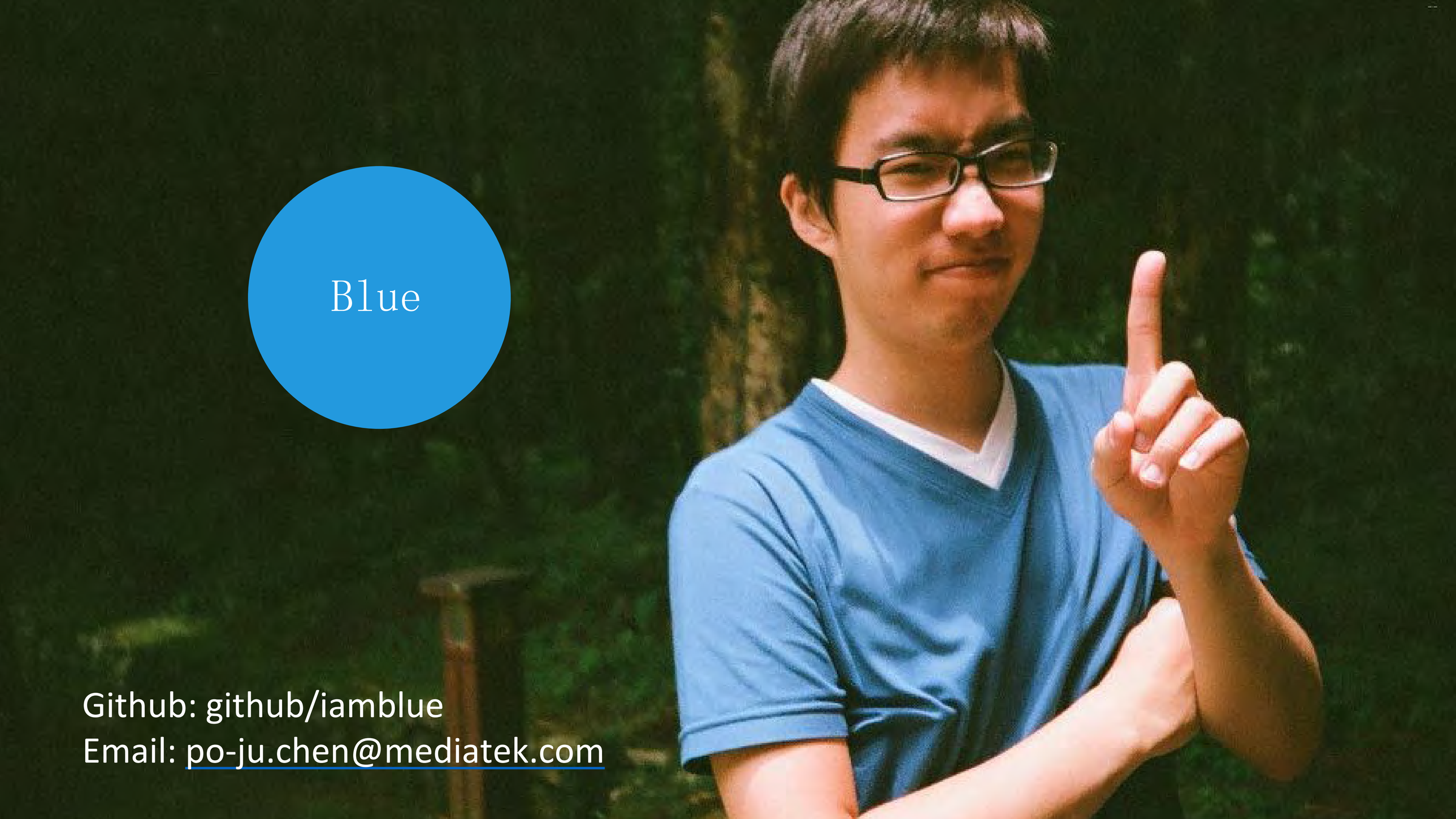
WOT
World Of Tech

2016年8月26日-27日　北京珠三角JW万豪酒店

World Of Tech 2016
移动互联网技术峰会
THE BEST MOBILE TECH IN HERE

主办：51CTO

MCU 與 Lightweight Javascript Engine 的邂逅



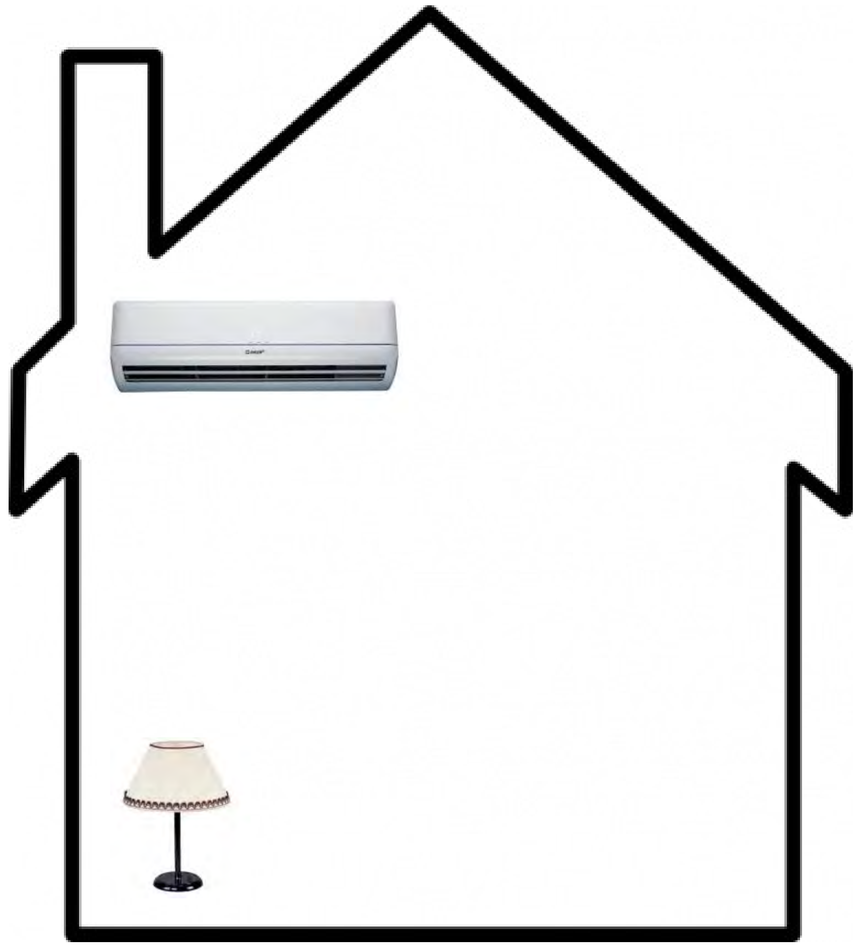
Github: github/iamblue
Email: po-ju.chen@mediatek.com

JavaScript is well-suited to embedded device programming

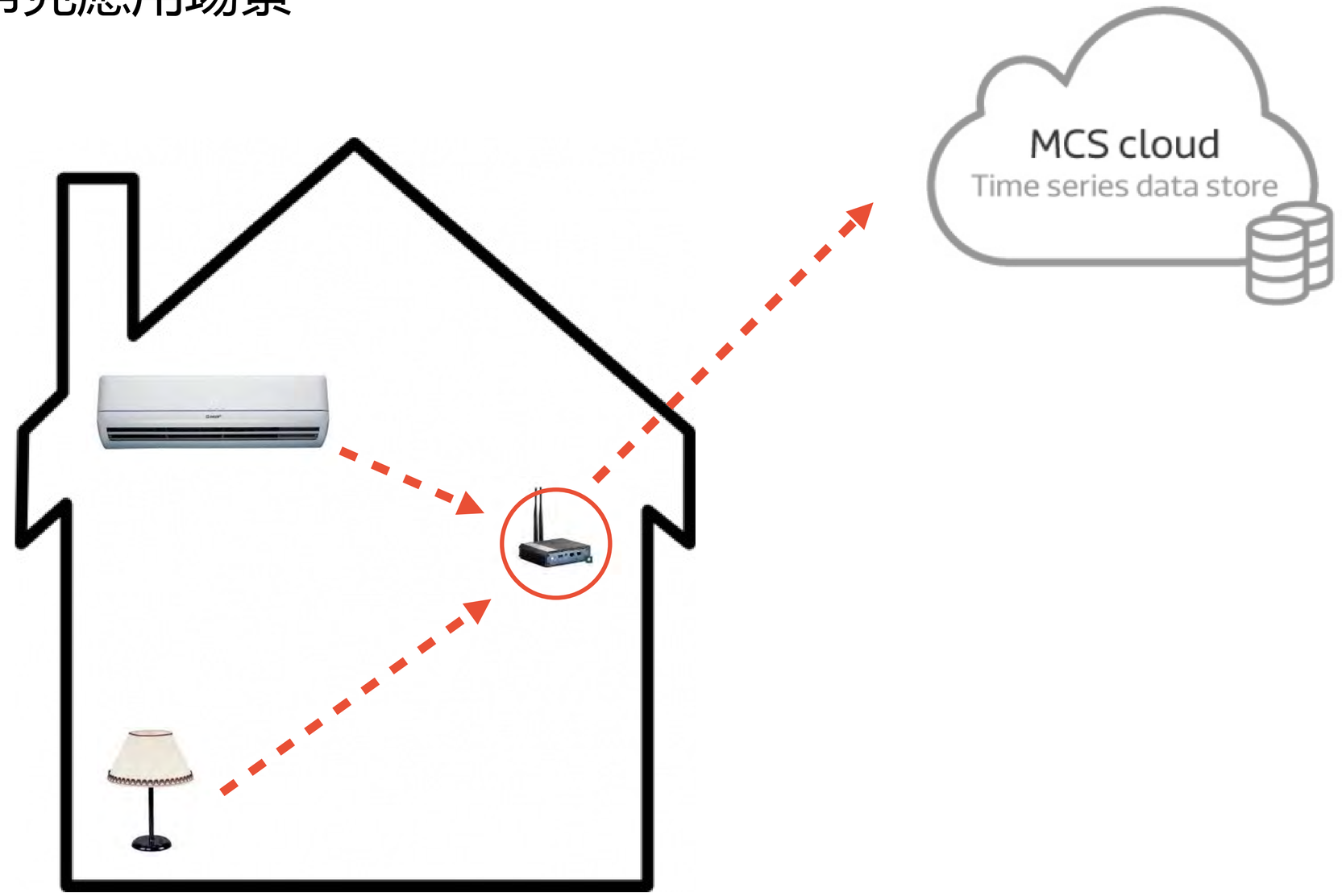
Javascript 語言特性

1.事件驅動
2.Non Blocking / Async → 特別是 IoT device 寫 C 常遇到的 Internet request Blocking
3.JSON parser → 不需要再花而外力氣去 parse JSON
4.弱型別語言 → 不需負擔太多學習成本
5.Node package management → 龐大的 community

