【國產 IC 開發套件】

DSI5188 教材說明

指導單位：經濟部工業局

主辦單位：財團法人資訊工業策進會

執行單位：物聯網智造基地

合作單位：網聯通訊股份有限公司

目　　錄

圖目錄

表目錄

一、 計畫緣起及目標

產學研生態鏈結物聯網智造基地計畫以四年期為發展目標，計畫緊扣政府亞洲·矽谷推動方案、智慧製造、創新經濟發展等方案，配合國家晶片設計與半導體前瞻科技…等相關科技研發或應用施政政策，基於臺灣產業既有優勢的硬體製造，結合未來的軟性趨勢（包含軟體以及創新服務系統等）。從堅強硬體出發，培養軟實力人才，透過收斂與串連產、學、研製造資源於物聯網智造基地 HUB，並結合智慧聯網實作專題競賽進行案件募集，進行國產開發套件推廣，實現可量產產品雛型生產鏈，並導入國內半導體相關解決方案。本計畫為了推動國產 IC 於創新物聯產業之應用，以國產 IC 晶片為核心，進行開發套件之設計與推動，降低 IoT 產品開發之技術門檻，期待進一步提升國產 IC 物聯方案之普及度。自 107 年度始，本計畫透過研究國內 IC 晶片之物聯網解決方案，提出最常被應用於物聯網創新服務之硬體元件模組，期望幫助新創團隊加速產品開發，並提供開發過程範例、產品開發指導等支援；同時，經由雲端平台的資料介接，可另行串接網路推播功能，讓產品的服務流程更加完善。108 年度，物聯網智造基地攜手 IC 業者，推出二款國產 IC 開發套件與應用教材，包括：運用瑞昱半導體 IC 晶片型號 RTL8711AM 之 DSI5168Wi-Fi 開發套件、運用聯發科技 IC 晶片型號 MT2625 之 DSI2598NBIoT 開發套件，同步搭配多場實作工作坊，促進國內 IC 產業發展數據應用與衍生性服務，促進國產 IC 開發方案普及化，並產出 IoT 雛型案例。

實作開發上有鑑於本計畫所推動之產品導向為物聯網相關領域之產品輔導，因此設備端如何透過晶片聯網為產品端的重要課題，實際於國內進行相關物聯網網通晶片盤點後，為延續 FY108 實際開發之 DSI5168(Wi-Fi)及 DSI2598(NBIoT)之成功教材典範與提供更多樣性的開源設計開發方案，FY109 國產開發套件擬以新唐 NuMicro-M487 與網聯 OPL1000 兩款 IC 作為今年度協助業者發展推動之標的，相較其他國產晶片，新唐科技本款 IC 對應 Arduino 開發環境與應用社群相對完整與友善，開發板腳位設計符合開源社群定義，可讓開發者於研發中彈性整合之感測器、通訊協議、驅動裝置等擴充模組，達到快速開發暨驗證測試之目的，同時新創團隊可於物聯網數據平台做研發前端測試，產品雛型完成後可接續應用新唐科技預整合好之國際物聯網雲端平台(例如 AWS 跟 AZURE)之服務鏈，快速接軌國際；網聯科技 OPL1000 為嵌入式多標準 2.4GHz 無線連接解決方案，將 Wi-Fi 和藍芽智慧集成到一個單一的 SoC，使 IoT 系統互連設計更輕鬆，方便開發者以更大的靈活性串接物聯網數據平台，適合需要低功耗、高性能、靈活串接的新興物聯網產品應用，同時可擴展網路與最低系統成本。

二、 國產 IC 開發套件 DSI5188

本款套件為使用網聯通訊 Netlink OPL1000 串接物聯網數據平台，因而命名為 DSI5188，期待運用其開發板超低功耗、高性能、高效能低成本等特性，搭載 Wi-Fi 與 Bluetooth 雙頻連網通訊功能，未來可廣泛應用於穿戴式裝置，例如健康照護、體感監測、智能鎖、智能秤、Wi-Fi 定位、智慧燈具開關、透傳模式(橋接模式)等相關領域，可大幅降低產品成本與功能多樣性。

