ARM® Cortex®-M

32-bit Microcontroller

NuMicro® Family
NuMaker Brick
Design Guide

For additional information or questions, please contact: Nuvoton Technology Corporation.
www.nuvoton.com

目錄

圖目錄

表目錄

1. 簡介

NuMaker Brick 平台的設計是以提供物聯開發者一個感測器高度集成的開發平台，以協助快速開發為目的。模組本身有各自獨立的功能，透過模組的組合使得 NuMaker Brick 平台能有高度的使用彈性，創建滿足使用者的特定應用產品。此外 NuMaker Brick 平台也了解到使用者永遠有自己的創新想法，創建自己設計的專用模組，因此 NuMaker Brick 平台提供了一塊拓展版，提供使用者自行連接感測器，依照需求增加所需要的感測模組，進一步擴充 NuMaker Brick 平台功能。根據經驗，一般新增模組對於使用者而言，最大的挑戰並非是模組本身的新增功能設計，而是與其他模組的整合問題，通常需要修改其他各個模組的程式才能與新增模組溝通，有鑒於此，NuMaker Brick 平台的設計採用統一的通訊協議，新增模組只需要依照通訊協議的規範進行設計，再加上少許的主控端新增協議就能與平台原生的各個模組互相識別與通訊，不但大幅減少新模組開發的複雜度，也幫助開發者更專注於新創模組本身的軟硬體設計，加速產品開發時程。

本文件循序漸進的引導使用者在 NuMaker Brick 平台上新增一個自行創建的模組。從平台基本的架構觀念介紹，到軟體設計概念與通訊協議等，鉅細靡遺，協助開發者建立一個完整的平台運作概念， 並且以一個實際範例的細部解說，讓開發者從概念與到具體的實施方法都完整的結合在一起，作為日後 NuMaker Brick 創新設計的基礎。

2. NUMAKER BRICK 平台程式介紹

2.1. NuMaker Brick 平台概念

NuMaker Brick 平台是由一個主控端與其他感測模組所構成，感測模組只需被動等待主控端呼叫後。傳送或接收資料給主控端，而主控端需要蒐集並處理各感測模組回傳的資料，如圖 2-1

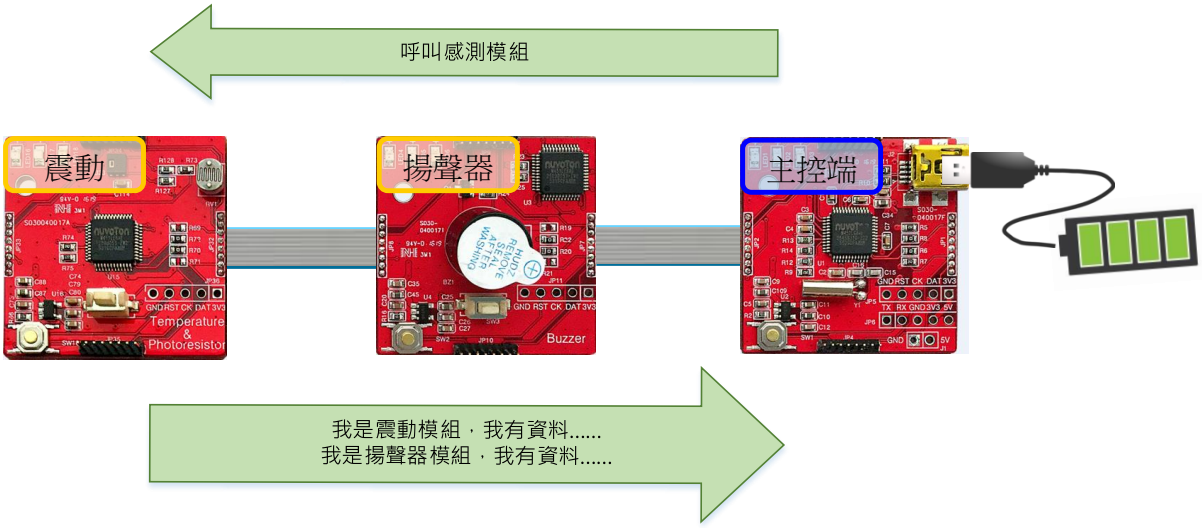

圖 2-1 NuMaker Brick 平台溝通簡介

在這一章將進入整個 NuMaker Brick 平台的程式，介紹各程式的主要功用。主程式放在 main.c 中，分成兩部分：setup() 以及 loop()。setup() 是對裝置做初始化的動作，loop() 是 NuMaker Brick 平台執行時程式所放置的地方，接下來對這兩部份分別做介紹，接著進入整個平台的架構做分析。

為了模組化程式，將每個模組能執行的功能進行分類：

1. 模組初始化：在 setup 中，模組各自需要的初始化功能
2. 模組週期執行程式：在 loop 中，模組需要週期執行的功能
3. 模組及時執行程式：在 loop 中，模組持續檢查或執行的功能

如圖 2-2，使用者可依照需求再將程式加入其中

```
APFN_FUNC_T pfnDevFunc[MAX_TID_DEV] =
{
    /*
    Functions for each device:
    {Initial, Period, Pulling, Report}
    Initial : Init and configure device
    Period  : Process device sensor data or set sensor once per 0.1 seconds.
    Pulling : Process device sensor data or set sensor frequently.
    Report  : Report process data to master
    */

    /* for embedded device boards */
```

模組初始化　　模組周期執行程式　　模組及時執行程式

```
    {Battery_Init,    MasterControl,    NULL,              report_battery},
    {Buzzer_Init,     NULL,             Buzzer_Control,    report_buzzer},
    {Led_Init,        NULL,             Led_Control,       report_led},
    {AHRS_Init,       AHRS_Control,     NULL,              report_ahrs},
    {SonarInit,       SonarDetect,      SonarTimeOutCheck, report_sonar},
    {HTU21D_Init,     WaitHTU21D,       GetHTU21DTemp,     report_temp},
    {Gas_Init,        GetGas,           NULL,              report_gas},
    {IR_Init,         IR_Control,       IR_Check,          report_ir},
    {key_init,        NULL,             NULL,              report_key},

    /* for custom device boards
    fill your four board functions here to specified ID. */

    {Lr_Init,  GetLr,  NULL,  report_resDev9},    // For device ID 9
    {NULL,     NULL,   NULL,  report_resDev10},   // For device ID 10
    {NULL,     NULL,   NULL,  report_resDev11},   // For device ID 11
    {NULL,     NULL,   NULL,  report_resDev12},   // For device ID 12
    {NULL,     NULL,   NULL,  report_resDev13},   // For device ID 13
    {NULL,     NULL,   NULL,  report_resDev14},   // For device ID 14
};
```

使用者可自定義

圖 2-2 NuMaker Brick 程式模組分類圖

2.2. setup()

圖 2-3為 setup() 的流程圖：

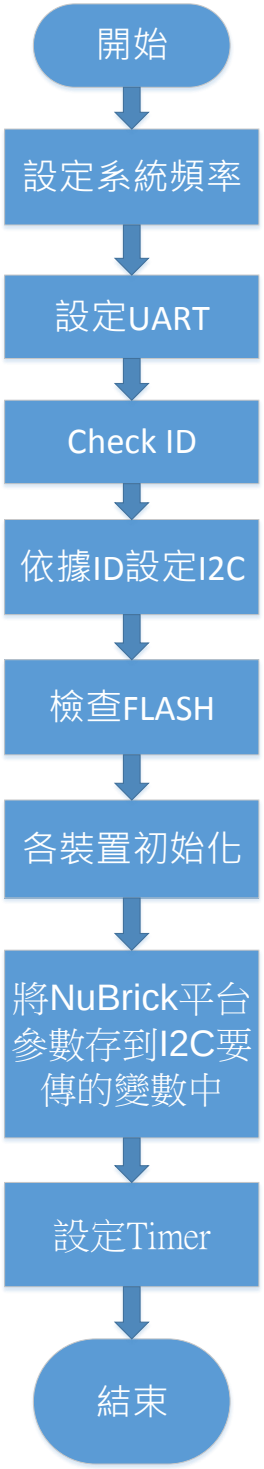

圖 2-3 NuMaker Brick 平台初始化流程圖

setup() 流程說明如下，程式內容如圖 2-4

1. 設定系統頻率 72 MHz
2. 設定 UART baud rate 115200 bps
3. 檢查 ID：

   辨認該模組是主控端或哪個感測模組
4. 依據 ID 設定 I^2C：

   根據檢查 ID 時的結果作 I^2C 的設定
5. FLASH 初始化
6. 各裝置初始化，在這裡分成四步驟：
   A. 判斷裝置 ID 是否符合，若符合則執行初始化
   B. 執行感測模組的初始化
   C. 執行 FLASH 的資料檢查，根據檢查 ID 的結果，檢查 FLASH 內是否已經存有對應的資料
   D. 設定週期，依據不同模組的資料設定之後執行的週期迴圈時間
7. 將模組的資料存到 I^2C 要傳的變數中：
   A. 主控端準備下一次對感測模組要傳的資料
   B. 感測模組準備下一次主控端要接收的資料

```
void setup()
{
        int i;

        setupSystemClock();                                   1.設定系統頻率
        setupUART();                                          2.設定 UART
        setup_system_tick(SYSTEM_TICK_FREQ);
        ID_Init();                                            3.檢查ID
        GetID();

        /* Init AHRS I2C */                                   4.依據ID設定I2C
        /* ----Initialize I2C slave mode--- */
        if(devNum == 0)
                I2C_MS_Master_Init();
        else
                I2C_MS_Slave_Init();

        TIDMstFirstInitFIN = 0;
        TIDMstInitFIN = 0;
        TIDMstStage = 0;
        TIDMstInitDevState = 1;
        FlashInit();                                          5.FLASH初始化

        // Setup function pointers for default sensor boards.  6.各裝置初始化
        for(i=0; i<MAX_TID_DEV; i++)
        {
                if (pTidList[i])
                SetDeviceFunction(pTidList[i], &(pfnDevFunc[i]));
        }


        // Call board init function
        if (pTidList[devNum]->func.pfnSetup)
                pTidList[devNum]->func.pfnSetup();

        // Load TID feature from flash
        GetFlashTID(&(pTidList[devNum]->Feature), devNum);

        // Set Timer1 to (1000/value) in (Hz)
        Timer1Init(1000 / (pTidList[devNum]->Feature.data1.value));

        /* TID initialize */                                  7.將模組資料存到I2C要傳的變數中
        SlvDataInit();
        MstDataInit();
}
```

圖 2-4 setup 程式說明圖

2.2.1. 感測模組初始化的內容

最後必須要針對感測模組初始化的程式內容作介紹，如下面程式碼以 Battery_init() 當作範例。在初始化的程式中，第一部分是原本感測模組的初始化，接著是感測模組與主控端要傳遞的資料設定。資料設定的方式如下。

1. 首先訂定需要傳遞的資料，光敏感測模組資料如下
   A. 睡眠周期
   B. 警報觸發數值
   C. 光敏電壓
   D. 警報

2. 狀態分類：特性、輸出、輸入
   A. 特性：感測模組的一些設定，不會常常需要從主控端傳遞或接收的資料。
   B. 輸出：主控端要傳遞給感測模組的資料。
   C. 輸入：主控端需要從感測模組接收的資料。
      以電池為例，它的特性有兩筆：睡眠周期、警報觸發數值。
      輸入有兩筆：電池電量、警報。

3. 設定每一筆資料的內容，每一筆資料內包含：最大值、最小值、位置、長度、資料內容。
   A. 最大值及最小值 (maximum, minimum)：下圖棕色框內每一筆資料皆有，依資料需求設定。
   B. 位置 (arg)：表示資料在該分類的位置，位置按順序遞增。
   C. 長度 (datalen)：表示該筆資料的長度，每 8 bits 為一單位。
   D. 資料內容：即為該筆資料的數值。
   E. 資料數量(dataNum)：每一種類的資料最後皆會有一格保留給該種類資料的數量，例如輸入資料有兩筆，因此數值為 2。

```
void Battery_Init()
{
    SYS_UnlockReg();
    /* Enable EADC module clock */
    CLK_EnableModuleClock(EADC_MODULE);
    /* EADC clock source is 72MHz, set divider to 8, ADC clock is 72/8 MHz */
    CLK_SetModuleClock(EADC_MODULE, 0, CLK_CLKDIV0_EADC(8));
    SYS_LockReg();
    /* Configure the GPB0 - GPB3 ADC analog input pins.  */
    SYS->GPB_MFPL &= ~SYS_GPB_MFPL_PB1MFP_Msk;
    SYS->GPB_MFPL |= SYS_GPB_MFPL_PB1MFP_EADC_CH1;

    GPIO_DISABLE_DIGITAL_PATH(PB, BIT1);

    /* Set the ADC internal sampling time, input mode as single-end and enable the A/D converter */
    EADC_Open(EADC, EADC_CTL_DIFFEN_SINGLE_END);
    EADC_SetInternalSampleTime(EADC, 6);

    /* Configure the sample module 0 for analog input channel 1 and software trigger source.*/
    EADC_ConfigSampleModule(EADC, 1, EADC_SOFTWARE_TRIGGER, 1);

    /* Clear the A/D ADINT0 interrupt flag for safe */
    EADC_CLR_INT_FLAG(EADC, 0x2);

    /* Enable the sample module 0 interrupt.  */
    EADC_ENABLE_INT(EADC, 0x2);//Enable sample module A/D ADINT0 interrupt.
    EADC_ENABLE_SAMPLE_MODULE_INT(EADC, 1, 0x2);//Enable sample module 0 interrupt.

    BatDev.DevDesc.DevDesc_leng = 26;                    //Report descriptor
    BatDev.DevDesc.RptDesc_leng = 36;                    //Report descriptor
    BatDev.DevDesc.InRptLeng = 5;                        //Input report
    BatDev.DevDesc.OutRptLeng = 0;                       //Output report
    BatDev.DevDesc.GetFeatLeng = 6;                      //Get feature
    BatDev.DevDesc.SetFeatLeng = 6;                      //Set feature
    BatDev.DevDesc.CID = 0;
    //manufacturers ID
    BatDev.DevDesc.DID = 0;
    //Product ID
    BatDev.DevDesc.PID = 0;
    //Device firmware revision
    BatDev.DevDesc.UID = 0;
    //Device Class type
    BatDev.DevDesc.UCID = 0;                             //reserve
    /* Feature */
    BatDev.Feature.data1.minimum = 0;
    BatDev.Feature.data1.maximum = 1024;
    BatDev.Feature.data1.value = 100;                    //Sleep period
    BatDev.Feature.data2.minimum = 0;
    BatDev.Feature.data2.maximum = 100;
    BatDev.Feature.data2.value = 50;                     //Battery alerm value
    BatDev.Feature.arg[0] = 1;
    BatDev.Feature.arg[1] = 2;
    BatDev.Feature.datalen[0] = 2;
    BatDev.Feature.datalen[1] = 2;
    BatDev.Feature.dataNum = 2;
    /* Input */
    BatDev.Input.data1.minimum = 0;
    BatDev.Input.data1.maximum = 100;
    BatDev.Input.data1.value = 100;                      //Battery value
    BatDev.Input.data2.minimum = 0;
    BatDev.Input.data2.maximum = 1;
    BatDev.Input.data2.value = 0;                        //Over flag
    BatDev.Input.arg[0] = 1;
    BatDev.Input.arg[1] = 2;
    BatDev.Input.datalen[0] = 2;
    BatDev.Input.datalen[1] = 1;
    BatDev.Input.dataNum = 2;
    /* Output */
    BatDev.Output.dataNum = 0;
}
```

