



掌握音樂的未來：

智慧娛樂

音響全新登場

嵌入式智慧系統創新應用就業養成班


第 48 期 第五組

您準備好體驗音樂的全新世界了嗎？我們的智慧音響不僅帶來高品質的音樂享受，還將音樂與科技完美結合，為您帶來前所未有的聽覺饗宴。

指導老師：詹民進 老師

此部分負責組員：陳麒元

中華民國 112 年 8 月



一、前言：

「在現代快節奏的生活中，您是否在放鬆娛樂時渴望一個能夠同時提供高品質音樂和無與倫比體驗的環境？我們的全新智慧娛樂音響將為您帶來驚喜和改變；並藉由燈光讓生活中的每一刻都有其獨特的氛圍和情感，因此我們的智慧氣氛燈將成為您的創意工具，讓您根據心情隨心打造各種光影場景。同時，在這個多變的時代，安全問題變得日益嚴峻。我們的智慧警衛不僅是一個產品，更是您安全的守護者，為您打造全方位的保護網。」



二、 價值展示

我們相信，音樂可以成為您運動的最佳伴侶。我們的智慧音響不僅提供了卓越的音質，更讓您的娛樂放鬆過程更加愉快和振奮。您可以隨意切換歌曲，調整音量，或是播放放鬆的頻率音樂，甚至是可以自彈自唱，達到徹底放鬆狀態；也可以利用氣氛燈的切換，徹底融入在這個只有自己的時刻。

而當您沈浸在自我的小世界時，也不用擔心會有其他人的打擾或是干擾，當你切換至警訊衛兵模式時，就會幫助注意周遭環境，當已有人靠近時會發出燈光及聲音的警訊通知。

三、 產品特點：



卓越音質：憑藉先進的音頻技術，我們的娛樂音響將帶給您深沉而清晰的音樂體驗，讓您全情沉浸在節奏之中。



多種燈效：智慧氣氛燈擁有多種燈光效果，包括柔和燈光、彩虹燈效、節奏跳動等，營造出不同的情感氛圍。



音樂互動：我們的氣氛燈支援音樂同步功能，讓您的燈光隨著節奏變化，打造極具震撼的視聽效果。



即時警訊：我們的智慧警衛具即時警報功能，一旦偵測到可疑動態，立即發出警訊，確保您可第一時間做出應對。

四、 功能展示：



1. 當燈光調控模式時，若利用紅外線偵測到人時開啟七彩霓虹燈閃爍，若無人則暫停
2. 在闖入偵測模式，若利用紅外線偵測到人時蜂鳴器及燈光發出警告訊號，若無人則暫停
3. 在音樂娛樂的部分，除了可以播放預設的背景及歌曲音樂之外，還可以在娛樂間中自彈自唱，進入娛樂的殿堂。

五、 團隊堅持：

我們深知音樂對於您的情感和心靈的重要性。因此，我們的智慧音響不僅注重技術，更追求卓越的音質。堅持四大目標，讓每一個音符每個休息的瞬間，都如同鑽石一般晶瑩剔透，讓您沉浸在音樂的海洋中。」



六、 客戶見證：

- 「'這是我使用過的最好的音響，音質真的太出色了！無論在跑步還是娛樂時，都讓我感到更有動力。」
- 「'從使用了這個智慧音響後，我的音樂體驗得到了質的飛躍。無論在家中還是辦公室，它都為我帶來了無比的享受。'- 張先生」
- 「在寒冷的冬夜，您可以調整為柔和的暖色調，讓溫暖的燈光點綴整個房間，讓您感到宜人舒適。」

