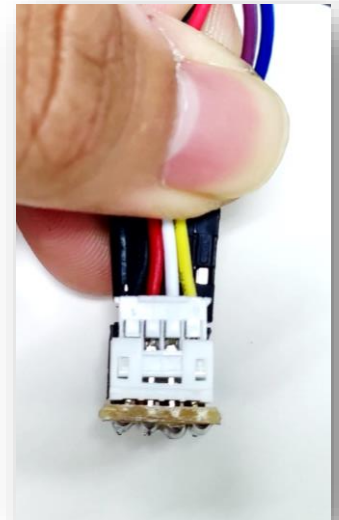
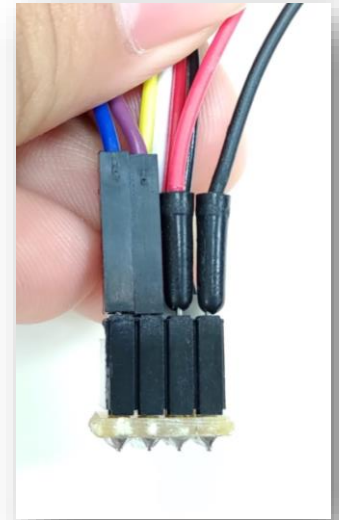


Lab7

- TIMER, I2C

A Little Reminder

- [課程教材連結](#)
- [影片講解連結](#)
- 章節以及目錄可以多加使用
- 黃、藍、綠分別表示三個Lab
- 連接模組時可以使用轉接頭



目錄

Lecture 1-Ultra
Sonic Ranger

Lecture 2-LCD

Lecture 1 – Ultra Sonic Ranger

- 複習 Timer、GPIO的使用
- 應用 Timer 透過超音波測距模組得到距離

Lecture 1

— 概覽

超音波測距模組介紹 – P.6

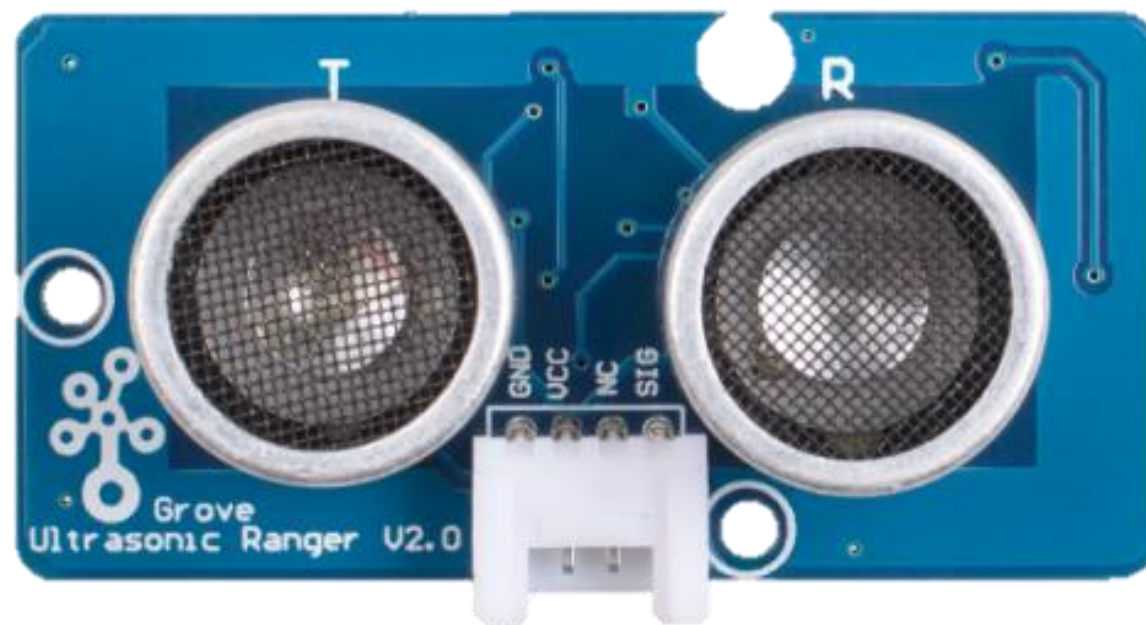
Timer 回顧 – P.13

GPIO與print回顧 – P.22

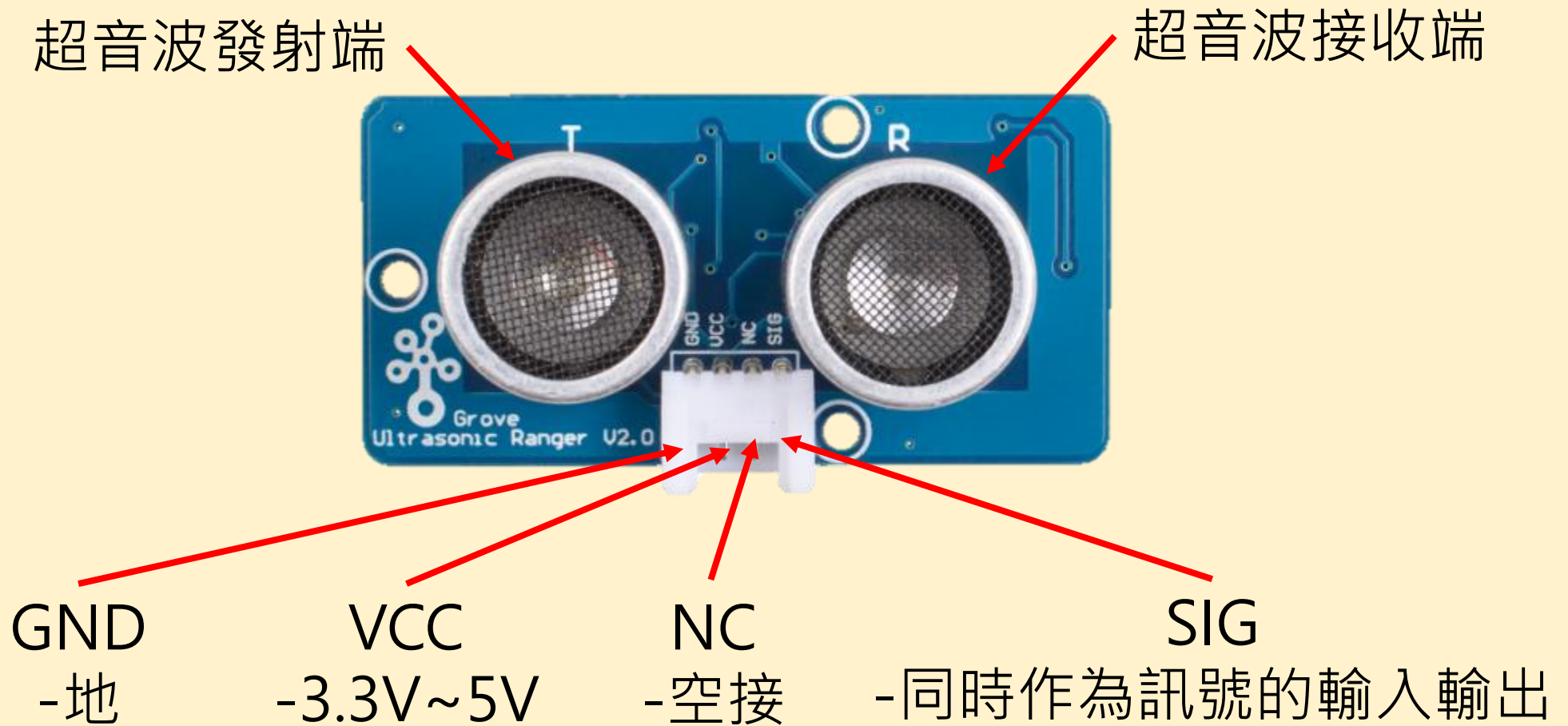
動手實作 – P.26

超音波測距模組

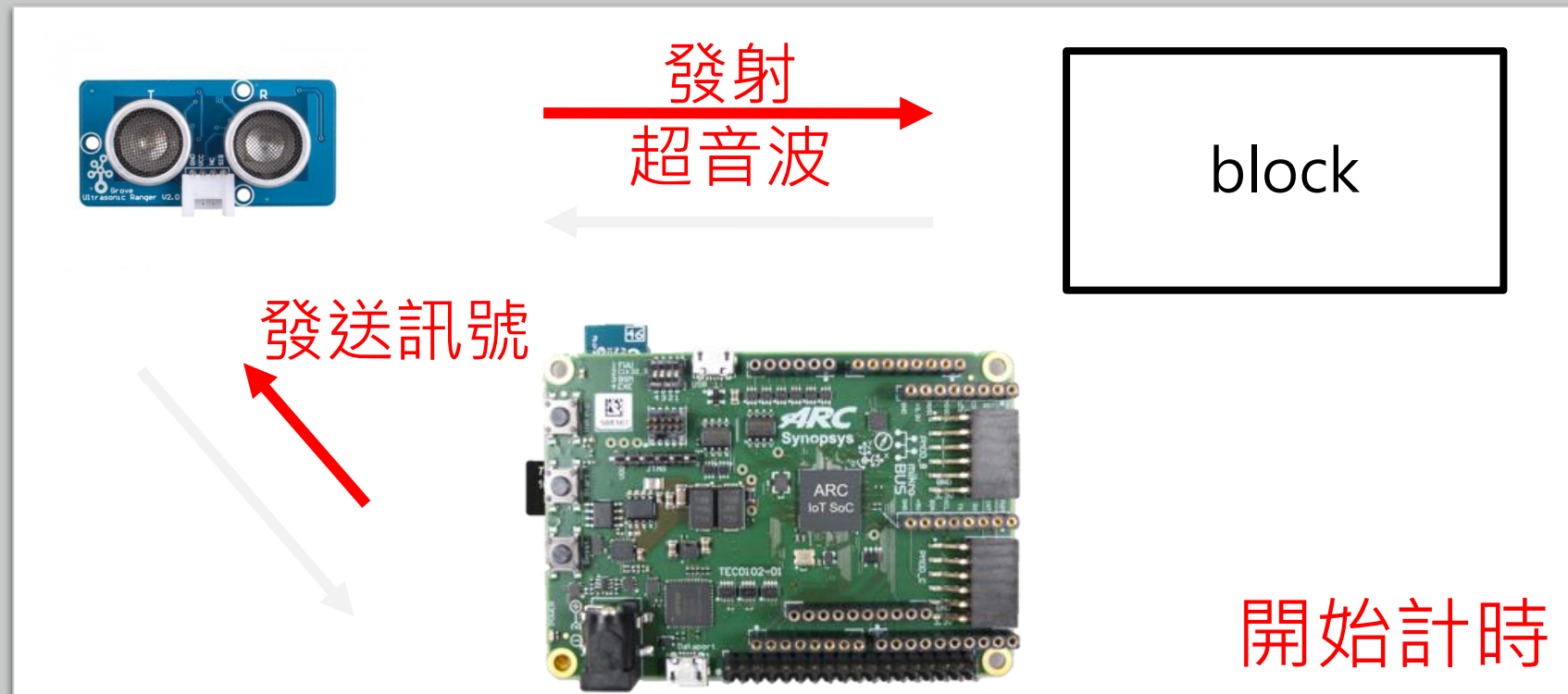
- 提供測量距離使用
- 官方測量範圍：3cm~350cm
- 工作電壓：3.3V~5V