原本感測模組的初始化

平台參數的初始化

資料格式

睡眠周期

特性

警報觸發數值

資料位置

資料長度

資料數量

電池電量

輸入

警報

資料位置

資料長度

資料數量

資料數量

圖 2-5 感測模組初始化程式

新的感測模組的初始化即完成。

2.3. loop()

接著是 loop()，無論是主控及感測模組的主要程式皆放置於此處。在執行階段需要將整個流程分成兩部分，第一部分是感測模組的工作，第二部分則是主控端的反應。

首先將 loop 分成兩個區塊，週期區塊及非週期區塊。週期區塊是每隔設定的時間執行一次，例如振動感測模組每隔 0.1 秒更新一次資料，或主控端每 0.1 秒準備與感測模組通訊等。非週期區塊則是程式會一直進入並執行的區域，通常用來做檢查用以及即時的反應用，例如聲納持續接收回傳的距離資訊、或 LED 燈檢查是否需要閃爍等等。圖 2-6 分別代表 loop 的主控端及感測模組流程圖

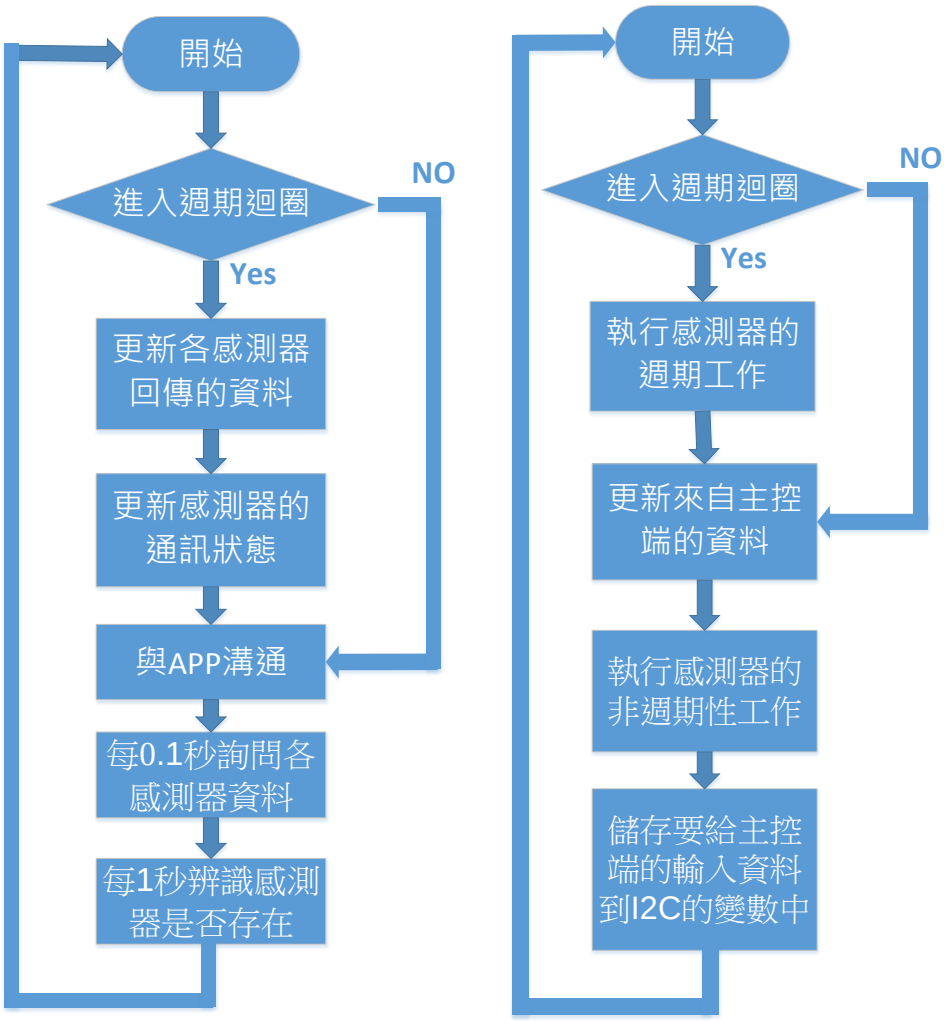

圖 2-6 主控端以及感測模組 loop 程式流程圖

2.3.1. 持續執行的程式內容

以下先依照整個程式的流程做介紹，如圖 2-7

週期執行區塊 (在 if(TMR1INTCount != RecentTimeCounter) 內)

1. 主控端
   A. 與APP間溝通
   B. 更新各感測模組的輸出值
   C. 主控端更新模組間通訊設定 (使用 I^2C)
      主控端每隔一段時間便會與各感測模組通訊，接收最新的資料。
2. 感測模組：
   A. 執行感測模組的週期工作

即時執行區塊(在 if(TMR1INTCount != RecentTimeCounter) 外)：

3. 主控端
   A. 每 0.1 秒由 I^2C 詢問各感測模組資料，包括傳遞特性 (Set Feature)、接收特性 (Get Feature)、傳遞輸出 (Set output) 及接收輸入 (Get input)
   B. 每1秒由主控端 I^2C 重頭詢問辨識感測模組端是否存在，包括接收感測模組描述 (Device Descriptor) 以及回報描述 (Report Descriptor)
4. 感測模組：
   A. 更新來自主控端的輸出訊號
   B. 執行感測模組的非週期工作
   C. 儲存要給主控端的輸入資料到 I^2C 的變數中

```
void loop()
{
    CommandProcess();  1.A 與APP間溝通
    TaskScheduler();
    PowerControl();

    // ********************************************************
    //                Get data for once erery 0.1s
    // ********************************************************
    if(TMR1INTCount != RecentTimeCounter)
    {
        MainTimeMSCounter = getTickCount();

        if (pTidList[devNum]->func.pfnPeriod)
            pTidList[devNum]->func.pfnPeriod();   1.B 更新各感測模組的輸出值
                                                  1.C 主控端更新模組間通訊設定
        RecentTimeCounter = TMR1INTCount;         2.A 執行感測模組的週期工作
    }

    // =============TID Master handle received data===============
    if(devNum == 0)
    {
        if(TIDMstInitFIN==0)
        {
            TIDMst_GetDev();   3-A、主控端每0.1秒詢問各感測模組資料
        }
        else if(TIDMstInitFIN==1)
        {
            TIDMstDevTRx();    3-B、主控端每1秒重頭詢問感測模組的狀態
        }
    }
    // ====================  SLAVE PART  ======================
    // -----------TID slave handle received data--------------
    else
    {
        if(I2CMS_SlvRxFin==1)
        TID_SlvRxUpdate();   4.A 更新來自主控端的輸出訊號
    }

    // -----------Slave part excute subfunction--------------

    if (pTidList[devNum]->func.pfnPulling)
        pTidList[devNum]->func.pfnPulling();   3.B 執行感測模組的非週期工作

    // --------------TID Slave Store data-----------------
    if(devNum != 0)
    SlvDataStore();   4.C 儲存要給主控端的輸入資料到 I2C 的變數中
}
```

圖 2-7 loop 程式說明

2.3.2. 持續執行的程式內容細節

在這一節將從不同模組的角度來看程式的細節，分成主控端以及感測模組。首先介紹感測模組，NuMaker Brick 平台在執行階段提供兩種不同執行的程序，以滿足不同狀況的需求，

- 第一種是週期執行的功能，將程式放在新增的 ID 區域便會週期的執行程式，如圖 2-8

```
// ********************************************************
//                Get data for once erery 0.1s
// ********************************************************
if(TMR1INTCount != RecentTimeCounter)
{
    MainTimeMSCounter = getTickCount();

    if (pTidList[devNum]->func.pfnPeriod)
        pTidList[devNum]->func.pfnPeriod();

    RecentTimeCounter = TMR1INTCount;
}
```

2.A 執行感測模組的週期工作

圖 2-8 loop 感測模組的週期性工作程式說明

此程式對應到本章開始所所說明的模組化程式位置如下，如圖 2-9

```
APFN_FUNC_T pfnDevFunc[MAX_TID_DEV] =
{
    /*
       Functions for each device:

        {Initial, Period, Pulling, Report}

        Initial : Init and configure device
        Period  : Process device sensor data or set sensor once per 0.1 seconds.
        Pulling : Process device sensor data or set sensor frequently.
        Report  : Report process data to master
    */
                        感測模組的週期工作
    /* for embedded device boards */

    {Battery_Init,      MasterControl,  NULL,               report_battery},
    {Buzzer_Init,       NULL,           Buzzer_Control,     report_buzzer},
    {Led_Init,          NULL,           Led_Control,        report_led},
    {AHRS_Init,         AHRS_Control,   NULL,               report_ahrs},
    {SonarInit,         SonarDetect,    SonarTimeOutCheck,  report_sonar},
    {Init_DHT11_PWM0,   DHT11Control,   DHT11GetDATA,       report_temp},
    {Gas_Init,          GetGas,         NULL,               report_gas},
    {IR_Init,           IR_Control,     IR_Check,           report_ir},
    {key_init,          NULL,           NULL,               report_key},

    /* for custom device boards
       fill your four board functions here to specified ID. */

    {Lr_Init, GetLr, NULL, NULL},   // For device ID 9
    {NULL, NULL, NULL, NULL},   // For device ID 10
    {NULL, NULL, NULL, NULL},   // For device ID 11
    {NULL, NULL, NULL, NULL},   // For device ID 12
    {NULL, NULL, NULL, NULL},   // For device ID 13
    {NULL, NULL, NULL, NULL}    // For device ID 14
};
```

圖 2-9 感測模組的週期工作模組化區塊

以下舉振動感測模組為例子

```
/* 執行階段的程式 */
Void AHRS_Control()
{
        /* 將程式至於此處 */
        /* 以下是振動感測模組的範例 */
        else if(devNum == 3)
        {
        SensorsRead(SENSOR_ACC|SENSOR_GYRO,1);
        nvtUpdateAHRS(SENSOR_ACC|SENSOR_GYRO);
        AhrsRead(AHRSDev.Feature.data2.value, AHRSDev.Feature.data3.value, AHRSDev.Output.data1.value);

        }
}
```

- 第二部分則是提供給需要一直檢查或執行的功能，只需要放在

  if(TMR1INTCount != RecentTimeCounter){} 外面即可。如圖 2-10

```
// -----------Slave part excute subfunction--------------
if (pTidList[devNum]->func.pfnPulling)
    pTidList[devNum]->func.pfnPulling();
```

3.B 執行感測模組的非週期工作

圖 2-10 loop 感測模組的即時工作程式說明

此程式對應到本章開始所所說明的模組化程式位置如下，如圖 2-11

```
APFN_FUNC_T pfnDevFunc[MAX_TID_DEV] =
{
    /*
       Functions for each device:

        {Initial, Period, Pulling, Report}

        Initial : Init and configure device
        Period  : Process device sensor data or set sensor once per 0.1 seconds.
        Pulling : Process device sensor data or set sensor frequently.
        Report  : Report process data to master
    */

    /* for embedded device boards */

    {Battery_Init,      MasterControl,  NULL,               report_battery},
    {Buzzer_Init,       NULL,           Buzzer_Control,     report_buzzer},
    {Led_Init,          NULL,           Led_Control,        report_led},
    {AHRS_Init,         AHRS_Control,   NULL,               report_ahrs},
    {SonarInit,         SonarDetect,    SonarTimeOutCheck,  report_sonar},
    {Init_DHT11_PWM0,   DHT11Control,   DHT11GetDATA,       report_temp},
    {Gas_Init,          GetGas,         NULL,               report_gas},
    {IR_Init,           IR_Control,     IR_Check,           report_ir},
    {key_init,          NULL,           NULL,               report_key},

    /* for custom device boards
       fill your four board functions here to specified ID. */

    {Lr_Init, GetLr, NULL, NULL},   // For device ID 9
    {NULL, NULL, NULL, NULL},   // For device ID 10
    {NULL, NULL, NULL, NULL},   // For device ID 11
    {NULL, NULL, NULL, NULL},   // For device ID 12
    {NULL, NULL, NULL, NULL},   // For device ID 13
    {NULL, NULL, NULL, NULL}    // For device ID 14
};
```

感測模組的即時工作

圖 2-11 感測模組的即時工作模組化區塊

- 設定完感測模組之後，對主控端進行設定。主控端要做的事情非常的單純，目標是要讓主控端能對新增的感測模組做反應，包含設定新增裝置能觸發的對象，以及能夠觸發新裝置的對象。修改