- 「準備好改變您的運動體驗了嗎？立刻掃描下方連結，線上體驗智慧音響，享受優質音樂和無比舒適的運動。」

<https://www.youtube.com/watch?v=t0RlmHF2i5g>



立即體驗智能娛樂的未來！點擊下方連結，帶你實際享受

您的智能家居為您的生活帶來無與倫比的便利和享受。



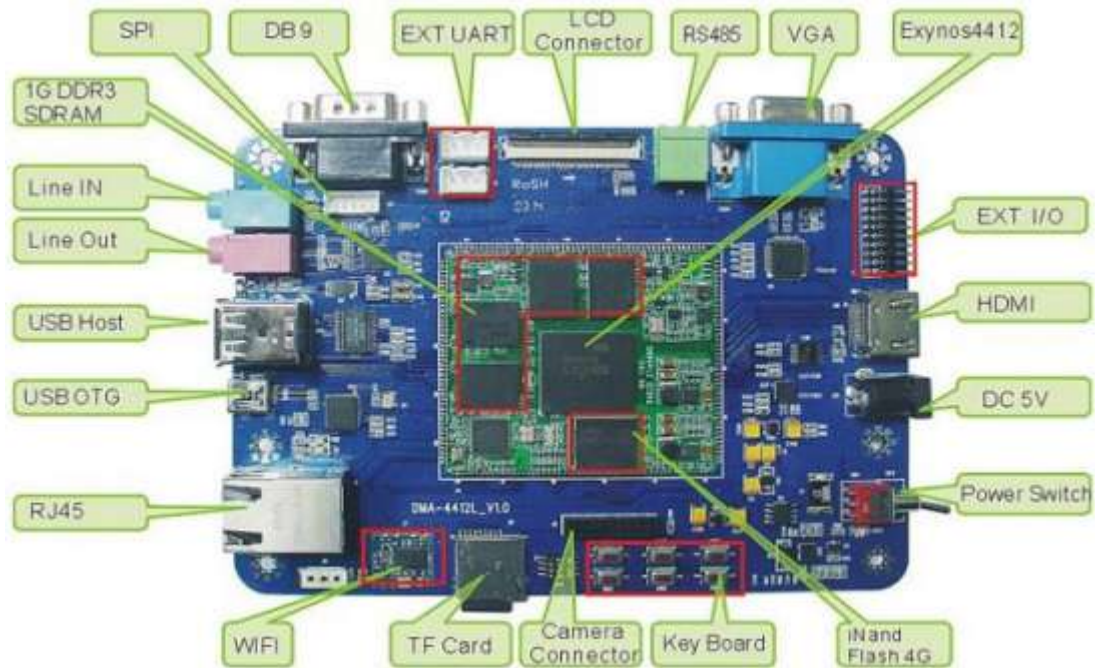
七、 支援與聯繫：

如果您對我們的產品有任何疑問或需要支援，請隨時聯繫我們的 MEME48-第五組服務團隊，我們將竭誠為您提供幫助。

八、 產品與硬體說明：

(一) 硬體介紹

1. 開發版：DMA-4412L



- (1). 中央處理器：Samsung Exynos 4412-SCP
(核心為 ARM Cortex A9)
- (2). 作業系統：Linux3.0.8 / Android4.2/4.4
- (3). 內建高性能 ARM-Mali-400 MP 3D 圖形引擎和 2D 圖形引擎。
- (4). 內建 NFC，Mpeg-1/2/4/VC-1/H. 263/H. 264 的硬體編碼與解碼。
- (5). 支援 5 吋高解析度 LCD+電容式多點觸控板（IIC 介面）。
- (6). USB Wi-Fi 無線上網、藍芽 BT2.0+EDR 傳輸功能。
- (7). 支援攝像功能，IIC CMOS 及 USB CMOS 介面。
- (8). 3G Modem 通話、上網、簡訊功能(USB 外接介面)。
- (9). HDMI 1.4，1080p/60Hz 高清輸出。
- (10).GPS 配合 Google MAP 定位及導航使用。

DMA-4412L 是一款 ARM Coretex-A9 四核開發平台，採用

~~Samsung Exynos4412 (Exynos4412 Quad) 設計。主頻提升至~~

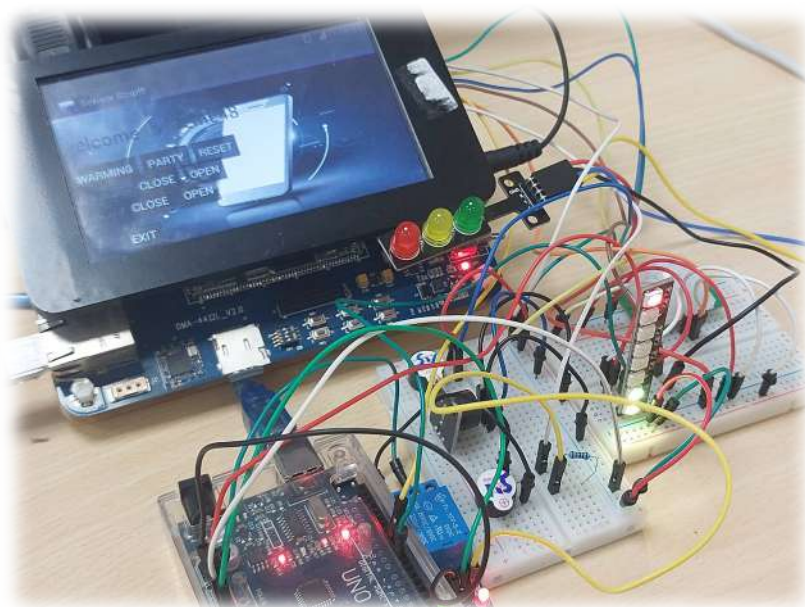
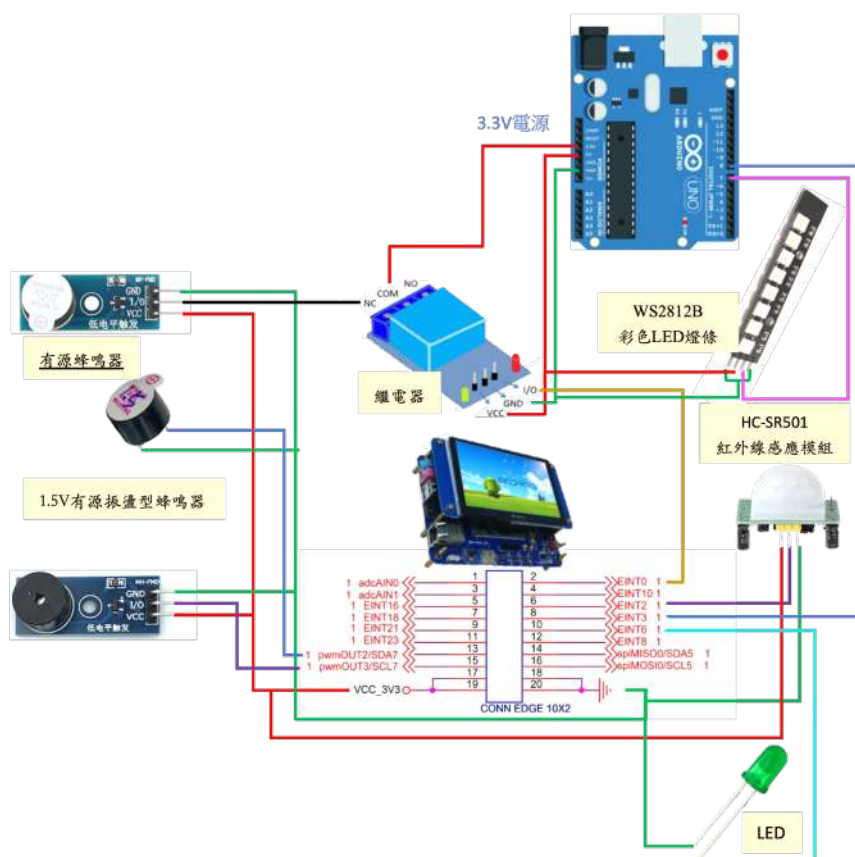
1.5 GHz，其自帶 128/64 位元內部匯流排結構，32/32KB 的資料/指令一級緩存，1024KB 的二級緩存，可以實現 2000DMIPS（每秒 2 億指令集）的高性能運算能力。採用 32nm 低功率制程，運算能力和功耗方面分別比 45nm 工藝的 Exynos 4210 提升 25% 和降低 40% 左右，另外其內建高性能 ARM Mali-400 MP 3D 圖形引擎和 2D 圖形引擎，多邊形生成率為 44M 三角形/秒，圖元填充率可達 1.6G 圖元/秒，支援 DX9、SM3.0、OpenGL2.0 等 PC 級別顯示技術，支援全高清、多標準的視頻編碼，流暢播錄 1080p 視頻檔，內建 MFC，支援 MPEG-1/2/4、VC-1、H.263、H.264 編解碼，支援數位 TV 輸出，完美的展現了 Samsung Exynos4412 晶片的功能。