本教材規劃分為三個部分：DSI5188 介紹、物聯網數據平台串接說明、DSI 5188 應用範例。分別說明如下：

(一) DSI5188 介紹

DSI 5188 結合 OPL1000 開發板串接物聯網數據平台，成為一套物聯網創新軟硬整合方案，POL1000 具有雙 ARM® Cortex 核心，且為一嵌入式 2.4GHz 無線連接解決方案，它將 Wi-Fi 802.11b 和 Bluetooth Smart® 集成到一個系統單晶片中，與僅支援單一通訊協定的解決方案相比，其突破性的架構，共享 Wi-Fi 和 Bluetooth，基頻和 MAC，因此可以實現業界最低的功耗，最高的性能，以及最低的系統成本。此外，它提供了可擴展的網絡，使設計互連 IoT 系統和連接雲端更加輕鬆、靈活。

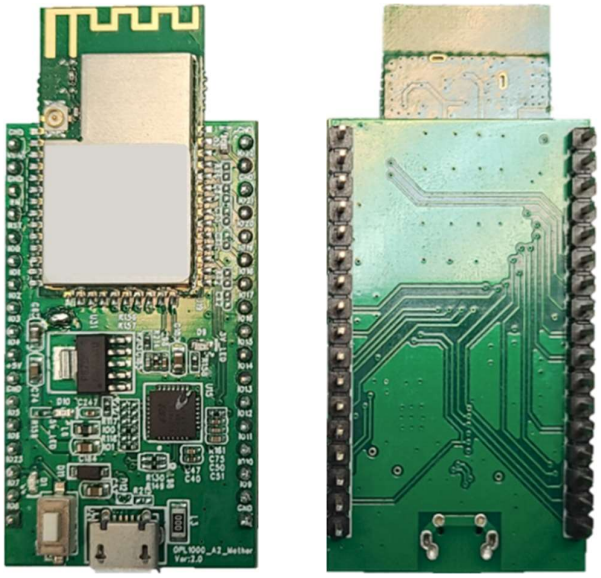

圖 1、DSI5188 (OPL1000)正反面示意圖

1. DSI5188 特色

具有雙 ARM Cortex M0 / M3 核心，支援 Wi-Fi 802.11b 傳輸速率達 11 Mbps，同時支援 Bluetooth 5.0 LE 傳輸速率達 2 Mbps，擁有集成電源管理單元、完整硬體安全加密引擎，具有訊息加密、消息認證傳輸協定等功能。具有 GPIO、ADC、SPI、UART、I2C、PWM 週邊，其中最大特色為超低功耗，可使用 CR2032 鈕扣電池操作 Wi-Fi 及 BLE 雙模。

2. DSI5188 硬體規格

表 1、硬體功能簡表

DSI 5188 硬體功能	
Chipset	Wi-Fi 802.11b、Bluetooth 5.0 LE 通訊晶片
MCU	ARM® Cortex-M0、Cortex-M3
I/O	18
ADC	10
SPI	2
UART	2
I2C	1
PWM	6