在 TIDMstUpdateDevState()中，在 main.c 中可以找到 pfnDevFunc 的宣告中有 MasterControl 的程式，TIDMstUpdateDevState()包含在 MasterControl 之中，位置如圖 2-12。

```
APFN_FUNC_T pfnDevFunc[MAX_TID_DEV] =
{
    /*
       Functions for each device:

        {Initial, Period, Pulling, Report}

        Initial : Init and configure device
        Period  : Process device sensor data or set sensor once per 0.1 seconds.
        Pulling : Process device sensor data or set sensor frequently.
        Report  : Report process data to master
    */

    /* for embedded device boards */

    {Battery_Init,      MasterControl,  NULL,               report_battery},
    {Buzzer_Init,       NULL,           Buzzer_Control,     report_buzzer},
    {Led_Init,          NULL,           Led_Control,        report_led},
    {AHRS_Init,         AHRS_Control,   NULL,               report_ahrs},
    {SonarInit,         SonarDetect,    SonarTimeOutCheck,  report_sonar},
    {Init_DHT11_PWM0,   DHT11Control,   DHT11GetDATA,       report_temp},
    {Gas_Init,          GetGas,         NULL,               report_gas},
    {IR_Init,           IR_Control,     IR_Check,           report_ir},
    {key_init,          NULL,           NULL,               report_key},

    /* for custom device boards
       fill your four board functions here to specified ID. */

    {Lr_Init, GetLr, NULL, NULL},   // For device ID 9
    {NULL, NULL, NULL, NULL},   // For device ID 10
    {NULL, NULL, NULL, NULL},   // For device ID 11
    {NULL, NULL, NULL, NULL},   // For device ID 12
    {NULL, NULL, NULL, NULL},   // For device ID 13
    {NULL, NULL, NULL, NULL}    // For device ID 14
};
```

```
void MasterControl(void)
{
    /* Master Store data */
    TIDMstUpdateDevState();
    /* Master recheck device */
    if(TIDMstFirstInitFIN==1)
    {
        if(TMR1TimerCounter > 10)
        {
            if(I2CMstEndFlag==1)
            {
                TIDMstInitFIN=0;
                TMR1TimerCounter=0;
                I2C_Close(I2C_MS_PORT);
                I2C_MS_Master_Restart();
            }
        }
        else
        {
            TIDMstInitFIN = 1;
            TMR1TimerCounter++;
        }
    }
    GetBattery();
}
```

圖 2-12 主控端對感測模組的回饋設定

TIDMstUpdateDevState()範例程式如下：

```
void TIDMstUpdateDevState()
{
        /* 振動感測模組回傳的資料為1 */
        if((AHRSDev.Input.data2.value==1) && (TIDMstInitDevState & (1<<AHRSDEV)))
        {
                /* 揚聲器與振動感測模組有關連 */
                if(BuzDev.dTod.dTHR==1)
                {
                        /* 主控端要揚聲器開始響鈴 */
                        BuzDev.Output.data1.value = 1;                  //Song
                        TIDMstDevState[BUZZERDEV] |= 2;      //buzzer operate：output(2)
                }
        }
}
```

以振動感測模組當例子，在這個程式中所使用的功能如下：

假設振動感測模組回傳的資料為 1：if(AHRSDev.Input.data2.value==1)

揚聲器與振動感測模組有關連：if (BuzDev.dTod.dTHR==1)

主控端要揚聲器開始響鈴：BuzDev.Output.data1.value = 1

2.4. 程式檔案介紹

NuMaker Brick 程式可以分成兩個部分，第一個部分是 NuMaker Brick 平台的架構以及與手機溝通的通訊協定，如表 2-1 NuMaker Brick 平台架構。第二部分則是各個感測器的副程式，表 2-2 。

	程式名稱	程式內容
I²C	i2c_ms	平台的 I²C 通訊設定
	i2cdev	I²C 通訊的子程式
平台參數	tid	平台參數的設定
	tidmst	主控端的平台參數儲存方式
	tiddev	模組的平台參數儲存方式
ID check	devCheck	模組的 ID check
APP 通訊	report	與 APP 間通訊
	report_AP	與 APP 通訊子程式

表 2-1 NuMaker Brick 平台架構

	程式名稱	程式內容
電池感測模組	battery	使用 ADC 偵測電池
LED 模組	led	使用 PWM 驅動 LED
揚聲器模組	buzzer	使用 PWM 驅動揚聲器
	music	音樂的 table
瓦斯感測模組	gas	使用 ADC 偵測瓦斯
紅外線感測模組	Ir	用 PWM 驅動發射紅外線及使用 capture 接收紅外線
	PWM0P2	紅外線的 PWM 設定及中斷執行
聲納感測模組	sonar	使用 PWM capture 接收聲納訊號
溫度感測模組	temp_HTU21D	用 I²C 通訊接收溫度與濕度訊號
	i2c_HTU21D	溫濕度 IC 的 I²C 通訊設定
振動感測模組	AHRS	用 I²C 通訊接收振動訊號
	calibrate	校正振動感測器
	mpu6050	振動感測器的 I²C 通訊設定
	sensors	振動感測器的資料整理
按鈕模組	KEY	用 GPIO 接收按鈕狀態
	GPC_IRQ	使用中斷接收按鈕狀態
	GPE_IRQ	使用中斷接收按鈕狀態
週期設定	timerIRQ1	設定個模組的週期執行程式的頻率

表 2-2 各模組程式

3. NUMAKER BRICK 擴展平台通訊層

在上一章概述了程式的功能，這一章將討論 NuMaker Brick 平台如何實現這樣的架構。此平台分成三部分：

1. 主控端：主控端的功能是將各感測模組回傳的資料做整理並對其他感測模組端發出訊號。
2. 感應模組端：市面上常用的感測模組。
3. 通訊層：通訊層分散在主控端及感測模組中，為了完成之前所提到的兩個目標而建立，設備的擴充性與資料的一致性。因此在這裡對市面上常用的感測模組資料做一個規範，針對各種不同的感測模組種類做規劃，使得舊有的感測模組也能經過一次性的擴充便可加入整個系統中，讓主控端能對其蒐集資訊及發出命令。

整個平台的工作理念是主控端對感測模組端蒐集資訊，做出決定後再對其他的感測模組發出命令。因此最重要的是讓主控端認識感測模組所發出的資訊，即使是舊的感測模組經過一次性的更新之後也可以被主控端認識，及感測模組可以接收主控端發出的命令。

在這一章將介紹整個擴展平台的資料傳遞。第一節將說明整個平台如何完成前面所說兩個問題：

1. 資料的分類：對每筆資料的設定一致的格式 (包含最大值、最小值、及目前數值)。
2. 模組的擴充性：對每個感測模組的資料都區分成三類 (輸入、輸出、特性)，而每一類的資料還有記錄每筆資料的長度 (datalen) 及順序 (arg)。

第二節將說明主控端是如何運作及傳送的資料格式，最後一節將介紹每個感測模組所要傳遞的資訊內容。

NuMaker Brick 擴展平台

整個平台的目標是由主控端跟感測模組蒐集資訊，再對其他感測模組發出命令，而主控端跟感測模組的通訊方式是由 I^2C 傳遞資訊，將傳遞資料的過程定義為通訊層，此定義的目的是為了要將整個通訊過程與原本感測模組就有的程式做區分。通訊層對於感測模組的每筆資料做規範，每個感測模組內資料的分類，如表 3-1 感測模組資訊分類

	資料分類	資料內容
感測模組	特性	最大值
		最小值
		長度
		順序
		數值內容
	輸出	最大值

		最小值
		長度
		順序
		數值內容
	輸入	最大值
		最小值
		長度
		順序
		數值內容

表 3-1 感測模組資訊分類

每筆資料皆有以下的資料：

1. 資料的範圍：包含資料的最大值及最小值
2. 資料的長度：資料在程式內的長度
3. 資料的順序：該筆資料在類別中位置
4. 資料的內容：該筆資料的數值

通訊層也有對感測模組的整體資料進行分類，此分類使得溝通時可以避免傳遞不需要的資料，定義資料是單向的 （由主控端傳給感測模組或感測模組傳給主控端） 還是雙向的 （感應端與主控端可以互傳），分類如下：

1. 輸入：感測模組端的資訊，用來回傳給主控端。如偵測到的距離、感測到的振動值等。
2. 輸出：主控端發送給感測模組端的命令。如命令揚聲器發出聲響的資料、命令 LED 開始閃爍的資料等。
3. 特性：感測模組所特有的資料。如揚聲器的頻率、聲納發出警示的距離等。

此分類不管對於主控端或感測模組皆是一樣的描述。不會在感測模組端就由輸入變成輸出。

3.1. NuMaker Brick 通訊流程

了解通訊層的規範後回到整個通訊流程，溝通的流程是整個系統最重要的部分，可以分類如下：

1. 甚麼感測器 (第一階段)：此階段感測器會回覆基本資訊給主控端。
2. 有甚麼資料 (第二階段)：此階段感測器會回覆資料格式給主控端。
3. 傳遞資料 (第三階段)：此階段可以互相傳遞資料內容，分成四種：接收輸入，傳送輸出、接收特性、傳送特性。

整個通訊流程可表示如圖 3-1 所示。

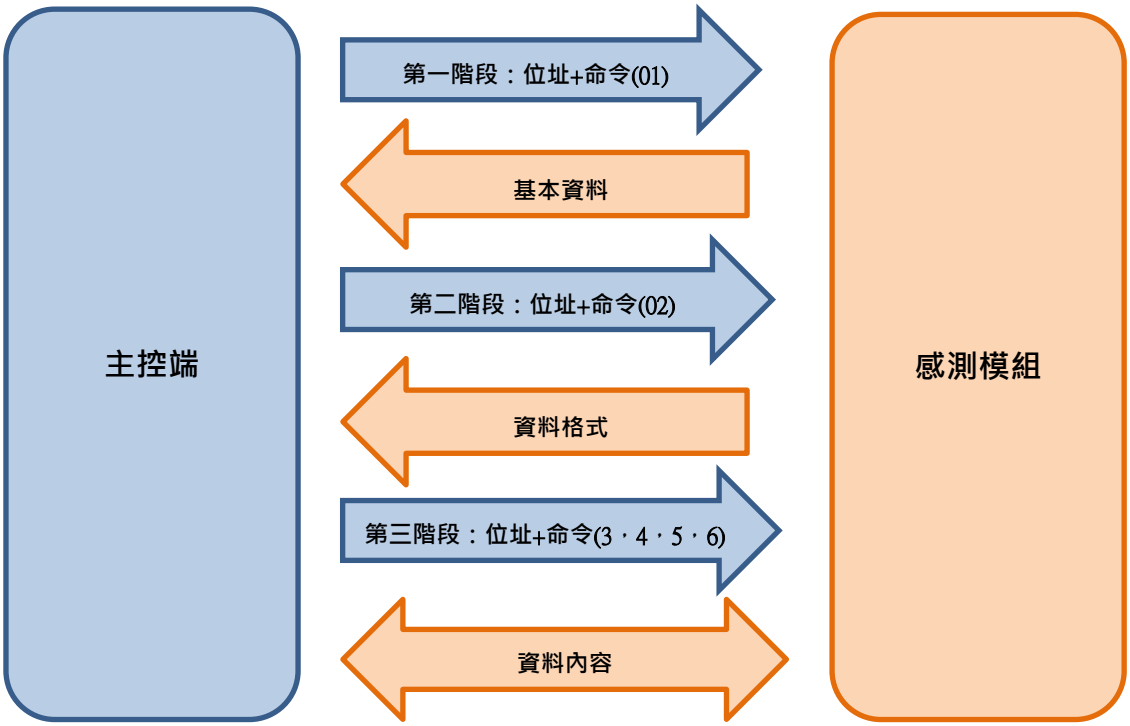

圖 3-1 主控端與感測模組通訊流程

整個系統開始運作時必須要由主控端辨認出感測模組，才可以開始傳輸或接收資料，而主控端對感測模組的溝通方法如下：

1. 主控端必須辨認連接上何種感測模組：經由 I^2C 對設定的位址做呼叫，若有回應則知道何種感測模組有連接上。

2. 主控端命令感測模組回應基本資訊 (第一階段)，如資料長度、ID 等。

基本資訊 (Device Descriptor)：命令為 01

Start	Address	W	ACK	Command(01)	ACK	Command (00)		ACK	Sr
	Address	R	ACK	TID Descriptor	ACK	. . .	TID Descriptor	NACK	

Note. 參考表 3-2 感測模組基本資訊

3. 主控端命令感測模組回應資料格式 (第二階段)，如是否有該筆資料、單一筆資料的長度以及單一筆資料的數值範圍。

資料格式 (Report Descriptor)： 命令為 02

Start	Address	W	ACK	Command(02)	ACK	Command (00)		ACK	Sr
	Address	R	ACK	Report Descriptor	ACK	. . .	Report Descriptor	NACK	

Note. 參考表 3-3 回傳資料格式

4. 開始傳送或接收資料內容(第三階段)，分成四種：接收輸入，傳送輸出、接收特性、傳送特性。

- 回報輸入 (Get Input) ： 命令為 03

Start	Address	W	ACK	Command(03)	ACK	Command (00)		ACK	Sr
	Address	R	Data (length field,LSB)		ACK	Data (length field,MSB)		ACK	
			Data (Report Field,LSB)		ACK	. . .	Data (Report Field,MSB)	NACK	

Note. 參考表 3-4 確定裝置時所需要的資料

- 設定輸出 (Set Output) ： 命令為 04

Start	Address	W	ACK	Command(04)	ACK	Command (00)		ACK
			Data (length field,LSB)		ACK	Data (length field,MSB)		ACK
			Data (Report Field,LSB)		ACK	. . .	Data (Report Field,MSB)	ACK	P

Note. 參考表 3-4 確定裝置時所需要的資料

- 回報特性 (Get Feature) ： 命令為 05

Start	Address	W	ACK	Command(05)	ACK	Command (00)		ACK	Sr
	Address	R	Data (length field,LSB)		ACK	Data (length field,MSB)		ACK	
			Data (Report Field,LSB)		ACK	. . .	Data (Report Field,MSB)	NACK	