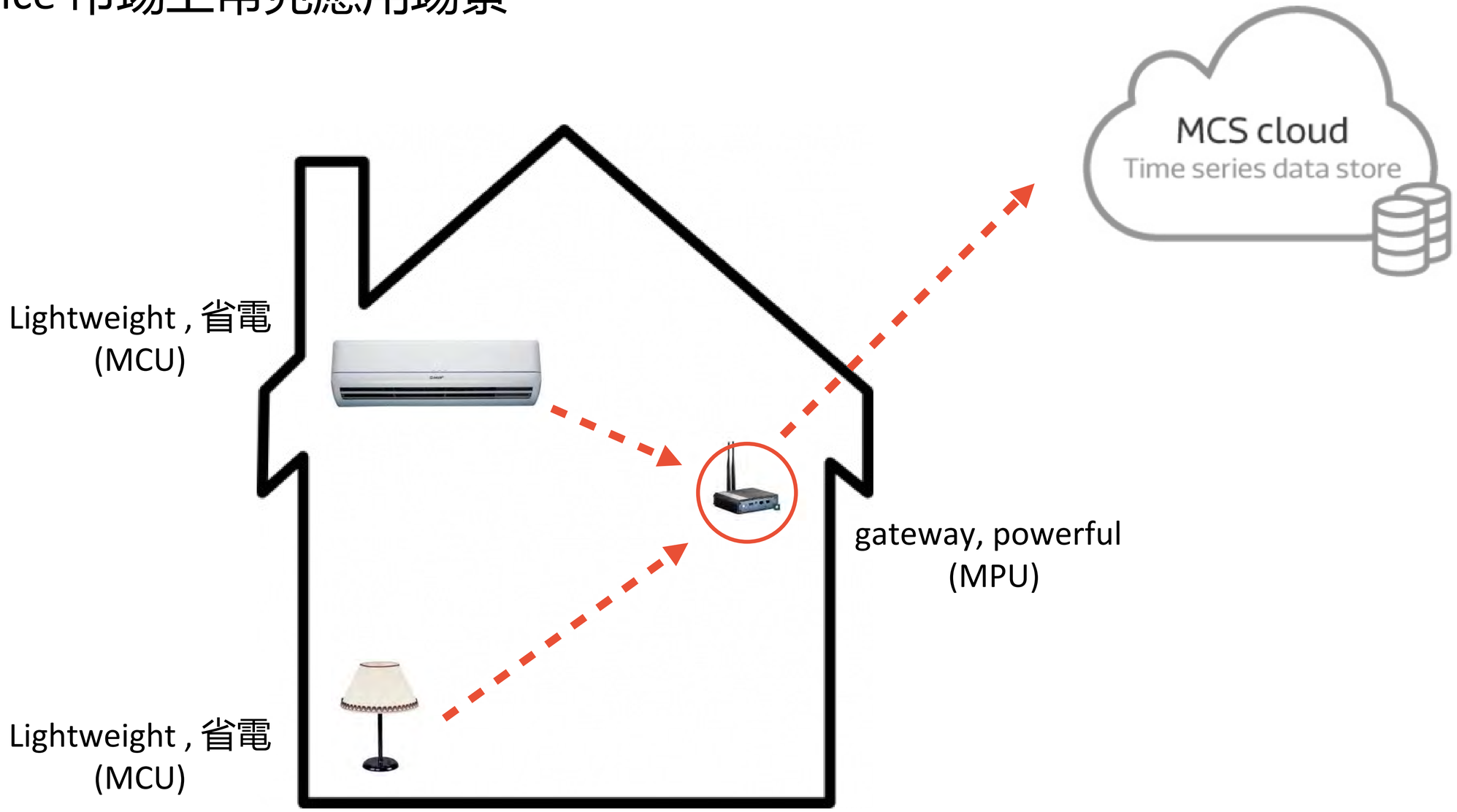
IoT device 市場上常見應用場景



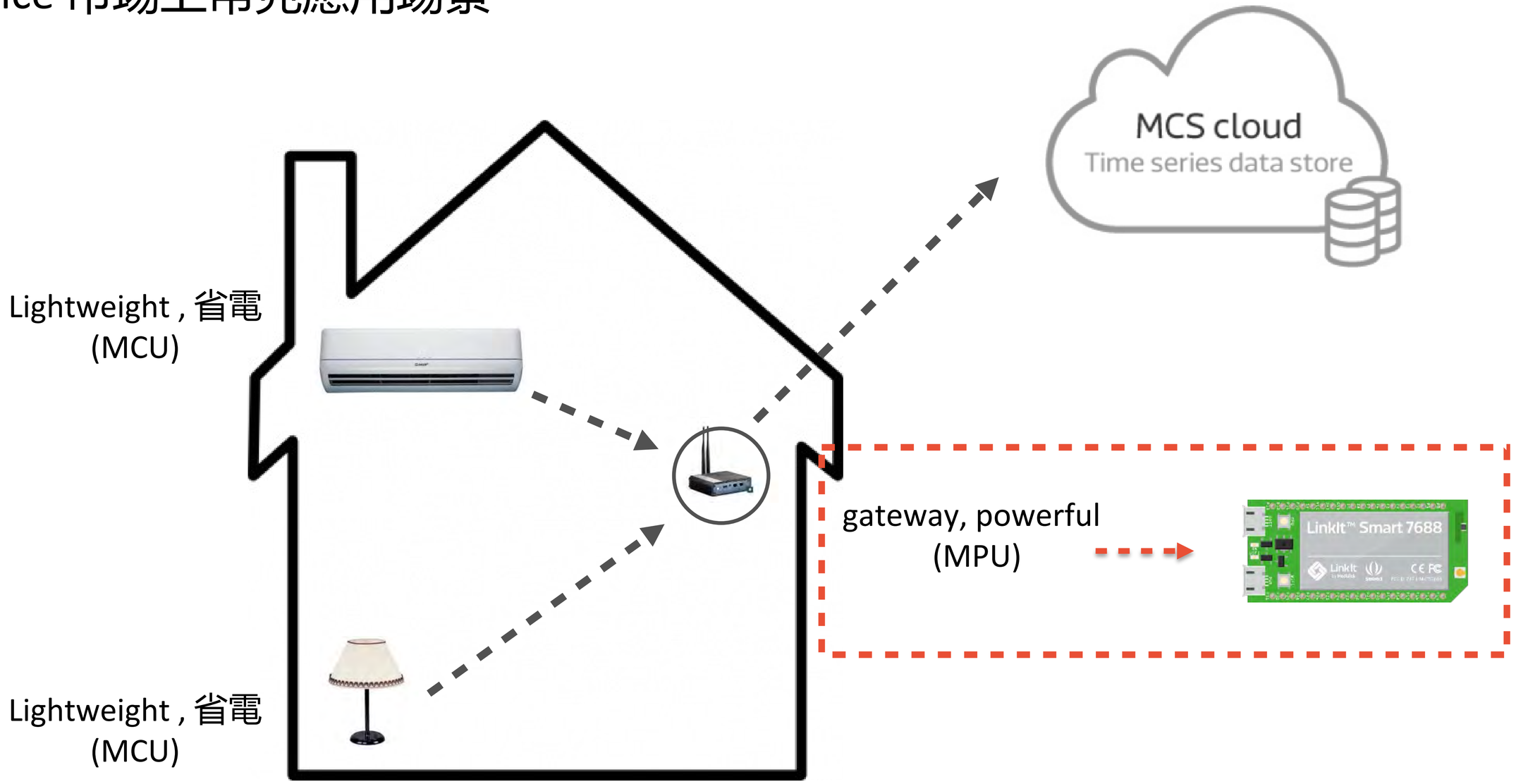
IoT device 市場上常見應用場景



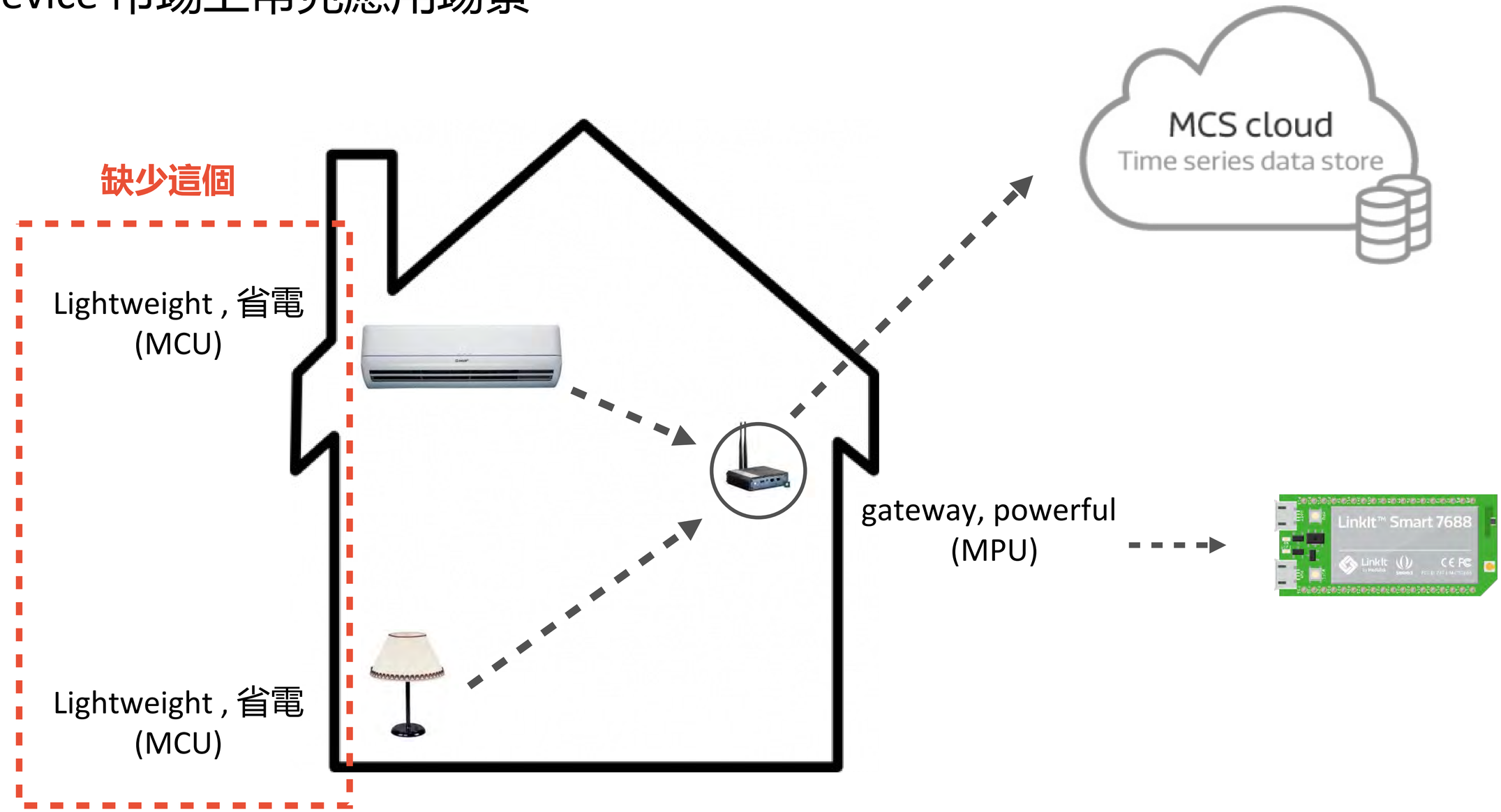
IoT device 市場上常見應用場景



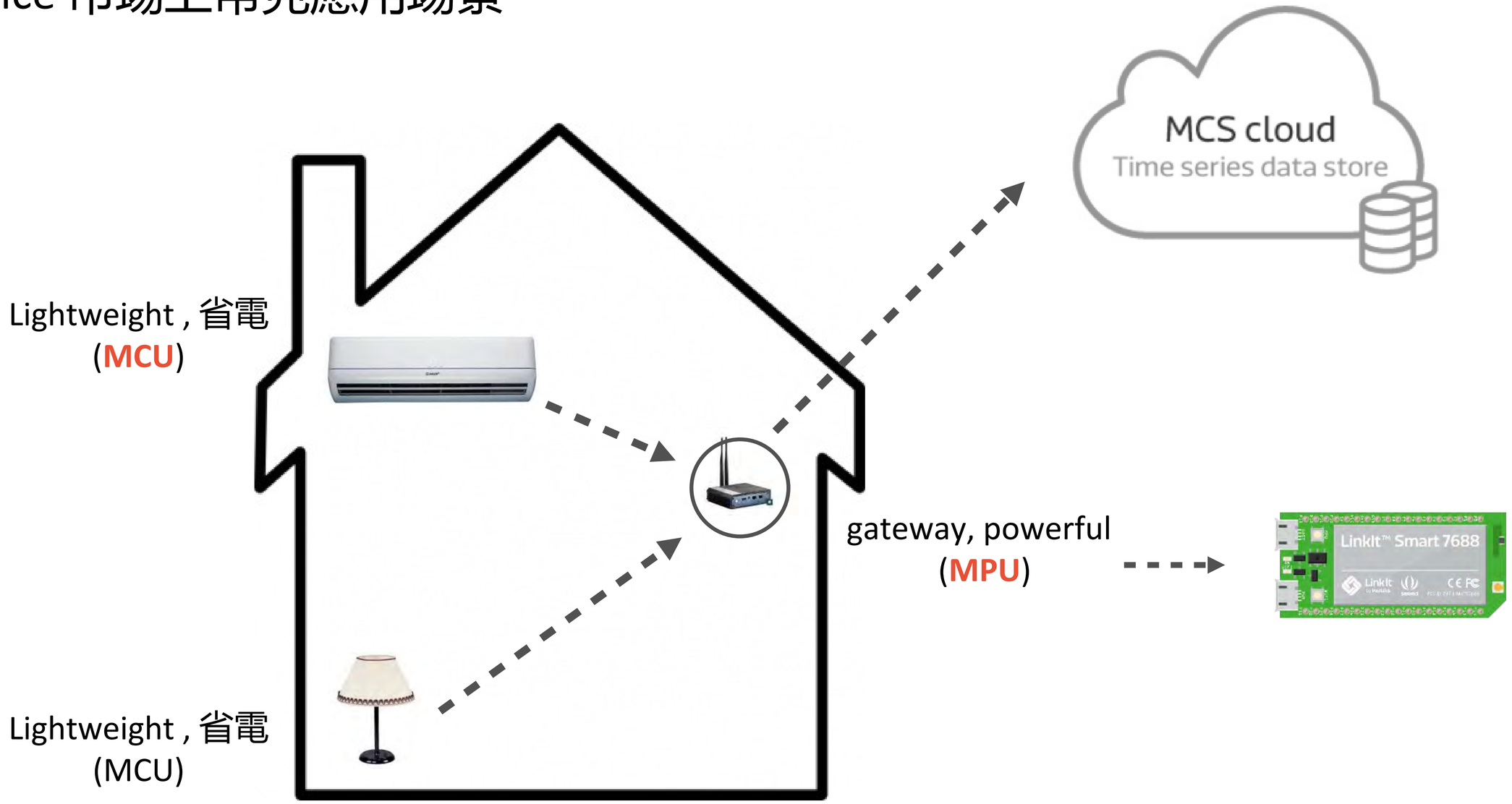
IoT device 市場上常見應用場景



IoT device 市場上常見應用場景

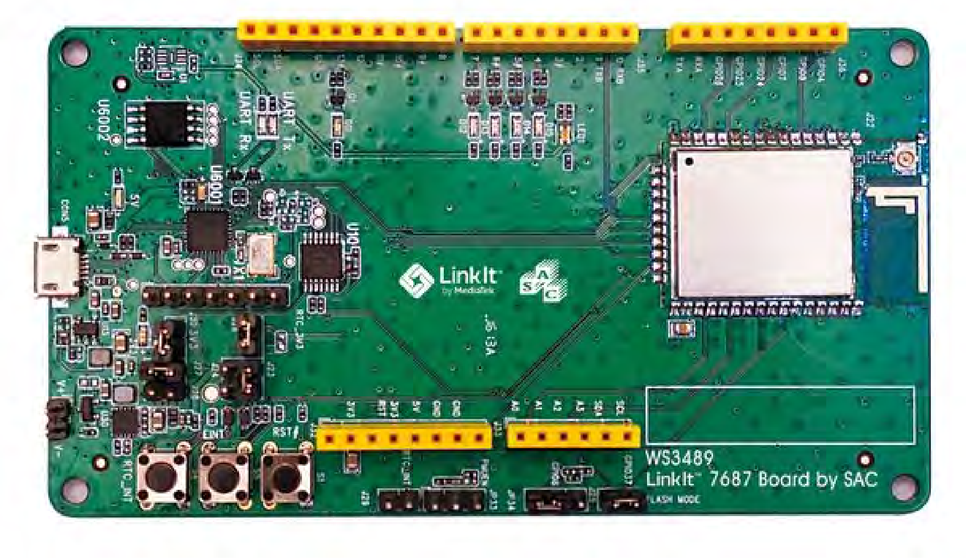


IoT device 市場上常見應用場景



兩者體驗上的差異在哪？

2016/4 release MT7687



- **ARM Cortex-M4F MCU (192MHz)**.
- 1x1 802.11b/g/n Wi-Fi .
- **Real-Time OS (freeRTOS)**
- Versatile peripheral connectivity, including UART, I2C, SPI, I2S, PWM, IrDA and auxiliary ADC.
- **256KB RAM**
- Embedded SRAM/ROM and **2MB serial flash** in package.
- Integrated security engine (AES and 3DES/SHA).
- 8 x 8mm 68-pin QFN package.

http://ec.wpgo2o.com/getAdDetailPage?AdId=46&Catg=C&ItemId=3086999&ParentId=45

