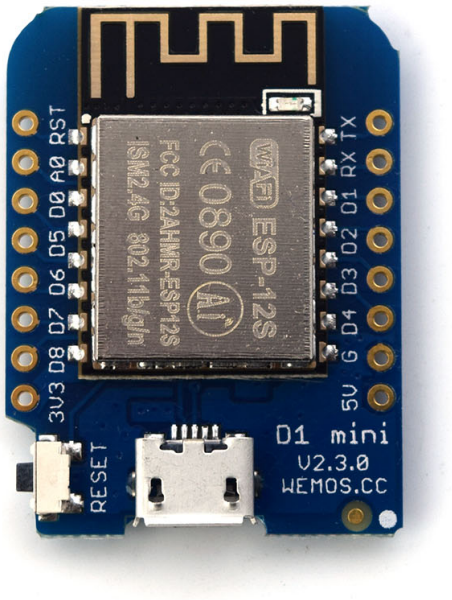
Arduino_測試



學習目的:

- 驗證D1 mini硬體是否沒問題
- 安裝Arduino IDE
- 安裝D1 mini晶片軟體
- Arduino IDE的設定
- 匯入Libraries
- 加入程式

安裝Arduino IDE

Arduino開發工具下載

Download the Arduino IDE

ARDUINO 1.8.5

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.
This software can be used with any Arduino board.
Refer to the Getting Started page for Installation instructions.

Windows Installer
Windows ZIP file for non admin install

Windows app Get

Mac OS X 10.7 Lion or newer

Linux 32 bits
Linux 64 bits
Linux ARM

Release Notes
Source Code
Checksums (sha512)

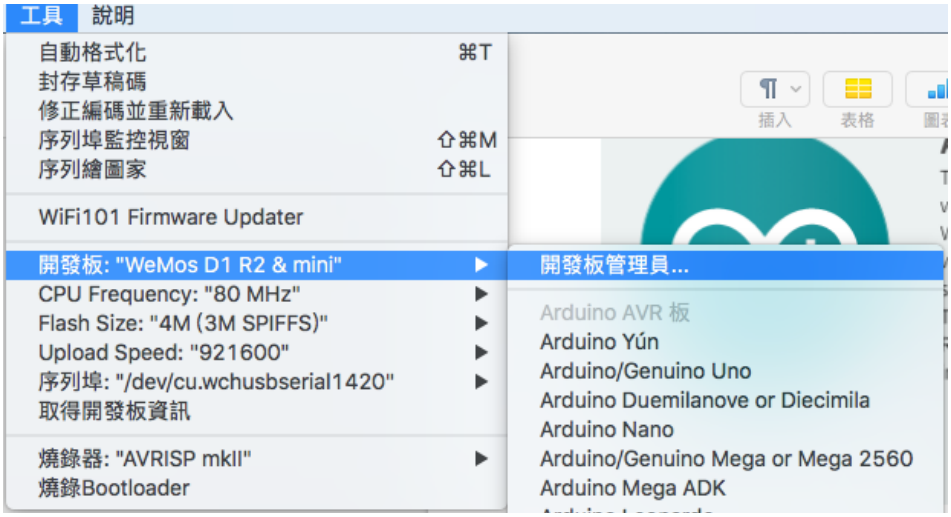
安裝D1 mini晶片軟體

在mac上執行 CH34x_Install_V1.3.pkg
Arduino IDE的設定

ESP 8266 套件安裝:

- 開啟 Arduino IDE > Preferences視窗
- 在 額外的板子管理員網址 輸入下面網址

- http://arduino.esp8266.com/stable/package_esp8266com_index.json



- 在工具Tools > 板子(Board menu) > 板子管理員安裝esp8266套件
  搜尋:esp8266關鍵字，並安裝。

- 至工具>開發板>選擇WeMos D1 R2 & mini
- 使用USB線D1 mini
- 至工具>序列埠>選擇適當的硬體(每一台都不大一樣)

匯入Libraries

請複製/libraries 到 文件/Arduino/libraries

加入程式

```
void setup() {
pinMode(LED_BUILTIN, OUTPUT); // D1 mini LED
}
void loop() {
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);
  digitalWrite(LED_BUILTIN, HIGH);
  delay(2000);
}
```

驗證D1 mini硬體是否沒問題

D1 mini的藍色led燈會一閃一閃

```
function logEvent(name, params) {
  if (!name) {
    return;
  }

  if (window.AnalyticsWebInterface) {
    // Call Android interface
    window.AnalyticsWebInterface.logEvent(name,
JSON.stringify(params));
  } else if (window.webkit
      && window.webkit.messageHandlers
      && window.webkit.messageHandlers.firebase) {
    // Call iOS interface
    var message = {
      command: 'logEvent',
      name: name,
      parameters: params
    };
    window.webkit.messageHandlers.firebase.postMessage(message);
  } else {
    // No Android or iOS interface found
    console.log("No native APIs found.");
  }
}
```

```
function setUserProperty(name, value) {
  if (!name || !value) {
    return;
  }

  if (window.AnalyticsWebInterface) {
    // Call Android interface
    window.AnalyticsWebInterface.setUserProperty(name,
value);
  } else if (window.webkit
      && window.webkit.messageHandlers
      && window.webkit.messageHandlers.firebase) {
    // Call iOS interface
    var message = {
      command: 'setUserProperty',
      name: name,
      value: value
   };

window.webkit.messageHandlers.firebase.postMessage(messa
ge);
  } else {
    // No Android or iOS interface found
    console.log("No native APIs found.");
  }
}
```

```
func userContentController(_ userContentController:
WKUserContentController,
                           didReceive message:
WKScriptMessage) {
  guard let body = message.body as? [String: Any] else
{ return }
  guard let command = body["command"] as? String else
{ return }
  guard let name = body["name"] as? String else
{ return }

  if command == "setUserProperty" {
    guard let value = body["value"] as? String else
{ return }
    FIRAnalytics.setUserPropertyString(value, forName:
name)
  } else if command == "logEvent" {
    guard let params = body["parameters"] as? [String:
NSObject] else { return }
    FIRAnalytics.logEvent(withName: name, parameters:
params)
  }
}
```

ViewController.swift