Note. 參考表 3-4 確定裝置時所需要的資料

- 設定特性 (Set Feature) ： 命令為 06

Start	Address	W	ACK	Command(06)	ACK	Command (00)		ACK
			Data (length field,LSB)		ACK	Data (length field,MSB)		ACK
			Data (Report Field,LSB)		ACK	. . .	Data (Report Field,MSB)	ACK	P

Note. 參考表 3-4 確定裝置時所需要的資料

通訊過程分成三階段的原因是由於每個資料皆保持了最大的彈性，可以任意設定單筆資料的長度、範圍、及是否有此筆資料。

了解了通訊層所規定的格式及主控端便是感測模組的裝置大致流程後，下一節將深入了解主控端辨識及收發資料給感測模組端的流程。

3.2. 主控端所需要的資料

整個平台最重要的腳色是主控端，由主控端辨識裝置、接收感測模組資料後設定資料格式、傳遞或接收資料內容。資料的傳遞是由 I^2C 做通訊的介面，可以分成六種格式：

3.2.1. 基本資訊

主控端命令感測模組回傳基本資訊(第一階段)，在此階段每筆資料長度皆為 2 bytes，此時主控器接收感測模組的基本設定，如廠商 ID、版本號，其中感測模組的輸入資料長度、輸出資料長度、特性資料長度是用來設定接下來的傳輸資料格式階段。此時要傳的資料內容如下表 3-2 感測模組基本資訊。

資料名稱	資料內容	資料長度	功能說明
開始旗標	257 (1, 1)	2 位元組	
基本資料長度	26	2 位元組	第一階段 (感測模組基本資訊) 的資料長度，是 I^2C 傳值時所需要的設定
資料格式長度	變動	2 位元組	第二階段 (感測模組資料格式) 的資料長度
回報輸入長度	變動	2 位元組	第三階段感測模組回報輸入的資料長度
設定輸出長度	變動	2 位元組	第三階段主控端命令輸出的資料長度
回報特性長度	變動	2 位元組	第三階段感測模組回報特性的資料長度
設定特性長度	變動	2 位元組	第三階段主控端設定感測模組特性的資料長度
公司代號	變動	2 位元組	出產感測模組的公司代號
設備代號	變動	2 位元組	感測模組的設備代號
產品代號	變動	2 位元組	感測模組的產品代號
韌體版本	變動	2 位元組	感測模組的韌體版本
設備類別代號	變動	2 位元組	感測模組的設備類別代號
保留位	變動	2 位元組	
保留位	變動	2 位元組	

表 3-2 感測模組基本資訊

3.2.2. 資料格式

主控端命令感測模組回傳資料格式 (第二階段)，在確認裝置的基本資料及每個分類的資料長度後便可以開始確認單筆資料的長度及範圍。此階段先傳整筆資料的長度。接著先傳類型為特性的資料，先傳送 257 代表接下來的資料類型為特性。之後開始傳單筆資料，單筆資料的資料內容首先傳資料的位址(第一筆：5， 第二筆：17， 第三筆：29， 第四筆：41， 第五筆：53， 第六筆：65， 第七筆：77， 第八筆：89， 第九筆：101， 第十筆：113)，再來傳資料的長度(通常是 1~2 位元組)，接著再宣告下一筆資料是資料的最小值及最小值數值的長度 (8+1 或 8+2)，再來是最小值數值，接著再宣告下一筆資料是資料的

最大值及最大值數值的長度 (10+1 或 10+2)，再來是最大值數值。以上是單筆資料的內容，傳完之後可以傳下一筆資料。傳完特性的資料後，開始傳輸入的資料，先傳送 258 代表接下來的資料類型為特性。之後再傳輸入的資料，單筆資料傳的格式同上面的格式。傳完輸入的資料後再傳輸出的資料，單筆資料傳的格式同上面的格式。

資料名稱	資料內容	資料長度	功能說明
資料格式長度	變動		這一筆資料的長度， 是 I^2C 傳值時所需要的設定
資料類別 (特性)	257 (1, 1)	2 位元組	宣告以下資料的資料類別：特性
資料位址	5	1 位元組	宣告這筆資料的位址 (5, 17, 29, 41, 53, 65, 77, 89, 101, 113)
資料長度	1~2	1 位元組	宣告這筆資料的長度(1位元組~2位元組)
通知下一筆是資料的最小值及最小值資料長度	9 or 10	1 位元組	宣告下一欄是資料的最小值 (8) 還有下一欄的長度 (1or2)
資料的最小值	變動	1~2位元組	資料的最小值
通知下一筆是資料的最大值及最大值資料長度	13 or 14	1 位元組	宣告下一欄是資料的最小值 (12) 還有下一欄的長度 (1or2)
資料的最大值	變動	1~2位元組	資料的最大值
資料位址	17	1 位元組	宣告這筆資料的位址 (5, 17, 29, 41, 53, 65, 77, 89, 101, 113)
資料長度	1~2	1 位元組	宣告這筆資料的長度(1位元組~2位元組)
通知下一筆是資料的最小值及最小值資料長度	9 or 10	1 位元組	宣告下一欄是資料的最小值 (8) 還有下一欄的長度 (1or2)
資料的最小值	變動	1~2位元組	資料的最小值
通知下一筆是資料的最大值及最大值資料長度	13 or 14	1 位元組	宣告下一欄是資料的最小值 (12) 還有下一欄的長度 (1or2)
資料的最大值	變動	1~2位元組	資料的最大值
…	…	…	…
資料類別 (輸入)	258(1, 2)		宣告以下資料的資料類別：輸入
資料位址	5	1 位元組	宣告這筆資料的位址 (5, 17, 29, 41, 53, 65, 77, 89, 101, 113)
資料長度	1~2	1 位元組	宣告這筆資料的長度 (1位元組~2位元組)
通知下一筆是資料的最小值及最小值資料長度	9 or 10	1 位元組	宣告下一欄是資料的最小值 (8) 還有下一欄的長度 (1or2)
資料的最小值	變動	1~2位元組	資料的最小值
通知下一筆是資料的最大值及最大值資料長度	13 or 14	1 位元組	宣告下一欄是資料的最小值 (12) 還有下一欄的長度 (1or2)
資料的最大值	變動	1~2位元組	資料的最大值
資料位址	17	1 位元組	宣告這筆資料的位址 (5, 17, 29, 41, 53, 65, 77, 89, 101, 113)
資料長度	1~2	1 位元組	宣告這筆資料的長度(1位元組~2位元組)
通知下一筆是資料的最小值及最小值資料長度	9 or 10	1 位元組	宣告下一欄是資料的最小值 (8) 還有下一欄的長度 (1or2)
資料的最小值	變動	1~2位元組	資料的最小值

通知下一筆是資料的最大值及最大值資料長度	13 or 14	1 位元組	宣告下一欄是資料的最小值 (12) 還有下一欄的長度 (1or2)
資料的最大值	變動	1~2位元組	資料的最大值
…	…	…	…
資料類別 (輸出)	259 (1, 3)		宣告以下資料的資料類別 (輸出)
資料位址	5	1 位元組	宣告這筆資料的位址 (5, 17, 29, 41, 53, 65, 77, 89, 101, 113)
資料長度	1~2	1 位元組	宣告這筆資料的長度(1位元組~2位元組)
通知下一筆是資料的最小值及最小值資料長度	9 or 10	1 位元組	宣告下一欄是資料的最小值 (8) 還有下一欄的長度 (1or2)
資料的最小值	變動	1~2位元組	資料的最小值
通知下一筆是資料的最大值及最大值資料長度	13 or 14	1 位元組	宣告下一欄是資料的最小值 (12) 還有下一欄的長度 (1or2)
資料的最大值	變動	1~2位元組	資料的最大值
資料位址	17	1 位元組	宣告這筆資料的位址 (5, 17, 29, 41, 53, 65, 77, 89, 101, 113)
資料長度	1~2	1 位元組	宣告這筆資料的長度(1位元組~2位元組)
通知下一筆是資料的最小值及最小值資料長度	9 or 10	1 位元組	宣告下一欄是資料的最小值 (8) 還有下一欄的長度 (1or2)
資料的最小值	變動	1~2位元組	資料的最小值
通知下一筆是資料的最大值及最大值資料長度	13 or 14	1 位元組	宣告下一欄是資料的最小值 (12) 還有下一欄的長度 (1or2)
資料的最大值	變動	1~2位元組	資料的最大值
…	…	…	…

表 3-3 回傳資料格式

此時主控端必須針對這些資料做以下的整理：

在收到資料位址後，將資料長度儲存在對應類別的 datalen 中，並對 arg 寫入目前數值 (如收到 29 的位址，長度為 2，則對第三筆資料的 datalen 寫入 2，arg 寫入 3)，以利於之後收資料數值時判斷用。

3.2.3. 資料內容

傳送或接收資料內容：確定每筆資料的長度及範圍後便可以做傳收值的動作，此時有四種選擇，對應不同的資料類別 (輸入、輸出、特性)

- 傳送輸出命令感測模組動作
- 讀取輸入接收感測模組資料
- 傳送特性修改感測模組特性
- 讀取特性接收感測模組特性

資料名稱	資料內容	資料長度	功能說明
回報輸入長度 **(命令輸出長度)** **(回報特性長度)** **(設定特性長度)**	變動	2 位元組	這一筆資料的長度， 是 I^2C 傳值時所需要的設定
第一筆資料數值	變動	1~2 位元組	該分類中的第一筆資料數值
第二筆資料數值	變動	1~2 位元組	該分類中的第二筆資料數值
...	…	…	…

表 3-4 確定裝置時所需要的資料

根據該類別的 arg 做判斷，若不為零則代表該筆資料是有意義的，若為零則代表已經結束。

4. 新增感測模組

在這一章介紹如何在 NuMaker Brick 平台上新增一個感測模組，新增感測模組十分簡單，幾乎不需要考慮平台的溝通設定。假設已經有個可以獨立正常工作的感測模組要加入 NuMaker Brick 平台，以下將分步驟介紹如何完成此工作。首先將整個流程分成硬體及軟體部分。圖 4-1 是新增感測模組的硬體示意圖，在這一章以新增光敏感測模組為例子，硬體會介紹將獨立工作的感測模組移到 NuMaker Brick 拓展版上必要的步驟，完成整個流程後光敏感測模組完成。

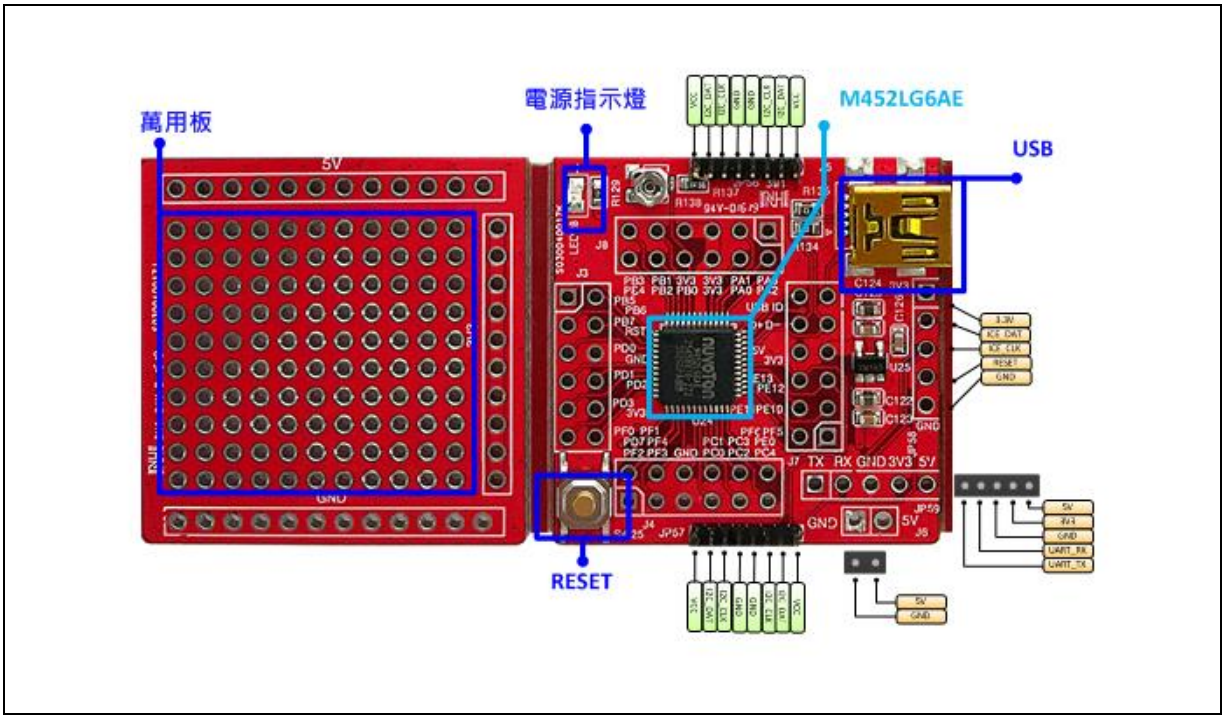

圖 4-1 拓展板實體圖

4.1. 新增感測模組硬體設定

使用者將原有的感測模組硬體移植到 NuMaker Brick 拓展板時，可以很方便的使用拓展板上的各式腳位及功能，以下將舉光敏感應模組為例子介紹 NuMaker Brick 拓展板的功能。

下圖是 NuMaker Brick 拓展板新增感測模組時會用到的電路圖，使用的是新唐的 M452 微控器。在這個拓展板上所有的腳位都有外接接頭，使得微控制器的使用性可以發揮到最大，M452 微控器的功能包含 UART、smart card、SPI、I^2C、I^2S、USB、PWM、AMCP、DAC、ADC 等。