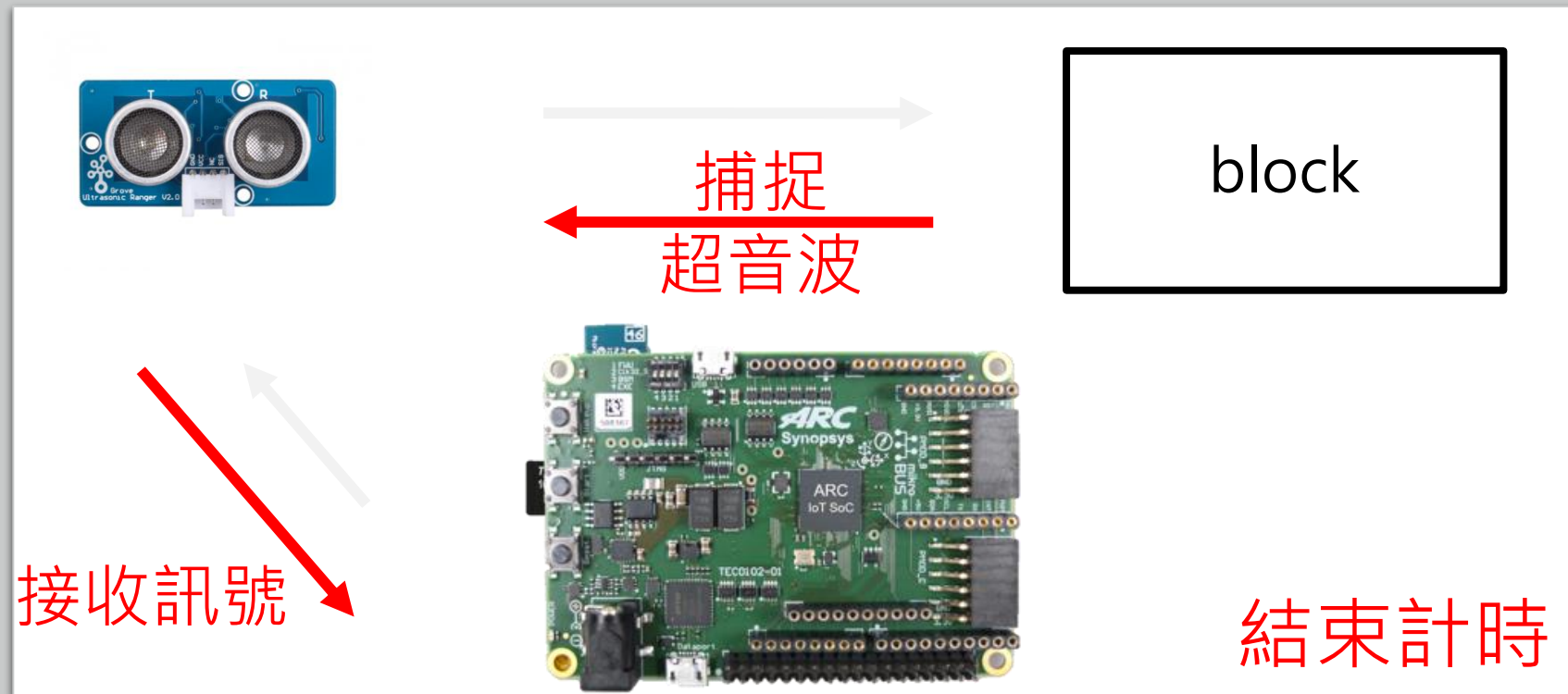
超音波測距模組-元件功能



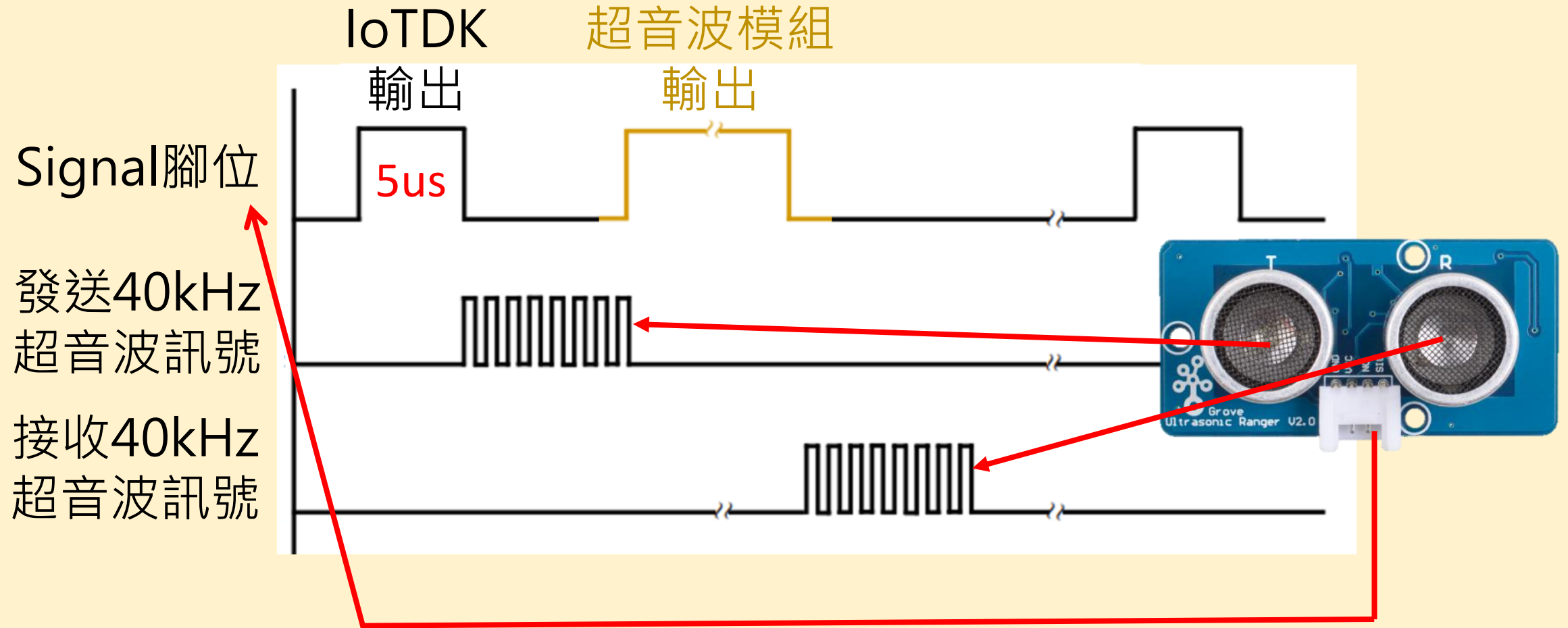
超音波測距 - 原理(1/3)



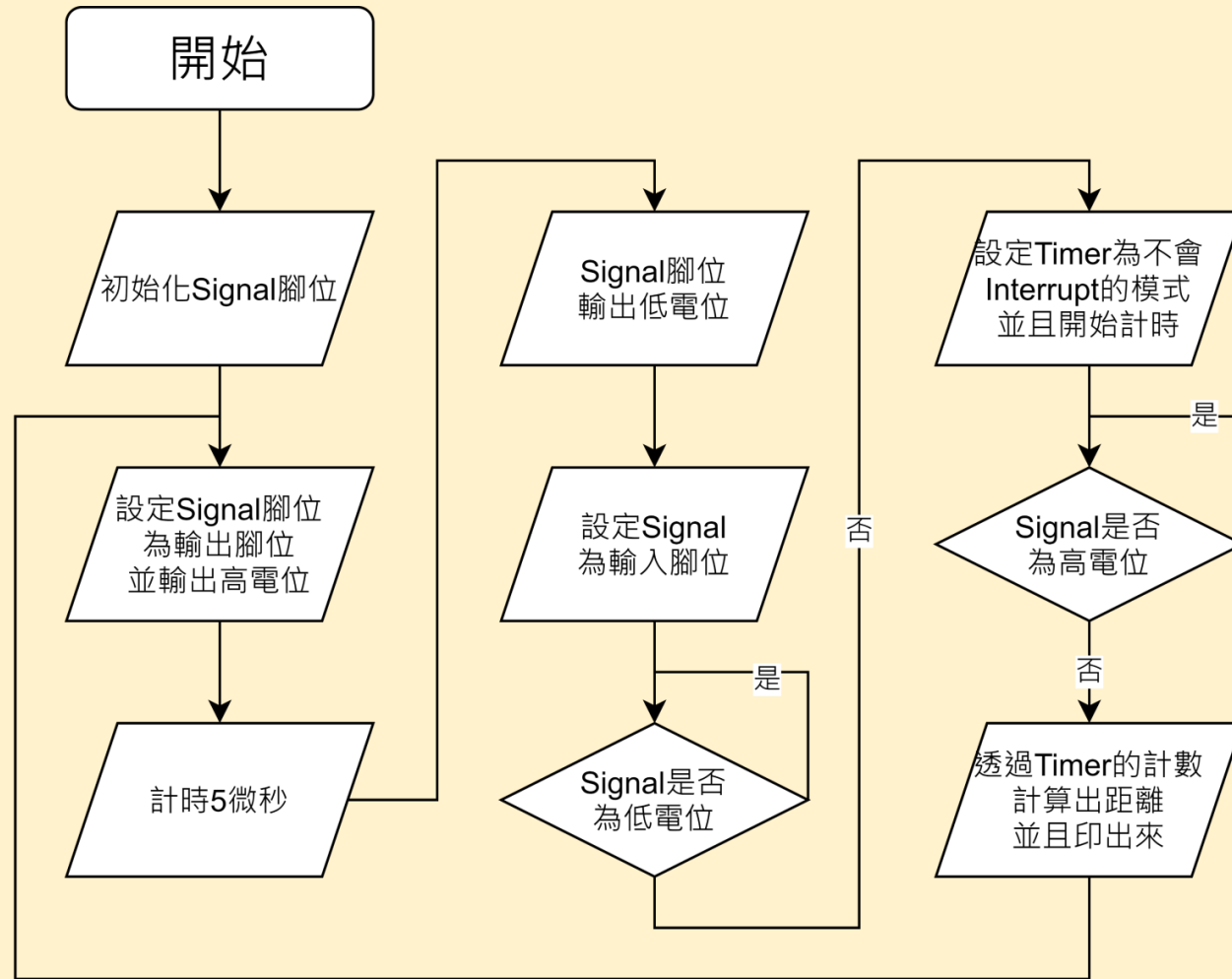
超音波測距 - 原理(2/3)



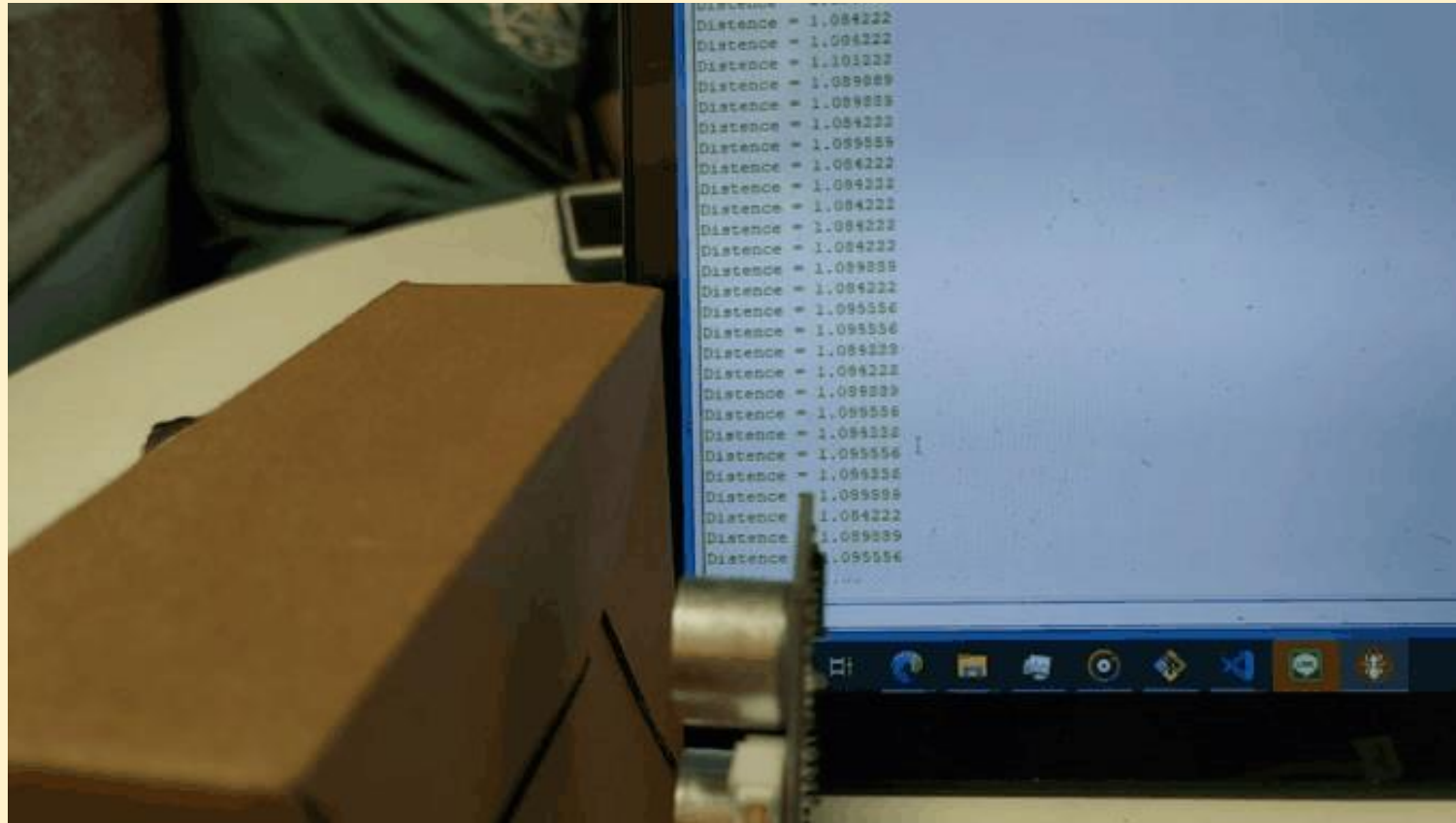
超音波測距 - 原理(3/3)




