51	檔案資料夾
Adafruit_MSA301	2021/8/22 下午 05:51	檔案資料夾
Adafruit_NeoPixel	2021/8/22 下午 05:51	檔案資料夾
Adafruit_Sensor-master	2021/4/20 下午 11:52	檔案資料夾
Adafruit_TCS34725-master	2020/9/27 下午 03:14	檔案資料夾
Adafruit-PWM-Servo-Driver-Library-m...	2021/4/20 下午 11:52	檔案資料夾
Arduino-IRremote-master	2021/4/20 下午 11:52	檔案資料夾
ArduinoJson	2020/9/27 下午 03:14	檔案資料夾
DFRobotDFPlayerMini	2020/9/27 下午 03:14	檔案資料夾
esp32-lcd-master	2021/4/20 下午 11:52	檔案資料夾

開發板網址： https://dl.espressif.com/dl/package_esp32_index.json





草稿碼簿的位置：C:\Users\fsm\Documents\Arduino

編輯器語言：台語 (Chinese (Taiwan)) （需要重新啟動 Arduino）

編輯器字型大小：12

介面縮放率：☑ 自動 100 % （需要重新啟動 Arduino）

Theme: Default theme （需要重新啟動 Arduino）

顯示詳細輸出：☐ 編譯 ☐ 上傳

編譯器警告：無

☐ 顯示行數 ☐ 啟用程式碼摺疊功能

☑ 上傳後驗證程式碼 ☐ 使用外部編輯器

☑ 啟動時檢查有無更新 ☑ 驗證或上傳時先存檔

☐ Use accessibility features

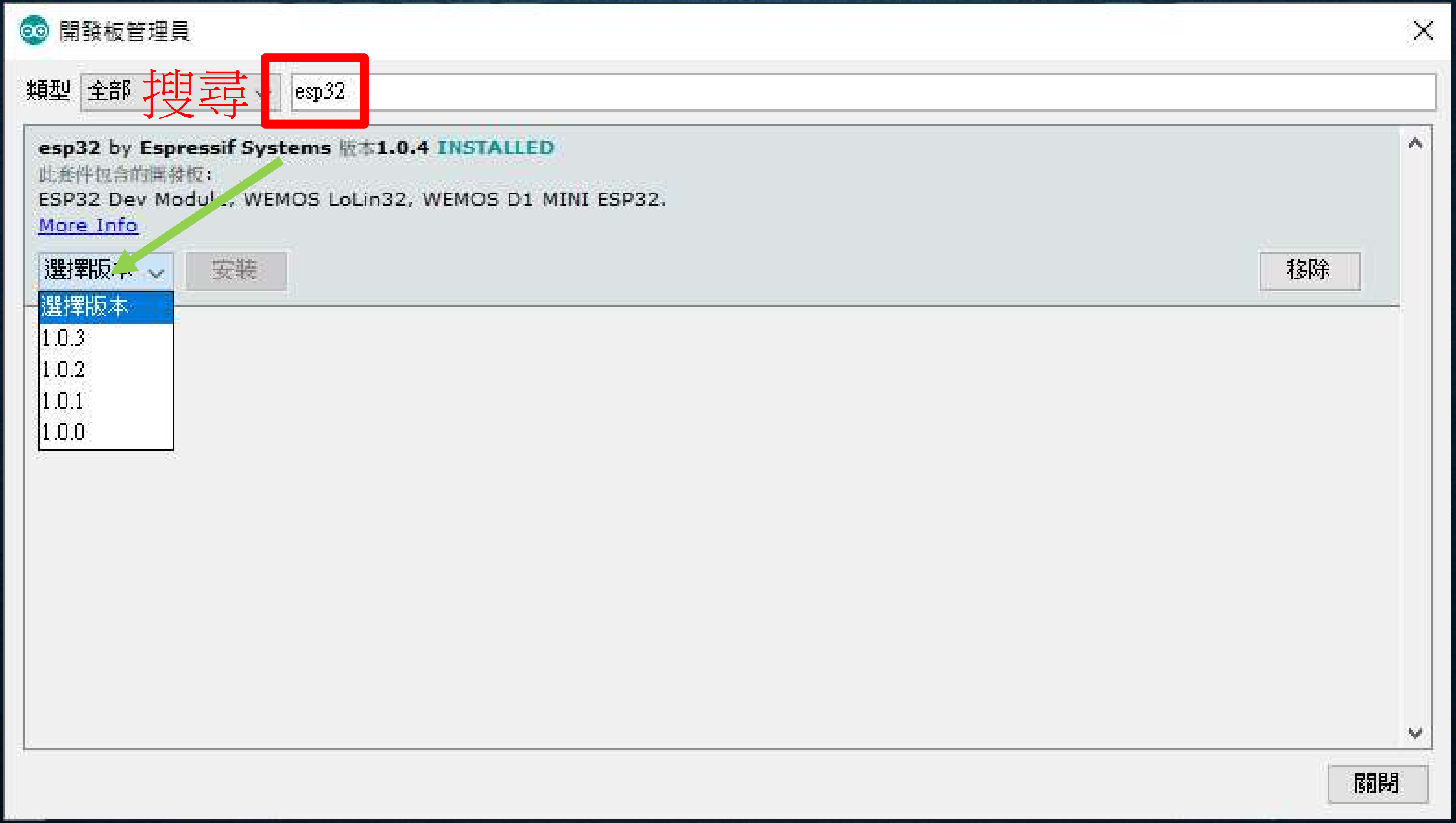
額外的開發板管理員網址：https://dl.espressif.com/dl/package_esp32_index.json