3. 腳位說明

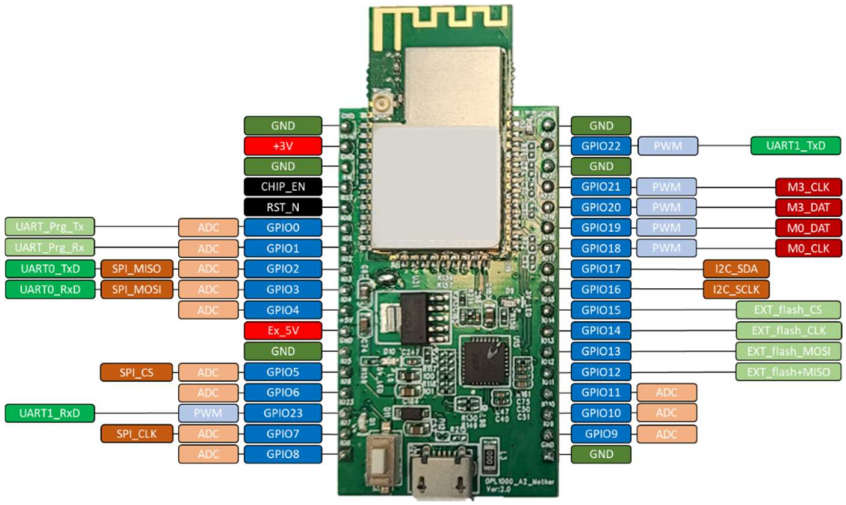

圖 2、DSI5188 腳位圖

4. 燒錄教學

(1) 至 GitHub 的 OPL1000A2-SDK 專案下載並開啟 OPL1000 Download Tool (於\Tool\Download\download_RELEASE.exe)。
(2) 於 Pack 標籤設定 Script 為 FW_Pack\PatchData.txt 存放路徑，M0 Bin 為 FW_Pack/opl1000_m0.bin 存放路徑，M3 Bin 為編譯應用(以 GCC 編譯為例，於各專案目錄下輸入 make 指令)產生 opl1000_app_m3.bin 之存放路徑，設定完成後按下 Pack。
(3) 點選 Download，三秒內按板上 Reset 按鈕。(Download 標籤中 Patch Bin 即自動帶入 M0、M3 Bin 合併後產生 opl1000.bin 之存放路徑。)

(二) 物聯網數據平台串接說明

1. 教材說明

本範例使用 DSI 5188 開發版，連接 Wi-Fi 網路，將資料上傳至物聯網數據平台。

2. 教材材料

- DSI 5188 開發板 1 個
- Micro USB 線 1 條

3. 程式說明

(程式碼：請至 https://reurl.cc/N6X765 下載 ch2 資料夾)

下載 OPL1000A2-SDK 專案，於 SDK\APS_PATCH\examples 路徑下新增 training 資料夾，複製路徑 SDK\APS_PATCH\examples\protocols\https_request 資料夾至 training 資料夾內，並重新命名為 ch2。

(1) https_client_request.h 程式碼說明

第 22-23 行：修改 WI-FI_SSID、WI-FI_PASSWORD 為欲連線 Wi-Fi 的 SSID 及密碼。

表 2、22-23 程式碼示意圖

22	#define WI-FI_SSID "Opulinks-TEST-AP"
23	#define WI-FI_PASSWORD "1234abcd"

(2) https_client_request.c 程式碼說明

第 57 行：修改 SERVER_NAME 定義。

第 58 行：新增 WEB_URL 定義，將[AccessTokens]修改為物聯網數據平台使用的存取權杖，現為 20 字元英文字串。

第 59-67 行：將 GET_REQUEST 改為 POST_REQUEST，並修改其定義為上傳 json 格式字串，其中 PM2.5 為於物聯網數據平台輸入的時間序列名稱。(有關存取權杖及時間序列，詳見物聯網數據平台網頁教學.pptx ：https://iforum.ideaschain.com.tw/iforum/devtool/board.do?board=3)。

第 69-72 行：新增 random_int 函數定義，其調用 rand 函數以產生指定範圍內的亂數，其中第 64 行定義請求體長度為 14 bytes，而錯誤的請求體長度會導致超時，或訊息不完全。

表 3、57-72 程式碼示意圖

57	#define SERVER_NAME "iiot. ideaschain.com.tw"
58	#define WEB_URL "http://iiot. ideaschain.com.tw/api/v1/[AccessTokens]/telemetry"
59	#define POST_REQUEST "POST " WEB_URL " HTTP/1.1\r\n"\
60	"Content-Type: application/json; charset=utf-8\r\n"\
61	"Host: "SERVER_NAME"\r\n"\
62	"User-Agent: OPL1000 Opulinks\r\n"\
63	"Connection: close\r\n"\
64	"Content-Length: 14\r\n"\
65	"\r\n"\
66	"{\"PM2.5\":\"%d\"}"\
67	"\r\n"
68	
69	static int random_int(int min, int max)
70	{
71	return min + rand() % (max+1 - min);

72	}

第 451 行：調用 lwip_network_init 函數執行 TCP/IP、網路介面與 DHCP 客戶端處理初始化。

第 454 行：調用 lwip_net_ready 函數等待連線與取得 IP。

第 456 行：新增 srand 函數，以 time(NULL)為參數，改變亂數種子。

第 458-465 行：while 迴圈內調用 ssl_client_start 函數，等待十秒後再度執行。

表 4、第 448-466 程式碼示意圖

```
448  void app_entry(void *args)
449  {
450      /* Tcpip stack and net interface initialization,    dhcp client process
     initialization. */
451      lwip_network_init(WI-FI_MODE_STA);
452
453      /* Waiting for connection & got IP from DHCP server */
454      lwip_net_ready();
455
456      srand(time(NULL));
457
458      while (1)
459      {
460          ssl_client_start();
461          for (int countdown = 10; countdown >= 0; countdown--){
462              printf("%d...    \r\n", countdown);
463              osDelay(1000);
464          }
465      }
466  }
```