超音波測距－實驗目標



超音波測距－結果預覽





溫故知新

- 回憶Lab6中的timer

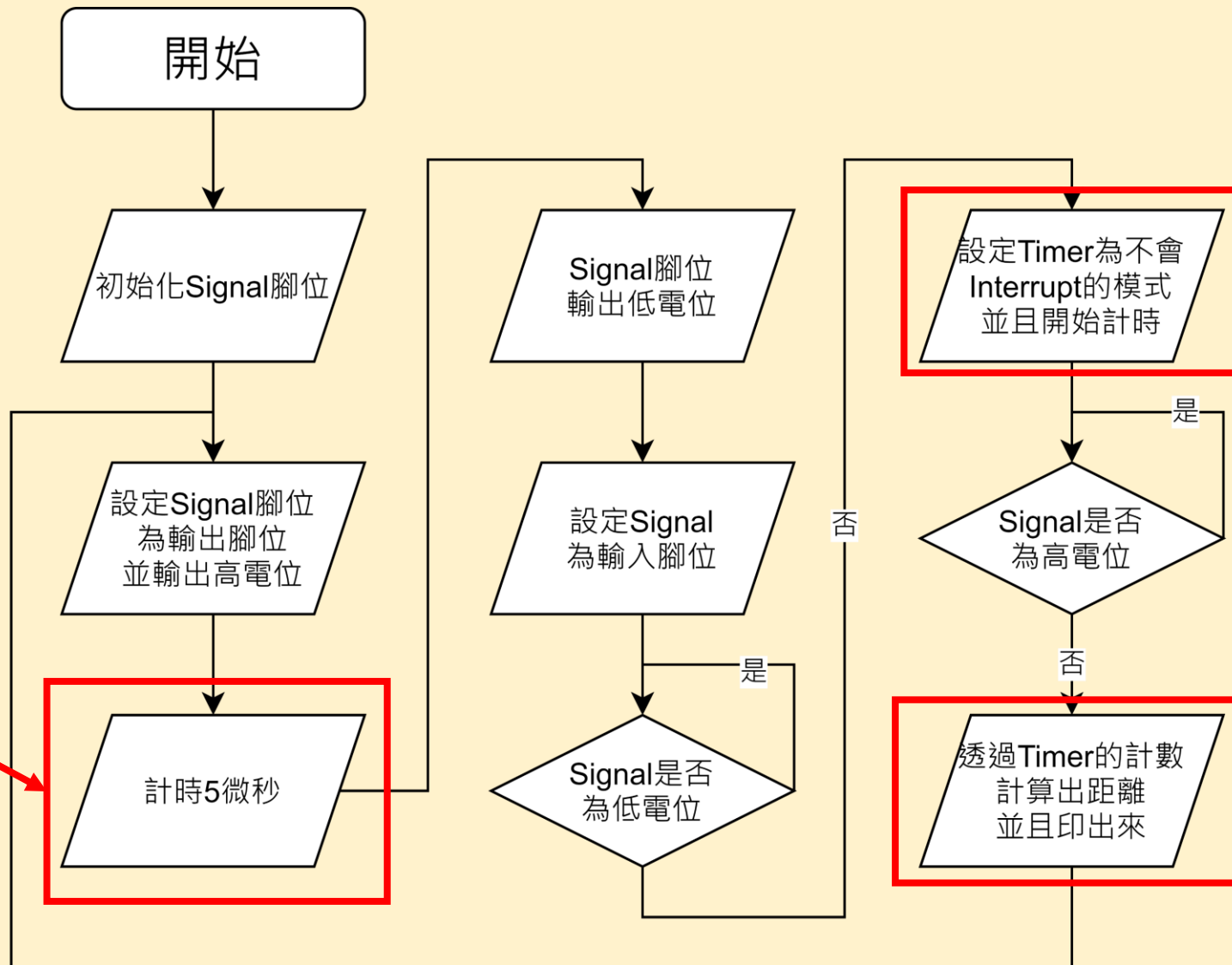
Timer回顧

- Timer 是 IoTDK 內建的暫存器
- 每個CPU Cycle會增加1
- 可以選擇在到達某數值後給予CPU中斷訊號



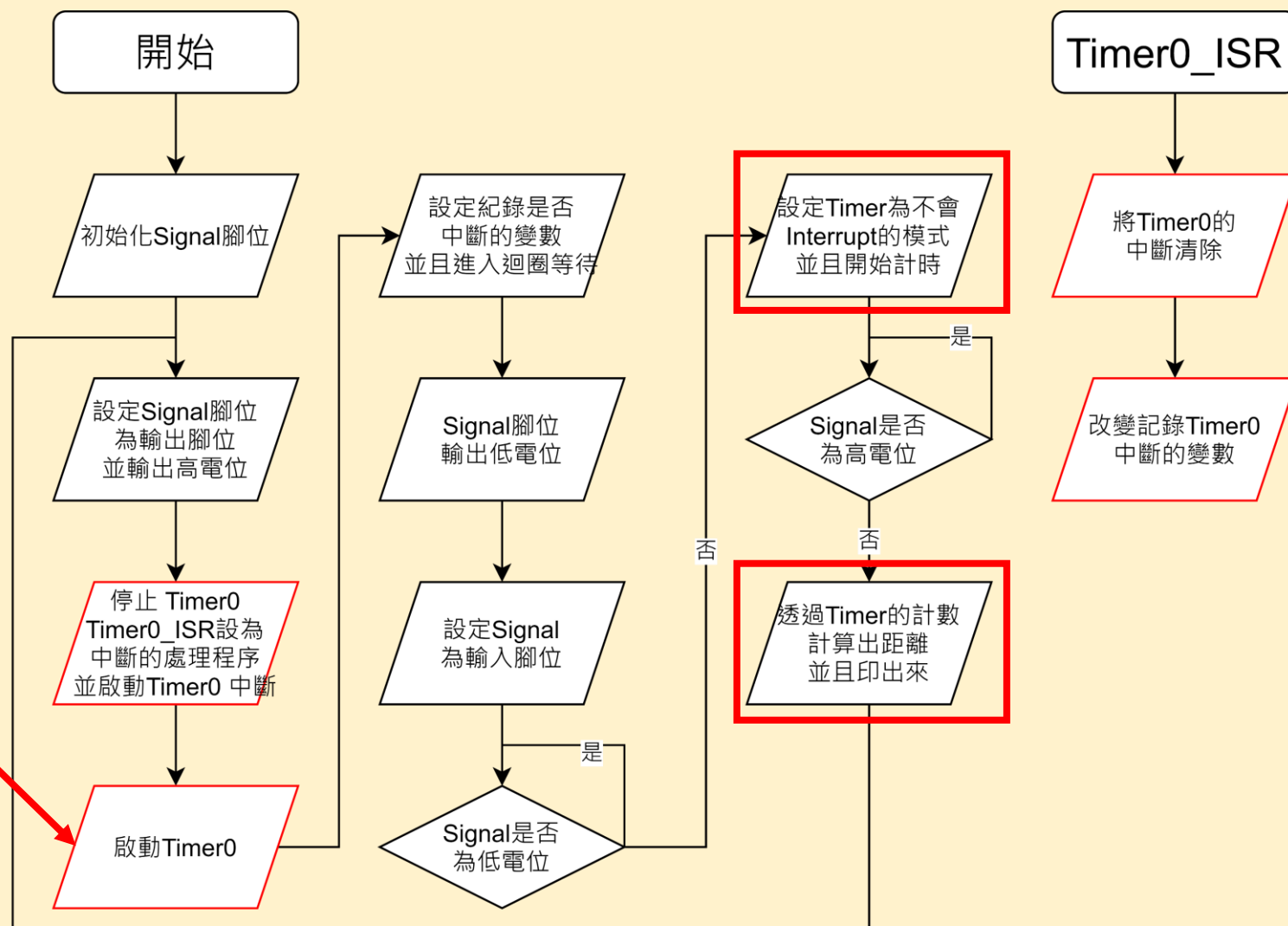
使用Timer的時機

將這個行為展開



使用Timer的時機

144M Cycles/s
→ ?? Cycles/5us



IoTDK上的Timer(1/5) – inc/arc/arc_exception.h

- Interrupt 相關函數

- 為了arc_timer.h裡面的time_int_clear以及模式中所設定的TIMER_CTRL_IE，需要先開啟Timer的中斷功能，才能讓Timer在計數到上限值時中斷，而以下的int_handler_install則可以設定在Timer0 interrupt發生時的ISR，而int_enable可以讓Timer0 interrupt開啟。

```
537 extern void cpu_unlock_restore(const uint32_t status);  
538 extern int32_t int_handler_install(const uint32_t intno, INT_HANDLER handler);  
539 extern INT_HANDLER int_handler_get(const uint32_t intno);
```

```
524 extern int32_t int_disable(const uint32_t intno);  
525 extern int32_t int_enable(const uint32_t intno);  
526 extern int32_t int_enabled(const uint32_t intno);
```

停止 Timer0
Timer0_ISR設為
中斷的處理程序
並啟動Timer0 中斷

IoTDK上的Timer(2/5) – inc/arc/arc_exception.h

- Interrupt 參數定義

- 而在上述兩個函式中所看到的**const uint32_t intno**，則可以用**INTNO_TIMER0**參數帶入，代表著Timer0的中斷在中斷表上的號碼。

```
61     #else
62     #define INTNO_TIMER0 16
63     #endif
```

IoTDK上的Timer(3/5) – inc/arc/arc_timer.h

- Timer 控制函數
 - 這次主要會用到的是timer_start, timer_stop, timer_current, time_int_clear
 - 分別是啟動、停止跟讀取現在的計數值
 - 以及清除目前的中斷訊號

```
106 extern int32_t timer_present(const uint32_t no);
107 extern int32_t timer_start(const uint32_t no, const uint32_t mode, const uint32_t val);
108 extern int32_t timer_stop(const uint32_t no);
109 extern int32_t timer_current(const uint32_t no, void* val);
110 extern int32_t timer_int_clear(const uint32_t no);
111 extern void timer_init(void);
112 extern void arc_delay_us(uint32_t usecs);
```

IoTDK上的Timer(4/5) – inc/arc/arc_timer.h

- Timer 參數定義
 - 共有TIMER_0/TIMER_1可供使用
 - 當看到uint32_t no這樣的參數就可以填入**TIMER0**

```
52  #define TIMER_0      0    /*!< macro name for arc internal timer 0 */
53  #define TIMER_1      1    /*!< macro name for arc internal timer 1 */
54  #define TIMER_RTC     2    /*!< macro name for arc internal RTC */
```

- Timer 模式定義
 - 共有4種模式
 - 當看到uint32_t mode這樣的參數就可以填入

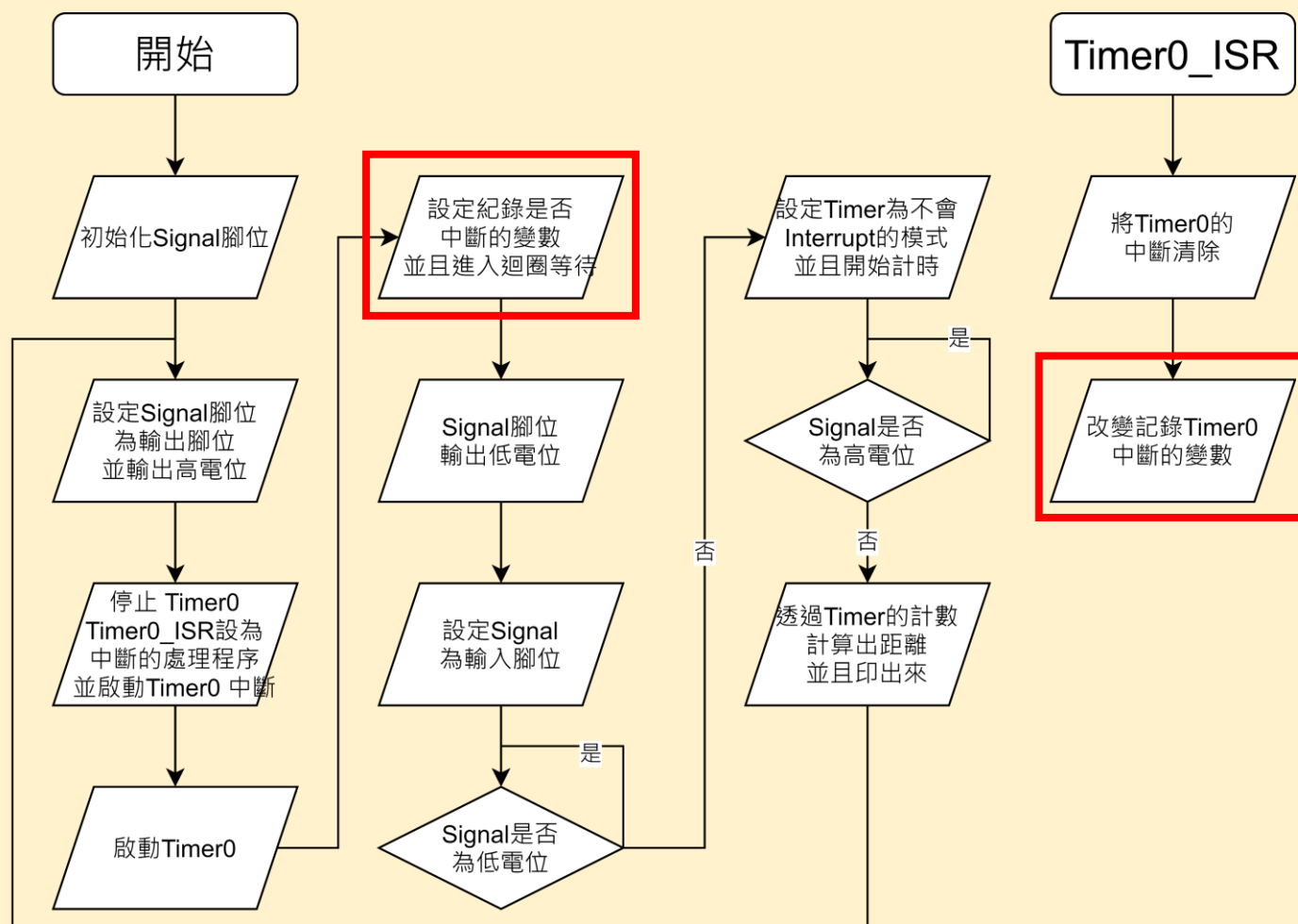
```
95  #define TIMER_CTRL_IE      (1 << 0)    /*!< Interrupt when count reaches limit */
96  #define TIMER_CTRL_NH      (1 << 1)    /*!< Count only when CPU NOT halted */
97  #define TIMER_CTRL_W       (1 << 2)    /*!< watchdog enable */
98  #define TIMER_CTRL_IP      (1 << 3)    /*!< interrupt pending */
```

IoTDK上的Timer(5/5) - volatile

volatile – 修飾詞
強制compiler編譯出讓
CPU從記憶體拿變數的指令，
避免對該變數的優化。

使用方式：

```
34  */
35  volatile bool
36  /**
```



溫故知新

- GPIO操控與UART輸出函式

IoTDK上的GPIO(1/3) –

device/ip/ip_hal/inc/dev_gpio.h,

board/iotdk/drivers/ip/subsystem/gpio/dfss_gpio_obj.h

- GPIO 初始化函數

- 先使用gpio_get_dev，並且把得到的DEV_GPIO_PTR存下來

```
499  */
500  extern DEV_GPIO_PTR gpio_get_dev(int32_t gpio_id);
501
```

- GPIO 初始化參數

- 有下列的GPIO埠可以選擇，需要將其填入int32_t gpio_id中。

```
41  #define DFSS_GPIO_8B0_ID      14  /* GPIO 8 ID macro (io_gpio_8b0) */
42  #define DFSS_GPIO_8B1_ID      15  /* GPIO 9 ID macro (io_gpio_8b1) */
43  #define DFSS_GPIO_8B2_ID      16  /* GPIO 10 ID macro (io_gpio_8b2) */
44  #define DFSS_GPIO_8B3_ID      17  /* GPIO 11 ID macro (io_gpio_8b3) */
45  #define DFSS_GPIO_4B0_ID      18  /* GPIO 4 ID macro (io_gpio_4b0) */
46  #define DFSS_GPIO_4B1_ID      19  /* GPIO 5 ID macro (io_gpio_4b1) */
47  #define DFSS_GPIO_4B2_ID      20  /* GPIO 6 ID macro (io_gpio_4b2) */
48  #define DFSS_GPIO_4B3_ID      21  /* GPIO 7 ID macro (io_gpio_4b3) */
```


IoTDK上的GPIO(2/3) – device/ip/ip_hal/inc/dev_gpio.h

- GPIO 控制函數
 - 對DEV_GPIO_PTR指向的物件進行操作

```
417 typedef struct dev_gpio {  
418     DEV_GPIO_INFO gpio_info; /*!< gpio device informati  
419     int32_t (*gpio_open) (uint32_t dir); /*!< open gpio dev  
420     int32_t (*gpio_close) (void); /*!< close gpio device  
421     int32_t (*gpio_control) (uint32_t ctrl_cmd, void *param);  
422     int32_t (*gpio_write) (uint32_t val, uint32_t mask); /*  
423     int32_t (*gpio_read) (uint32_t *val, uint32_t mask); /*  
424 } DEV_GPIO, * DEV_GPIO_PTR;
```