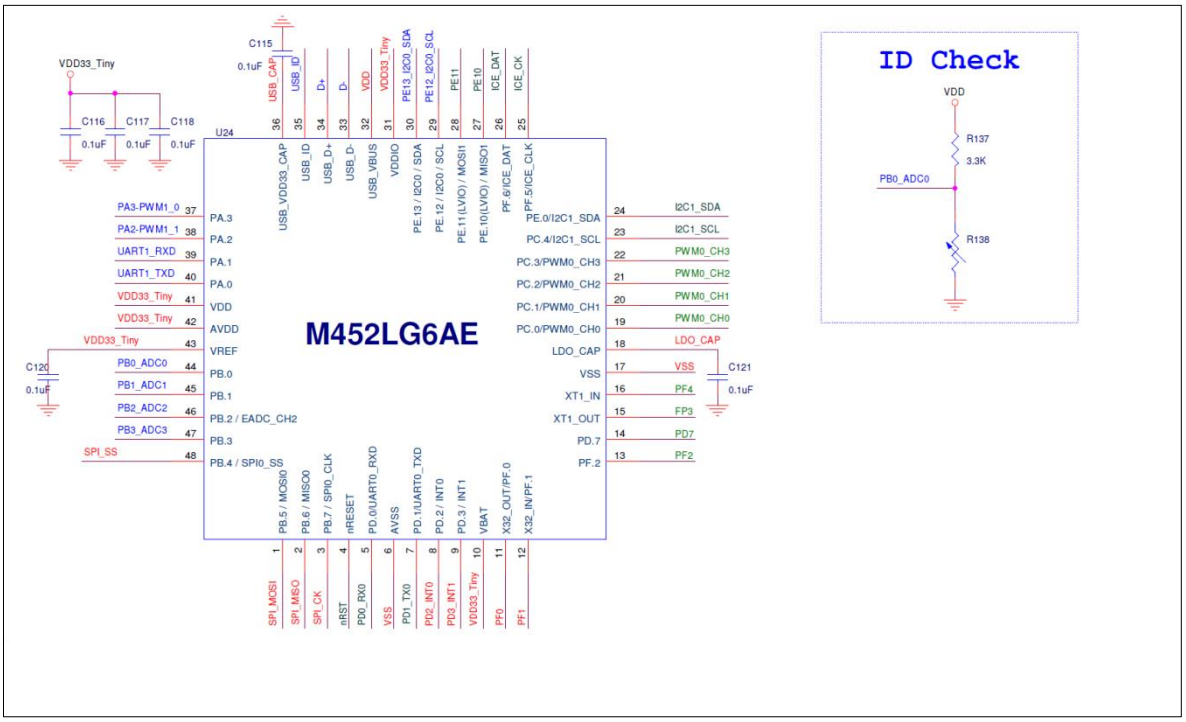

圖 4-2 NuMaker Brick 拓展板微控器及ID Check電路圖

在新增感測模組時首先需要注意的地方是模組 ID 識別的部分，在上圖 ID check 的 R138 是一個可變電阻，為了能讓程式辨認出這是新的裝置，電阻值設為 5.1 KΩ (此時新增第 9 模組)。使用時可先上電，量測至所需電壓 (2V) 即可。若需要新增的模組不只一個，也可以外接新的新唐 M451 系列微控器，腳位 PB.0 必須要使用 ADC 功能讀電壓值，電路圖可參考的 ID check 功能，表 4-1 新裝置 ID 對應電阻值為各個 ID 所帶需要的 ADC 電壓值以及電阻建議值，ID 從第九個開始。

ID	目標電壓值範圍	連到 VDD 電阻值	連到 GND 電阻值	實際電壓值
9	1.9V ~ 2.05V	3.3KΩ	5.1KΩ	1.99V
10	2.1V ~ 2.25V	3.3KΩ	6.8KΩ	2.2V
11	2.3V ~ 2.45V	3.3KΩ	9.1KΩ	2.41V
12	2.5V ~ 2.65V	3.3KΩ	12KΩ	2.59V
13	2.7V ~ 2.85V	3.3KΩ	18KΩ	2.8V
14	2.9V ~ 3.05V	3.3KΩ	33KΩ	3V

表 4-1 新裝置ID對應電阻值

設定完 ID 辨識部分之後，接著便可以開始將原本獨立的模組加入拓展板上，拓展板上有額外的空間可以加入所需要的模組，再將模組所需要的電源以及需要連接到微控器的腳位接到相應的接頭上便可以完成感測模組的移植， 舉例在新增光敏感測模組時的電路圖，如圖 4-3 光敏感測模組電路圖。

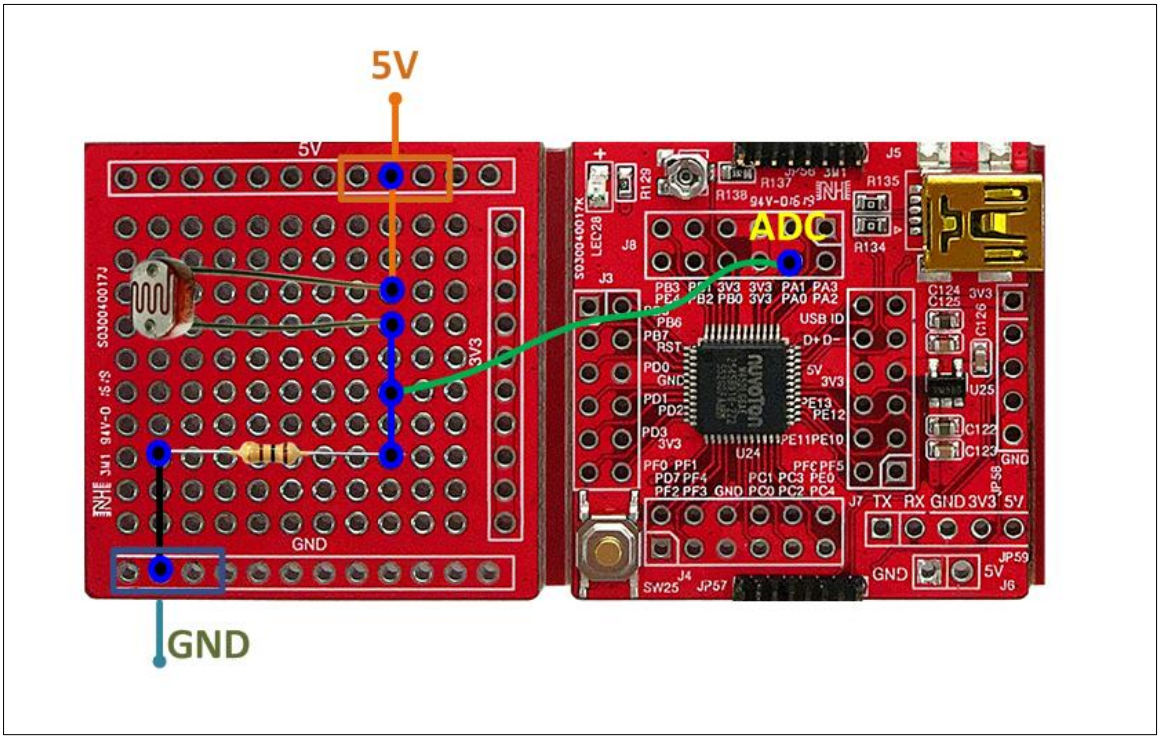

圖 4-3 光敏感測模組電路圖

4.2. 軟體擴充說明

完成硬體的擴充之後介紹軟體的移植。軟體分成兩部分，初始化階段以及執行階段。

初始化階段首先要對感測模組做啟動時的辨別身分，接著做 NuMaker Brick 平台介面資料初始化以及感測模組本身的初始化。

執行階段要修改的功能分成兩部分，第一階段是新增主控端對新增裝置所對應的行為，第二階段是將感測模組原本的工作移到程式中，並且將 NuMaker Brick 的資料與原本的程式結合。下表對軟體的每階段作個表格整理，由於程式修改時主控端必須要做一些相對的回應給感測模組，因此不僅僅感測模組的程式需要修改，主控端也需要作整理。**錯誤! 找不到參照來源。**將每個階段的步驟分成主控端以及感測模組避免混淆。

軟體	主控端	感測模組
初始化階段	無須修改	ID Check 平台資料初始化 加入感測模組初始化程式
執行階段	新增感測模組的對應行為	加入感測模組執行程式 將平台資料合併至原本程式中

表 4-2 軟體新增感測模組說明

而在 NuMaker Brick 已經將這三個階段簡化至一步，只要將相對應的程式放到確切的位置即可，如圖 4-4 將初始化放在第九個裝置的第一格，週期執行程式放在第二格，即時執行程式放在第三格。

```
APFN_FUNC_T pfnDevFunc[MAX_TID_DEV] =
{
    /*
       Functions for each device:

        {Initial, Period, Pulling, Report}

        Initial : Init and configure device
        Period  : Process device sensor data or set sensor once per 0.1 seconds.
        Pulling : Process device sensor data or set sensor frequently.
        Report  : Report process data to master
    */

    /* for embedded device boards */

    {Battery_Init,      MasterControl,  NULL,               report_battery},
    {Buzzer_Init,       NULL,           Buzzer_Control,     report_buzzer},
    {Led_Init,          NULL,           Led_Control,        report_led},
    {AHRS_Init,         AHRS_Control,   NULL,               report_ahrs},
    {SonarInit,         SonarDetect,    SonarTimeOutCheck,  report_sonar},
    {Init_DHT11_PWM0,   DHT11Control,   DHT11GetDATA,       report_temp},
    {Gas_Init,          GetGas,         NULL,               report_gas},
    {IR_Init,           IR_Control,     IR_Check,           report_ir},
    {key_init,          NULL,           NULL,               report_key},

    /* for custom device boards
       fill your custom board functions here to specified ID. */
    {Lr_Init,  GetLr,  NULL,  NULL},   // For device ID 9
    {NULL, NULL, NULL, NULL},   // For device ID 10
    {NULL, NULL, NULL, NULL},   // For device ID 11
    {NULL, NULL, NULL, NULL},   // For device ID 12
    {NULL, NULL, NULL, NULL},   // For device ID 13
    {NULL, NULL, NULL, NULL}    // For device ID 14
};
```

初始化 週期執行 即時執行

圖 4-4 新增模組程式放置位置

4.2.1. 軟體初始化階段

在此討論初始化階段的流程，由於初始化程式只要填入確切的位置即可，在此介紹 NuMaker Brick 平台在初始化時所需要修改的資料。

- 模組初始化

  對新增的裝置進行初始化，必須新增程式，範例如圖 4-5 光敏感測模組的初始化內容，

  初始化內容修改 (Lr_init() 說明)：

  必須要針對初始化的部分作修改，將光敏感測模組的初始化：Lr_init() 當作範例

  在初始化的程式中，第一部分是原本光敏感測模組的初始化，這裡只要將原本光敏感測模組的初始化移植過來即可不需要做改動，接著是光敏感測模組與主控端要傳遞的資料設定。

  建議資料設定的方式如下，

  A. 首先訂定需要傳遞的資料，光敏感測模組需要的資料如下：

     睡眠周期、警報觸發數值、光敏電壓、警報。

  B. 接著做分類：特性、輸出、輸入。

     特性有兩筆：睡眠周期、警報觸發數值。

     輸入有兩筆：光敏電壓、警報。

  C. 接著設定每一筆資料的內容，每一筆資料內包含了：

     最大值、最小值、位置、長度、資料內容。

I. 最大值及最小值 (maximum, minimum)：下圖棕色框內每一筆資料皆有，照資料需求設定。

II. 位置 (arg)：下圖黃色框內設定，位置按順序遞增。

III. 長度 (datalen)：下圖黃色框內設定，先預設 2 表示有 16 位元即可，後面章節再介紹。

IV. 資料內容：即為該筆資料的數值。

V. 資料數量 (dataNum) ：每一種類的資料最後皆會有一筆資料需要填入該種類資料的數量，例如在這裡輸入資料有兩筆，因此數值為 2。

實際光敏感測模組初始化程式如圖 4-5所示。

```
void lr_Init()
{
    SYS_UnlockReg();
    /* Enable EADC module clock */
    CLK_EnableModuleClock(EADC_MODULE);
    /* EADC clock source is 72MHz, set divider to 8, ADC clock is 72/8 MHz */
    CLK_SetModuleClock(EADC_MODULE, 0, CLK_CLKDIV0_EADC(8));
    SYS_LockReg();
    /* Configure the GPB0 - GPB3 ADC analog input pins.  */
    SYS->GPB_MFPL &= ~SYS_GPB_MFPL_PB1MFP_Msk;
    SYS->GPB_MFPL |= SYS_GPB_MFPL_PB1MFP_EADC_CH1;

    GPIO_DISABLE_DIGITAL_PATH(PB, BIT1);

    /* Set the ADC internal sampling time, input mode as single-end and enable the A/D converter */
    EADC_Open(EADC, EADC_CTL_DIFFEN_SINGLE_END);
    EADC_SetInternalSampleTime(EADC, 6);

    /* Configure the sample module 0 for analog input channel 1 and software trigger source.*/
    EADC_ConfigSampleModule(EADC, 1, EADC_SOFTWARE_TRIGGER, 1);

    /* Clear the A/D ADINT0 interrupt flag for safe */
    EADC_CLR_INT_FLAG(EADC, 0x2);

    /* Enable the sample module 0 interrupt.  */
    EADC_ENABLE_INT(EADC, 0x2);//Enable sample module A/D ADINT0 interrupt.
    EADC_ENABLE_SAMPLE_MODULE_INT(EADC, 1, 0x2);//Enable sample module 0 interrupt.

    ResDev9.DevDesc.DevDesc_leng = 26;                  //Report descriptor
    ResDev9.DevDesc.RptDesc_leng = 36;                  //Report descriptor
    ResDev9.DevDesc.InRptLeng = 5;                      //Input report
    ResDev9.DevDesc.OutRptLeng = 0;                     //Output report
    ResDev9.DevDesc.GetFeatLeng = 6;                    //Get feature
    ResDev9.DevDesc.SetFeatLeng = 6;                    //Set feature
    ResDev9.DevDesc.CID = 0;                            //manufacturers ID
    ResDev9.DevDesc.DID = 0;                            //Product ID
    ResDev9.DevDesc.PID = 0;                            //Device firmware revision
    ResDev9.DevDesc.UID = 0;                            //Device Class type
    ResDev9.DevDesc.UCID = 0;                           //reserve
    /* Feature */
    ResDev9.Feature.data1.minimum = 0;                  //Sleep period
    ResDev9.Feature.data1.maximum = 1024;
    ResDev9.Feature.data1.value = 100;
    ResDev9.Feature.data2.minimum = 0;                  //Light resistant alerm value
    ResDev9.Feature.data2.maximum = 100;
    ResDev9.Feature.data2.value = 20;
    ResDev9.Feature.arg[0] = 1;
    ResDev9.Feature.arg[1] = 2;
    ResDev9.Feature.datalen[0] = 2;
    ResDev9.Feature.datalen[1] = 2;
    ResDev9.Feature.dataNum = 2;
    /* Input */
    ResDev9.Input.data1.minimum = 0;                    //light resistance value
    ResDev9.Input.data1.maximum = 100;
    ResDev9.Input.data1.value = 100;
    ResDev9.Input.data2.minimum = 0;                    //Over flag
    ResDev9.Input.data2.maximum = 1;
    ResDev9.Input.data2.value = 0;
    ResDev9.Input.arg[0] = 1;
    ResDev9.Input.arg[1] = 2;
    ResDev9.Input.datalen[0] = 2;
    ResDev9.Input.datalen[1] = 1;
    ResDev9.Input.dataNum = 2;
    /* Output */
    ResDev9.Output.dataNum = 0;
```