Porting Javascript engine on MCU device 非常新奇

但 .... 困難重重

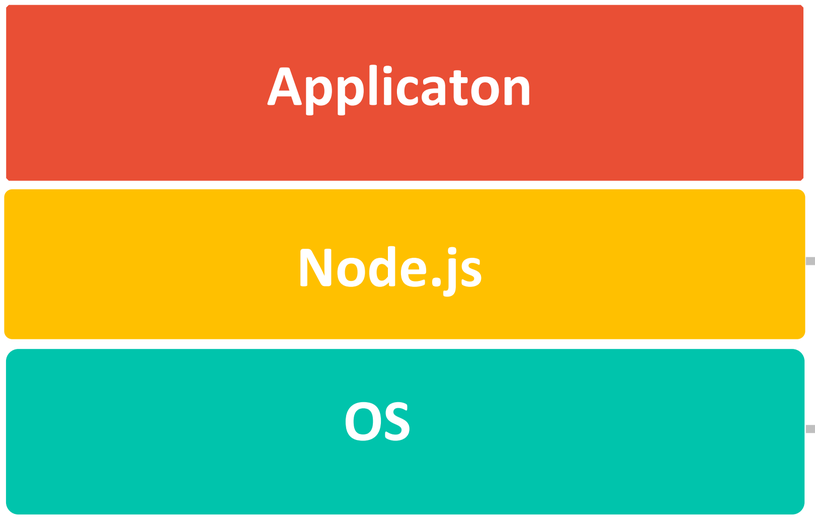
Node.js 現況問題

以實際使用面來看 (以當初 porting LinkIt smart 7688 為例)

先從一般基本跑得動 Node.js 情境來談

想像一下各位現有的 server 如何跑 Node.js?

架構



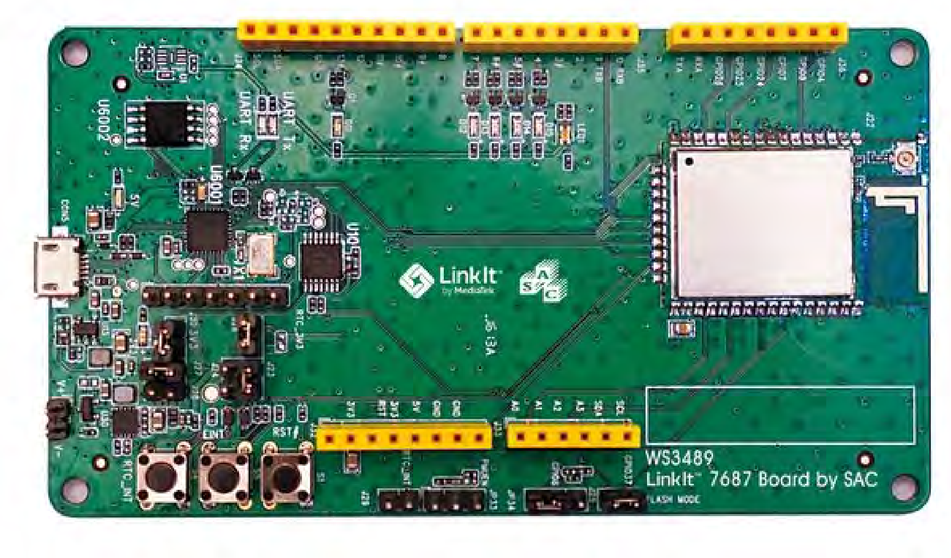
規格

硬碟至少要 1X mb, RAM 建議至少要 64 mb

Linux, openWRT...etc

Reference: Node.js 與 LinkIt smart 7688 的邂逅

2016/4 release MT7687



因此在這應硬體需求下，需要去設計更 Lightweight 的 javascript engine

- **ARM Cortex-M4F MCU (192MHz).**
- 1x1 802.11b/g/n Wi-Fi
- **Real-Time OS (freeRTOS)**
- Versatile peripheral connectivity, including UART, I2C, SPI, I2S, PWM, IrDA and auxiliary ADC.
- **256KB RAM**
- Embedded SRAM/ROM and **2MB serial flash** in package.
- Integrated security engine (AES and 3DES/SHA).
- 8 x 8mm 68-pin QFN package.