- GPIO 控制參數

```
177 /*  
178 #define GPIO_CMD_SET_BIT_DIR_INPUT      DEV_SET_SYSCMD(0)  
179 #define GPIO_CMD_SET_BIT_DIR_OUTPUT    DEV_SET_SYSCMD(1)  
180 /**
```

Port[x] → 1 < x: 輸出
0 < x: 輸入

Port[x] → 1 < x: 選擇腳位

Port[x] → 1 < x: 高電位
0 < x: 低電位

IoTDK上的printf(3/3) — inc/embARC_debug.h

- EMBARC_PRINTF/printf 可以做為格式化輸出的函式
 - 使用EMBARC_PRINTF(“%f”, variable_name);
 - 或是printf(“%f”, variable_name);

```
49  #ifndef EMBARC_PRINTF
50      #ifdef MID_COMMON
51          #include "xprintf.h"
52          #define EMBARC_PRINTF xprintf
53      #else
54          #include <stdio.h>
55          #define EMBARC_PRINTF printf
56      #endif
57 #endif
```



自行嘗試時間 Lecture3-UltraSonic

超音波測距-軟體設定

- 建議檔案位置

./embarc_osp/Lab1-UltraSonic-Practice

- 如果不想設定Timer上限，在timer_start時將val設為MAX_COUNT即可

```
7  /* Define Max Number for 32-bit Integer */
8  #define MAX_COUNT 0xffffffff
9
```

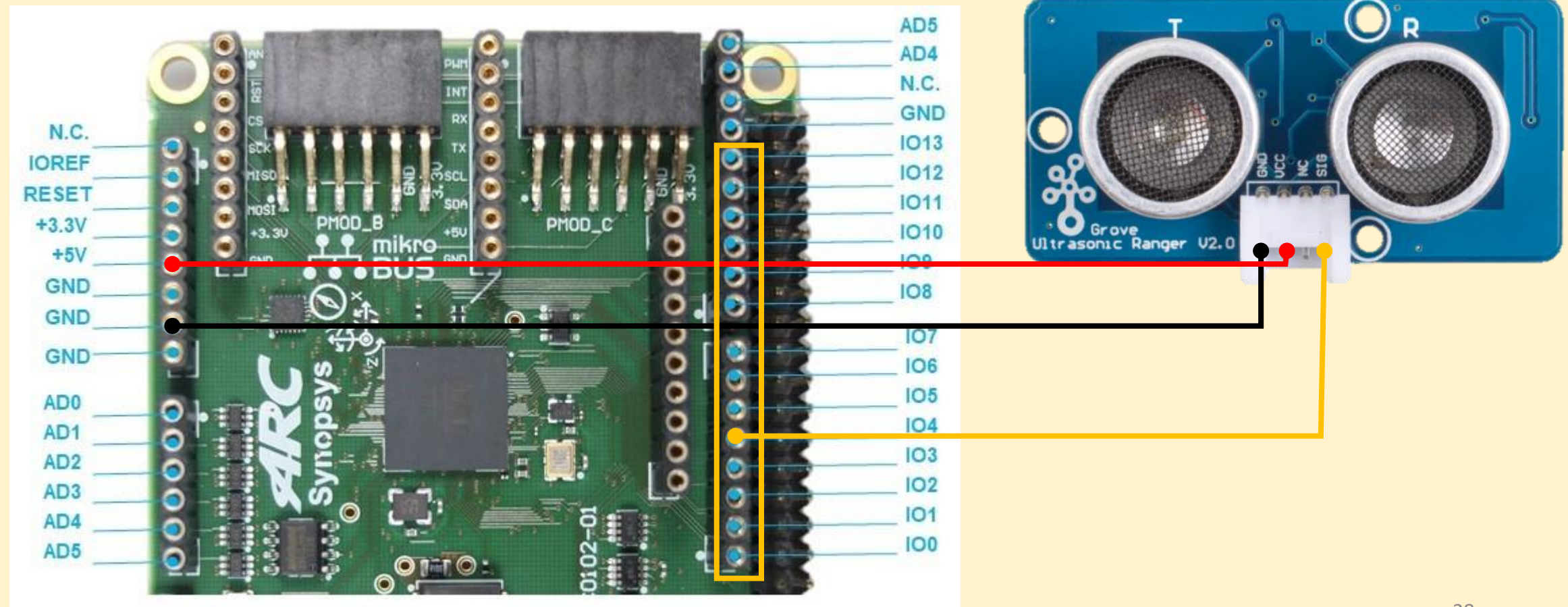
- 在程式裡已經有宣告Timer0_ISR的原型，更改main.c#146的函式即可

```
40  /* 宣告Timer0_ISR的原型 */
41  void Timer0_ISR();
42
```

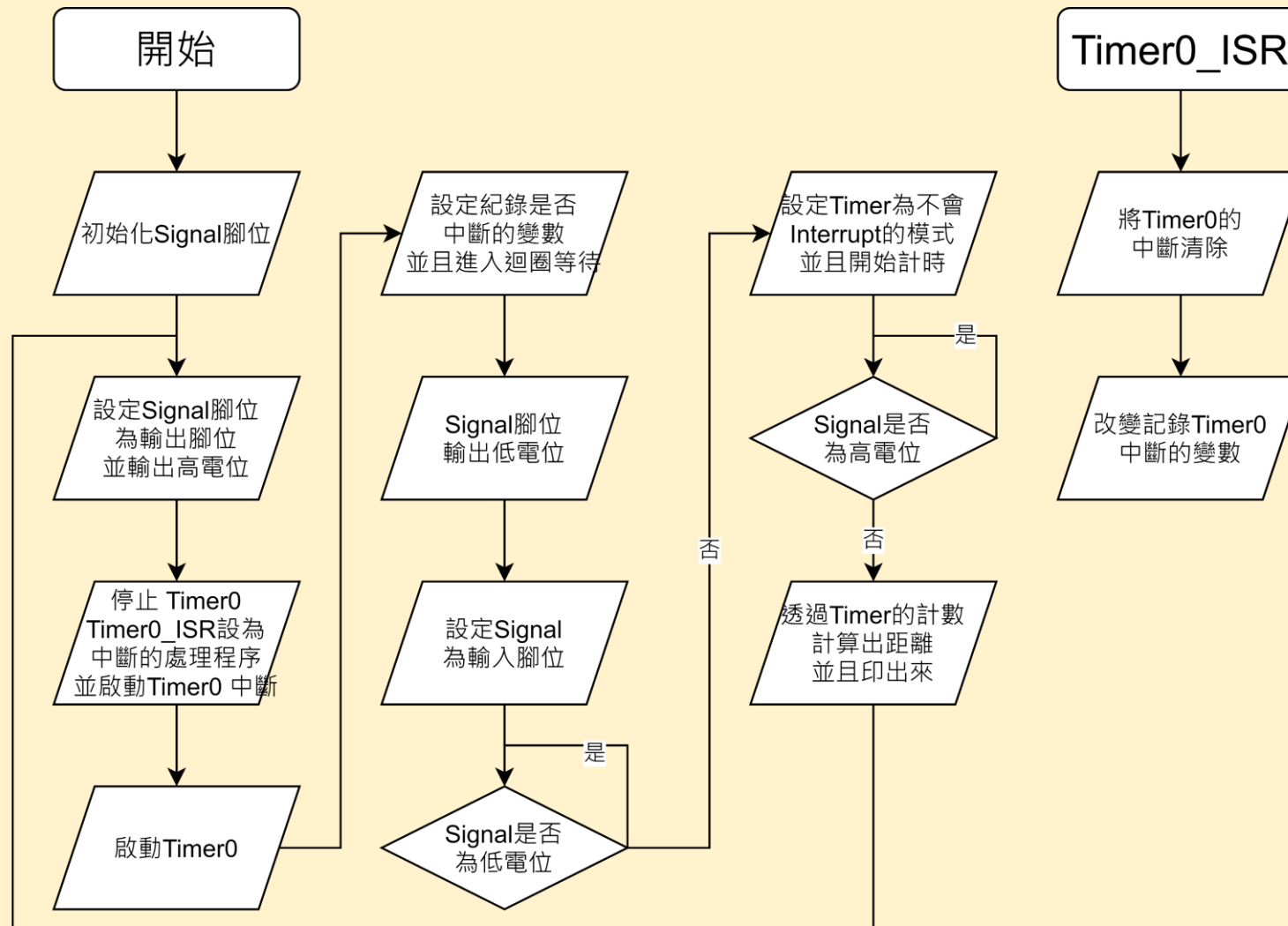
```
148 void Timer0_ISR() {
149     /**
150      * 先將TIMER_0的 interrupt 清除
151      * 設定該設定的變數
152      * {
153      */
154
155     /**
156      * }
157      */
158 }
159
160
```