第 117 行：修改 buf 陣列大小為 1024 字元。

第 268 行：新增調用 random_int 函數產生亂數，並賦值給 val 變數。

第 269 行：修改調用 sprintf 將 POST_REQUEST 定義與 val 變數組合並填入 buf 字元陣列，其回傳字元陣列長度，並賦值給 len 變數。

第 270 行：調用 mbedtls_ssl_write 函數將 buf 字元陣列上傳。

表 5、第 112-339 程式碼示意圖

```
112  ssl_client_start(void)
113  {
...  /*---略---*/
117      unsigned char buf[1024];
...  /*---略---*/
268      int val = random_int(15,45);
269      len = sprintf( (char *) buf, POST_REQUEST, val);
270      while ((ret = mbedtls_ssl_write(&ssl, buf, len)) <= 0)
271      {
...  /*---略---*/
278      }
...  /*---略---*/
339  }
```

4. 教材成果演示

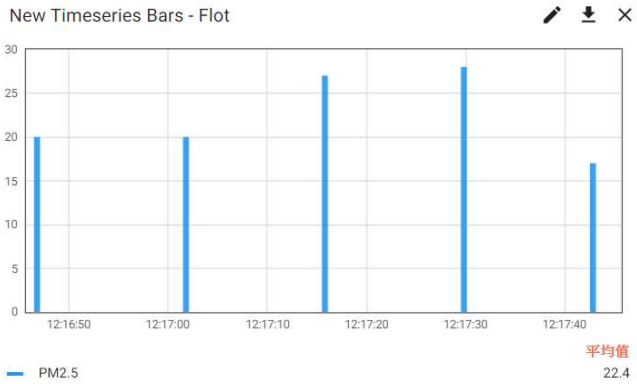

圖 3、每十秒亂數產生虛擬 PM2.5 感測值上傳

(三) DSI 5188 應用範例一：環境煙霧及可燃氣體監測

1. 教材說明

本應用使用 MQ-2 氣體感測器探測液化氧、丁烷、丙烷、甲烷、酒精、氫氣、煙霧等氣體，並將探測結果(含可燃氣與乾淨空氣下之電阻比值)透過 DSI 5188 的 Wi-Fi 上傳至物聯網數據平台。

MQ-2 感測器的原理為內部氣敏材料的電導率會隨空氣中可燃氣體濃度的增加而增大，而內部電路將電導率轉換為相對應的輸出信號，再經過非線性轉換後即可知道空氣中可燃氣體的濃度，可探測範圍為 300~10000ppm。

2. 教材材料

- DSI 5188 開發版 1 個
- Micro USB 線 1 條
- 杜邦線 3 條
- MQ-2 氣體偵測感測器 1 個

3. 接線圖

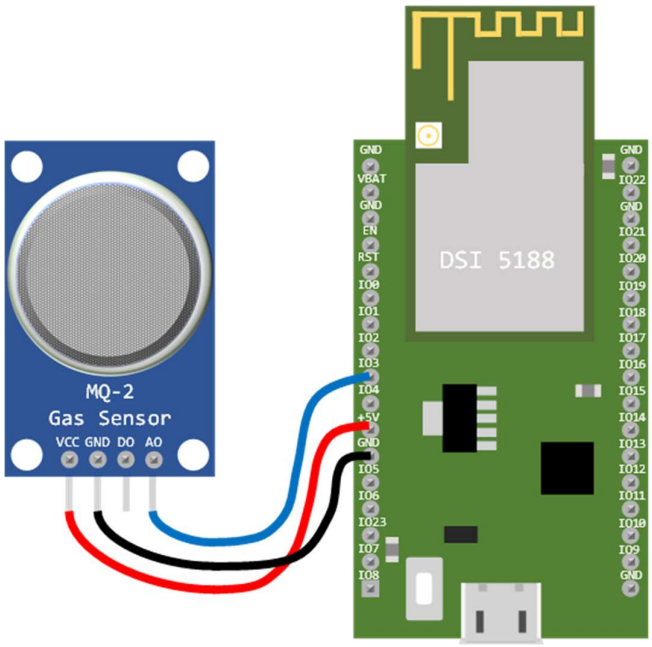

圖 4、M487 與 LED 接線組裝

表 6、硬體功能簡表

MQ-2		DSI 5188
VCC	5V 電源	+5V
GND	接地	GND
DO	數位訊號	
AO	類比訊號	GPIO3(類比數位轉換)