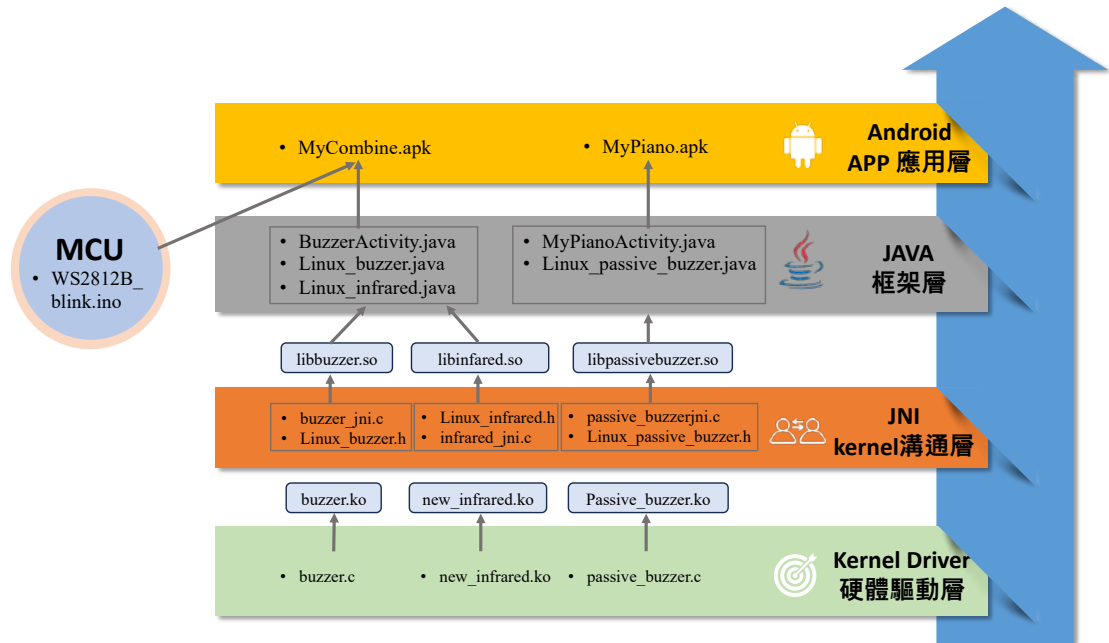
2. 人體紅外線感測器

產品型號	HC-SR501人體感應模塊
工作電壓範圍	直流電壓4.5-20V
靜態電流	<50uA
電平輸出	高3.3 V /低0V
觸發方式	L不可重複觸發/H重複觸發(默認重複觸發)
延時時間	0.5-200S(可調)可製作範圍零點幾秒-幾十分鐘
封鎖時間	2.5S(默認)可製作範圍零點幾秒-幾十秒
電路板外形尺寸	32mm*24mm
感應角度	<100度錯角
工作溫度	-15~+70度
感應距離尺寸	直徑:23mm(開設)

(二) 線路圖



九、軟體架構說明



(一) Kernel 程式設計

1. Buzzer (有源蜂鳴器)部分

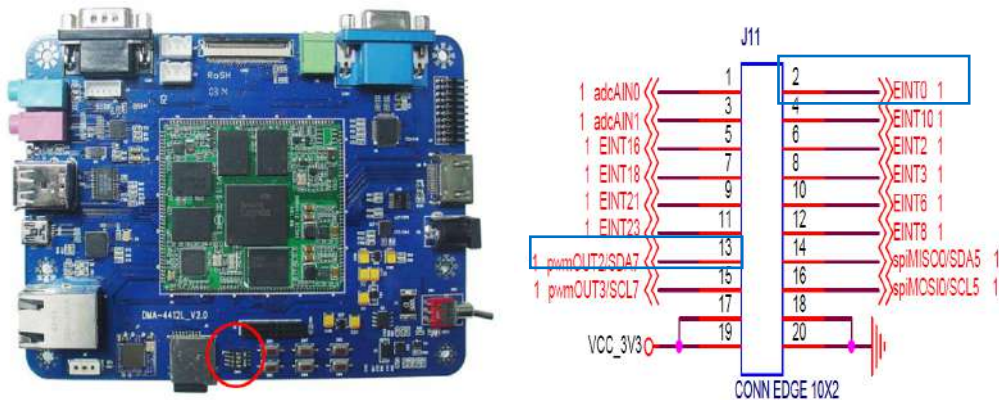


Figure7. Exynos 4412 外觀及接腳圖

(1). 使用接腳

在有源蜂鳴器上，預計設計成兩個蜂鳴器搭配開啟，在腳位的部分，分配的腳位為 Pin 2 及 Pin 13 腳，說明如下圖所示：

```
#define PIN          EXYNOS4_GPX0(0) //pin2
#define PIN2         EXYNOS4_GPD0(2) //pin13
```

(2). 使用主編號及裝置名稱

```
#define DEVICE_MAJOR 235
#define DEVICE_NAME  "buzzer"
```

(3). 程式進入點

進入核心底層後，Server 會先進入 `init()` 內將暫存器開啟、建立需使用的模組（buzzer），及給予 Major 號碼提供註冊；再離開 Kernel 時，會進入 `exit()` 裡面將原本建立及註冊的資料做一個清除的動作。

```
static struct class *buzzer_class;

static int __init buzzer_init(void)
{
    int retval;

    retval = register_chrdev(DEVICE_MAJOR, DEVICE_NAME, &buzzer_fops);
    if(retval < 0)
    {
        printk(KERN_INFO "%s: registering device %s with major %d failed with %d\n",
            __func__, DEVICE_NAME, DEVICE_MAJOR, DEVICE_MAJOR );
        return retval;
    }

    printk("Buzzer driver register success!\n");

    buzzer_class = class_create(THIS_MODULE, "buzzer");
    if (IS_ERR(buzzer_class))
    {
        printk(KERN_WARNING "Can't make node %d\n", DEVICE_MAJOR);
        return PTR_ERR(buzzer_class);
    }

    device_create(buzzer_class, NULL, MKDEV(DEVICE_MAJOR, 0), NULL, DEVICE_NAME);
    printk("Buzzer driver make node success!\n");
    return 0;
} « end buzzer_init »

static void __exit buzzer_exit(void)
{
    device_destroy(buzzer_class, MKDEV(DEVICE_MAJOR, 0));
    class_destroy(buzzer_class);
    printk("Buzzer driver remove node success!\n");
    gpio_free(PIN);
    gpio_free(PIN2);
    unregister_chrdev(DEVICE_MAJOR, DEVICE_NAME);
    printk("Buzzer driver release success!\n");
}

module_init(buzzer_init);
module_exit(buzzer_exit);

static struct file_operations buzzer_fops =
{
    owner      :   THIS_MODULE,
    read       :   buzzer_read2,
    write      :   buzzer_write,
    unlocked_ioctl :   buzzer_ioctl,
    open       :   buzzer_open,
    release    :   buzzer_release,
};
```