超音波測距-硬體連接

- 建議硬體連接-IO4經過使用測試



超音波測距-程式流程



Lecture 2 – LCD

- 了解I2C協定
- 學會如何使用 LCD
- 透過I2C API控制LCD

Lecture2

－ 概覽

LCD模組介紹 – P.32

I2C 簡介 – P.36

動手實作- P.45

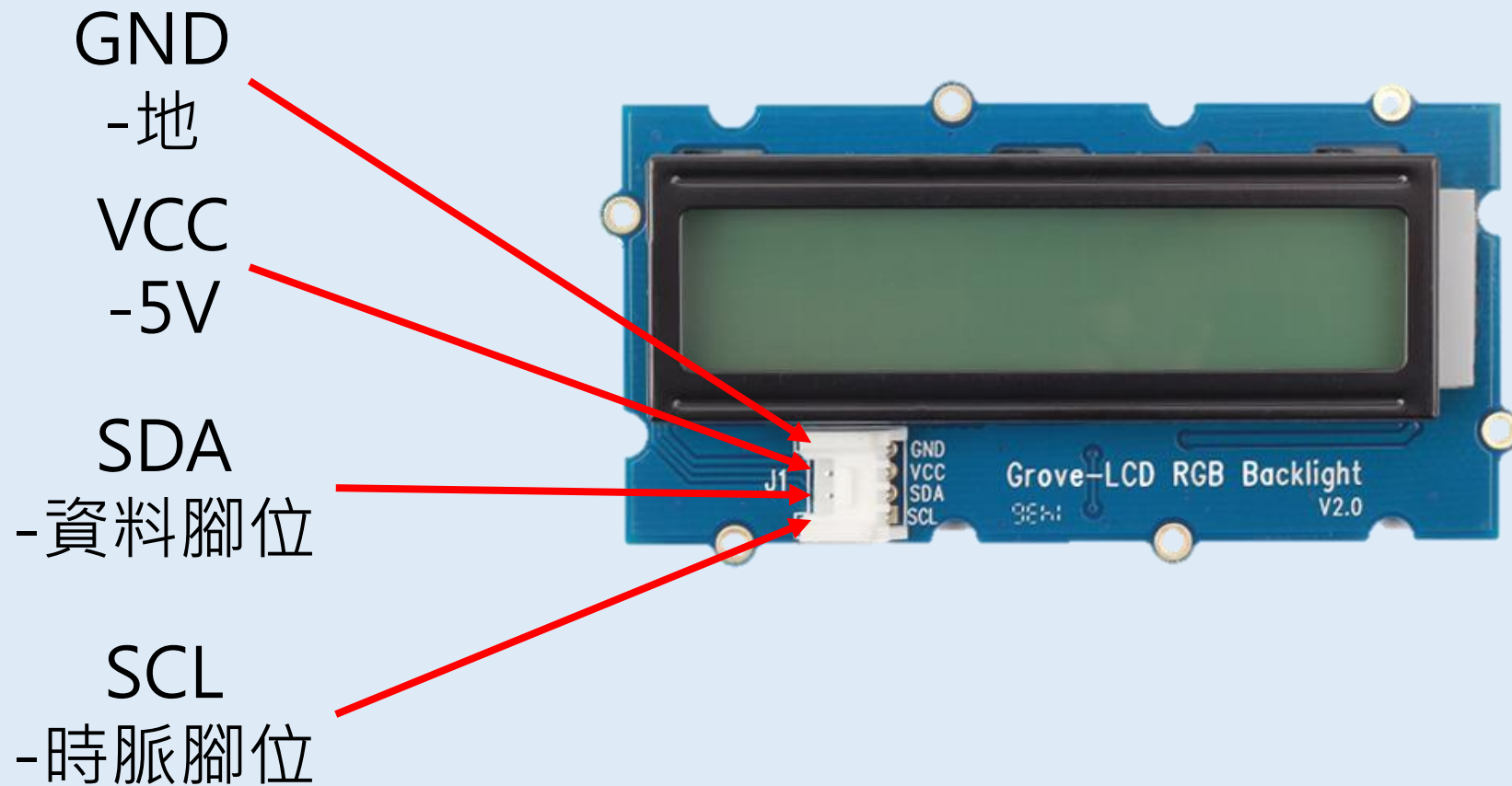
lic1602lcd說明 – P.57

LCD模組(1/4)-簡介

- 提供顯示資訊使用
- 共有 2x16 個字元可顯示
- 具有三色背光



LCD模組(2/4)-腳位介紹



LCD模組(3/4)-RGB背光



LCD模組(4/4)-共2x16字元顯示

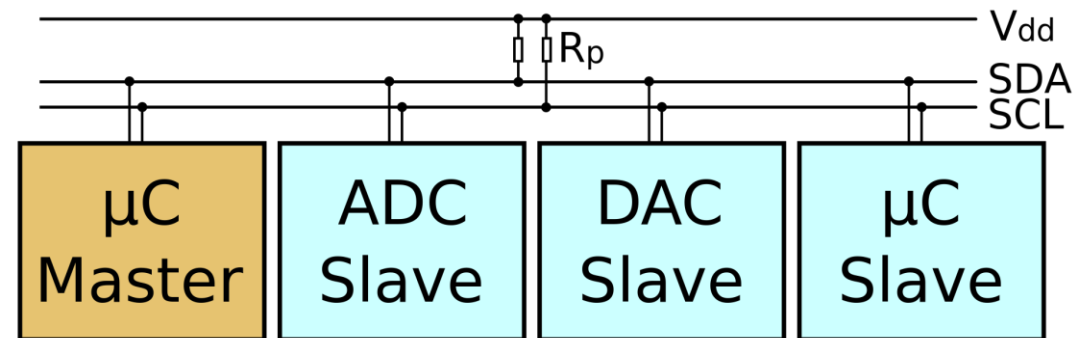




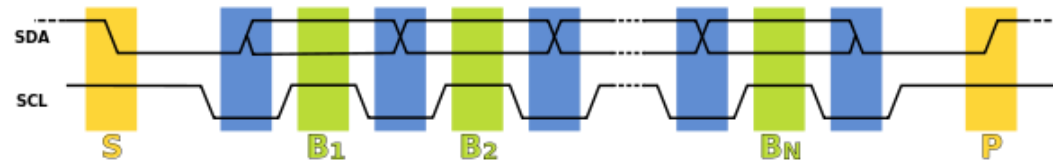
如何操縱LCD模組呢？

I2C簡介

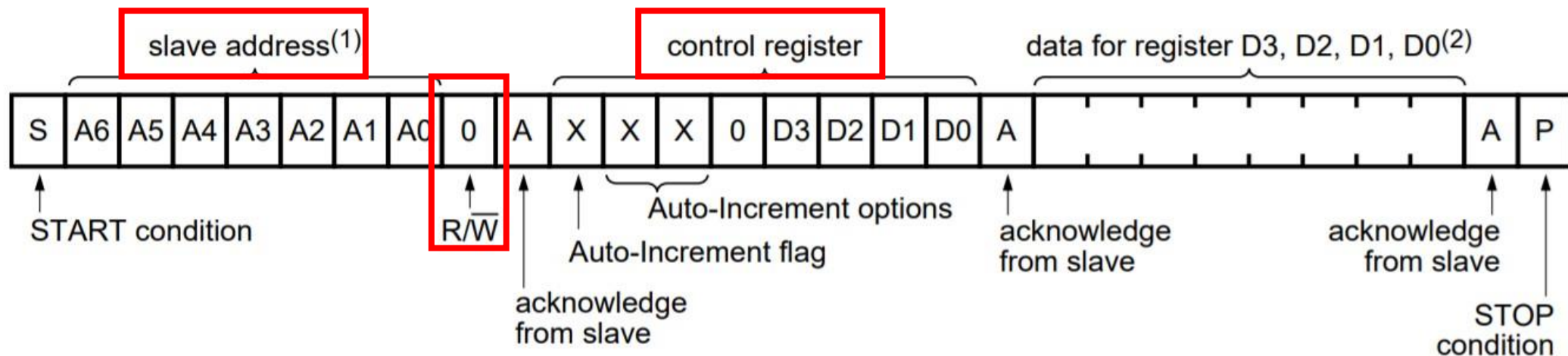
- Inter-Integrated Circuit的縮寫
- 可以有多個Slave與Master連接上線，不過同時只能有一組Master與Slave相互溝通
- 所有連接線都為Wired And – 只要有其中一個裝置為低電位，整條傳輸線皆為低電位



<https://zh.wikipedia.org/wiki/I%C2%B2C#/media/File:I2C.svg>



https://upload.wikimedia.org/wikipedia/commons/thumb/6/64/I2C_data_transfer.svg/600px-I2C_data_transfer.svg.png



002aab418

https://files.seeedstudio.com/wiki/Grove_LCD_RGB_Backlight/res/PCA9633.pdf

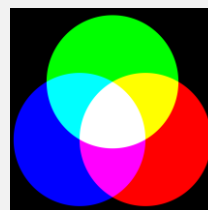
分辨溝通對象

- 透過Slave Address區分與誰溝通
- 透過Control Register值來區分控制項目
- 最後再把控制所需的值傳入

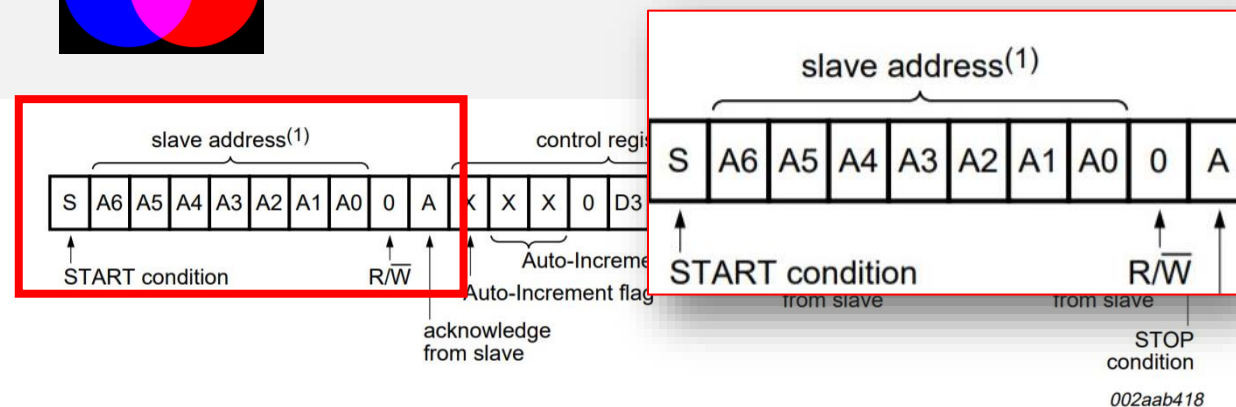
兩個分開的 控制器



LCD顯示面板 – 0x3E



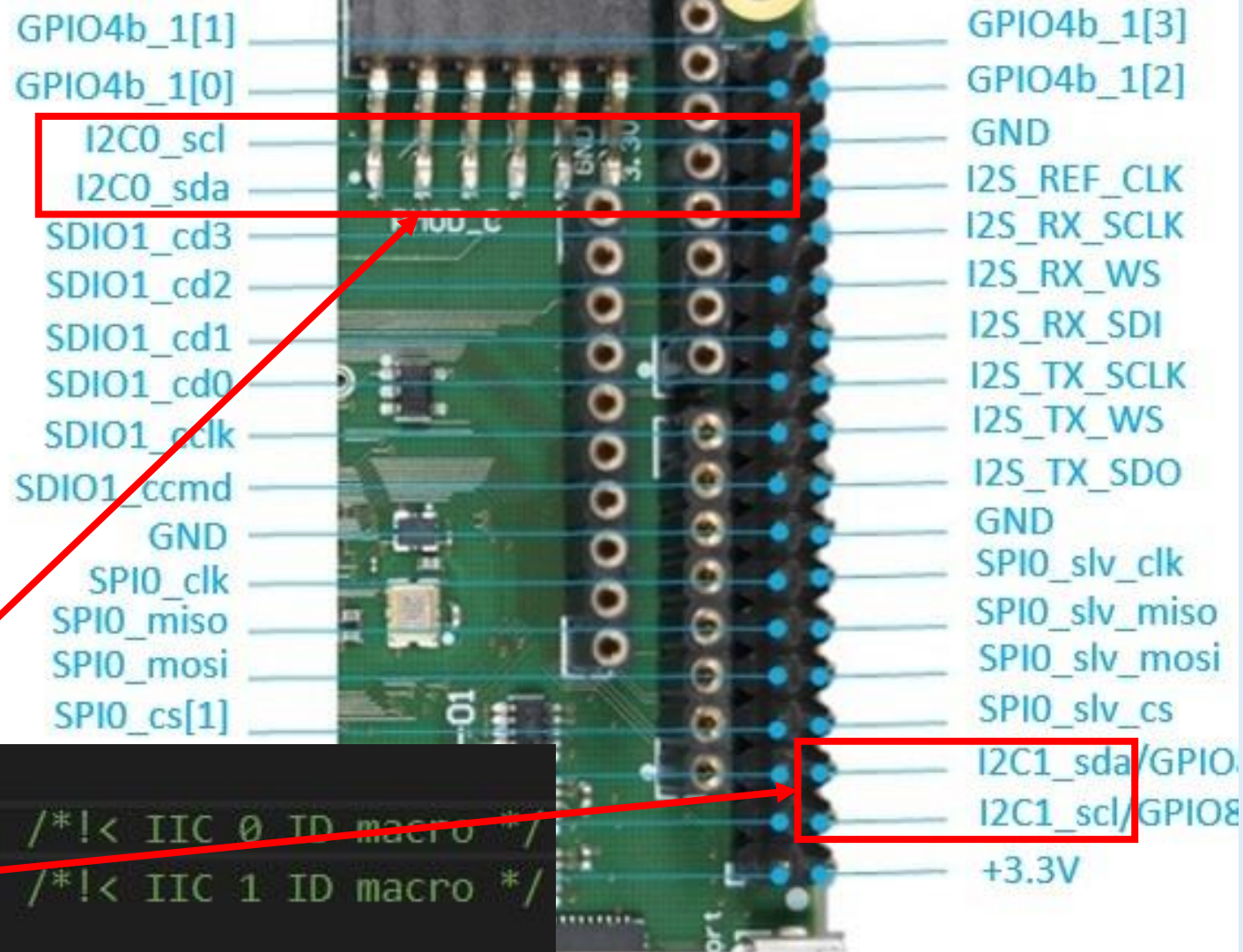
RGB LED背光 – 0x62



IoTDK上的I2C(1/5)

board/iotdk/drivers/ip/subsystem/iic/dfss_iic_obj.h

```
40
41 #define DFSS_IIC_0_ID 0 /*!< IIC 0 ID macro */
42 #define DFSS_IIC_1_ID 1 /*!< IIC 1 ID macro */
43
```



IoTDK上的I2C(2/5) – device/ip/ip_hal/inc/dev_iic.h

- 得到可操控I2C物件的方式 – **iic_get_dev**
 - 由於I2C包含許多控制暫存器，因此OSP裡他被封裝變成**DEV_IIC**
 - 我們傳入上一頁的**DFSS_IIC_x_ID**便可以得到其**DEV_IIC_PTR**的指標

```
510
511  /**
512   * \brief  get an \ref dev_iic "iic device" by iic device id.
513   * For how to use iic device hal refer to \ref DEVICE_HAL_IIC_DEVSTRUCT "Fu
514   * \param[in]  iic_id  id of iic, defined by user
515   * \retval  !NULL  pointer to an \ref dev_iic "iic device structure"
516   * \retval  NULL   failed to find the iic device by \ref iic_id
517   * \note      need to implemented by user in user code
518   */
519  extern DEV_IIC_PTR iic_get_dev(int32_t iic_id);
520
```

IoTDK上的I2C(3/5) – device/ip/ip_hal/inc/dev_iic.h

- DEV_IIC內含的函式

```

425 typedef struct dev_iic {
426     DEV_IIC_INFO iic_info;                /*!< iic device information */
427     int32_t (*iic_open) (uint32_t mode, uint32_t param);    /*!< open iic device in master/slave mode, \
428     |                                     when in master mode, param stands for speed mode, \
429     |                                     when in slave mode, param stands for slave address */
430     int32_t (*iic_close) (void);            /*!< close iic device */
431     int32_t (*iic_control) (uint32_t ctrl_cmd, void *param); /*!< control iic device */
432     int32_t (*iic_write) ([const void *data, uint32_t len]); /*!< send data by iic device (blocking method) */
433     int32_t (*iic_read) (void *data, uint32_t len);         /*!< read data from iic device (blocking method) */
434 } DEV_IIC, * DEV_IIC_PTR;

```

IoTDK上的I2C(4/5) – 開啟I2C

device/ip/ip_hal/inc/dev_iic.h, device/inc/dev_common.h

- 以master mode下標準速度開啟

```
98  /*
99  * macros for device working mode
100  */
101  #define DEV_MASTER_MODE (0)
102  #define DEV_SLAVE_MODE (1)
103  /** @} */
```

```
71  /** IIC Bus possible speed mode
72  typedef enum iic_speed_mode {
73      IIC_SPEED_STANDARD = 0, /*
74      IIC_SPEED_FAST      = 1, /*
75      IIC_SPEED_FASTPLUS  = 2, /*
76      IIC_SPEED_HIGH      = 3, /*
77      IIC_SPEED_ULTRA     = 4  /*
78  } IIC_SPEED_MODE;
```

```
int32_t (*iic_open) (uint32_t mode, uint32_t param);
```

The diagram illustrates the mapping of parameters from the code snippets to the `iic_open` function signature. A red arrow originates from the `DEV_MASTER_MODE` macro (value 0) in the first code block and points to the `mode` parameter in the function signature. A yellow arrow originates from the `IIC_SPEED_STANDARD` enum value (0) in the second code block and points to the `param` parameter in the function signature.