4. 在物聯網數據平台上新增裝置

教材程式：請至 https://reurl.cc/N6X765 下載 ch3 資料夾

複製 SDK\APS_PATCH\examples\peripherals\auxadc 資料夾至 training 資料夾內，並重新命名為 ch3，再複製 ch2 資料夾內 https_client_request.h 及.c 兩個檔案至 ch3 資料夾內。

(1) main_patch.c 程式碼說明

第 55 行：新增包含 https_client_request.h 標頭檔。

第 58 行：修改 WAIT_TIME_MS 為 10000，即 10 秒。

第 71~74 行：修改 S_MessageQ 結構型別，內有 float 型別的 ratio 變數。

第 81~82 行：新增外部變數，為記憶體區塊設定相關參數。按下紅色箭頭處的連結來新增裝置

表 7、第 55-82 程式碼示意圖

```
55    #include "https_client_request.h"
…
58    #define WAIT_TIME_MS       10000
…
71    typedef struct
72    {
73          float ratio;
74    } S_MessageQ;
…
81    extern uint8_t* g_ucaMemPartAddr;
```

82	extern uint32_t g_ulMemPartTotalSize;

第 129 行：調用 SysInit_EntryPoint 函數，進行冷啟動系統初始化。

第 148 行：設定 AT 指令切換 AT/Debug UART。

第 151~154 行：修改配置記憶體邊界。

第 156 行：指定應用程式進入點為使用者定義 Main_AppInit_patch 函數。

表 8、第 126-129 程式碼示意圖

126	void __Patch_EntryPoint(void)
127	{
128	// don't remove this code
129	SysInit_EntryPoint();
…	
…	//---略---
…	
147	// update the switch AT UART / dbg UART function
148	at_cmd_switch_uart1_dbguart = Main_AtUartDbgUartSwitch;
149	
150	// modify the heap size, from 0x43C000 to 0x44F000
151	g_ucaMemPartAddr = (uint8_t*) 0x43C000;
152	g_ulMemPartTotalSize = 0x13000;
153	
154	Sys_SetUnsuedSramEndBound(0x43C000);
155	// application init
156	Sys_AppInit = Main_AppInit_patch;
157	}