(4). 開啟裝置

在 open 函數中先使用變數來檢查 PIN 變數是否可以使用，加上 if() 判斷式，若不能創立則發出 "PIN1 fail" 的訊息，並將資料釋放 (gpio_free)。

```
static int buzzer_open(struct inode *inode, struct file *filp)
{
    int ret;

    printk("open in kernel\n");
    /* GPIO 000 */

    ret=gpio_request(PIN,DEVICE_NAME);
    ret=gpio_request(PIN2,DEVICE_NAME);

    if(ret<0){
        printk(KERN_EMERG "PIN fail !\n");
        printk( KERN_INFO "%s: %s unable to get TRIG gpio\n", DEVICE_NAME, __func__ );
        gpio_free(PIN);
        gpio_free(PIN2);
        return ret;
    }

    // Set gpios directions
    //s3c_gpio_cfgpin(PIN, S3C_GPIO_OUTPUT);
    //gpio_set_value(EXYNOS4_GPB(0),1);

    if( gpio_direction_output( PIN, 0 ) < 0 )    // Set pin 2 as output with default value 0
    {
        printk( KERN_INFO "%s: %s unable to set TRIG gpio as output\n", DEVICE_NAME, __func__ );
        ret = -EBUSY;
        return(ret);
    }
    if( gpio_direction_output( PIN2, 0 ) < 0 )    // Set pin 2 as output with default value 0
    {
        printk( KERN_INFO "%s: %s unable to set TRIG gpio as output\n", DEVICE_NAME, __func__ );
        ret = -EBUSY;
        return(ret);
    }
    return 0;
}
« end buzzer_open »

static int buzzer_release(struct inode *inode, struct file *filp)
{
    printk("buzzer release\n");
    return 0;
}
```

(5). 腳位控制

有關蜂鳴器驅動程式是使用 GPD2 的 I/O 腳位來實現，透過 ioctl 的指令中的 cmd 去控制蜂鳴器開或關與發出長短音，以下是部分程式碼。

```
static long buzzer_ioctl(struct file *file, unsigned int cmd, unsigned long arg)
{
    int ret = 0;
    int num;
    printk("cmd=%d\n",cmd);
    switch(cmd)
    {
        case BUZZER_ON:
            printk("on\n");
            ret = copy_from_user(&num, (int*)arg, sizeof(int));
            if (ret != 0)
            {
                printk("gpio_ioctl: copy_from_user failed\n");
                return(-EFAULT);
            }
            printk("num = %d\n",num);
            buzzer_on(num);
            break;
        case BUZZER_OFF:
            printk("of\n");
            ret = copy_from_user(&num, (int *)arg, sizeof(int)); //arg = 3-->num = 3
            if (ret != 0)
            {
                printk("gpio_ioctl: copy_from_user failed\n");
                return(-EFAULT);
            }
            printk("num = %d\n",num);
            buzzer_off(num);
            break;
    }
    « end switch cmd »
    return 0;
}
« end buzzer_ioctl »
```

根據接收的訊息，決定開關

而以下是在 ioctl 中有調用到的函數。

```
static void buzzer_off(int buzzer_num)
{
    //int gpj2dat,gpj3dat;
    switch(buzzer_num)
    {
        case 1://buzzer1
            printk("buzzer_off\n");
            gpio_direction_output(PIN,0);
            break;
        case 2://buzzer3
            printk("buzzer_off\n");
            gpio_direction_output(PIN2,0);
            break;
        default:
            break;
    }
}

static void buzzer_on(int buzzer_num)
{
    //int gpj2dat,gpj3dat;
    switch(buzzer_num)
    {
        case 1://buzzer1
            printk("buzzer_on\n");
            gpio_direction_output(PIN,1);
            break;
        case 2://buzzer3
            printk("buzzer_on\n");
            gpio_direction_output(PIN2,1);
            break;
        default:
            break;
    }
}
```


(6). Test 執行測試檔

建立一 C 語言執行測試檔案，來與 Kernel 互相連結，在 test 檔裡可以清楚看到蜂鳴器需要使用的 operation 是 open 函數及 read 函數；而 open 在 Kernel 裡的 /dev 下 buzzer 此裝置後，Exynos 4412 才可使用 pin 腳（pin 2、pin 13）進行發送及接收訊號；在 while 迴圈中使用 read 函數將超音波感測器的 feed back 回饋訊號傳到變數 buf，再將其列印到 console 上，其中 feed back 訊號已經由 Kernel 中換算成距離單位，若可以印出距離單位的數值，表示開啟裝置、讀取數值成功，傳至 console 也成功，若否，則 ret 檢查用的變數，會 exit。

```
15 int main(int argc, char *argv[])
16 {
17     int fd;
18     int val = -1;
19     int num = 1;
20     fd= open(DEVICE_BLTEST,O_RDONLY); //opendev
21     if(fd<0){
22         perror("can not open device");
23         exit(1);
24     }
25     while(1){
26         printf("please select number to control buzzer\n");
27         printf("1:buzzer-1: 2:buzzer-2: 3:exit : ->");
28         scanf("%d", &val);
29         printf("\n");
30         if(val !=3){
31             printf("select led on or off: 1:ON 3:OFF: ->");
32             scanf("%d", &num);
33             printf("\n");
34         }
35         switch(val){
36             case 1:
37                 if(num == BUZZER_ON)
38                     ioctl(fd,BUZZER_ON,&val);
39                 else if(num == BUZZER_OFF)
40                     ioctl(fd,BUZZER_OFF,&val);
41
42                 break;
43             case 2:
44                 if(num == BUZZER_ON)
45                     ioctl(fd,BUZZER_ON,&val);
46                 else if(num == BUZZER_OFF)
47                     ioctl(fd,BUZZER_OFF,&val);
48                 break;
49             case 3:
50                 close(fd);
51             default:
52                 return 0;
53         }
54     }
55     //close(fd);
56     return 0;
57 }
58
```