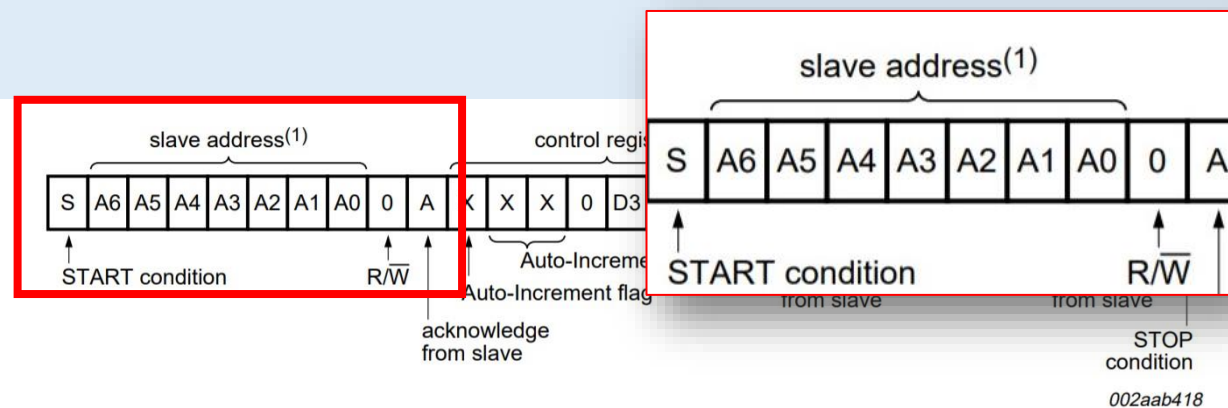
IoTDK上的I2C(5/5) – 設定slave裝置位址

device/ip/ip_hal/inc/dev_iic.h

- 在uint32_t ctrl_cmd中可以填入的參數

```
340  /**
341   * Set target slave device address for selecting slave device
342   * - Param type : uint32_t
343   * - Param usage : target slave address value
344   * - Return value explanation :
345   */
346  #define IIC_CMD_MST_SET_TAR_ADDR DEV_SET_IIC_MST_SYSCMD(2)
```

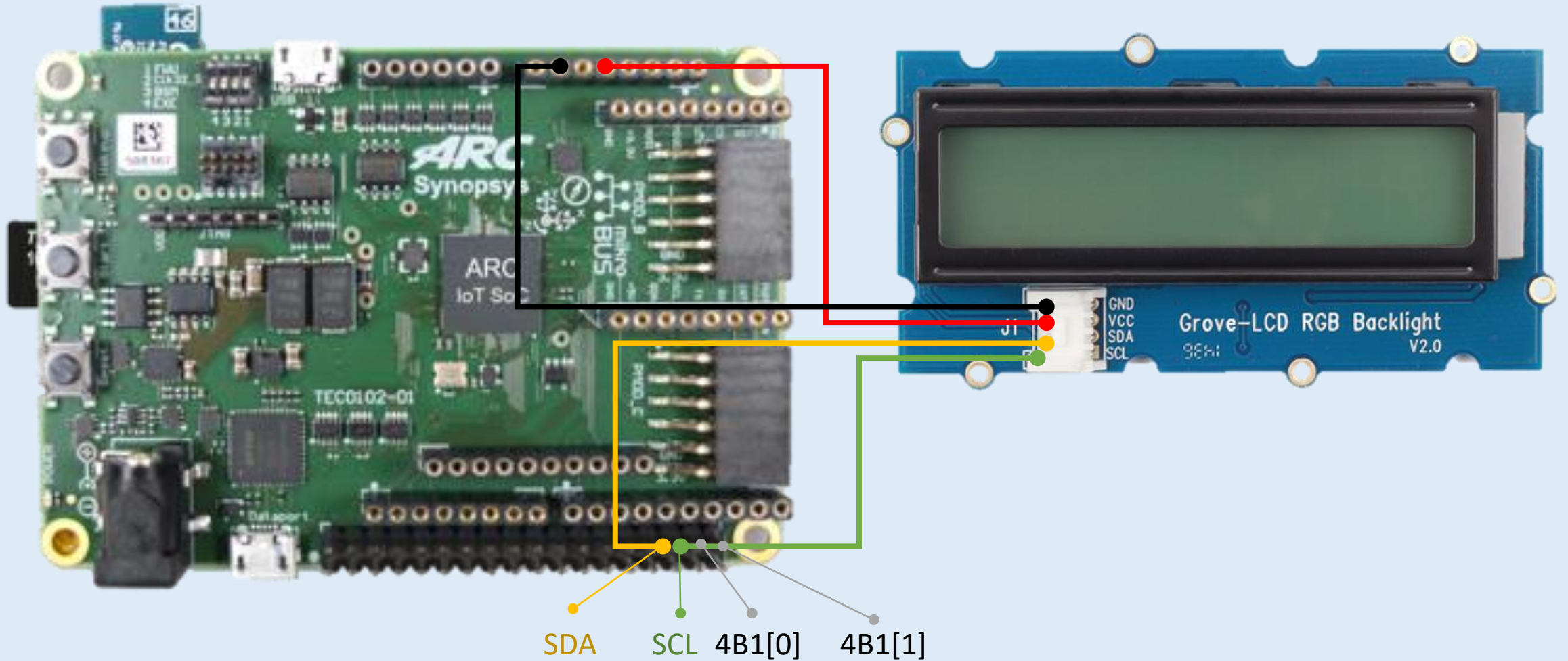
```
int32_t (*iic_control) (uint32_t ctrl_cmd, void *param);
```



動手實作-顯示字元

Lab2-I2C_LCD

硬體連接



超音波測距-軟體部分

- 建議檔案位置
 - ./embarc_osp/Lab2-I2C_LCD-Practice
- 照著註解提示完成
 - #112 int main(void)
 - #154 Lcd_Write(const char Chr)

即可

```
106  /* 宣告一個 DEV_IIC_PTR 的變數來存放 I2C 物件的指標 */
107  DEV_IIC_PTR iic;
```

自行完成部分說明(1/8)

- 在Lab2-I2C_LCD-Practice/main.c中可以看到這樣一個需要填寫的部分
- 這部分是在初始化最上面所宣告的*iic*物件指標

```
112 int main(void) {
113     /**
114     * 透過 iic_get_dev(uint32_t id) 傳入DFSS_IIC_x_ID 得到 DEV_IIC_PTR 的物件指標
115     * 並且將他 assign 給 上面的 iic 變數
116     * {
117     */
118
119     /**
120     * }
121     */
122
123     /**
124     * 透過 DEV_IIC 裡面的 iic_open 將 iic 開啟成 DEV_MASTER_MODE
125     * 同時給予 IIC_SPEED_STANDARD 的傳輸速度即可
126     * {
127     */
128
129     /**
130     * }
131     */
132
133     /* 為了讓同學能比較清楚看見出現的字，所以我利用 Lcd_Init_with_I2C_Dev 去初始化背景
134     Lcd_Init_with_I2C_Dev(iic);
135
136     Lcd_Write('H');
137     Lcd_Write('e');
138     Lcd_Write('l');
139     Lcd_Write('l');
140     Lcd_Write('o');
141     Lcd_Write(',');
142     Lcd_Write('W');
143     Lcd_Write('o');
144     Lcd_Write('r');
145     Lcd_Write('l');
146     Lcd_Write('d');
147     Lcd_Write('!');
```


自行完成部分說明(2/8)

- 第一段的 **DEV_IIC_PTR iic_get_dev(uint32_t id)** 需要傳入的是在**第41頁**所提到的 **DFSS_IIC_x_ID**，端看同學的腳位插在哪邊填入不同x，回傳值寫入上方main.c/107行中的**iic**變數即可。
- 第二段的 **iic_open(uint32_t mode, void* param)**，則如**第43頁**所提到是將I2C 打開，**mode**可以分成 **DEV_MASTER_MODE** 跟 **DEV_SLAVE_MODE**，因為LCD為Slave，所以IoTDK需要做為**Master**才能連結它；**param**在Master模式下是選擇不同速度的，填入**IIC_SPEED_STANDARD**即可。

自行完成部分說明(3/8)

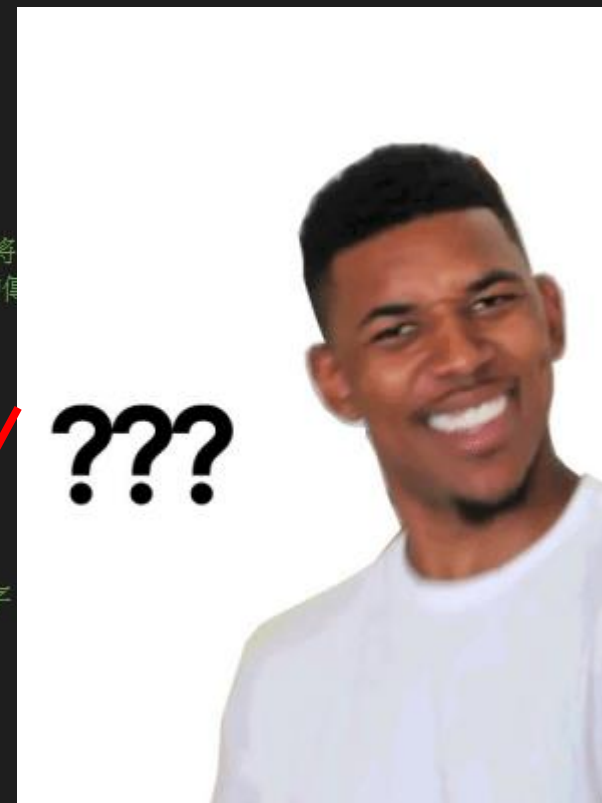
- You're halfway now. 同學應該可以看到LCD成功打開，並且呈現以下狀態。



自行完成部分說明(4/8)

- 那底下的`Hello,World!`是在做甚麼呢？

```
112 int main(void) {
113     /**
114      * 透過 iic_get_dev(uint32_t id) 傳入DFSS_IIC_x_ID 得到 DEV_IIC_PTR 的物件指標
115      * 並且將他 assign 給 上面的 iic 變數
116      * {
117      */
118
119     /**
120      * }
121      */
122
123     /**
124      * 透過 DEV_IIC 裡面的 iic_open 將
125      * 同時給予 IIC_SPEED_STANDARD 的傳
126      * {
127      */
128
129     /**
130      * }
131      */
132
133     /* 為了讓同學能比較清楚看見出現的字
134     Lcd_Init_with_I2C_Dev(iic);
135
136     Lcd_Write('H');
137     Lcd_Write('e');
138     Lcd_Write('l');
139     Lcd_Write('l');
140     Lcd_Write('o');
141     Lcd_Write(',');
142     Lcd_Write('W');
143     Lcd_Write('o');
144     Lcd_Write('r');
145     Lcd_Write('l');
146     Lcd_Write('d');
147     Lcd_Write('!');
```



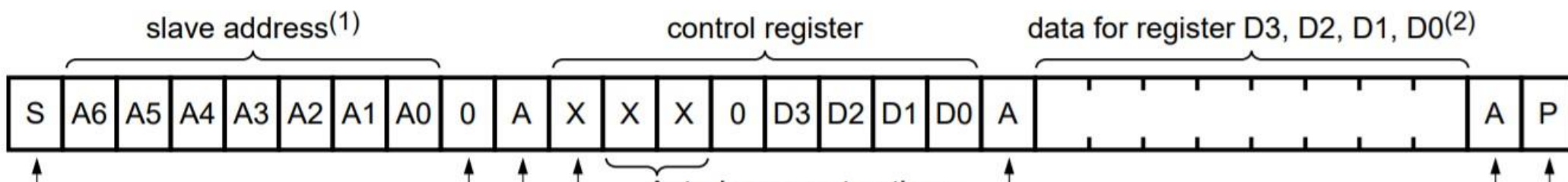
自行完成部分說明(4/8)

- 在 main.c#154行可以看到這個函式，裡面有需要填寫的部分。
- 這部分是在進行I2C資料的送出，並且讓送出的字元可以顯示在LCD螢幕上

```
154 void Lcd_Write(const char Chr) {  
155     /**  
156     * 先利用iic_control設定 IIC_CMD_MST_SET_TAR_ADDR  
157     * 將Master的目標位址設定成LCD的地址 - 0x3E  
158     * {  
159     */  
160  
161     /**  
162     * }  
163     */  
164  
165     /**  
166     * 將0x40與傳進來的Chr變數準備好，變成一個array  
167     * 注意因為每次傳送都是一個byte，所以型別寫uint8_t/char都可以  
168     * {  
169     */  
170  
171     /**  
172     * }  
173     */  
174  
175     /**  
176     * 透過iic_write寫出去  
177     * {  
178     */  
179  
180     /**  
181     * }  
182     */  
183 }
```

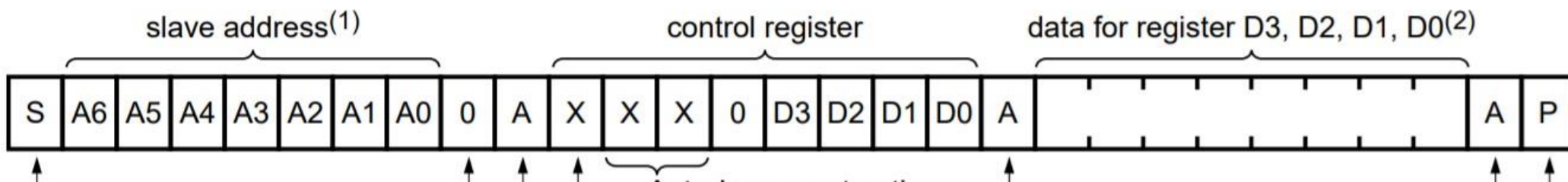
自行完成部分說明(5/8)

- 讓我們回顧一下第38頁的I2C資料格式。
 - 第一個是Slave Address，在Lab2-I2C_LCD-Practice/main.c#45中有定義出**LCD_ADDRESS**為0x3E，可以大膽用
 - 因此透過第44頁所提到的**iic_control(uint32_t ctrl_cmd, void* param)**，傳入**IIC_CMD_MST_SET_TAR_ADDR**以設定I2C Master 所要傳給的Slave Address，並且設定為**LCD_ADDRESS** 即可。



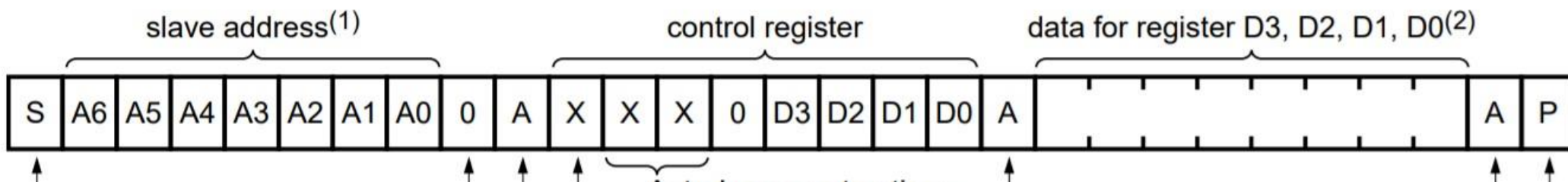
自行完成部分說明(6/8)

- 第二個是Control Register的Address，而控制LCD輸出的暫存器地址為**0x40**。
- 不過除了Slave Address是透過設定而自動生成的，**Control Register Address**跟緊接著的**Data**都被視為資料來傳送。
- 因此要先把Control Register Address跟Data包成一個array。
- Data的部分就是由外部傳進來的字元。



自行完成部分說明(7/8)

- 最後就是透過42頁提到的*iic_write(const void* data, uint32_t len)*將Control Register Address跟Data一同送出。
- 記得*uint32_t len*的部分是2。



自行完成部分說明(8/8)