第 326-328 行：新增訊息佇列及記憶池相關變數。

第 328 行：新增宣告 Main_AppMessageQSend 函數，其作用為發送訊息至佇列。

第 334 行：新增宣告 tMessageDef 變數。

第 335 行：新增宣告 tMemPoolDef 變數。

第 348 行：新增調用 AppInit 函數，其作用為初始化並啟動 Wi-Fi。

第 351~358 行：新增建立訊息佇列。

第 361~368 行：新增建立記憶池。

表 9、第 323-369 程式碼示意圖

```
323 static osThreadId g_tAppThread_1;
324 static void Main_AppThread_1(void *argu);
325
326 osMessageQId g_tAppMessageQ;
327 osPoolId g_tAppMemPoolId;
328 S_MessageQ *ptMsgPool;
329 static osStatus Main_AppMessageQSend(S_MessageQ *ptMsg);
330
331 static void Main_AppInit_patch(void)
332 {
333     osThreadDef_t tThreadDef;
334     osMessageQDef_t tMessageDef;
335     osPoolDef_t tMemPoolDef;
336
337     tThreadDef.name = "App_1";
338     tThreadDef.pthread = Main_AppThread_1;
339     tThreadDef.tpriority = OS_TASK_PRIORITY_APP;            //
    osPriorityNormal
340     tThreadDef.instances = 0;         // reserved, it is no used
341     tThreadDef.stacksize = OS_TASK_STACK_SIZE_APP;         // (512),
    unit: 4-byte, the size is 512*4 bytes
342     g_tAppThread_1 = osThreadCreate(&tThreadDef, NULL);
343     if (g_tAppThread_1 == NULL)
344     {
345         printf("To create the thread for AppThread_1 is fail.\n");
        }
346
347     AppInit();
348
349     // create the message queue for AppMessageQ
350     tMessageDef.queue_sz = APP_MESSAGE_Q_SIZE;                //
351 number of elements in the queue
        tMessageDef.item_sz = sizeof(S_MessageQ);                 // size of
352 an item
        tMessageDef.pool = NULL;          // reserved, it is no used
```

```
353         g_tAppMessageQ = osMessageCreate(&tMessageDef,
354 g_tAppThread_1);
            if (g_tAppMessageQ == NULL)
355         {
356             printf("To create the message queue for AppMessageQ is
357 fail.\n");
            }
358
359         // create the memory pool for AppMessageQ
360         tMemPoolDef.pool_sz = APP_MESSAGE_Q_SIZE;                    //
361 number of items (elements) in the pool
            tMemPoolDef.item_sz = sizeof(S_MessageQ);                  // size of
362 an item
            tMemPoolDef.pool = NULL;                                     //
363 reserved, it is no used
            g_tAppMemPoolId = osPoolCreate(&tMemPoolDef);
364         if (g_tAppMemPoolId == NULL)
365         {
366             printf("To create the memory pool for AppMessageQ is
367 fail.\n");
            }
368 }
369
```

第 374-377 行：新增宣告變數，floVoltageCal 為校正期間平均電壓，R0 為空氣無揮發氣體時之基準電阻值。ratio 為比值有無揮發氣體時之電阻值比值，tMsg 為訊息結構。

第 386-396 行：連續讀取 ADC 腳位電壓值 100 次求平均，換算基準電阻值 R0。

第 398-418 行：循環讀取 ADC 腳位電壓值，並計算電阻值比值，調用 Main_AppMessageQSend 函數將數值發送至訊息佇列，每十秒完成一次循環。

表 10、第 371-419 程式碼示意圖

```
371 static void Main_AppThread_1(void *argu)
372 {
373     float floVoltage;
374     float floVoltageCal;
375     float R0;
376     float ratio;
377     S_MessageQ tMsg;
378
379     Hal_Aux_Init();
380     //Hal_Aux_PatchInit();
381
382     g_ubHalAux_Pu_WriteDirect = 1;
383     Hal_Aux_AdcCal_Init();
384
385     // Get a average data by testing 100 times
386     for(int x = 0 ; x < 100 ; x++)
387     {
388         Hal_Aux_IoVoltageGet(AUX_GPIO_IDX, &floVoltage);
389         floVoltageCal = floVoltageCal + floVoltage;
390     }
391
392     g_ubHalAux_Pu_WriteDirect = 0;
393
394     floVoltageCal = floVoltageCal/100.0;
395
396     R0 = (5.0-floVoltageCal)/floVoltageCal/9.8;
397
398     while (1) {
399         g_ubHalAux_Pu_WriteDirect = 1;
400         Hal_Aux_AdcCal_Init();
401         uint8_t ret = Hal_Aux_IoVoltageGet(AUX_GPIO_IDX,
    &floVoltage);
402         g_ubHalAux_Pu_WriteDirect = 0;
403
404         if (ret == HAL_AUX_OK) {
```

```
405                 ratio =(5.0-floVoltage)/floVoltage/R0;
406                 printf("Get io %d voltage : %.1f \r\n", AUX_GPIO_IDX,
floVoltage);
407                 printf("Ratio :%.1f \r\n",ratio);
408
409                 // send the result to AppThread_1
410                 tMsg.ratio = ratio;
411                 Main_AppMessageQSend(&tMsg);
412             } else {
413                 printf("error io ...\r\n");
414             }
415
416             osDelay(WAIT_TIME_MS);
417             printf("\r\n");
418         }
419     }
```

第 421-451 行：新增 Main_AppMessageQSend 函數，參考範例 gpio 中 Main_AppMessageQSend 函數，將第 424 行宣告註解。