2. Infrared(紅外線感測器)部分

(1). 使用接腳

在紅外線感測器上，分配的腳位為 Pin 6、Pin 8 及 Pin 10 腳

```
#define PIN      EXYNOS4_GPX0(2) //pin6
#define BLING    EXYNOS4_GPX0(3) //pin8
#define LED      EXYNOS4_GPX0(6) //LED    pin10
```

(2). 使用主編號及裝置名稱

```
#define DEVICE_MAJOR 233
#define DEVICE_NAME "infrared"
```

(3). 程式進入點

進入核心底層後，Server 會先進入 init()內將暫存器開啟、建立需使用的模組 (buzzer)，及給予 Major 號碼提供註冊；再離開 Kernel 時，會進入 exit()裡面將原本建立及註冊的資料做一個清除的動作。

```
static struct file_operations infrared_fops =
{
    owner      : THIS_MODULE,
    read       : infrared_read,
    open       : infrared_open,
    unlocked_ioctl : infrared_ioctl,
    release    : infrared_release,
};
```

使用 infrared_init 函式進行裝置建立與註冊

```
static struct class *infrared_class;

static int __init infrared_init(void)
{
    int ret;
    /* 註冊 DEVICE_MAJOR, DEVICE_NAME */
    ret = register_chrdev(DEVICE_MAJOR, DEVICE_NAME, &infrared_fops);
    if (ret < 0)
    {
        printk(KERN_WARNING "Can't get major %d\n", DEVICE_MAJOR);
        return ret;
    }

    printk("Infrared driver register success!\n");

    infrared_class = class_create(THIS_MODULE, "Infrared");
    if (IS_ERR(infrared_class))
    {
        printk(KERN_WARNING "Can't make node %d\n", DEVICE_MAJOR);
        return PTR_ERR(infrared_class);
    }

    device_create(infrared_class, NULL, MKDEV(DEVICE_MAJOR, 0), NULL, DEVICE_NAME);
    printk("Infrared driver makes node successfully!\n");
    return 0;
} « end infrared_init »

static void __exit infrared_exit(void)
{
    device_destroy(infrared_class, MKDEV(DEVICE_MAJOR, 0));
    class_destroy(infrared_class);
    printk("Infrared driver remove node successfully!\n");
    gpio_free(PIN);
    unregister_chrdev(DEVICE_MAJOR, DEVICE_NAME);
    printk("Infrared driver released successfully!\n");
}

module_init(infrared_init);
module_exit(infrared_exit);
```

```
static int infrared_release(struct inode *inode, struct file *filp)
{
    /* ----- GPIO 000 ----- */
    gpio_free(PIN);
    gpio_free(BLING);
    gpio_free(LED);
    printk("infrared device release\n");
    return 0;
}
```

(4). 開啟裝置

在 open 函數中先使用變數來檢查 PIN 變數是否可以使用，加上 if() 判斷式，若不能創立則發出 "PIN1 fail" 的訊息，並將資料釋放 (gpio_free)。

```
static int infrared_open(struct inode *inode, struct file *filp)
{
    /* GPIO 000 */
    int ret;
    printk("open in kernel\n");
    /* GPIO 000 */
    gpio_free(PIN);
    ret = gpio_request(PIN, DEVICE_NAME);
    if (ret < 0) {
        printk(KERN_EMERG "%d failed !\n", PIN);
        printk(KERN_INFO "%s: %s unable to get TRIG gpio\n", DEVICE_NAME, __func__);
        gpio_free(PIN);
        return ret;
    }
    // Set gpios directions
    // s3c_gpio_cfgpin(PIN, S3C_GPIO_INPUT); // NPIN 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
    gpio_direction_input(PIN);
    /* ----- GPIO 000 ----- */
    printk("<1>-----BLING open-----\n");
    gpio_free(BLING);
    ret = gpio_request(BLING, "BLING");
    if (ret < 0) {
        printk("<1> ---ret error---");
        ret = -EBUSY;
        return (ret);
    }
    gpio_direction_output(BLING, 0);
    /* ----- GPIO 000 ----- */
    printk("<1>-----LED open-----\n");
    gpio_free(LED);
    ret = gpio_request(LED, "LED");
    if (ret < 0) {
        printk("<1> ---ret error---");
        ret = -EBUSY;
        return (ret);
    }
    gpio_direction_output(LED, 0);
    return 0;
} « end infrared_open »
```

(5). 腳位控制

有關蜂鳴器驅動程式是使用 GPD2 的 I/O 腳位來實現，透過 ioctl 的指令中的 cmd 去控制蜂鳴器開或關與發出長短音，以下是部分程式碼。

```
static long infrared_ioctl(struct file *file, unsigned int cmd, unsigned long arg){
    int ret;
    //int num;
    printk("cmd=%d\n",cmd);
    switch (cmd){
        case 0:
            mode = cmd;
            printk("<1>----case 0_ warming condition----\n");

            break;
        case 1:
            mode = cmd;
            printk("<1>----case 1_ party condition----\n");
            break;
        default : // APP arg = 3 ---> mode = 3
            ret = copy_from_user(&mode, (int *)arg, sizeof(int));
            if (ret != 0){
                printk("<1>----gpio_ioctl copy from user failed----\n");
                return (-EFAULT);
            }

            break;
    }
    gpio_set_value(LED, 0);
    gpio_set_value(BLING, 0);
    return 0;
}
```

(6). 裝置讀取

在 open 結束後，則會到 read 函數中進行讀取數值的動作，放到 value 位置，並傳送至上層(copy_to_user)。

```
static ssize_t infrared_read(struct file *file, int* value, size_t size, loff_t *off)
{
    size_t ret;

    if (mode == 0){ //warming
        int data;
        printk("<1> reading _data = %d\n", gpio_get_value(PIN));
        local_irq_disable();
        data = gpio_get_value(PIN);
        gpio_set_value(LED, gpio_get_value(PIN));
        local_irq_enable();
        ret = copy_to_user(&value, (int *)data, sizeof(data));
        //ret = copy_to_user(value, data, sizeof(data));
        if (ret<0){
            printk("<1> ---ret error---");
        }
    }else if (mode == 1){ //party
        int data;
        printk("<1> reading _data = %d\n", gpio_get_value(PIN));
        local_irq_disable();
        data = gpio_get_value(PIN);
        gpio_set_value(BLING, gpio_get_value(PIN));
        local_irq_enable();
        ret = copy_to_user(&value, (int *)data, sizeof(data));
        //ret = copy_to_user(value, &data, sizeof(data));
        if (ret<0){
            printk("<1> ---ret error---");
        }
    }else{
        gpio_set_value(LED, 0);
        gpio_set_value(BLING, 0);
    }
    return 0;
} « end infrared_read »
```