- 最終成品應該如下圖。



不過直接透過I2C來操
控太複雜了

現成的
iic1602lcd.h/.c

✓ Lab3-Practice

> obj_iotdk_10

C iic1602lcd.c U

C iic1602lcd.h U

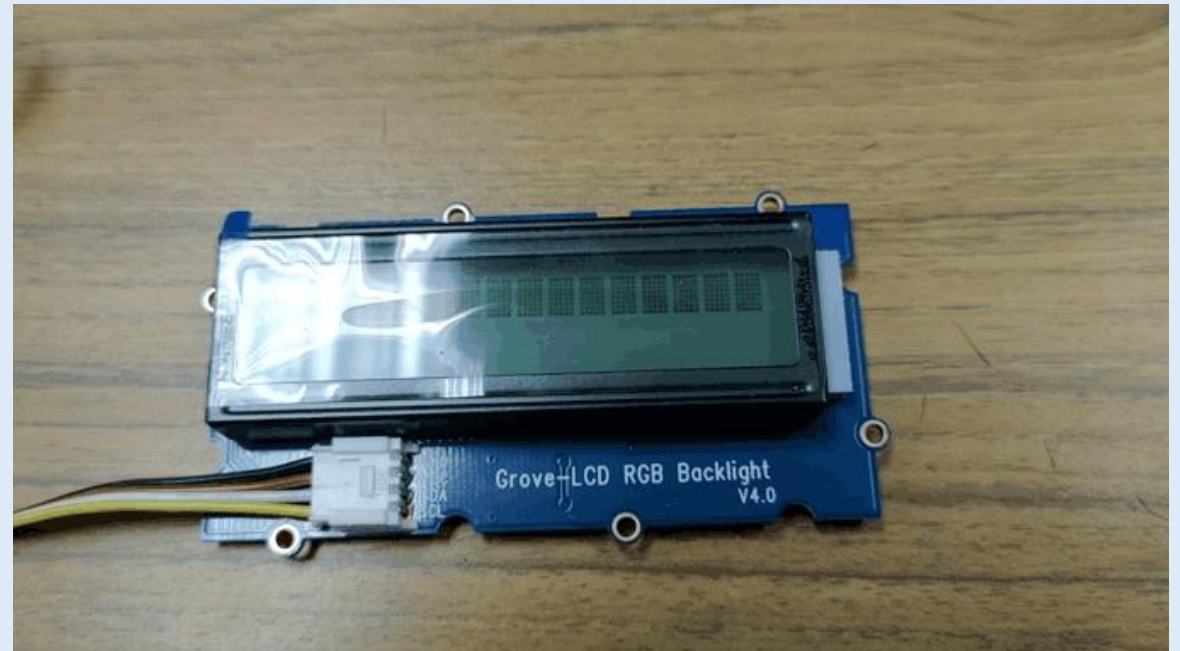
C main.c U

M makefile U

現成的iic1602lcd.h/.c(1/10) – `pLCD_t LCD_Init(uint32_t iic_id)`

- 透過iic_id來初始化螢幕，會將螢幕清空並且背光設定成白色。

```
44  pLCD_t lcd_obj;  
45  
46  int main(void)  
47  {  
48      lcd_obj = LCD_Init(DFSS_IIC_0_ID);  
49  
50      while (1)  
51      {  
52      }
```



現成的iic1602lcd.h/.c(2/10) – **void .printf(formatted string)**

- 跟普通printf沒兩樣，只是這是附屬於LCD_t裡面的函式，並且會把內容印到LCD上。

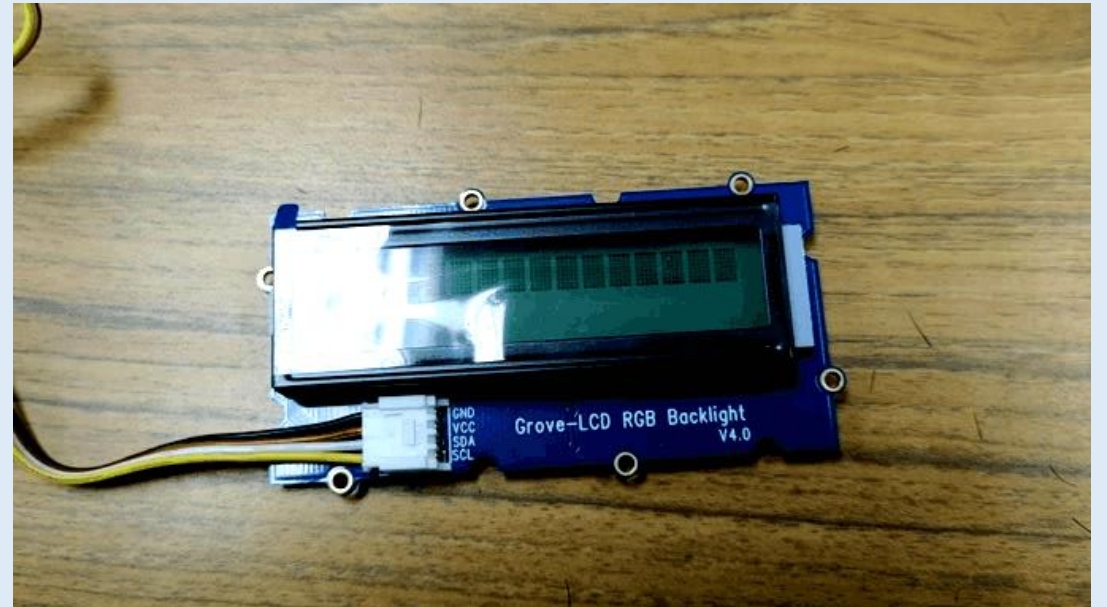
```
44  pLCD_t lcd_obj;  
45  
46  int main(void)  
47  {  
48      lcd_obj = LCD_Init(DFSS_IIC_0_ID);  
49      char *tmpStr = "NCKUEE CASLAB";  
50      int num = 1;  
51      lcd_obj->printf("%s %d", tmpStr, num);  
52  
53      while (1)  
54      {  
55      }
```



現成的iic1602lcd.h/.c(3/10) – **void .clear()**

- 清空螢幕上所顯示的字元

```
44  pLCD_t lcd_obj;  
45  
46  int main(void)  
47  {  
48      lcd_obj = LCD_Init(DFSS IIC 0 ID);  
49      lcd_obj->printf("Proto");  
50      board_delay_ms(1000, 0);  
51      lcd_obj->clear();  
52  
53      while (1)  
54      {  
55      }
```



現成的iic1602lcd.h/.c(4/10) – **void .set_Cursor(ON/OFF)**

- 顯示游標(字旁邊的底線)

```
46  int main(void)
47  {
48      lcd_obj = LCD_Init(DFSS IIC 0 ID);
49      lcd_obj->set_Cursor(ON);
50      for (int i = 0; i < 10; i++)
51      {
52          lcd_obj->printf("%d", i);
53          board_delay_ms(100, 0);
54      }
55
56      while (1)
57      {
```



現成的iic1602lcd.h/.c(5/10) – **void .set_Blink(ON/OFF)**

- 顯示游標位置閃爍的黑線

```
44  pLCD_t lcd_obj;  
45  
46  int main(void)  
47  {  
48      lcd_obj = LCD_Init(DFSS_IIC_0_ID);  
49      lcd_obj->set_Blink(ON);  
50      for (int i = 0; i < 10; i++)  
51      {  
52          lcd_obj->printf("%d", i);  
53          board_delay_ms(100, 0);  
54      }  
55  
56      while (1)  
57      {
```



現成的iic1602lcd.h/.c(6/10) – **void .home()**

- 讓游標回到左上角

```
44  pLCD_t lcd_obj;  
45  
46  int main(void)  
47  {  
48      lcd_obj = LCD_Init(DFSS_IIC_0_ID);  
49      lcd_obj->set_Blink(ON);  
50      for (int i = 0; i < 10; i++)  
51      {  
52          lcd_obj->printf("%d", i);  
53          board_delay_ms(100, 0);  
54      }  
55      lcd_obj->home();  
56      lcd_obj->printf("ABCD");
```



現成的iic1602lcd.h/.c(7/10) – `void .set_CursorPos(col(0-15), row(0-1))`

- 設定游標位置，第一個數字為長邊，第二個為短邊

```
44  pLCD_t lcd_obj;  
45  
46  int main(void)  
47  {  
48      lcd_obj = LCD_Init(DFSS_IIC_0_ID);  
49      lcd_obj->set_CursorPos(14, 1);  
50      lcd_obj->printf("HI");  
51      lcd_obj->set_CursorPos(5, 0);  
52      lcd_obj->printf("ABCD");  
53      board_delay_ms(2000, 0);  
54      lcd_obj->set_Display(OFF);  
55      board_delay_ms(2000, 0);  
56      lcd_obj->set_Display(ON);  
57  
58      while (1)
```



現成的iic1602lcd.h/.c(8/10) – **void .set_Display(ON/OFF)**

- 設定LCD是否顯示

```
44  pLCD_t lcd_obj;  
45  
46  int main(void)  
47  {  
48      lcd_obj = LCD_Init(DFSS_IIC_0_ID);  
49      lcd_obj->set_CursorPos(14, 1);  
50      lcd_obj->printf("HI");  
51      lcd_obj->set_CursorPos(5, 0);  
52      lcd_obj->printf("ABCD");  
53      board_delay_ms(2000, 0);  
54      lcd_obj->set_Display(OFF);  
55      board_delay_ms(2000, 0);  
56      lcd_obj->set_Display(ON);  
57  
58      while (1)
```



現成的iic1602lcd.h/.c(9/10) – `void .set_Color(RED/GREEN/BLUE/WHITE)`

- 設定LCD的背光顏色

```
44  pLCD_t lcd_obj;  
45  
46  int main(void)  
47  {  
48      lcd_obj = LCD_Init(DFSS_IIC_0_ID);  
49      lcd_obj->set_Color(RED);  
50      board_delay_ms(1000, 0);  
51      lcd_obj->set_Color(GREEN);  
52      board_delay_ms(1000, 0);  
53      lcd_obj->set_Color(BLUE);  
54      board_delay_ms(1000, 0);  
55      lcd_obj->set_Color(WHITE);  
56  
57      while (1)
```




現成的iic1602lcd.h/.c(10/10) –
**void .set_RGB(RED/GREEN/BLUE,
uint8_t value)**

- 設定LCD的背光某一通道的值

```
44  pLCD_t lcd_obj;  
45  
46  int main(void)  
47  {  
48      lcd_obj = LCD_Init(DFSS_IIC_0_ID);  
49      lcd_obj->set_RGB(RED, 0x70);  
50      board_delay_ms(1000, 0);  
51      lcd_obj->set_RGB(GREEN, 0x70);  
52      board_delay_ms(1000, 0);  
53      lcd_obj->set_RGB(BLUE, 0x70);  
54  
55      while (1)
```