http://ec.wpgo2o.com/getAdDetailPage?AdId=46&Catg=C&ItemId=3086999&ParentId=45

Lightweight Javascript engine 技術趨勢

Community 最早盛行的 Javascript Engine: Tiny.js

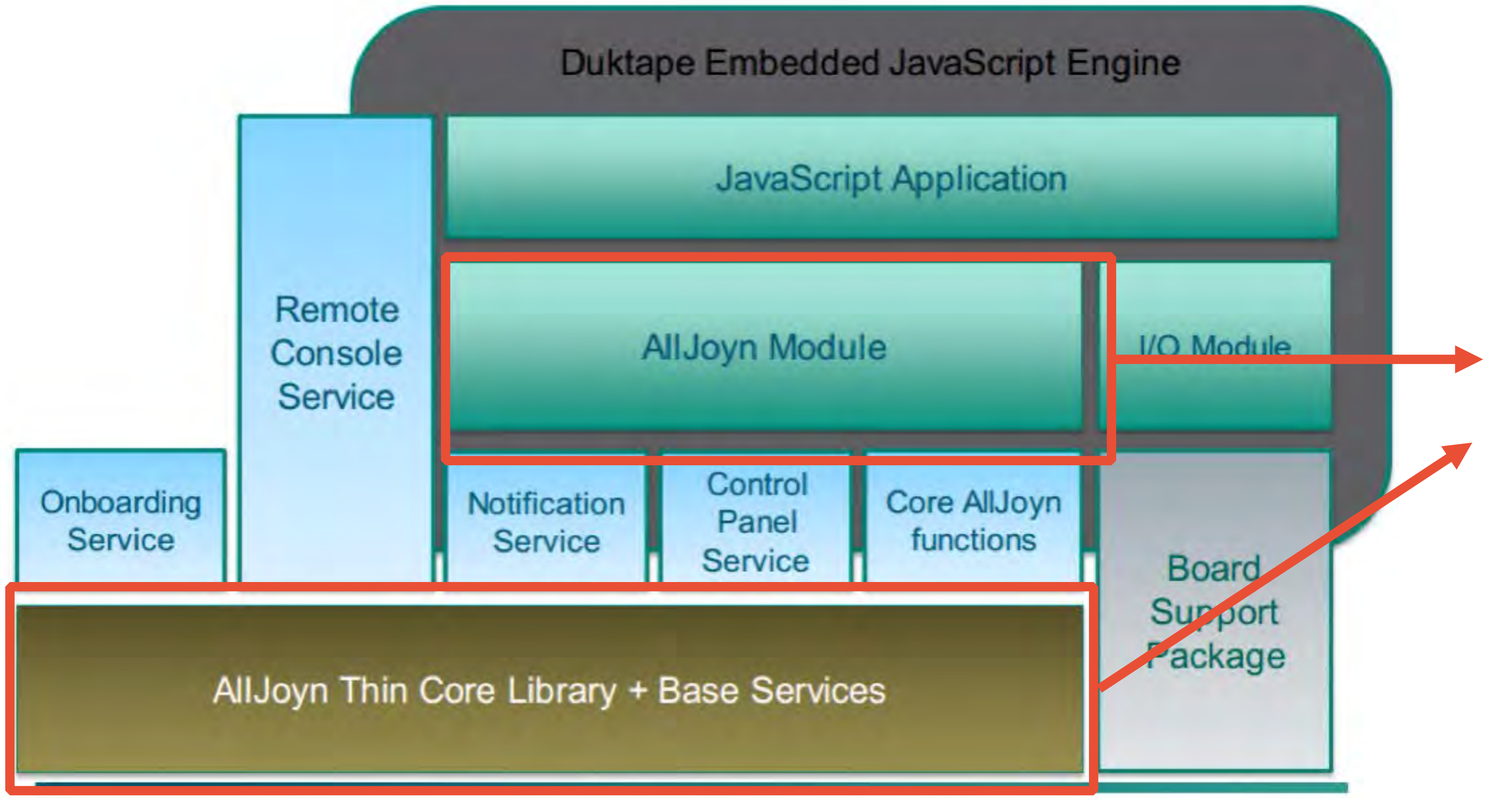
- First Run on STM32 javascript engine
- Extremely Simple (~2000 line) javascript interpreter
- Github: https://github.com/gfwilliams/tiny-js
- 原理: 邊 parse 邊解析（不構成 AST） https://www.zhihu.com/question/36674924

基於 Tiny.js 做出來的產品 Espruino

2012 年各家晶片大廠研究 porting Lightweight Javascript engine on MCU 議題

- 高通 (Allseen 聯盟): alljoyn.js (duktape) → 贊助開發
- Samsung (OIC 聯盟): IoT.js (jerryscript) → 自主研發
- ESP: Espruino, V7 → 社群贊助，但 ESP 8266 由於硬體資源不足，目前大家習慣是 lua 開發

Alljoyn.js (Duktape engine)



Alljoyn (AllSeen 聯盟)
在 module 佈局廣主打這塊

Source code : https://git.allseenalliance.org/cgit/core/alljoyn-js.git/
Reference: https://wiki.allseenalliance.org/_media/training/programming_alljoyn.js.pdf

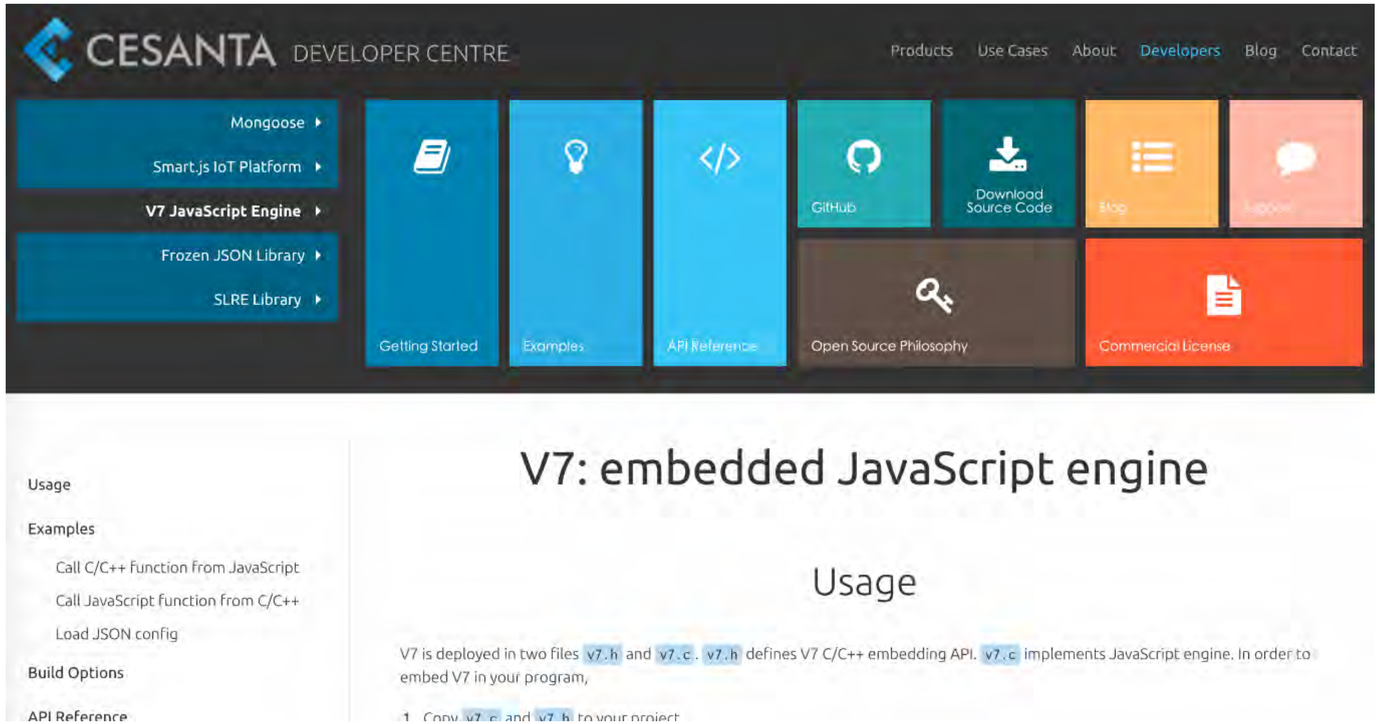
Duktape Javascript engine

- The first full support ES5 javascript engine
- Run time memory suggestion: 96 KB
- Flash space suggestion: 256KB
- 提供各種 RAM or ROM build 機制選項
- 把 build , debugger 機制給考慮進 engine 之中
- 較好的跨平台 Build 機制

除此之外

- 當時時空背景：wifi + MCU 市場尚未起來
- AllSeen 聯盟主推較 powerful 的 MCU
- 因此較低的 RAM/ROM 不是 Duktape engine 設計初衷

Smart.js (v7 engine)

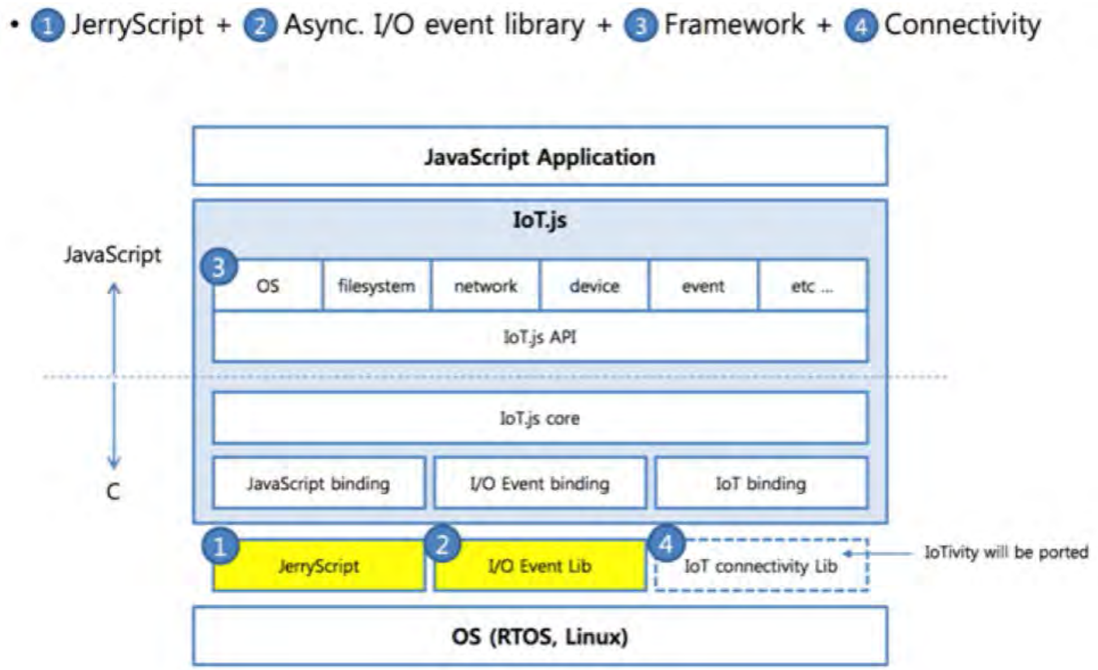


- support ES5 Javascript engine
- 專門給較小 RAM/ROM device 設計
- 轉 bytecode 部分是針對 AST 優化

(效能不錯，但耗的 RAM 較高)

Reference: Smart.js 官方網站

Samsung IoT.js (Jerryscript engine)