表 11、第 55-82 程式碼示意圖

```
421 static osStatus Main_AppMessageQSend(S_MessageQ *ptMsg)
422 {
423     osStatus tRet = osErrorOS;
424     //S_MessageQ *ptMsgPool;
425
426     // allocate the memory pool
427     ptMsgPool = (S_MessageQ *)osPoolCAlloc(g_tAppMemPoolId);
428     if (ptMsgPool == NULL)
429     {
430         printf("To allocate the memory pool for AppMessageQ is
    fail.\n");
431         goto done;
432     }
433
434     // copy the message content
435     memcpy(ptMsgPool, ptMsg, sizeof(S_MessageQ));
436
437     // send the message
438     if (osOK != osMessagePut(g_tAppMessageQ, (uint32_t)ptMsgPool,
    osWaitForever))
439     {
440         printf("To send the message for AppMessageQ is fail.\n");

441         // free the memory pool
442         osPoolFree(g_tAppMemPoolId, ptMsgPool);
443         goto done;
444     }
445
446     tRet = osOK;
447
448 done:
449     return tRet;
450 }
451
```

(2) https_client_request.c

第 59~67 行：修改 POST_REQUEST 定義，json 字串內之屬性名稱改為 ppm。

第 69~72 行：修改 S_MessageQ 結構型別內容，內有名為 ratio 的 float 型別變數，其中第 64 行定義請求體長度修改為 13 bytes。

第 74~76 行：宣告訊息佇列、記憶池相關外部變數。

表 12、第 59-76 程式碼示意圖

59	#define POST_REQUEST "POST " WEB_URL " HTTP/1.1\r\n"\
60	"Content-Type: application/json; charset=utf-8\r\n"\
61	"Host: "SERVER_NAME"\r\n"\
62	"User-Agent: OPL1000 Opulinks\r\n"\
63	"Connection: close\r\n"\
64	"Content-Length: 13\r\n"\
65	"\r\n"\
66	"{\"ppm\":\"%.1f\"}"\
67	"\r\n"
68	
69	typedef struct
70	{
71	float ratio;
72	} S_MessageQ;
73	
74	extern S_MessageQ *ptMsgPool;
75	extern osMessageQId g_tAppMessageQ;
76	extern osPoolId g_tAppMemPoolId;

第 277 行：將調用 randon 函數註解。

第 278 行：將 sprintf 傳入參數改為 ptMsgPool->ratio。

表 13、第 121-348 式碼示意圖

121	static int ssl_client_start(void)
122	{
…	/*---略---*/
277	//int val = random_int(15,45);
278	len = sprintf((char *) buf, POST_REQUEST, ptMsgPool->ratio);
	/*---略---*/
…	}
348	

第 460 行：調用 lwip_network_init 函數執行 TCP/IP、網路介面與 DHCP 客戶端處理初始化。

第 463 行：調用 lwip_net_ready 函數等待連線與取得 IP。

第 465 行：新增 osEvent 結構型別的變數 tEvent，其結構內容包含系統狀態、事件值、事件定義。

第 469 行：等待取得 AppMessageQ 訊息佇列中的事件訊息。

第 471-475 行：判斷事件狀態，如果非 osEventMessage 則印出錯誤訊息。

第 478 行：取出訊息內容，並賦值給 ptMsgPool。

第 480 行：調用 ssl_client_start 函數將訊息內容上傳。

第 483 行：釋放記憶池資源。

表 14、第 457-485 程式碼示意圖

```
457 void app_entry(void *args)
458 {
459     /* Tcpip stack and net interface initialization,    dhcp client process
    initialization. */
460     lwip_network_init(WI-FI_MODE_STA);
461
462     /* Waiting for connection & got IP from DHCP server */
463     lwip_net_ready();
464
465     osEvent tEvent;
466
467     while (1) {
468         // receive the message from AppMessageQ
469         tEvent = osMessageGet(g_tAppMessageQ, osWaitForever);
470
471         if (tEvent.status != osEventMessage)
472         {
473             printf("To receive the message from AppMessageQ is
    fail.\n");
474             continue;
475         }
476
477         // get the content of message
478         ptMsgPool = (S_MessageQ *)tEvent.value.p;
479
480         ssl_client_start();
481
482         // free the memory pool
483         osPoolFree(g_tAppMemPoolId, ptMsgPool);
484     }
485 }
```