Lab3

測距與近迫警示

測距與近迫警示

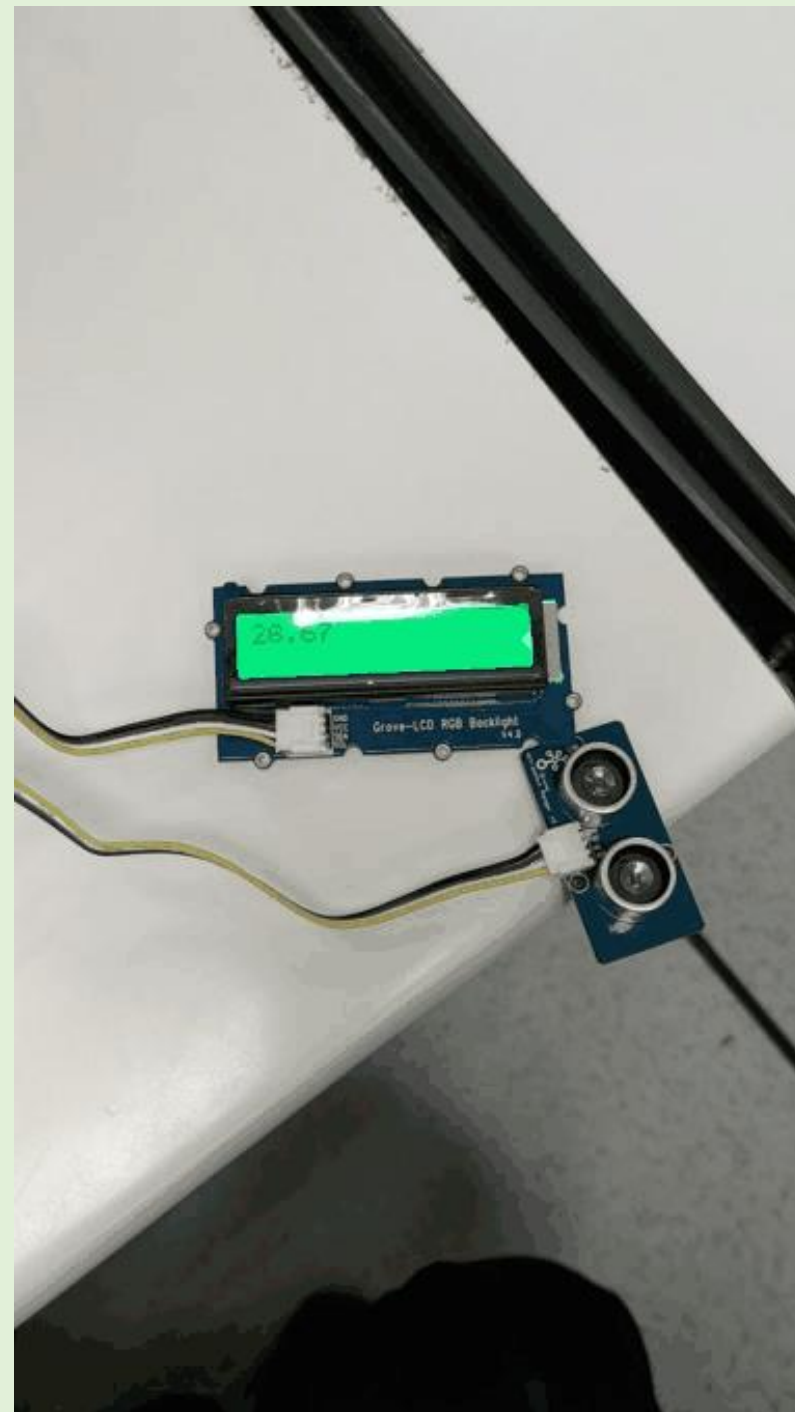
透過超音波測距模組測量
距離

透過LCD模組顯示

當測量距離低於指定數值
後改變顏色

測距與近迫警示 - 成品展示

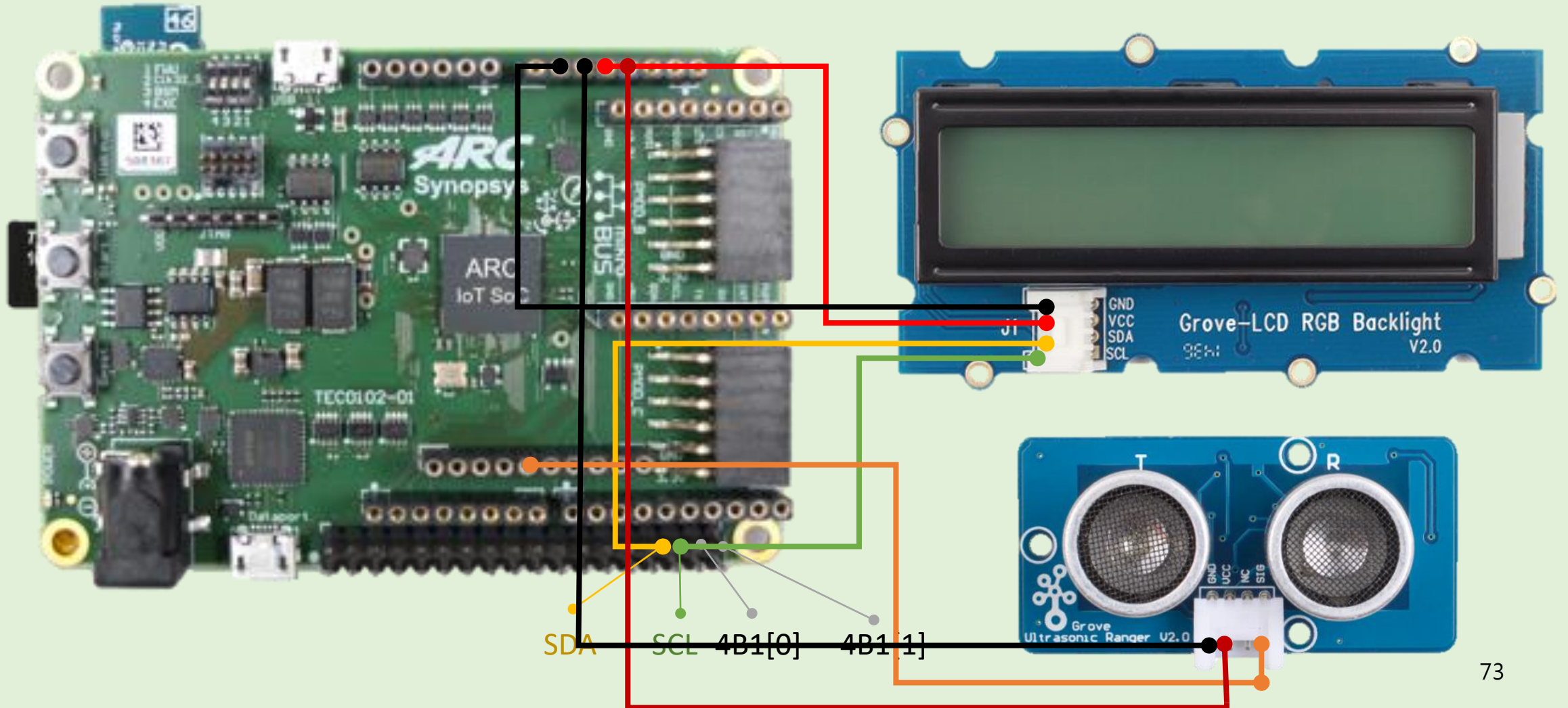
- LCD顯示對超音波模組的距離
- 當低於15公分時會切換顏色



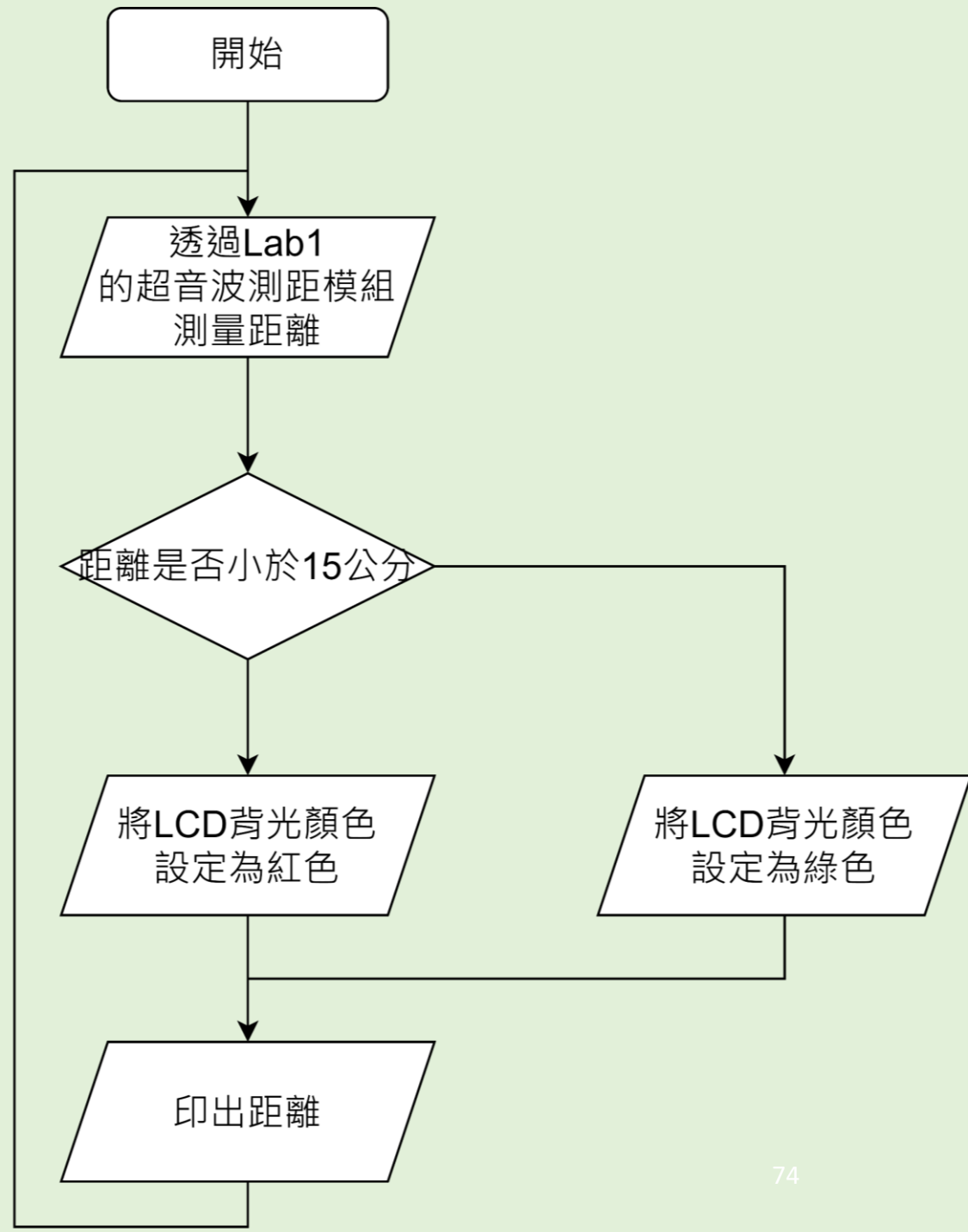
超音波測距-軟體部分

- 建議檔案位置
./embarc_osp/Lab3-Practice
- 照著註解提示或是流程圖即可完成

測距與近迫警示－硬體連接

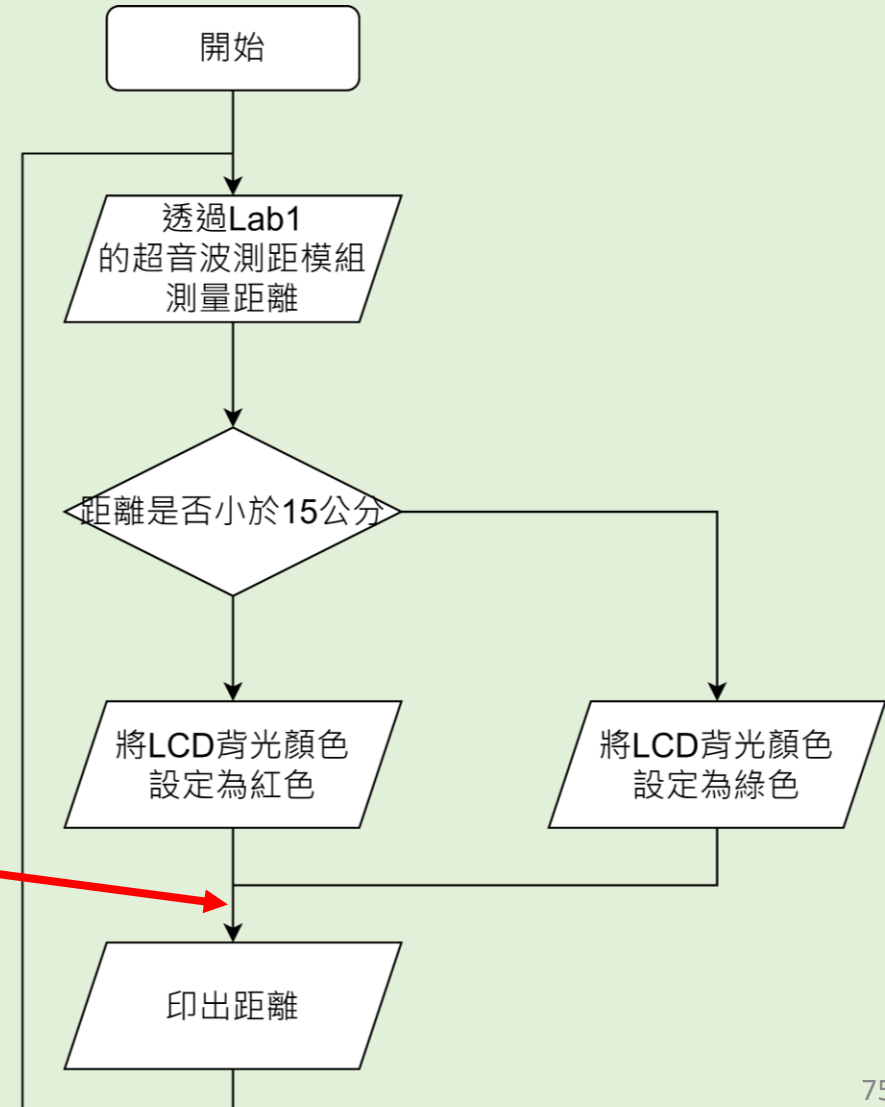


測距與近迫警示 - 程式流程



測距與近迫警示－需注意部分

需清空螢幕
並且將游標設定回0, 0位置

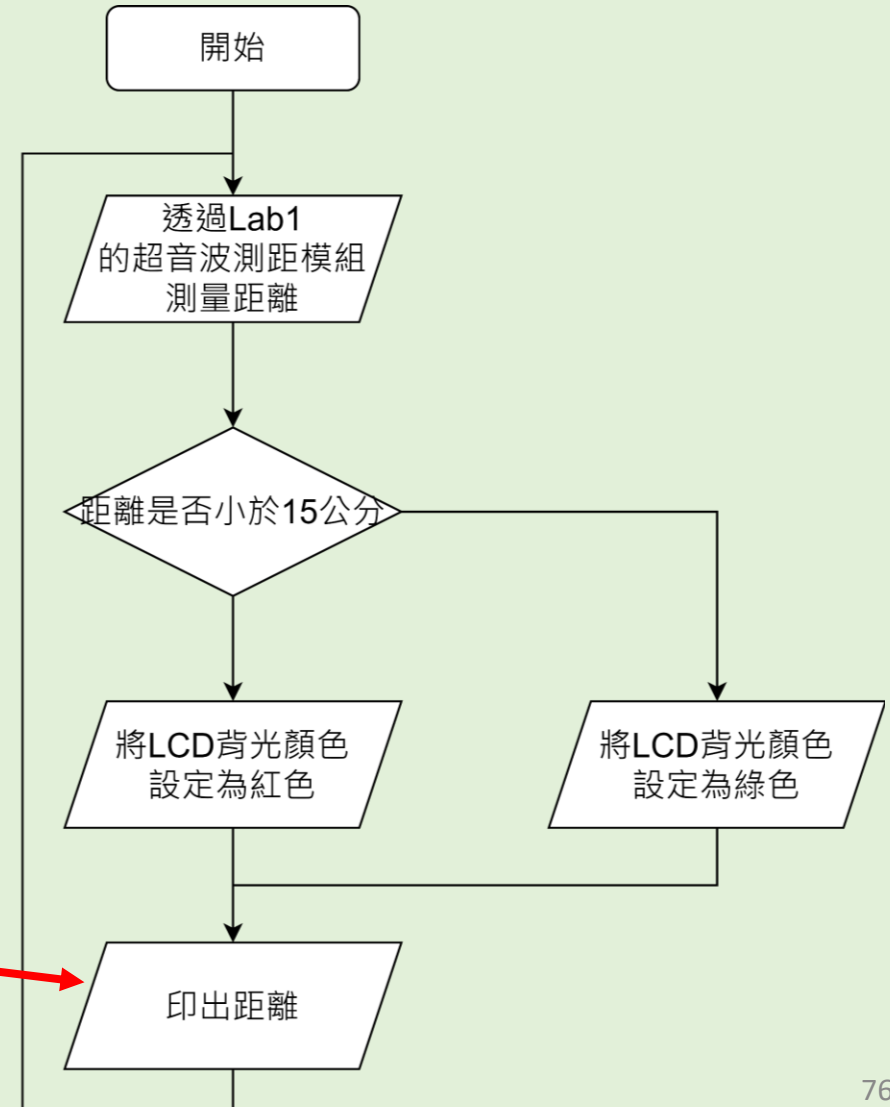


測距與近迫警示 – 需注意部分

無法使用 `printf(“%f”, distance)`

必須使用

`printf(“%d.%d”, (int)distance,
(int)(Distance * 100) % 100)`



Final Project

- 如果需要額外模組可以詢問助教

waynelin2339@gmail.com

Reference

- [超音波模組介紹](#)
- [超音波模組source code](#)
- [ARCompact ISA Programmer's Reference](#)
- [ARC IoTDK Pinout](#)
- [LCD 字元模組 Datasheet](#)
- [LCD 背光模組 Datasheet](#)



Thank you for
listening