圖 4-5 光敏感測模組的初始化內容

至此，一個新的感測模組的初始化便告一個段落，接下來將進入執行的階段。

4.2.2. 軟體執行階段

在執行階段需要將整個流程分成兩部分，第一部分是感測模組的修改，第二部分則是主控端的反應。

- 感測模組部分的修改

  感測模組的部分只要將原本單獨工作的感測模組程式加到 NuMaker Brick 平台即可，同樣的只要在確切的位置填入程式即可。

- 主控端部分的修改

  主控端是要讓主控端能對新增的感測模組做回饋，包含設定新增裝置能觸發的對象，以及能夠觸發新裝置的對象。位於 TIDMstUpdateDevState() 中，在 tismst.c 中，主程式位置如下：

```
/* 執行階段的程式 */
loop()
{
    if(TMR1INTCount != RecentTimeCounter)
    {
            /* 主控端檢查及反應感測模組資料的位置 */
            TIDMstUpdateDevState();
    }
}
```

光敏感測模組在這個程式中所使用的功能如下：

1. 假設光敏感測模組回傳的資料為 1
2. 揚聲器與光敏感測模組有關連
3. 主控端要揚聲器開始響鈴
4. LED 與光敏感測模組有關連
5. 主控端要 LED 開始閃爍

程式如下：

```
void TIDMstUpdateDevState()
{
        ……
        /* 光敏感測模組回傳的資料2為1 */
        else if((ResDev9.Input.data2.value == 1) && (TIDMstInitDevState & (1<<RESDEV9)))
        {
        /* BuzDev.dTod.dNINE表示揚聲器與光敏感測模組關連性 */
            if(BuzDev.dTod.dNINE==1)
            {
                /* BuzDev.Output.data1.value表示揚聲器的第一筆輸入資料(開始響鈴) */
                BuzDev.Output.data1.value = 1;          //Song
                TIDMstDevState[BUZZERDEV] |= 2;     //buzzer operate：  output(2)
            }
            /* LedDev.dTod.dNINE表示揚聲器與光敏感測模組關連性*/
            if(LedDev.dTod.dNINE==1)
            {
                /* LedDev.Output.data1.value表示LED的第一筆輸入資料(開始閃爍) */
                LedDev.Output.data1.value = 1;          //Period
                TIDMstDevState[LEDDEV] |= 2;          //led operate：       output(2)
            }
        }
}
```

如此一來便完成整個 NuMaker Brick 平台新增感測模組的流程。

5. 使用 ARDUINO 連接 NUMAKER BRICK

NuMaker Brick 也可使用 Arduino 當作新增的感測器，這一章首先介紹必須要使用到的程式。接著介紹如何新增感測器，同樣新增感測器分為硬體及軟體兩部分。

5.1. 必須呼叫的程式

在這一節介紹不管任何模組皆需要呼叫的平台架構程式，同樣的分成 setup 以及 loop 兩個區塊。

5.1.1. Setup

如圖 5-1 所示，同要得必須設定 UART、I²C 以及平台架構的初始化。

1. 設定 UART
2. I²C 初始化
3. 將模組資料存到 I²C 要傳的變數中

```
void setup()
{
    // put your setup code here, to run once:
    Serial.begin(9600);
    I2C_init();
    Led_Init();
    SlvDevInit(&LedDev);
    SlvDataStore();
}
```

1.設定UART

2.I²C初始化

3.將模組資料存到I2C要傳的變數中

圖 5-1 Arduino LED 模組初始化必要程式

5.1.2. Loop

如圖 5-2 所示，在感測模組必須要做兩件事情：

- 將主控端送來的資料作整理
- 將模組資料存到 I²C 要傳的變數中

```
void loop()
{
    TID_SlvRxUpdate();                    將主控端送來的資料作整理

    /* LED code */
    if(LedDev.Output.data2.value==0)
    {
        if(LedDev.Output.data1.value == 1)
        {
            Led_Blink_Start();
            LedDev.Output.data1.value=0;
        }
    }
    else
    {
        Led_Stop();
    }
    Led_Blink_Check();

    //update TID data
    SlvDataStore();                       將模組資料存到I2C要傳的變數中
}
```

圖 5-2 Arduino LED 模組執行區必要程式

這樣便在Arduino上完成的一個基本的NuMaker Brick架構

5.2. 新增模組

5.2.1. 硬體

首先介紹如何新增硬體，所使用的 Arduino 板子為 Arduino MEGA，所使用的模組為LED模組，電路圖如圖 5-3 所示，RGB 分別代表 LED 燈的三色。SDA 以及 SCL 代表要連接到 NuMaker Brick 上的 I^2C 的介面接線。Arduino 必須與 NuMaker Brick 共地。

Note.連接 Arduino LED 時，原本的 LED 模組必須要斷開

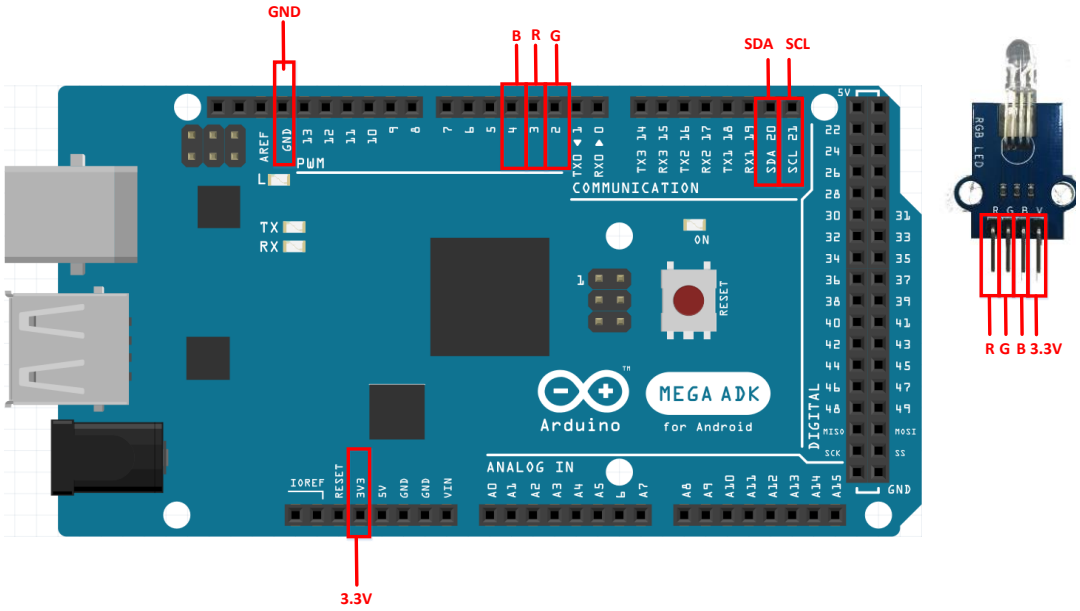

圖 5-3 Arduino LED 模組示意圖

5.2.2. 軟體

在新增模組時，分成 setup 以及 loop 區塊。

- Setup：

  在setup中需要新增模組的初始化程式，此程式與原本元 NuMaker Brick 的程式一樣，如圖 5-4所示。

```
void setup()
{
    // put your setup code here, to run once:

    Serial.begin(9600);

    I2C_init();

    Led_Init();

    SlvDevInit(&LedDev);
    SlvDataStore();
}
```

LED初始化

圖 5-4 Arduino LED 模組初始化新增程式

- loop：

在 loop 中需要增加兩種程式，如圖 5-5 所示。

1. 週期執行程式：在這次介紹的模組中沒有需要增加
2. 即時執行程式：直接將需要即時執行的程式加入 setup 不斷執行即可

```
void loop()
{
    TID_SlvRxUpdate();

    /* LED code */
    if(LedDev.Output.data2.value==0)
    {
        if(LedDev.Output.data1.value == 1)
        {
            Led_Blink_Start();
            LedDev.Output.data1.value=0;
        }
    }
    else
    {
        Led_Stop();
    }
    Led_Blink_Check();

    //update TID data
    SlvDataStore();
}
```

即時執行的程式

圖 5-5 Arduino LED 模組執行區新增程式

A. 附錄 APP 的設計概念

由於整個平台是由感測模組及警報所構成，在不同環境下可能需要不同的設定，如濕度標準、偵測距離標準等，但若每次修改都需要重新燒錄程式則整個流程會變得十分的繁瑣，因此計劃設計手機可以跟主控端溝通並命令主控端去修改感測模組的資料，如此一來對於使用者的便利性便會大幅提升。

APP 的主要目標有三種，在 APP 中設定四種頁面可供選擇：

1. 連接平台頁面：在此頁面座連接阜的設定，在此頁面未連接上平台時其他頁面皆不會顯示。
2. 首頁：協助使用者了解目前的感測模組資料，並且在進入此頁面後才會顯示其他有連接上的感測模組頁面，此頁面所接收到的資料長度皆為兩位元組。
3. 設備連接頁面：讓使用者可以修改各感測模組間連接設定，例如距離偵測是否可以觸發揚聲器、振動感測是否可以觸發 LED 燈閃爍等，此頁面所接收到的資料長度皆為兩位元組。
4. 各裝置頁面：讓使用者可以修改各感測模組資料，此頁面所接收到的資料長度可能會變動。

由於 APP 端是使用 UART 與主控端連接，在此介紹一些基本 APP 端可以發送的命令：

1. 開始傳值：@ss
2. 停止傳值：@sp
3. 設定傳輸格式：@tx

在後面的章節會在介紹一些額外的功能。

在這一章節將對這四個頁面做一一的介紹。

A.1. 首頁

連接上平台後首先進入首頁如下圖 A-1，這個頁面的功能主要分成兩種：第一、可以顯示目前主控端連接到的感測模組。第二、可以顯示感測模組的輸入資料狀態。接下來介紹主控端如何傳遞前述所提的兩種資料，注意到在此頁面每筆資料的長度不會變動，皆為 2 位元組。

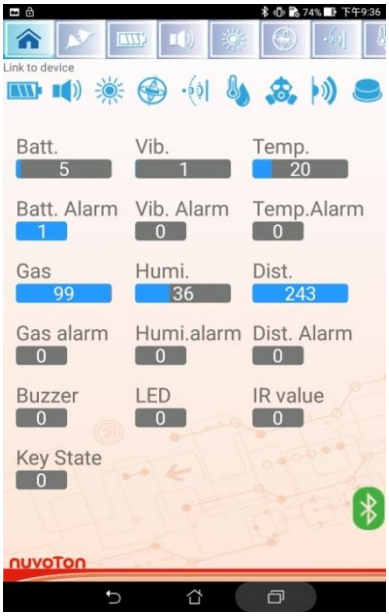

圖 A-1 首頁表示圖

將傳輸分成兩階段，在進入此頁面時為第一階段，此時傳送主控端的感測模組狀態 (是否有連接)，成功後進入第二階段，開始接收各感測模組的資料值。若有任一階段的資料有誤則重新回到第一階段重新接收感測模組狀態。

切換頁面的傳輸命令如下：

1. 傳 @tx，通知主控端切換頁面至首頁
2. 傳 @ss，通知主控端開始傳值
3. 若有錯誤傳 @sp，通知主控端暫停傳值
4. 傳 @tx，通知主控端重新切換頁面
5. 等 500 ms
6. 傳 @ss，通知主控端開始傳值

接著介紹 APP 收值的格式：

第一階段會收到的資料：

資料名稱	資料內容	資料長度	說明
通知開始旗標	257(1, 1)	2 位元組	判斷資料開始傳輸
第一階段資料長度	8	2 位元組	第一階段有4組資料，8位元組
感測模組數量	8	2 位元組	有設定的感測模組數量
連接上感測模組數量	變動	2 位元組	有連接到的感測模組
第二階段資料長度	30	2 位元組	第二階段要傳輸的資料量

表 A-1 首頁第一階段收到的資料

收到的值中，第四筆資料 (連接上感測模組數量)，在接收完後經過整理便可以得到上圖中 Link to device 狀態，第一個位元對應到 INDBATTERY (若為1則亮起)，第二個位元對應到 INDBUZZER (若為1則亮起)，第三個位元對應到 INDLED (若為1則亮起)，第四個位元對應到 INDAHRS (若為1則亮起)，第五個位元對應到 INDSONAR (若為1則亮起)，第六個位元對應到 INDTEMP. (若為1則亮起)，第七個位元對應到 (若為1則亮起)，第八個位元對應到 INDIR (若為1則亮起)。第五筆資料則用來確認第二階段收值是否正確，若錯誤則重新來過。第二階段會收到的資料格式如下：

資料名稱	資料內容	資料長度	說明
第二階段資料長度	30	2 位元組	若收到資料不等於30重回第一階段
電池電量	變動	2 位元組	
低電量警告	變動	2 位元組	1為電量過低
揚聲器執行狀態	變動	2 位元組	1為正在發出聲音
LED執行狀態	變動	2 位元組	1為正在閃爍
振動數值	變動	2 位元組	
振動警報	變動	2 位元組	1為觸發振動警報
聲納距離	變動	2 位元組	
距離警報	變動	2 位元組	1為觸發距離警報
溫度	變動	2 位元組	
濕度	變動	2 位元組	
溫度警報	變動	2 位元組	1為觸發溫度警報
濕度警報	變動	2 位元組	1為觸發濕度警報
瓦斯數值	變動	2 位元組	
瓦斯警報	變動	2 位元組	1為瓦斯警報