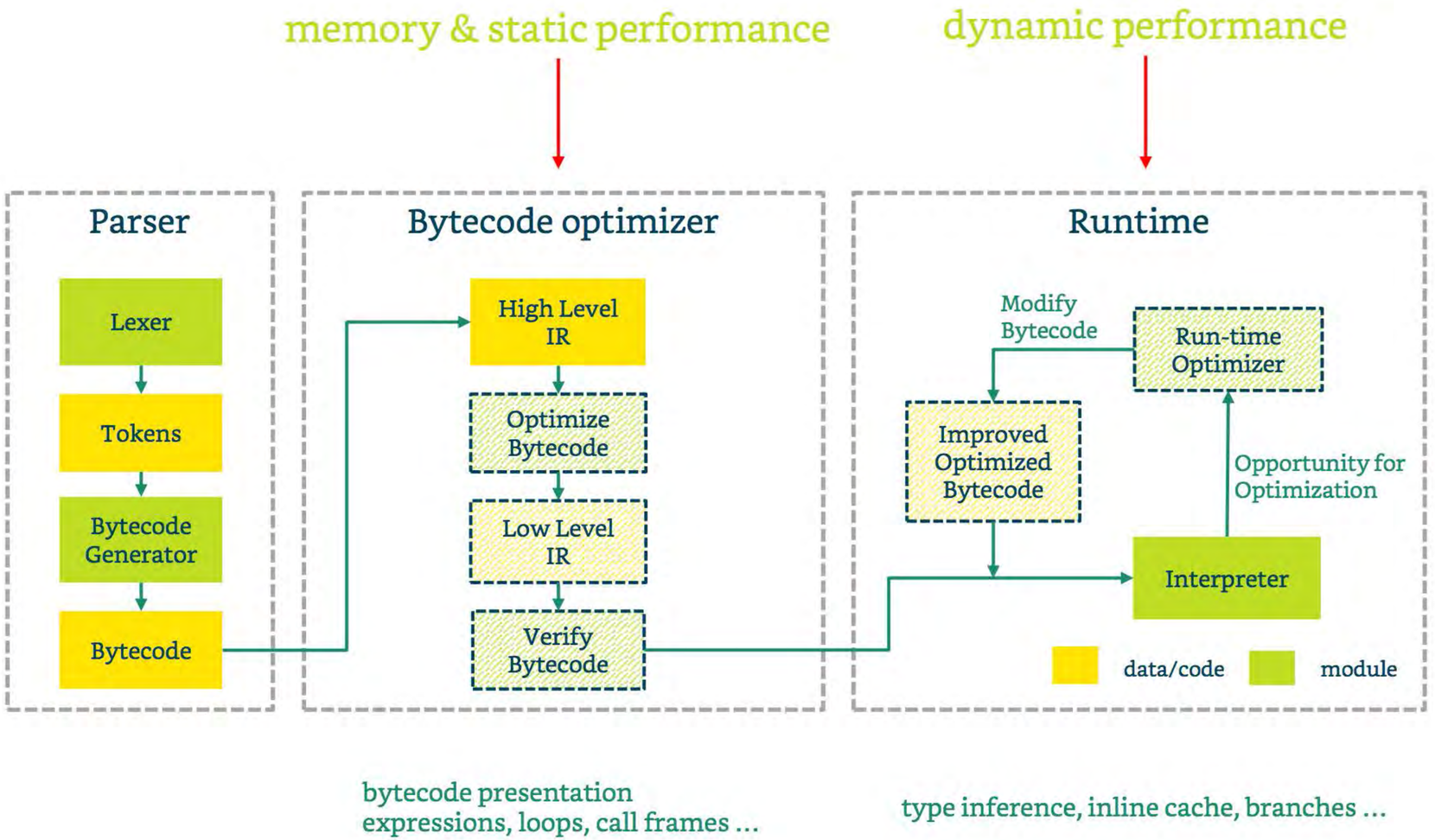


- 設計初期目標：針對 <100KB 之 RAM/ROM chip 設計的 engine
- 大幅針對 interpreter js engine 常見的通病：String (Duktape 最弱的地方), bytecode, memory 效能改進
- No AST, directly produce bytecode

Segment	Memory Footprint	Binary Size	Configuration	
Pico	8KB~16KB	~100KB	Compact JerryScript + Subset Profile	`2014
Nano	16KB~64KB	~200KB	JerryScript + Full Profile (ES v5.1)	`2015
Micro	64KB~256KB		JerryScript + Bytecode Optimization	
Light	256KB~	~300KB	JerryScript with JIT	`2016
Full	8MB~	10MB	V8	

Reference: Jerryscript 官方網站

Jerryscript Architecture



Reference: Jerryscript 官方網站

Jerryscript Architecture (for Example : String parser優化)

'String' orbject representation in JerryScript

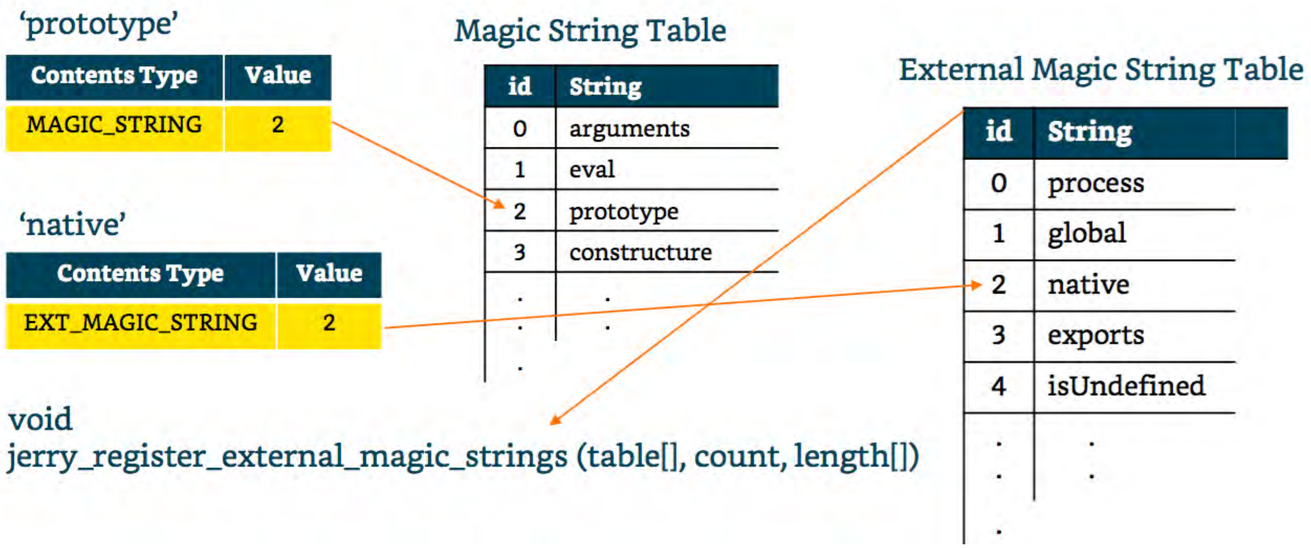
Container Type	15bits	15bits	2bits	Comment
LIT_TABLE	Literal index			Index of 'String' in Literal Table
HEAP_CHUNKS	Pointer to collection			'String' allocated as multiple chunks in heap memory
HEAP_NUMBER	Pointer to number			'String' stored as numeric representation of number object in heap. Converted to 'String' when it actually used
UINT32_IN_DESC	uint32_t value			'String' stored as numeric representation of 32bit integer. Immediate value. Converted to String when it actually used
CONCATENATION	Pointer to string1	Pointer to string2		2 concatenated String
MAGIC_STRING	Magic_string_id			Index of 'String' in Magic String Table
EXT_MAGIC_STRING	External Magic string id			Index of 'String' in External Magic String Table

Reference: Jerryscript 官方網站

Jerryscript Architecture (for Example : String parser優化)

External magic string

- Building pre-allocated string table for frequently used string literals in JavaScript source code
  - Reducing heap allocation for string object
  - Effective to pre-loaded large size framework code, such as built-in module of IoT.js



Reference: Jerryscript 官方網站

實作 benchmark

	LICENSE	ECMA 5 (leverage node.js community)	Native binding api	RAM (Run time memory usage)	ROM (binary size)	CM4 support
Jerryscript	Apache2	y (100%)	y	10~70KB	<100KB	y
v7	GPL	y (80%)	n	30~100KB	40-200KB	n (CM3)
Duktape	MIT	y(100%)	y	~96KB	200~256K	n

實作 benchmark (先比較 ECMA 與 LICENSE)

	LICENSE	ECMA 5 (leverage node.js community)	Native binding api	RAM (Run time memory usage)	ROM (binary size)	CM4 support
Jerryscript	Apache2	y (100%)	y	30~70KB	<100KB	y
v7	GPL	y (80%)	n	50~120KB	40-200KB	n (CM3)
Duktape	MIT	y(100%)	y	~96KB	200~256K	n

實作 benchmark (比較 RAM 與 ROM)

關鍵

	LICENSE	ECMA 5 (leverage node.js community)	Native binding api	RAM (Run time memory usage)	ROM (binary size)	CM4 support
Jerryscript	Apache2	y (100%)	y	30~70KB	<100KB	y
v7	GPL	y (80%)	n	30~100KB	40-200KB	n (CM3)
Duktape	MIT	y(100%)	y	~96KB	200~256K	n

光有 Javascript engine

這樣還不夠...

這樣還不夠 ….

- MCU build tool 環境各家不一致
- Debug/ Download code 各家不一樣
- LICENSE 問題：大部份廠商 SDK 皆有 dis-contribute 規範
- 沒有共同的 sharing project 標準
- 現成的 Js total solution 不夠輕量，不符合效能接近 C 的需求
- freeRTOS 設計，沒有 filesystem 設計

這樣還不夠 ….

- MCU build tool 環境各家不一致
- Debug/ Download code 各家不一樣
- LICENSE 問題：大部份廠商 SDK 皆有 dis-contribute 規範
- 沒有共同的 sharing project 標準
- **現成的 Js total solution 不夠輕量，不符合效能接近 C 的需求**
- freeRTOS 設計，沒有 filesystem 設計

現成的 Js total solution 不夠輕量，不符合效能接近 C 的需求

市場上同級的 Wifi MCU 競品規格

- RAM: 128~512KB
- flash: 2mb 居多，不然就是可額外掛載
- OS: 各家分佈不一，但大多數皆有 support freeRTOS

Samsung IoT.js 的問題

- 致力為成為 IoT 版的 Node.js ，Native API 與 Node.js 一致

Image 肥大介在 4 mb 之間

- 因為要符合跨 OS 需求，因此從新設計 libuv

有些 api RTOS 本身就已具備，且效能也比較快

Microlattice.js

Github: https://github.com/iamblue/microlattice

Gitbook: https://www.gitbook.com/book/iamblue/microlattice-js-for-linkit-rtos/details/zh-TW