在偏好設定檔裡還有更多設定值可直接編輯

C:\Users\fsm\AppData\Local\Arduino15\preferences.txt

（只能在Arduino未執行之時進行編輯）

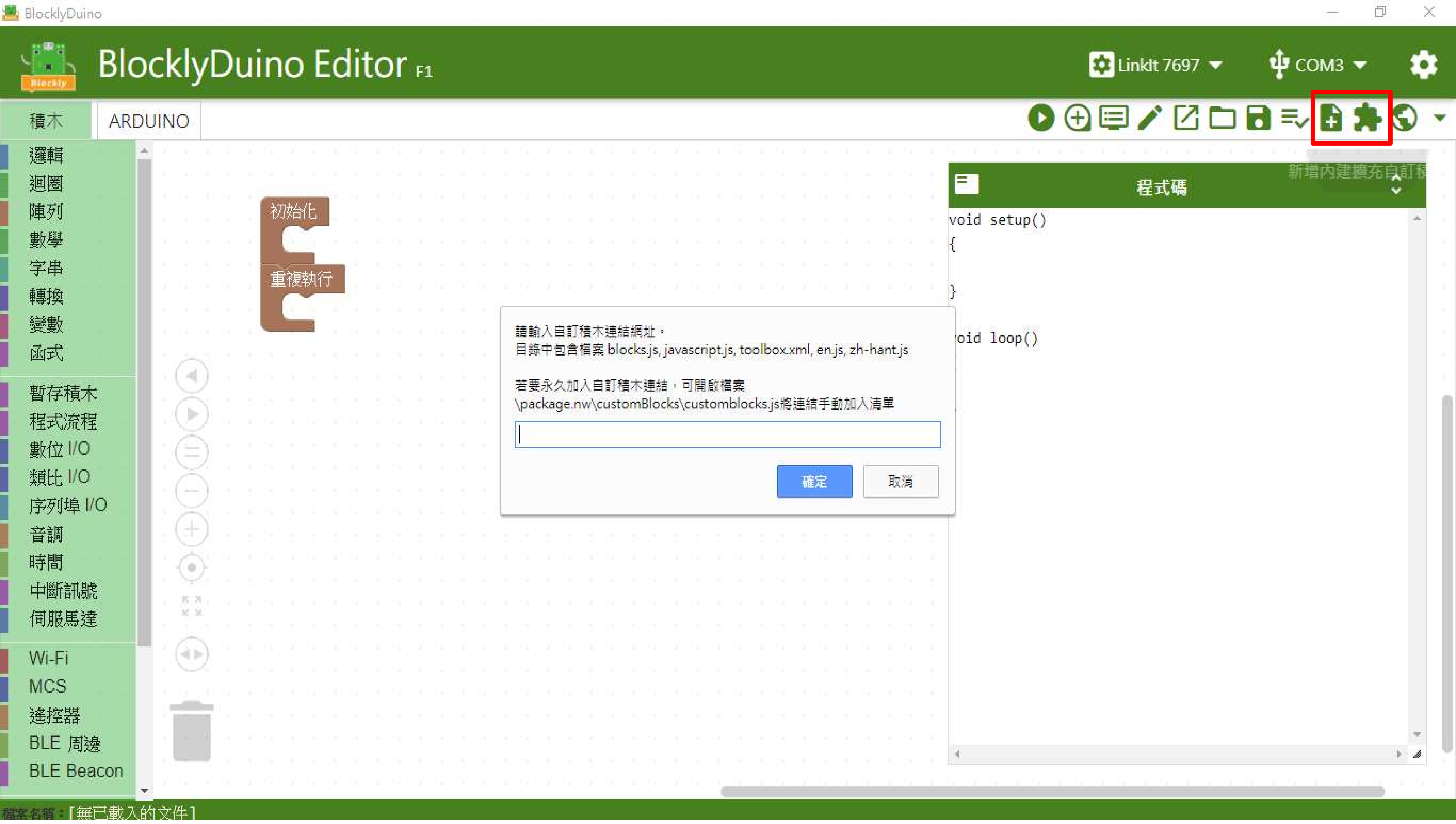
選擇安裝ESP32 SDK版本 1.0.4版



開發板管理員

類型 全部 搜尋 esp32

esp32 by Espressif Systems 版本1.0.4 INSTALLED
此套件包含的開發板:
ESP32 Dev Module, WEMOS LoLin32, WEMOS D1 MINI ESP32.
More Info



```
void setup()
{

}

void loop()
```

請輸入自訂積木連結網址。
目錄中包含檔案 blocks.js, javascript.js, toolbox.xml, en.js, zh-hant.js

若要永久加入自訂積木連結，可開啟檔案
\package.nw\customBlocks\customblocks.js將連結手動加入清單

自訂積木檔案掛載本機檔案

建立自訂積木資料夾。
BlocklyduinoF1\package.nw\customBlocks\自訂積木\
(內含自訂積木檔案blocks.js, javascript.js, toolbox.xml, en.js, zh-hant.js)

自訂積木連結設定檔
BlocklyduinoF1\package.nw\customBlocks\customblocks.js

```
var customBlocks = [
   ["https://circuspi.github.io/ICSHOP/","category_sep_custom"],
   ["customBlocks/自訂積木/","category_sep_custom"]
]
```

將自訂積木程式碼置入系統內建積木

積木定義 (blocks.js)
BlocklyduinoF1\package.nw\js\blocks_compressed.js
程式碼生成函式 (javascript.js)
BlocklyduinoF1\package.nw\js\arduino_compressed.js
工具箱目錄 (toolbox.xml)
BlocklyduinoF1\package.nw\category\category_F1.xml
英文語系變數 (en.js)
BlocklyduinoF1\package.nw\msg\en.js
繁體中文語系變數 (zh-hant.js)
BlocklyduinoF1\package.nw\msg\zh-hant.js