表 A-2 首頁第二階段收到的資料

A.2. 設備連接頁面

如圖3-3.1，在這個頁面可以顯示目前的感測模組彼此連接的狀態，將感測模組分成兩種類型，輸入 (電池、振動模組、聲納、溫溼度、瓦斯) 及輸出類 (揚聲器、LED)。在進入此頁面的收值部分分成兩方面：第一、顯示目前有連上平台的感測模組。第二、顯示目前感測模組間彼此的關係。將這兩方面分成三個階段，第一階段顯示目前有連接上平台的感測模組，第二階段顯示目前感測模組彼此的關係，第三階段接收感測模組彼此的關係但不更新顯示。切換頁面時傳送命令如下：

1. 進入頁面時傳送 @td，告訴主控端切到設備連接頁面
2. 傳 @ss，命令主控端開始傳值
3. 若有錯誤傳 @sp，通知主控端暫停傳值
4. 傳 @td，通知主控端重新切換頁面
5. 等 500 ms
6. 傳 @ss，通知主控端開始傳值

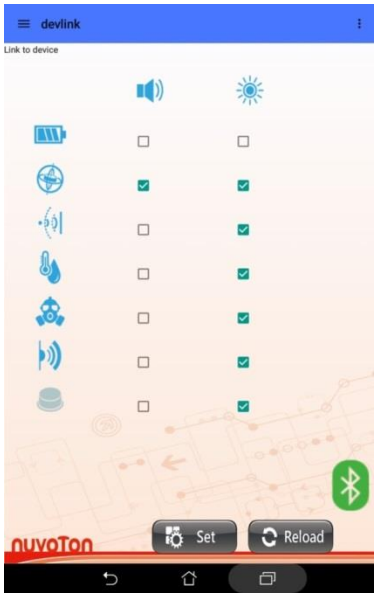

圖 A-2 裝置連接頁面

接收的資訊如下：

第一階段：

資料名稱	資料內容	資料長度	說明
通知開始旗標	257(1, 1)	2 位元組	判斷資料開始傳輸
第一階段資料長度	8	2 位元組	第一階段有4組資料，8位元組
感測模組數量	8	2 位元組	有設定的感測模組數量
連接上感測模組數量	變動	2 位元組	有連接到的感測模組
第二階段資料長度	6	2 位元組	第二階段要傳輸的資料量

表 A-3 設備連接頁面第一階段收到的資料

與首頁第一階段收到的資訊十分相似，第四筆資料用來做確認那些感測模組有連上平台。第五筆資料用來確認第二及第三階段的資訊是否正確。

第二階段：

資料名稱	資料內容	資料長度	說明
第二階段資料長度	6	2 位元組	若收到資料不等於6重回第一階段
揚聲器連接感測模組狀態	變動	2 位元組	
揚聲器連接感測模組狀態	變動	2 位元組	

表 A-4 設備連接頁面第二階段收到的資料

在二階段主要目的是顯示揚聲器及 LED 可以被那些感測模組控制 (1: 有連上，狀態要亮起)，揚聲器的第一位元表示與電池的連接狀態，第二及第三位元跳過，第四位元表示與振動器的連接狀態，第四位元表示與振動感測模組的連接狀態，第五位元表示與聲納的連接狀態，第六位元表示與溫溼度感測模組的連接狀態，第七位元表示與瓦斯感測模組的連接狀態。LED 燈亦同。

而在第三階段收到的資訊與第二階段完全一樣，但是第二階段要更新顯示的狀態第三階段不需要 (可由頁面中的 Reload 更新顯示的狀態)，如此做的目的是因為若持續更新頁面，則無法完成傳值的動作。

接著討論傳值的部分，傳值分成兩個部分，重新收值 (Reload) 及設定 (Set)。重新收值有兩步驟，傳 @ss 命令主控端繼續傳值，並更新顯示的狀態。由於在第三階段只收值但不更新顯示的狀態，在此提供可以不轉換頁面就可以更新狀態的功能。設定部分是將目前由 APP 端所更改過的感測模組連接狀態傳給主控端，命令主控端以新的連接狀態為準。其傳輸的流程如下

傳輸命令	說明
“@mz40”+1or0+”\r”	修改揚聲器與電池感測模組間狀態
“@mz43”+1or0+”\r”	修改揚聲器與振動感測模組間狀態
“@mz44”+1or0+”\r”	修改揚聲器與聲納感測模組間狀態
“@mz45”+1or0+”\r”	修改揚聲器與溫溼度感測模組間狀態
“@mz46”+1or0+”\r”	修改揚聲器與瓦斯感測模組間狀態
“@ml40”+1or0+”\r”	修改 LED 與電池感測模組間狀態
“@ml43”+1or0+”\r”	修改 LED 與振動感測模組間狀態
“@ml44”+1or0+”\r”	修改 LED 與聲納感測模組間狀態
“@ml45”+1or0+”\r”	修改 LED 與溫溼度感測模組間狀態
“@ml46”+1or0+”\r”	修改 LED 與瓦斯感測模組間狀態

表 A-5 設備連接頁面第二階段收到的資料

A.3. 感測模組頁面所需要的資料

裝置頁面共有七個頁面 (電池、揚聲器、LED、振動感測模組、聲納感測模組、溫溼度感測模組、瓦斯感測模組)，此時輸入及輸出類介面並不明顯，仍會分兩次介紹並稍作比較。感測模組頁面目的是 (1) 顯

示各感測模組目前的資料 (收值)，以及 (2) 可以對各感測模組的資料進行修改。之前有提過各感測模組的資料有三種，輸出、輸入及特性。在 APP 中將其簡化成兩類，控制類 (原本的輸出及特性) 及狀態類 (原本的輸入)，原因是原本的輸入類是由感測模組傳值給主控端，不能由主控端傳給感測模組。因此 APP 在對各感測模組的資料進行便只會對控制類進行修改而狀態類便可以略過。

接下來首先對於顯示各感測模組的資料進行介紹 (收值) 。分成四步驟，與感測模組與主控端的連接類似，第一階段先收裝置的基本訊息，第二階段收資料的格式，第三階段收資料的內容，第四階段同樣收資料的類容但只更新狀態類，控制類的項目不予更新 (可由頁面中的 Reload 更新)，如此做的目的是因為若持續更新頁面，則無法完成傳值的動作。

在切換到感測模組頁面 APP 端傳送的訊息如下：

1. 進入頁面時傳送 @tb (註)，告訴主控端切到揚聲器頁面
2. 傳 @ss，命令主控端開始傳值
3. 若有錯誤傳 @sp，通知主控端暫停傳值
4. 傳 @tb (註)，通知主控端重新切換頁面
5. 等 500 ms
6. 傳 @ss，通知主控端開始傳值

註：其他感測模組命令如表 A-6

感測模組	命令
電池	@tb
揚聲器	@tz
LED	@tl
振動感測模組	@ta
聲納感測模組	@ts
溫溼度感測模組	@tt
瓦斯感測模組	@tg

表 A-6 切換感測模組頁面各感測模組所要下的命令

圖 A-3便是揚聲器頁面，主要是顯示揚聲器的資料數值，將原本的資料分成兩類，控制類 (原本的輸出及資料)，狀態類 (原本的輸入)。

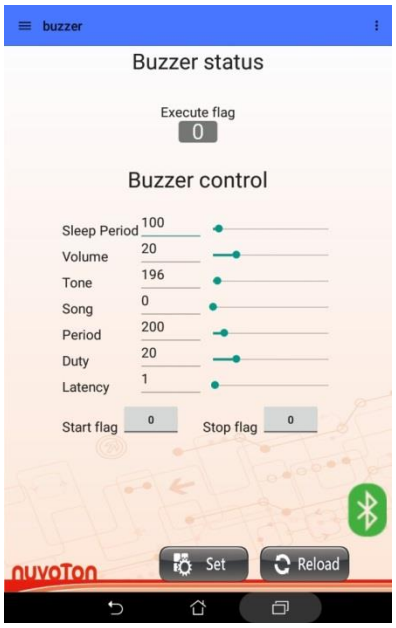

圖 A-3 揚聲器頁面

下圖 A-4 聲納感測模組頁面是聲納感測模組頁面，主要是顯示揚聲器的資料數值，將原本的資料分成兩類，控制類 (原本的輸出及資料)，狀態類 (原本的輸入)。

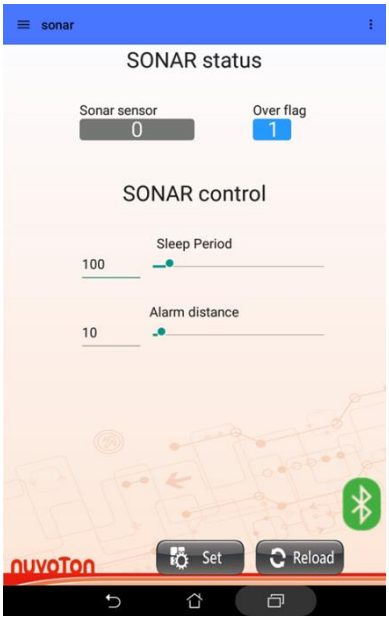

圖 A-4 聲納感測模組頁面

接著介紹在感測模組頁面的收值流程，第一步驟先收裝置的基本資訊，與主控端的收基本資訊幾乎一模一樣，若收到的開始旗標不符合 (不是257) 則有錯誤，資料內容如下：

資料名稱	資料內容	資料長度
通知開始旗標	257(1, 1)	2 位元組
基本資料長度	26	2 位元組
資料格式長度	變動	2 位元組

回報輸入長度	變動	2 位元組
設定輸出長度	變動	2 位元組
回報特性長度	變動	2 位元組
設定特性長度	變動	2 位元組
公司代號	0	2 位元組
設備代號	0	2 位元組
產品代號	0	2 位元組
韌體版本	0	2 位元組
設備類別代號	0	2 位元組
保留位	0	2 位元組
保留位	0	2 位元組

表 A-7 感測模組頁面第一階段的基本資料

第二步驟收感測模組的基本資訊，與主控端的收基本資訊幾乎一模一樣。在收到資料位址後，將資料長度儲存在對應類別的 datalen 中，並對 arg 寫入目前數值 (如收到 41 的位址，長度為 1，則對第四筆資料的 datalen 寫入 1，arg 寫入 4)，以利於之後收資料數值時判斷用，若收到的第一筆料不符合第一階段所收到的資料格式長度則有錯誤，資料內容如下：

資料名稱	資料內容	資料長度	功能說明
資料格式長度	變動		這一筆資料的長度， 是判斷第二階段資料正確是否所需要的設定
資料類別 (特性)	257(1, 1)	2 位元組	宣告以下資料的資料類別：特性
資料位址	5	1 位元組	宣告這筆資料的位址 (5, 17, 29, 41, 53, 65, 77, 89, 101, 113)
資料長度	1~2	1 位元組	宣告這筆資料的長度(1位元組~2位元組)
通知下一筆是資料的最小值及最小值資料長度	9 or 10	1 位元組	宣告下一欄是資料的最小值 (8) 還有下一欄的長度(1or2)
資料的最小值	變動	1~2位元組	資料的最小值
通知下一筆是資料的最大值及最大值資料長度	13 or 14	1 位元組	宣告下一欄是資料的最小值 (12) 還有下一欄的長度(1or2)
資料的最大值	變動	1~2位元組	資料的最大值
資料位址	17	1 位元組	宣告這筆資料的位址 (5, 17, 29, 41, 53, 65, 77, 89, 101, 113)
資料長度	1~2	1 位元組	宣告這筆資料的長度(1位元組~2位元組)
通知下一筆是資料的最小值及最小值資料長度	9 or 10	1 位元組	宣告下一欄是資料的最小值 (8) 還有下一欄的長度(1or2)
資料的最小值	變動	1~2位元組	資料的最小值
通知下一筆是資料的最大值及最大值資料長度	13 or 14	1 位元組	宣告下一欄是資料的最小值 (12) 還有下一欄的長度(1or2)
資料的最大值	變動	1~2位元組	資料的最大值
…	…	…	…
資料類別 (輸入)	258(1, 2)		宣告以下資料的資料類別：輸入

資料位址	5	1 位元組	宣告這筆資料的位址 (5, 17, 29, 41, 53, 65, 77, 89, 101, 113)
資料長度	1~2	1 位元組	宣告這筆資料的長度 (1位元組~2位元組)
通知下一筆是資料的最小值及最小值資料長度	9 or 10	1 位元組	宣告下一欄是資料的最小值 (8) 還有下一欄的長度(1or2)
資料的最小值	變動	1~2位元組	資料的最小值
通知下一筆是資料的最大值及最大值資料長度	13 or 14	1 位元組	宣告下一欄是資料的最小值 (12) 還有下一欄的長度(1or2)
資料的最大值	變動	1~2位元組	資料的最大值
資料位址	17	1 位元組	宣告這筆資料的位址 (5, 17, 29, 41, 53, 65, 77, 89, 101, 113)
資料長度	1~2	1 位元組	宣告這筆資料的長度(1位元組~2位元組)
通知下一筆是資料的最小值及最小值資料長度	9 or 10	1 位元組	宣告下一欄是資料的最小值 (8) 還有下一欄的長度(1or2)
資料的最小值	變動	1~2位元組	資料的最小值
通知下一筆是資料的最大值及最大值資料長度	13 or 14	1 位元組	宣告下一欄是資料的最小值 (12) 還有下一欄的長度(1or2)
資料的最大值	變動	1~2位元組	資料的最大值
…	…	…	…
資料類別 (輸出)	259(1, 3)		宣告以下資料的資料類別 (輸出)
資料位址	5	1 位元組	宣告這筆資料的位址 (5, 17, 29, 41, 53, 65, 77, 89, 101, 113)
資料長度	1~2	1 位元組	宣告這筆資料的長度(1位元組~2位元組)
通知下一筆是資料的最小值及最小值資料長度	9 or 10	1 位元組	宣告下一欄是資料的最小值 (8) 還有下一欄的長度(1or2)
資料的最小值	變動	1~2位元組	資料的最小值
通知下一筆是資料的最大值及最大值資料長度	13 or 14	1 位元組	宣告下一欄是資料的最小值 (12) 還有下一欄的長度(1or2)
資料的最大值	變動	1~2位元組	資料的最大值
資料位址	17	1 位元組	宣告這筆資料的位址 (5, 17, 29, 41, 53, 65, 77, 89, 101, 113)
資料長度	1~2	1 位元組	宣告這筆資料的長度(1位元組~2位元組)
通知下一筆是資料的最小值及最小值資料長度	9 or 10	1 位元組	宣告下一欄是資料的最小值 (8) 還有下一欄的長度(1or2)
資料的最小值	變動	1~2位元組	資料的最小值
通知下一筆是資料的最大值及最大值資料長度	13 or 14	1 位元組	宣告下一欄是資料的最小值 (12) 還有下一欄的長度(1or2)
資料的最大值	變動	1~2位元組	資料的最大值
…	…	…	…