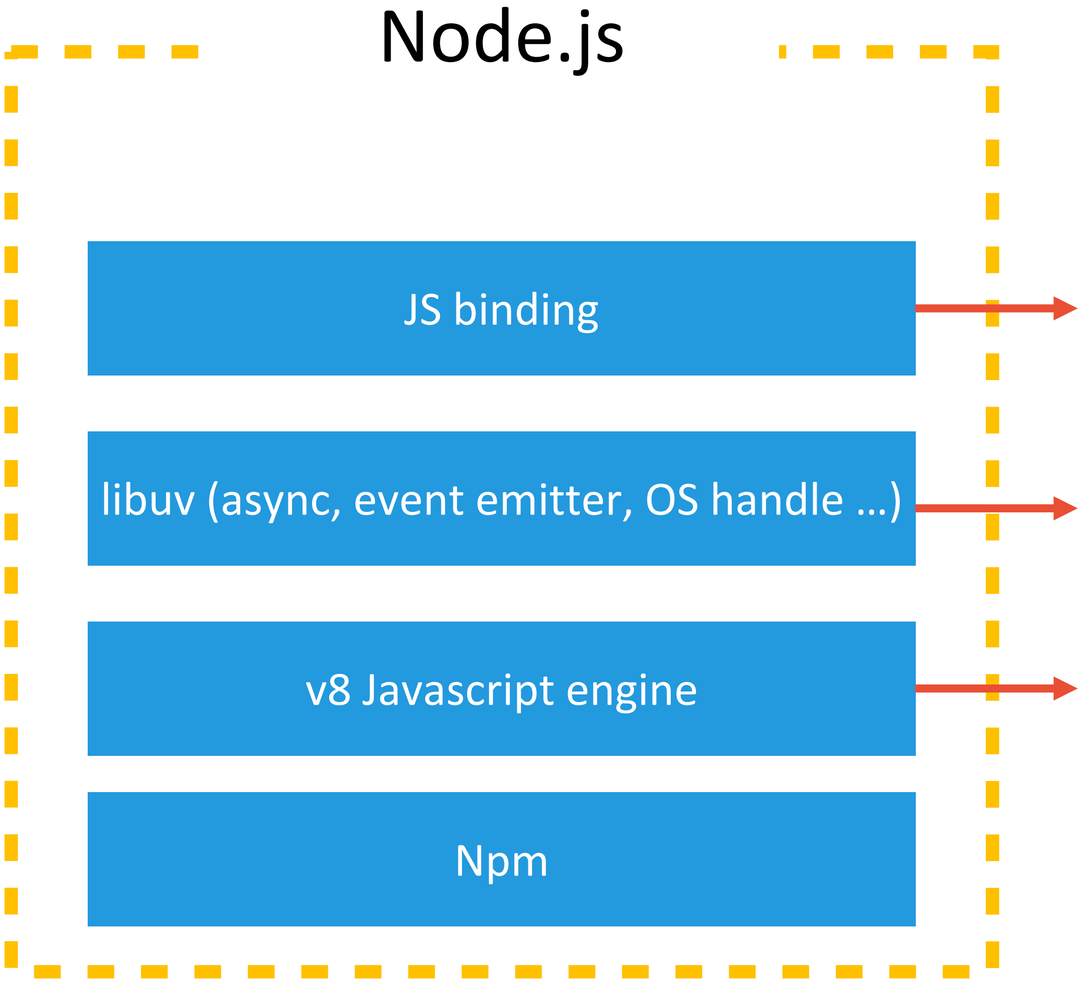
Microlattice.js 設計精神

- 中文意思：微晶格
- 英文意思：世界上最小最輕且非常堅固的金屬材料

設計理念

- 真正為 IoT device 所設計的 IoT 版的 Node.js.
- 針對 Javascript community 所熟悉的 coding style 追求接近 C 的效能
- 每一個細節包含 tool, module, engine core 都是可以拆分重組

與 Node. js 架構相較

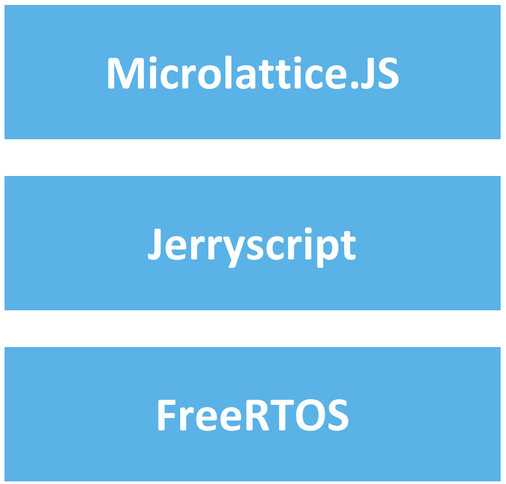


Node.js 提供了接近 20 種 api , 但是 Microlattice 設計 parser 機制每一種 Native API 都要用到才會被 compile 進 image

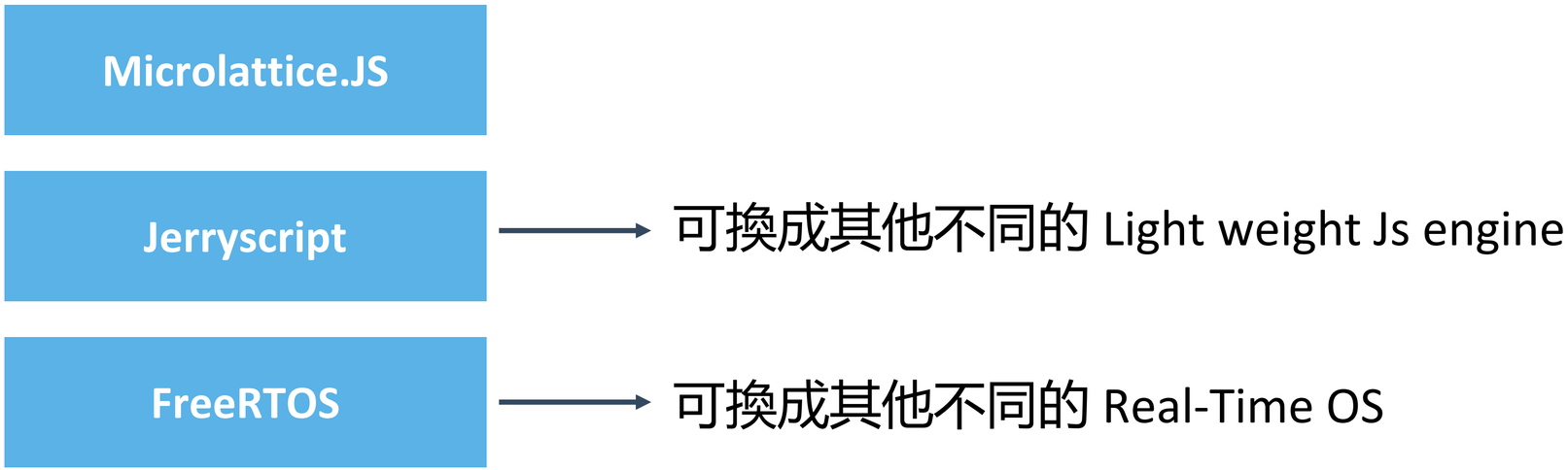
使用 RTOS API (初期專注 freeRTOS)

Jerryscript engine

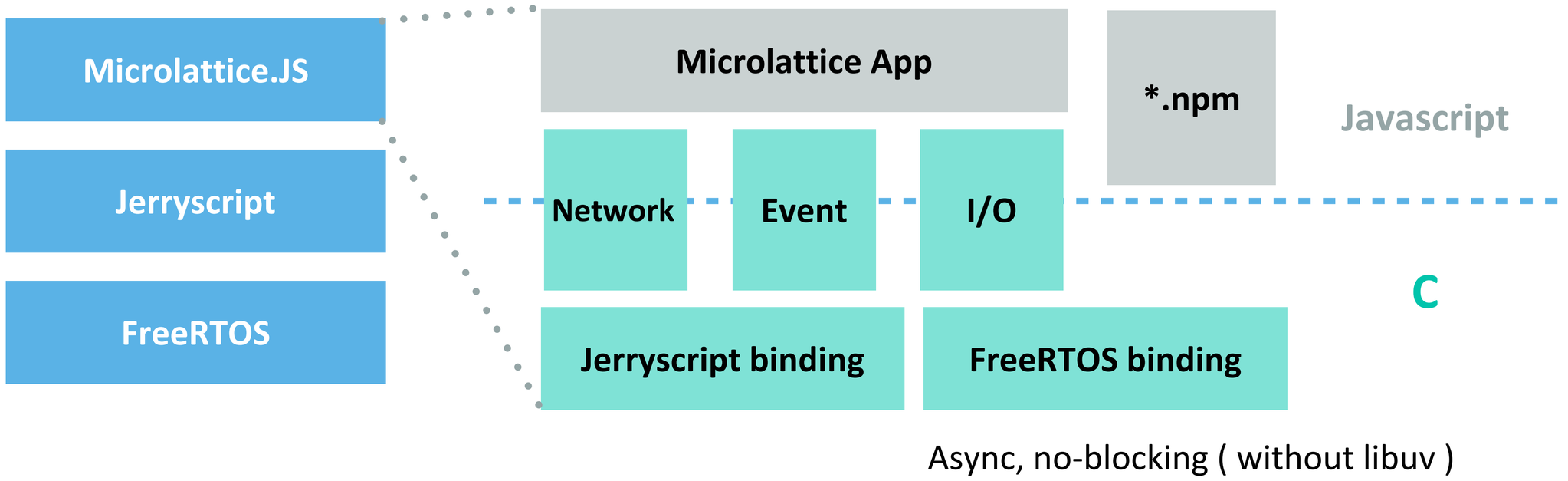
Microlattice.js Architecture



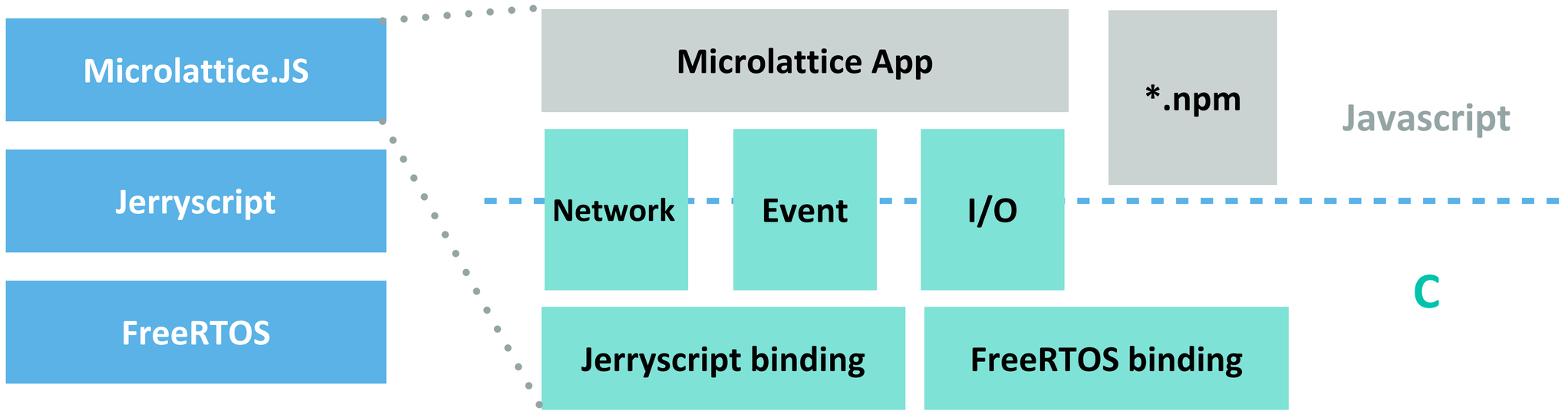
Microlattice.js Architecture



Microlattice.js Architecture



Microlattice.js Architecture



Async, no-blocking (without libuv)

support chip: MT7687 MT2523 MT7697

這樣還不夠 ….