(3) Makefile

新增以下定義，為編譯時所需檔案。

表 15、第 35-218 程式碼示意圖

35	DEFS += -DMBEDTLS_CONFIG_FILE="<config-opl-default.h>"
36	DEFS += -DHTTPCLIENT_SSL_ENABLE
111	INCDIR += -I ../../../../APS/middleware/third_party/mbedtls/include
	INCDIR += -I ../../../../APS_PATCH/middleware/third_party/mbedtls/include
188	
	M_SRC += ./https_client_request.c
218	

5. 教材成果演示

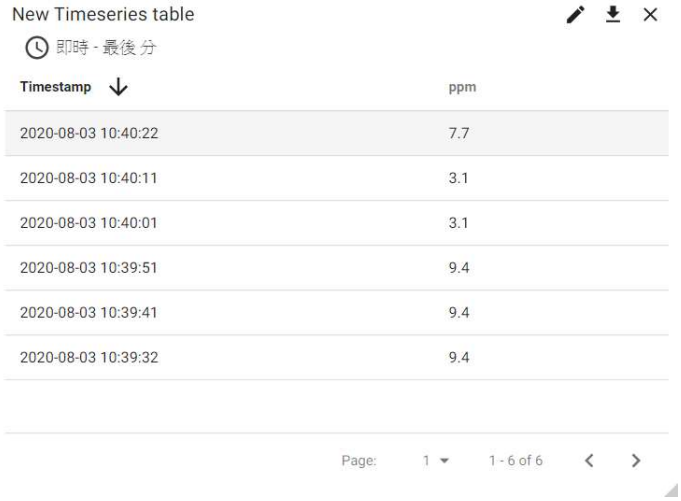

圖 5、物聯網數據平台儀錶板庫介面

此感測值為 Rs/R0 比值，與實際 ppm 為非線性關係，請參考下圖轉換為 ppm。

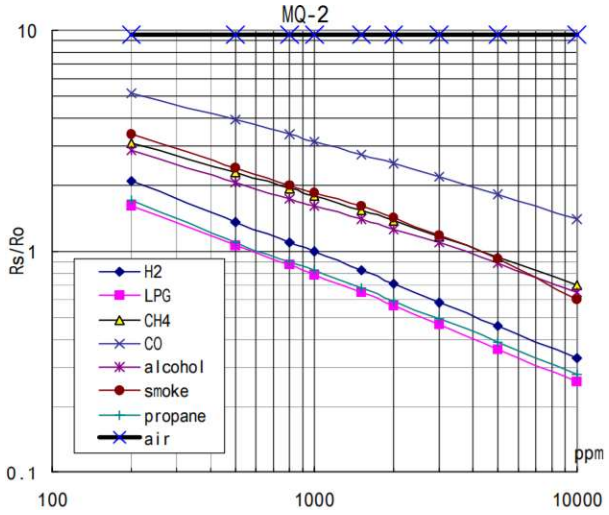

圖 6、MQ-2 氣敏元件的靈敏度特性

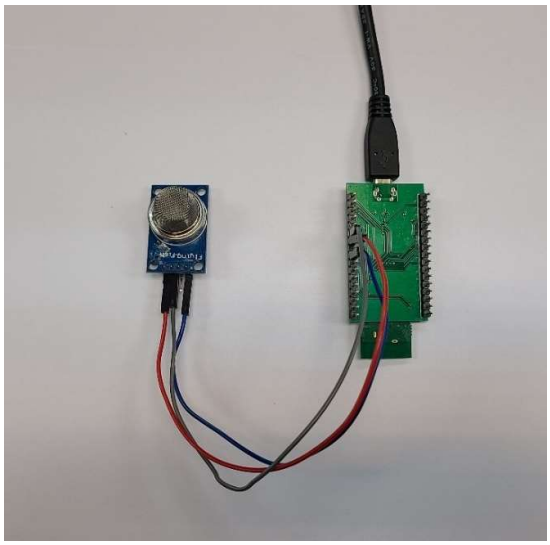

圖 7、電路組裝完成示意圖

三、 參考資料

1. 此教材使用之原始碼：https://github.com/Opulinks-Tech/OPL1000A2-SDK
2. 原廠提供文件：https://github.com/Opulinks-Tech/OpulinksTech-WIKI/wiki/product