(7). Test 執行測試檔

建立一 C 語言執行測試檔案，來與 Kernel 互相連結，在 test 檔裡可以清楚看到蜂鳴器需要使用的 operation 是 open 函數及 read 函數；而 open 在 Kernel 裡的 /dev 下 buzzer 此裝置後，Exynos 4412 才可使用 pin 腳（pin 2、pin 13）進行發送及接收訊號；在 while 迴圈中使用 read 函數將超音波感測器的 feed back 回饋訊號傳到變數 buf，再將其列印到 console 上，其中 feed back 訊號已經由 Kernel 中換算成距離單位，若可以印出距離單位的數值，表示開啟裝置、讀取數值成功，傳至 console 也成功，若否，則 ret 檢查用的變數，會 exit。

```
int main(int argc, char *argv[])
{
    int infret = -1;
    int val, condition;
    int freq = 1200;

    open_infrared();
    while(1)
    {
        printf("please select number for condition\n");
        printf("1:Motion detection 2:situational mode 3:exit: ->");
        scanf("%d", &condition);
        printf("\n");

        switch(condition){
            case 1:
                ioctl(infraredfd, 0, infrared_OPEN);
                break;
            case 2:
                ioctl(infraredfd, 1, infrared_OPEN);
                break;
            case 3:
                close_infrared();
            default:
                return 0;
        }

        //infraredfd detection
        infret = read(infraredfd, &val, sizeof(val));
        if(infret < 0){
            printf("Reading infrared data error!\n");
        }
        if (val == 1){
            printf("Alert ! Someone detected!\n");
            printf("LED ON! \n");
        }else{
            printf("no one!\n");
            printf("Buzzer OFF! \n");
        }
        sleep(2);
    } « end while 1 »

    return 0;
} « end main »

static void open_infrared(void){
    infraredfd = open(DEVICE_INFRARED, O_RDONLY); //Open infrared device
    if (infraredfd < 0){
        perror("Open Infrared device failed!!\n");
        exit(1);
    }
    //return infraredfd;
}

static void close_infrared(void){
    if (infraredfd > 0){
        close(infraredfd);
        infraredfd = -1;
    }
}
```


3. Passive Buzzer (無源蜂鳴器)部分

(1). 使用接腳

在無源蜂鳴器上，預計設計成兩個蜂鳴器搭配開啟，在腳位的部分，分配的腳位為 Pin 15 腳，說明如下圖所示：

```
#define MOTOR_PWM_GPIO      EXYNOS4_GPD0(3) //pin15
```

(2). 使用主編號及裝置名稱

```
#define DEVICE_MAJOR 238
#define DEVICE_NAME  "passive_buzzer"

#define MAGIC_NUMBER 'L'
#define PWM_IOCTL_SET_FREQ      _IO(MAGIC_NUMBER, 1)
#define PWM_IOCTL_STOP          _IO(MAGIC_NUMBER, 0)
#define PWM_IOCTL_INPUT_FREQ    _IO(MAGIC_NUMBER, 2)
#define PWM_IOCTL_SET_FREQ_TIME _IO(MAGIC_NUMBER, 3)
```

(3). 程式進入點

進入核心底層後，Server 會先進入 init()內將暫存器開啟、建立需使用的模組 (buzzer)，及給予 Major 號碼提供註冊；再離開 Kernel 時，會進入 exit()裡面將原本建立及註冊的資料做一個清除的動作。

```
static struct class *passive_buzzer_class;

static int __init passive_buzzer_init(void)
{
    int retval;
    //int ret;

    retval = register_chrdev(DEVICE_MAJOR, DEVICE_NAME, &passive_buzzer_fops);
    if(retval < 0)
    {
        printk(KERN_WARNING "Can't get major %d\n", DEVICE_MAJOR);
        return retval;
    }

    printk("passive_buzzer driver register success!\n");

    passive_buzzer_class = class_create(THIS_MODULE, "passive_buzzer");
    if (IS_ERR(passive_buzzer_class))
    {
        printk(KERN_WARNING "Can't make node %d\n", DEVICE_MAJOR);
        return PTR_ERR(passive_buzzer_class);
    }

    device_create(passive_buzzer_class, NULL, MKDEV(DEVICE_MAJOR, 0), NULL, DEVICE_NAME);
    printk("passive_buzzer driver make node success!\n");

    return 0;
} « end passive_buzzer_init »

static void __exit passive_buzzer_exit(void)
{
    gpio_free(MOTOR_PWM_GPIO);
    device_destroy(passive_buzzer_class, MKDEV(DEVICE_MAJOR, 0));
    class_destroy(passive_buzzer_class);
    printk("passive_buzzer driver remove node success!\n");
    unregister_chrdev(DEVICE_MAJOR, DEVICE_NAME);
    printk("passive_buzzer driver release success!\n");
}

module_init(passive_buzzer_init);
module_exit(passive_buzzer_exit);
```

(4). 開啟裝置

在 open 函數中先使用變數來檢查 PIN 變數是否可以使用，加上 if()判斷式，若不能創立則發出"PIN1 fail"的訊息，並將資料釋放（gpio_free）。

```
static int passive_buzzer_open(struct inode *inode, struct file *filp)
{
    /* GPIO 開啟 */

    int ret1 = -1;
    int ret2 = -1;
    ret1 = gpio_request( MOTOR_PWM_GPIO, DEVICE_NAME);
    if( ret1 < 0 ) // request pin 2
    {
        printk( KERN_INFO "%s: %s unable to get TRIG gpio\n", DEVICE_NAME, __func__ );
        ret1 = -EBUSY;
        gpio_free(MOTOR_PWM_GPIO);
        return(ret1);
    }

    ret2 = gpio_direction_output( MOTOR_PWM_GPIO, 0);
    if( ret2 < 0 ) // Set pin 2 as output with default value 0
    {
        printk( KERN_INFO "%s: %s unable to set TRIG gpio as output\n", DEVICE_NAME, __func__ );
        ret2 = -EBUSY;
        return(ret2);
    }

    s3c_gpio_cfgpin(MOTOR_PWM_GPIO, S3C_GPIO_OUTPUT);

    return 0;
} « end passive_buzzer_open »
```