- **MCU build tool 環境各家不一致**
- **Debug/ Download code 各家不一樣**
- LICENSE 問題：大部份廠商 SDK 皆有 dis-contribute 規範
- 沒有共同的 sharing project 標準
- 現成的 Js total solution 不夠輕量，不符合效能接近 C 的需求
- freeRTOS 設計，沒有 filesystem 設計

關於 Build tools

使用 Microlattice.js cli 來客製化需求

Microlattice.js CLI

- ml create
- ml burn　(Mass storage)
- ml debugger (OpenOCD)
  - ml init:7687
  - ml install:gcc
  - ml install:sdk
  - …

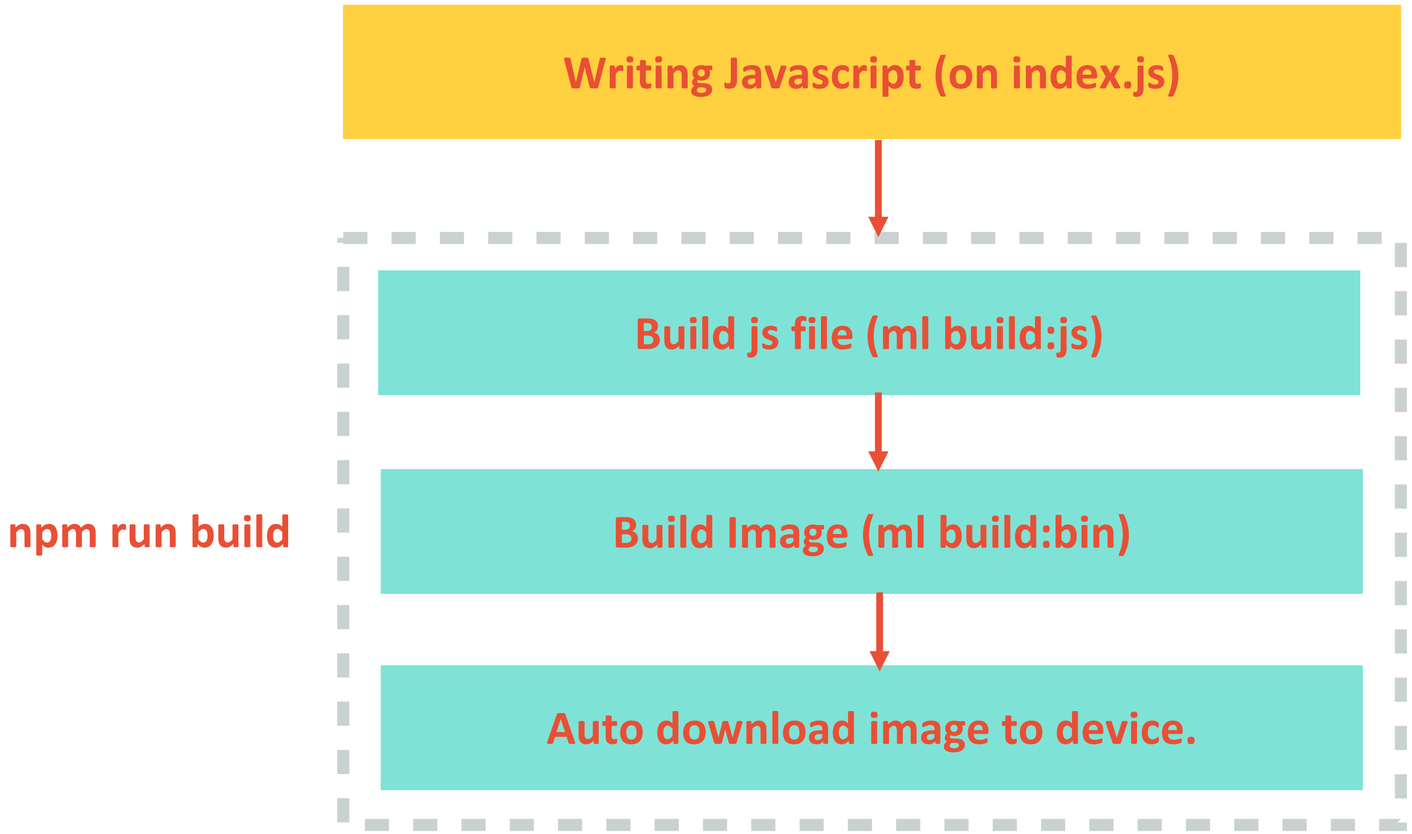
基本款通用的 command line

廠商可以客製化自己的 cli , 包在 ml-XXX-config 之中

以 7687 為例，全包在 ml-mt7687-config 中

https://github.com/iamblue/ml-mt7687-config/blob/master/index.js

Microlattice.js 一般燒 code 流程（ npm 化 ）

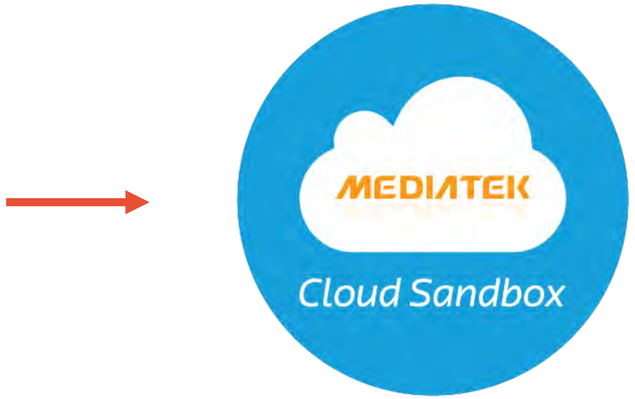


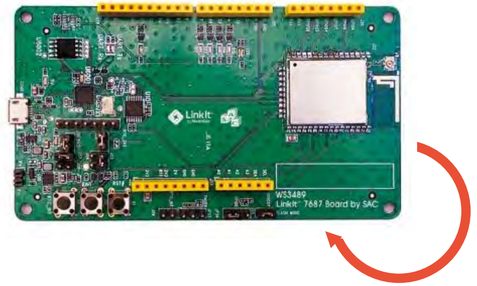
LiveReload 功能，藉由 MCS 輔助（ 隨 3.1.0 SDK release ）

耗時約 3s 內

Writing Javascript (on index.js)

npm run watch





Restart JS runtime

這樣還不夠 ….

- MCU build tool 環境各家很不一致
- Debug/ Download code 各家皆不一樣
- **LICENSE 問題：大部份廠商 SDK 皆有 dis-contribute 規範**
- 沒有共同的 sharing project 標準
- 現成的 Js total solution 不夠輕量，不符合效能接近 C
- freeRTOS 設計，沒有 filesystem 設計

關於 LICENSE 問題 - 讓各家在 command line 自行去定義

- Microlattice 預設會在 project 下有一個 sdk folder
建議各家將自己的 sdk 放在這路徑下去 build 出 image

- MT7687 做法
https://iamblue.gitbooks.io/microlattice-js-for-linkit-rtos/content/zh-TW//intro/create.html

這樣還不夠 ….

- MCU build tool 環境各家很不一致
- Debug/ Download code 各家皆不一樣
- LICENSE 問題：大部份廠商 SDK 皆有 dis-contribute 規範
- **沒有共同的 sharing project 標準**
- 現成的 Js total solution 不夠輕量，不符合效能接近 C
- 大部份 RTOS 設計，沒有 filesystem 設計

Sharing project 參考

- Microlattice gpio example project for MT7687

  https://github.com/iamblue/microlattice-gpio-example

這樣還不夠 ….

- MCU build tool 環境各家不一致
- Debug/ Download code 各家不一樣
- LICENSE 問題：大部份廠商 SDK 皆有 dis-contribute 規範
- 沒有共同的 sharing project 標準
- **現成的 Js total solution 不夠輕量，不符合效能接近 C**
- **freeRTOS 設計，沒有 filesystem 設計**

現成的 Js total solution 不夠輕量，不符合效能接近 C

- freeRTOS 設計，沒有 filesystem 設計 (在 build 過程設計 require module 機制)
- Native api binding 瘦身，用到即載入
- 針對 npm 加強使用 Native binding 時的效能優化

關於 npm，module management 效能優化

```
var wifi = require('ml-wifi');
var Wifi = new wifi({
  mode: 'station',
  auth: '',
  ssid: '',
  password: '',
});

Wifi.connect(function() {
  print('wifi connect!');
});
```

Microlattice.js
有設計輕量級的 require module 機制，可以在開發階段跟熟悉的語法一樣 require node module

```
var wifi = require('ml-wifi');
var Wifi = new wifi({
  mode: 'station',
  auth: '',
  ssid: '',
  password: '',
});

Wifi.connect(function() {
  print('wifi connect!');
});
```

Microlattice.js
有設計輕量級的 require module 機制，可以在開發階段跟熟悉的語法一樣 require node module

但由於效能上的考量 (特別是 256 KB 的 RAM 之中)
不建議每個 hal 層 api 都須經由 require 封裝後載入 (RAM 存取這些封裝函數也很吃重)

```
__wifi({
  mode: 'station', // default is station
  auth: 'PSK_WPA2',
  ssid: 'mcs1',
  password: '12345678',
});

global.eventStatus.on('wifiConnect', function(){
});
```

__XXX 為 binding api, 可以直接 call __XXX (or global.__XXX) 來對 C binding api 保有最佳的效能操作

適合較小 RAM 的 Device 開發

```
__wifi({
  mode: 'station', // default is station
  auth: 'PSK_WPA2',
  ssid: 'mcs1',
  password: '12345678',
});

global.eventStatus.on('wifiConnect', function(){
});
```

適合較小 RAM device

```
var wifi = require('ml-wifi');
var Wifi = new wifi({
  mode: 'station',
  auth: '',
  ssid: '',
  password: '',
});

Wifi.connect(function() {
  print('wifi connect!');
});
```

適合較大 RAM device

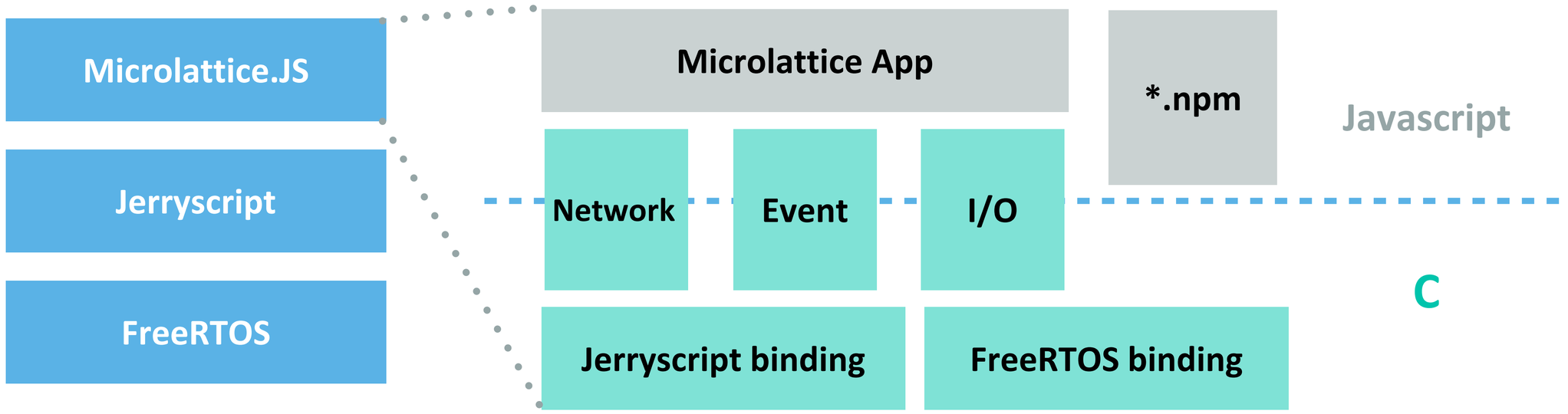
Support Javascript API List

- I/O
  - Pinmux
  - GPIO
  - ADC
  - PWM
  - Uart
- WIFI
  - AP/STATION/REAPTER mode
- Encryptor
  - MD5
  - SHA
  - AES (TBD)
  - DES (TBD)
- Internet protocol
  - MQTT
  - TCP
  - UDP
  - HTTP
  - HTTPS (TBD)
  - MQTTs (TBD)
  - HTTP2 (TBD)
- Event
  - Event emitter
- Tools
  - Timer (setTimeout/ setInterval / …)
  - Utils
  - Fota
  - Wdt
  - RebootScript

https://iamblue.gitbooks.io/microlattice-js-for-linkit-rtos/content/zh-TW//api/pinmux.html