表 A-8 感測模組頁面第二階段回傳資料格式

第三階段一次收下所有控制類及狀態類(原本的輸出、輸入、特性)的資料，若第一筆資料不等於第一階段的”回報輸入長度”+”設定輸出長度”+”回報特性長度”則錯誤，收值時根據該類別的 arg 做判斷，若不為零則代表該筆資料是有意義的，若為零則代表已經結束，進入下一種類別，資料內容如下：

資料名稱	資料內容	資料長度	說明
通知開始旗標	變動	2 位元組	回報輸入長度+設定輸出長度+回報特性長度
特性類資料第一筆	變動	1~2 位元組	
特性類資料第二筆	變動	1~2 位元組	
...	變動	1~2 位元組	由類別中的 arg 變數判斷該類別是否結束
輸入類資料第一筆	變動	1~2 位元組	
輸入類資料第二筆	變動	1~2 位元組	
...	變動	1~2 位元組	由類別中的arg變數判斷該類別是否結束
輸出類資料第一筆	變動	1~2 位元組	
輸出類資料第二筆	變動	1~2 位元組	
...	變動	1~2 位元組	由類別中的 arg 變數判斷該類別是否結束

表 A-9 感測模組頁面第三階段的基本資料

第四階段收到的資料與第三階段一模一樣，但只更新狀態類 (輸入類) 資料。

接著介紹在感測模組頁面的傳值流程，傳值部分分成兩個部分，重新收值 (Reload) 及設定 (Set)。重新收值傳送的訊息如下：

1. 重新回到第一階段
2. 傳送 @sp，暫停發送訊號
3. 傳送 @tb (隨感測模組不同變動)，通知主控端重新進入感測模組頁面
4. 等待 500 ms
5. 傳 @ss，命令主控端開始傳值

設定部分，將目前由APP端所更改過的感測模組連接狀態傳給主控端，命令主控端以新的連接狀態為準。其傳輸的流程如下：

傳輸命令	說明
“@m*10”+data+”\r”	修改電池的第一筆特性資料
“@m*11”+data+”\r”	修改電池的第二筆特性資料
...	由arg判斷感測模組還有沒有這類資料，若沒有則進入下一種類型
“@m*20”+data+”\r”	修改電池的第一筆輸入資料
“@m*21”+data+”\r”	修改電池的第二筆輸入資料
...	由 arg 判斷感測模組還有沒有這類資料，若沒有則進入下一種類型
“@m*30”+data+”\r”	修改電池的第一筆輸出資料
“@m*31”+data+”\r”	修改電池的第二筆輸出資料
...	由 arg 判斷感測模組還有沒有這類資料，若沒有則結束

註：@m*的*隨著不同的感測模組變動

表 A-10 修改感測模組資料的流程

感測模組	命令
電池	@mb
揚聲器	@mz
LED	@ml
振動感測模組	@ma
聲納感測模組	@ms
溫溼度感測模組	@mt
瓦斯感測模組	@mg

表 A-11 各感測模組修改資料所要下的命令

B. 附錄 感測模組資料內容

了解了主控端的模式之後再回來看感測模組所有的資料。感測模組與主控端間的連接分三個步驟：

1. 基本資料
2. 資料格式
3. 資料內容

而每個感測模組最重要的資料分成兩種：基本資料及資料。資料又可分類為三種：輸入、輸出及特性。目前感測模組共有7種：電池、揚聲器、LED、陀螺儀、聲納、溫溼度、瓦斯偵測。接下來便列舉出每個模組所具有的基本資料、輸出、輸入及特性資料，在表格內最大值及最小值括號內的數值對應到回傳資料格式中的宣告下一欄位為最大值或最小值的資料。

B.1. 電池模組：

資料名稱	資料內容	資料長度
I^2C位址	20	2 位元組
基本資料長度	26	2 位元組
資料格式長度	36	2 位元組
回報輸入長度	5	2 位元組
設定輸出長度	0	2 位元組
回報特性長度	6	2 位元組
設定特性長度	6	2 位元組
公司代號	0	2 位元組
設備代號	0	2 位元組
產品代號	0	2 位元組
韌體版本	0	2 位元組
設備類別代號	0	2 位元組
保留位	0	2 位元組
保留位	0	2 位元組

表 B-1 電池模組的基本資料

資料名稱	資料位址	資料長度	最小值(位址)	最大值(位址)
休眠週期	5	2位元組	0 ms (10)	1024 ms (14)
低電量警告準位	17	2位元組	0% (10)	100% (14)

表 B-2 電池模組的特性資料

資料名稱	資料位址	資料長度	最小值(位址)	最大值(位址)
電池電量	5	2位元組	0 (10)	1024 (14)
低電量警報旗標	17	2位元組	0 (9)	100% (14)

表 B-3 電池模組的輸入資料

B.2. 揚聲器模組

資料名稱	資料內容	資料長度
I²C位址	21	2 位元組
基本資料長度	26	2 位元組
資料格式長度	74	2 位元組
回報輸入長度	3	2 位元組
設定輸出長度	4	2 位元組
回報特性長度	12	2 位元組
設定特性長度	12	2 位元組
公司代號	0	2 位元組
設備代號	0	2 位元組
產品代號	0	2 位元組
韌體版本	0	2 位元組
設備類別代號	0	2 位元組
保留位	0	2 位元組
保留位	0	2 位元組

表 B-4 揚聲器模組的基本資料

資料名稱	資料位址	資料長度	最小值(位址)	最大值(位址)
休眠週期	5	2 位元組	0 ms (10)	2048 ms (14)
音量	17	1 位元組	0% (9)	100% (13)
頻率	29	2 位元組	0 Hz (10)	5000 Hz (14)
樂曲	41	1 位元組	0 (9)	2 (13)
週期	53	2 位元組	0ms (10)	2048 ms (14)
占空比	65	1 位元組	0% (9)	100% (13)
警報時間	77	1 位元組	0s (9)	60s (13)

表 B-5 揚聲器模組的特性資料

資料名稱	資料位址	資料長度	最小值(位址)	最大值(位址)
執行旗標	5	1 位元組	0 (9)	1 (13)

表 B-6 揚聲器模組的輸入資料

資料名稱	資料位址	資料長度	最小值(位址)	最大值(位址)
開始執行旗標	5	1 位元組	0 (9)	1 (13)
停止執行旗標	17	1 位元組	0 (9)	1 (13)

表 B-7 揚聲器模組的輸出資料

B.3. LED模組

資料名稱	資料內容	資料長度
I^2C 位址	22	2 位元組
基本資料長度	26	2 位元組
資料格式長度	74	2 位元組
回報輸入長度	3	2 位元組
設定輸出長度	4	2 位元組
回報特性長度	12	2 位元組
設定特性長度	12	2 位元組
公司代號	0	2 位元組
設備代號	0	2 位元組
產品代號	0	2 位元組
韌體版本	0	2 位元組
設備類別代號	0	2 位元組
保留位	0	2 位元組
保留位	0	2 位元組

表 B-8 LED 模組的基本資料

資料名稱	資料位址	資料長度	最小值(位址)	最大值(位址)
休眠週期	5	2 位元組	0 ms (10)	1024 ms (14)
亮度	17	1 位元組	0% (9)	100% (13)
顏色	29	2 位元組	0Hz (10)	2048 (14)
閃爍方式	41	1 位元組	0 (9)	2 (13)
週期	53	2 位元組	0 ms (10)	2048 ms (14)
占空比	65	1 位元組	0% (9)	100% (13)
警報時間	77	1 位元組	0s (9)	60s (13)

表 B-9 LED 模組的特性資料

資料名稱	資料位址	資料長度	最小值(位址)	最大值(位址)
執行旗標	5	1位元組	0 (9)	1 (13)

表 B-10 LED 模組的輸入資料

資料名稱	資料位址	資料長度	最小值(位址)	最大值(位址)
開始執行旗標	5	1位元組	0 (9)	1 (13)
停止執行旗標	17	1位元組	0 (9)	1 (13)

表 B-11 LED 模組的輸出資料

B.4. 陀螺儀模組

資料名稱	資料內容	資料長度
I^2C 位址	23	2 位元組
基本資料長度	26	2 位元組
資料格式長度	34	2 位元組
回報輸入長度	5	2 位元組
設定輸出長度	0	2 位元組
回報特性長度	5	2 位元組
設定特性長度	5	2 位元組
公司代號	0	2 位元組
設備代號	0	2 位元組
產品代號	0	2 位元組
韌體版本	0	2 位元組
設備類別代號	0	2 位元組
保留位	0	2 位元組
保留位	0	2 位元組

表 B-12 陀螺儀模組的基本資料

資料名稱	資料位址	資料長度	最小值(位址)	最大值(位址)
休眠週期	5	2位元組	0 ms (10)	1024 ms (14)
振動警告準位	17	1位元組	0% (9)	10 (13)

表 B-13 陀螺儀模組的特性資料

資料名稱	資料位址	資料長度	最小值(位址)	最大值(位址)
振動感測值	5	2位元組	0 (10)	720度 (14)
振動警報旗標	17	1位元組	0 (9)	1 (13)

表 B-14 陀螺儀模組的輸入資料

B.5. 聲納模組

資料名稱	資料內容	資料長度
I^2C 位址	24	2 位元組
基本資料長度	26	2 位元組
資料格式長度	36	2 位元組
回報輸入長度	5	2 位元組
設定輸出長度	0	2 位元組
回報特性長度	6	2 位元組
設定特性長度	6	2 位元組
公司代號	0	2 位元組
設備代號	0	2 位元組
產品代號	0	2 位元組
韌體版本	0	2 位元組
設備類別代號	0	2 位元組
保留位	0	2 位元組
保留位	0	2 位元組

表 B-15 聲納模組的基本資料

資料名稱	資料位址	資料長度	最小值 (位址)	最大值 (位址)
休眠頻率	5	2 位元組	0 ms (10)	1024 ms (14)
距離警告準位	17	2 位元組	0 cm (10)	200 cm (14)

表 B-16 聲納模組的特性資料

資料名稱	資料位址	資料長度	最小值 (位址)	最大值 (位址)
聲納感測值	5	2 位元組	0 (10)	400 cm (14)
聲納警報旗標	17	1 位元組	0 (9)	1 (13)

表 B-17 聲納模組的輸入資料

B.6. 溫溼度模組

資料名稱	資料內容	資料長度
I²C 位址	25	2 位元組
基本資料長度	26	2 位元組
資料格式長度	58	2 位元組
回報輸入長度	8	2 位元組
設定輸出長度	0	2 位元組
回報特性長度	8	2 位元組
設定特性長度	8	2 位元組
公司代號	0	2 位元組
設備代號	0	2 位元組
產品代號	0	2 位元組
韌體版本	0	2 位元組
設備類別代號	0	2 位元組
保留位	0	2 位元組
保留位	0	2 位元組

表 B-18 溫溼度模組的基本資料

資料名稱	資料位址	資料長度	最小值(位址)	最大值(位址)
休眠頻率	5	2 位元組	0 ms (10)	1024 ms (14)
溫度警告準位	17	2 位元組	0度 (10)	100度 (14)
濕度警告準位	29	2 位元組	0 % (10)	100% (14)

表 B-19 溫溼度模組的特性資料

資料名稱	資料位址	資料長度	最小值(位址)	最大值(位址)
溫度感測值	5	2 位元組	0 (10)	100度 (14)
濕度感測值	17	2 位元組	0 (10)	100% (14)
溫度警報旗標	29	1 位元組	0 (9)	1 (13)
濕度警報旗標	41	1 位元組	0 (9)	1 (13)

表 B-20 溫溼度模組的輸入資料

B.7. 瓦斯偵測模組

資料名稱	資料內容	資料長度
I^2C 位址	24	2 位元組
基本資料長度	26	2 位元組
資料格式長度	36	2 位元組
回報輸入長度	5	2 位元組
設定輸出長度	0	2 位元組
回報特性長度	6	2 位元組
設定特性長度	6	2 位元組
公司代號	0	2 位元組
設備代號	0	2 位元組
產品代號	0	2 位元組
韌體版本	0	2 位元組
設備類別代號	0	2 位元組
保留位	0	2 位元組
保留位	0	2 位元組

表 B-21 瓦斯偵測模組的基本資料

資料名稱	資料位址	資料長度	最小值(位址)	最大值(位址)
休眠頻率	5	2 位元組	0 ms (10)	1024 ms (14)
瓦斯警告準位	17	2 位元組	0 cm (10)	100% (14)

表 B-22 瓦斯偵測模組的特性資料

資料名稱	資料位址	資料長度	最小值(位址)	最大值(位址)
瓦斯感測值	5	2 位元組	0 (10)	100% (14)
瓦斯警報旗標	17	1 位元組	0 (9)	1 (13)

表 B-23 瓦斯偵測模組的輸入資料

REVISION HISTORY

Date	Revision	Description
2016.05.22	1.00	

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, “Insecure Usage”.

Insecure usage includes, but is not limited to： equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer’s risk, and in the event that third parties lay claims to Nuvoton as a result of customer’s Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.