(5). 腳位控制

有關蜂鳴器驅動程式是使用 GPD2 的 I/O 腳位來實現，透過 ioctl 的指令中的 cmd 去控制蜂鳴器開或關與發出長短音，以下是部分程式碼。

```

static long passive_buzzer_ioctl(struct file *file, unsigned int cmd, unsigned long arg, unsigned long duration)
{
    int delay1_us, sing_time;
    int q;
    int ret = 0;

    switch (cmd) {
        case PWM_IOCTL_SET_FREQ:
            if (arg == 0)
                return -EINVAL;

            printk("<1>passive_buzzer_CTL\n");
            delay1_us = (int)arg;
            printk("cmd = %d\n freq = %d\n", cmd, delay1_us);

            for(q=50;q>0;q--){
                gpio_direction_output(MOTOR_PWM_GPIO, 1);
                udelay(delay1_us);
                gpio_direction_output(MOTOR_PWM_GPIO, 0);
                udelay(PWM_IN_FREQ - delay1_us);
            }
            break;

        case PWM_IOCTL_STOP:
        case PWM_IOCTL_INPUT_FREQ:
            if (arg == 0)
                return -EINVAL;

            printk("<1>passive_buzzer_CTL\n");
            delay1_us = (int)arg;
            printk("cmd = %d\n freq = %d\n", cmd, delay1_us);

            for(q=10;q>0;q--){
                gpio_direction_output(MOTOR_PWM_GPIO, 1);
                ndelay(delay1_us);
                gpio_direction_output(MOTOR_PWM_GPIO, 0);
                udelay(PWM_IN_FREQ - delay1_us);
            }
            break;

        case PWM_IOCTL_SET_FREQ_TIME:
            printk("<1>passive_buzzer_SET\n");
            delay1_us = (int)arg;
            int ret;
            ret = copy_from_user(&sing_time, (int*)duration, sizeof(int));
            printk("sing_time = %d\n", sing_time);
            printk("cmd = %d\n freq = %d\nduration = %d\n", cmd, delay1_us, sing_time);

            for(q=sing_time;q>0;q--){
                gpio_direction_output(MOTOR_PWM_GPIO, 1);
                ndelay(delay1_us);
                gpio_direction_output(MOTOR_PWM_GPIO, 0);
                udelay(PWM_IN_FREQ - delay1_us);
            }
            break;

        default:
            break;
    }
    return 0;
}

```

(6). Test 執行測試檔

建立一 C 語言執行測試檔案，來與 Kernel 互相連結，在 test 檔裡可以清楚看到蜂鳴器需要使用的 operation 是 open 函數及 read 函數；而 open 在 Kernel 裡的 /dev 下 buzzer 此裝置後，Exynos 4412 才可使用 pin 腳（pin 2、pin 13）進行發送及接收訊號；在 while 迴圈中使用 read 函數將超音波感測器的 feed back 回饋訊號傳到變數 buf，再將其列印到 console 上，其中 feed back 訊號已經由 Kernel 中換算成距離單位，若可以印出距離單位的數值，表示開啟裝置、讀取數值成功，傳至 console 也成功，若否，則 ret 檢查用的變數，會 exit。

```

int main(int argc, char *argv[])
{
    int melody[] = {
        // HAPPY BIRTHDAY SONG
        // https://github.com/hibit-dev/buzzer/blob/master/src/other/happy\_birthday/happy\_birthday.ino

        NOTE_C4,4, NOTE_C4,8, NOTE_D4,4, NOTE_C4,4,
        NOTE_F4,4, NOTE_E4,2, NOTE_C4,4, NOTE_C4,8,
        NOTE_D4,4, NOTE_C4,4, NOTE_G4,4, NOTE_F4,2,
        NOTE_C4,4, NOTE_C4,8, NOTE_C5,4, NOTE_A4,4, NOTE_F4,4, NOTE_E4,4,
        NOTE_D4,4, REST,1, NOTE_A4,4, NOTE_A4,8, NOTE_A4,4, NOTE_F4,4,
        NOTE_G4,4, NOTE_F4,2,
    };

    int notes = sizeof(melody) / sizeof(melody[0]) / 2;
    fd = open("/dev/passive_buzzer", 0);
    if(fd<0){
        perror("can not open device file: /dev/passive_buzzer\n");
        exit(-1);
    }

    while(1){
        printf("please select number to control passive_buzzer\n");
        printf("1:Set Frequency 2:Echo from 200-5000HZ 3:sing a song 4:exit -> ");
        scanf("%d", &val);
        //printf("\n");
        switch(val){
            case 4:
                printf("EXIT\n");
                close(fd);
                return 0;
            case 1:
                printf("1: please set frequency \n");
                scanf("%d",&frequency);
                freq = (int)frequency*1000;
                //printf("4: please set duration \n");
                //scanf("%d",&duration);
                //duration = (int)duration*100;
                printf("-----\n");
                printf("freq = %d,duration = %d\n", freq,0);
                printf("-----\n");
                ioctl(fd, PWM_IOCTL_INPUT_FREQ, freq,0);
                //usleep(200000);
                break;
            case 2:
                for(i=200;i<=8000;i=i+50){
                    freq = (int)i*1000;
                    printf("-----\n");
                    printf("freq = %d,duration = %d\n", freq,0);
                    printf("-----\n");
                    ioctl(fd, PWM_IOCTL_INPUT_FREQ, freq,0);
                    //usleep(200000);
                }
                break;
            case 3:
                for (thisNote = 0; thisNote < notes * 2; thisNote = thisNote + 2) {
                    int tempo = 120; // change this to make the song slower or faster
                    int wholenote = (60000 * 4) / tempo;
                    int noteDuration;

                    // calculates the duration of each note
                    divider = melody[thisNote + 1];
                    if (divider > 0) {
                        // regular note, just proceed
                        noteDuration = wholenote / divider;
                    } else if (divider < 0) {
                        // dotted notes are represented with negative durations!!
                        noteDuration = wholenote / abs(divider);
                        noteDuration *= 1.5; // increases the duration in half for dotted notes
                    }

                    //for frequency
                    i=melody[thisNote];
                    freq = (int)i*1000;
                    //print word
                    printf("-----\n");
                    printf("freq = %d,duration = %d\n", freq,noteDuration);
                    printf("-----\n");
                    //ioctl(fd, PWM_IOCTL_SET_FREQ_TIME, freq,noteDuration);
                    ioctl(fd, PWM_IOCTL_INPUT_FREQ, freq,0);
                    usleep(noteDuration*400);
                }
            default:
                break;
        }
    }
    return 0;
}