Demo

Microlattice.js Architecture



Async, no-blocking (without libuv)

support chip: MT7687 MT2523 MT7697

Microlattice.js benchmark

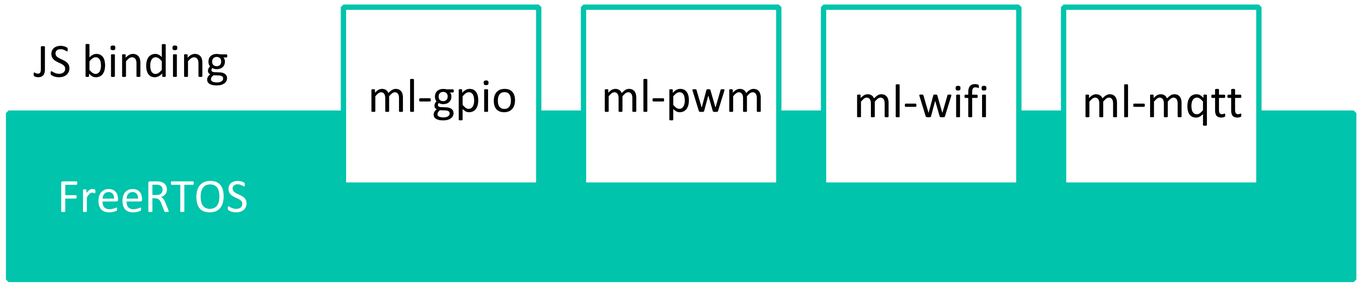
- RAM 使用率
  - SDK v3.0.0: Free heap -> ~80KB
  - SDK v.3.1.0~3.3.1: Free heap -> ~ 105 KB (有含 wifi)
  - SDK v.3.1.0~3.3.1: Free heap -> ~ 150 KB (不含 wifi)

- 每個 Native API 最多瞬間佔用 20 KB RAM
- ROM 使用率: Basic MTK SDK (iot_sdk) + Microlattice.js ~600KB

當前不足之處改進

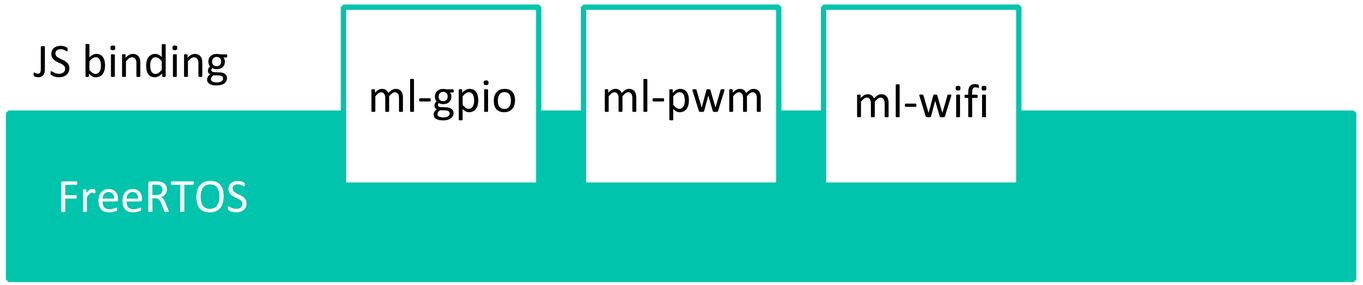
Microlattice.js 當前一些問題

現在: 基於 SDK 疊加
JS binding



Microlattice.js 當前一些問題

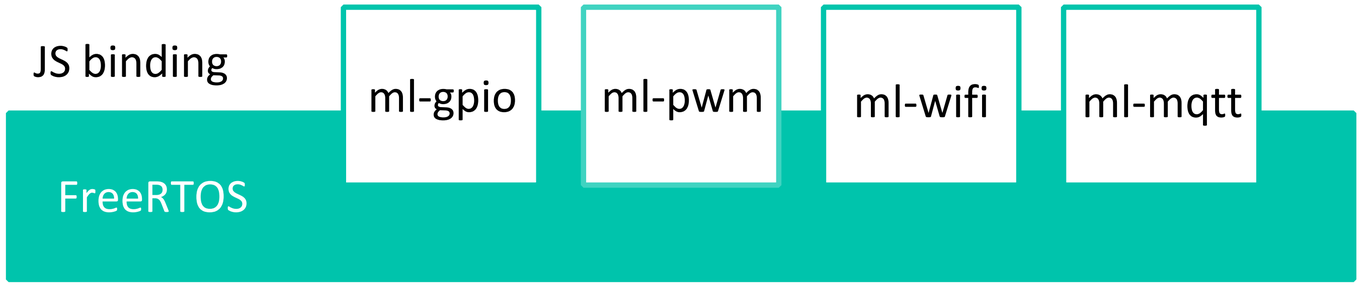
現在: 基於 SDK 疊加
JS binding



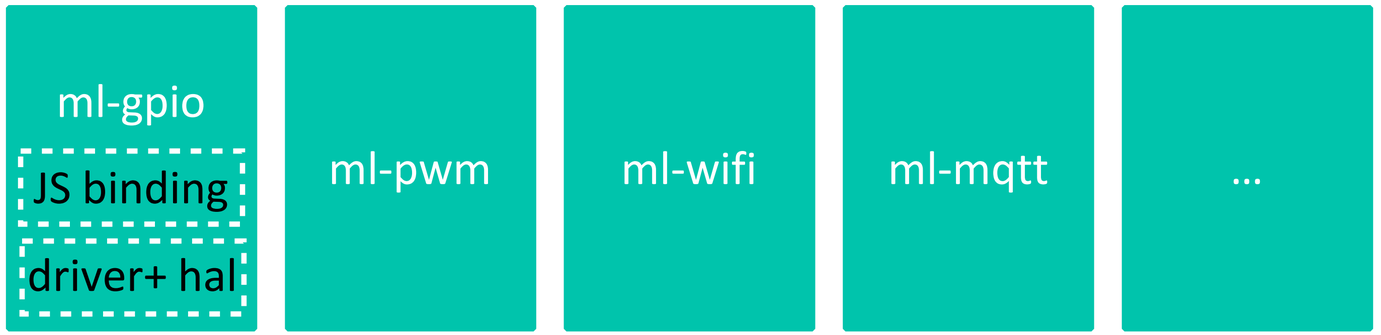
Total size 還是很大
且還有很多 dependency 問題

Microlattice.js 當前一些問題

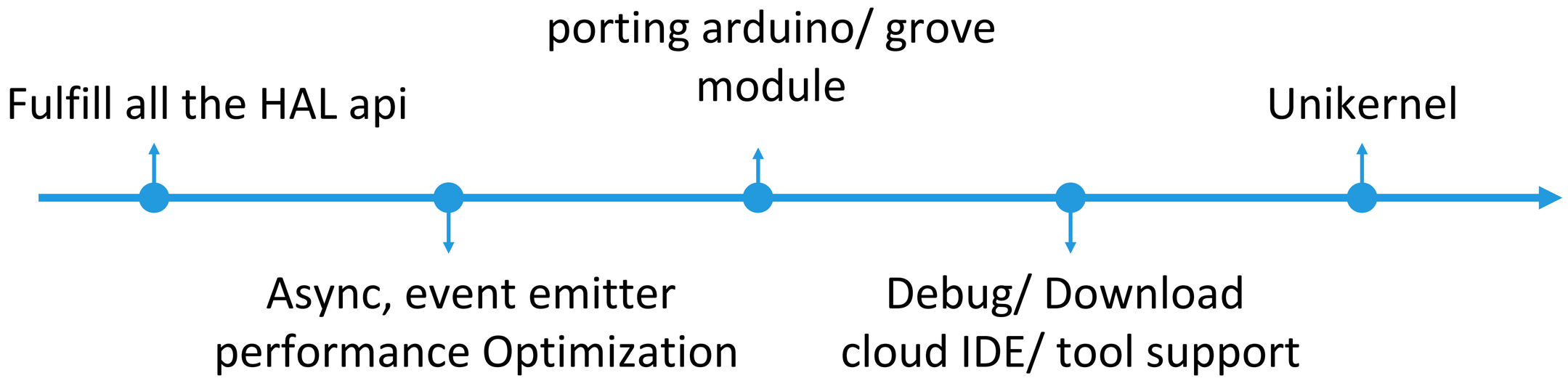
現在: 基於 SDK 疊加
JS binding



Unikernel 概念

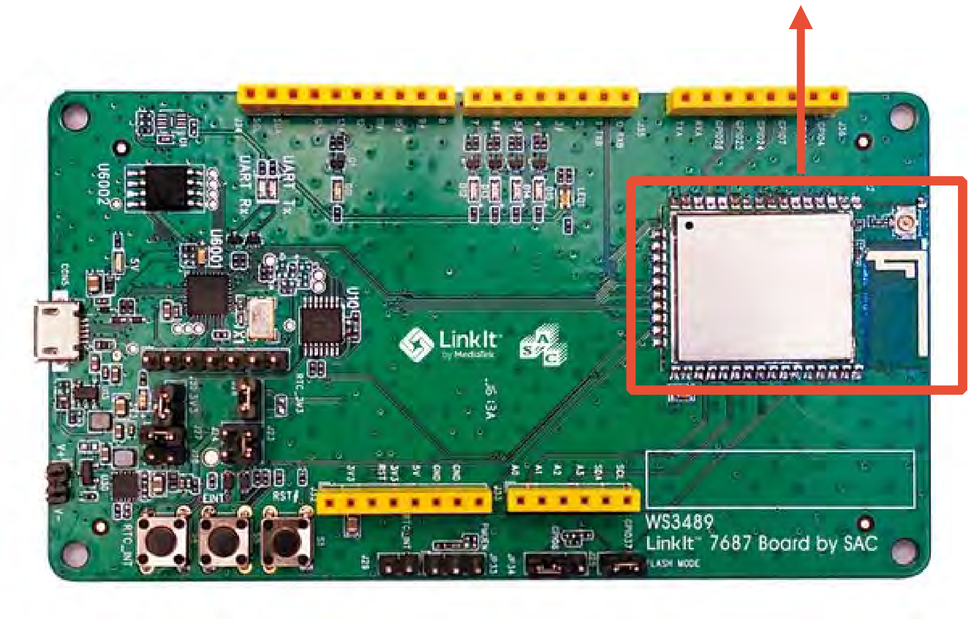


Microlattice.js RoadMap



MT7687 Datasheet

Module: $ **4.9**



HDK: $ **19.9**

- ARM Cortex-M4F MCU (192MHz).
- 1x1 802.11b/g/n Wi-Fi subsystem designed for power efficiency and robust connectivity.
- Versatile peripheral connectivity, including UART, I2C, SPI, I2S, PWM, IrDA and auxiliary ADC.
- Embedded SRAM/ROM and 2MB serial flash in package.
- Integrated security engine (AES and 3DES/SHA).
- 8 x 8mm 68-pin QFN package.

http://ec.wpgo2o.com/getAdDetailPage?AdId=46&Catg=C&ItemId=3086999&ParentId=45

About Microlattice.js

- E-Book:
https://www.gitbook.com/book/iamblue/microlattice-js-for-linkit-rtos

- Github:
https://github.com/iamblue/microlattice

Make it big with something small

特別感謝所有參與技術諮詢的所有同仁和朋友

RD： Anthony, KC(KengChu/有力人士), Pablo, smallp, Vincent
Microlattice.js 命名： Rosalind
文件翻譯： Angie

Thank you