```

(二) JNI 程式設計

Java Native Interface (JNI)是可以讓 Java applications 跟用其它語言 (諸如 C,C++)寫成的應用程式或程式庫互相呼叫使用。在 java 下的 C/C++ 程式如果要使用 Java 的資料型別，型別名稱前必須要加上「j」，例如：「int」變成「jint」，「long」變成「jlong」，「String」變成「jstring」，都是小寫字母。

1. Buzzer (有源蜂鳴器)部分

將原本的 C 語言轉成類似函式的功能，提供 Java 層來呼叫，也就是 Java Call C，以下是藉由 javah 指令，從 java 做出的 header(.h)檔案。

```
1  /* DO NOT EDIT THIS FILE - it is machine generated */
2  #include <jni.h>
3  /* Header for class tw_com_dmatek_buzzer_Linux_buzzer */
4
5  #ifndef _Included_tw_com_dmatek_buzzer_Linux_buzzer
6  #define _Included_tw_com_dmatek_buzzer_Linux_buzzer
7  #ifdef __cplusplus
8  extern "C" {
9  #endif
10
11  /*
12   * Class:      tw_com_dmatek_buzzer_Linux_buzzer
13   * Method:     openbuzzer
14   * Signature:  ()I
15   */
16  JNIEXPORT jint JNICALL Java_tw_com_dmatek_buzzer_Linux_1buzzer_openbuzzer
17      (JNIEnv *, jclass);
18
19  /*
20   * Class:      tw_com_dmatek_buzzer_Linux_buzzer
21   * Method:     closebuzzer
22   * Signature:  ()I
23   */
24  JNIEXPORT jint JNICALL Java_tw_com_dmatek_buzzer_Linux_1buzzer_closebuzzer
25      (JNIEnv *, jclass);
26
27  /*
28   * Class:      tw_com_dmatek_buzzer_Linux_buzzer
29   * Method:     send
30   * Signature:  (II)I
31   */
32  JNIEXPORT jint JNICALL Java_tw_com_dmatek_buzzer_Linux_1buzzer_send
33      (JNIEnv *, jclass, jint, jint);
34
35  #ifdef __cplusplus
36  }
37  #endif
38
```


而針對做出的.h 檔案，進行函數的實際實作，相關的檔案如下。

```
1  #include <string.h>
2  #include <jni.h>
3  //#include "tw_com_dmatek_buzzer_Linux_buzzer.h"
4  #include <stdio.h>
5  #include <stdlib.h> //system
6  #include <fcntl.h>
7  #include <errno.h>
8  #include <unistd.h>
9  #include <linux/delay.h>
10 #include <sys/ioctl.h>
11
12 #include "buzzer_jni.h"
13 #define BUZZER_TEST 3
14
15 #define DEVICE_BLTEST "/dev/buzzer"
16 int fd;
17 JNIEXPORT jint JNICALL Java_tw_com_dmatek_buzzer_Linux_lbuzzer_openbuzzer
18 (JNIEnv* env, jclass thiz)
19 {
20     fd= open(DEVICE_BLTEST,O_RDONLY);
21     return fd;
22 }
23 JNIEXPORT jint JNICALL Java_tw_com_dmatek_buzzer_Linux_lbuzzer_closebuzzer
24 (JNIEnv* env, jclass thiz)
25 {
26     close(fd);
27     return 0;
28 }
29
30 JNIEXPORT jint JNICALL Java_tw_com_dmatek_buzzer_Linux_lbuzzer_send
31 (JNIEnv* env, jclass thiz, jint buzzer_num, jint on_off)
32 {
33     ioctl(fd,on_off,&buzzer_num);
34     return 0;
35 }
```

對應的 driver 在 Kernel 中註冊的名稱。

接下來需要開啟 VM 來將資料做一個輸出轉檔的動作；首先要進入 jni 的資料夾中，檢查附檔名 Androidmk 檔及原檔是否正確；並下指令(ndk-build)讓原檔進行 Compile 動作，得到 libxxx.so 檔案。

編譯產生 so 檔後，還需要 eclipse 來模擬 app 使用層，並得到 apk 檔案；在要開啟的資料夾中，必須包含環境變數(env)及固定資料夾(src、res 等)，才可以去執行模擬。

2. infrared 部分

將原本的 C 語言轉成類似函式的功能，提供 Java 層來呼叫，也就是 Java Call C，以下是藉由 javah 指令，從 java 做出的 header(.h)檔案。

```
/* DO NOT EDIT THIS FILE - it is machine generated */
#include <jni.h>
/* Header for class tw_com_dmatek_buzzer_Linux_infrared */

#ifndef _Included_tw_com_dmatek_buzzer_Linux_infrared
#define _Included_tw_com_dmatek_buzzer_Linux_infrared
#ifdef __cplusplus
extern "C" {
#endif
/*
 * Class:      tw_com_dmatek_buzzer_Linux_infrared
 * Method:     openinfrared
 * Signature:  ()I
 */
JNIEXPORT jint JNICALL Java_tw_com_dmatek_buzzer_Linux_infrared_openinfrared
(JNIEnv *, jclass);

/*
 * Class:      tw_com_dmatek_buzzer_Linux_infrared
 * Method:     closeinfrared
 * Signature:  ()I
 */
JNIEXPORT jint JNICALL Java_tw_com_dmatek_buzzer_Linux_infrared_closeinfrared
(JNIEnv *, jclass);

/*
 * Class:      tw_com_dmatek_buzzer_Linux_infrared
 * Method:     ioctl_infrared
 * Signature:  (I)I
 */
JNIEXPORT jint JNICALL Java_tw_com_dmatek_buzzer_Linux_infrared_ioctl_infrared
(JNIEnv *, jclass, jint);

/*
 * Class:      tw_com_dmatek_buzzer_Linux_infrared
 * Method:     read_infrared
 * Signature:  ()I
 */
JNIEXPORT jint JNICALL Java_tw_com_dmatek_buzzer_Linux_infrared_read_infrared
(JNIEnv *, jclass);
#ifdef __cplusplus
}
#endif
#endif
```

而針對做出的.h 檔案，進行函數的實際實作，相關的檔案如下。

```
#include <string.h>
#include <jni.h>
#include <stdio.h>
#include <stdlib.h> //system
#include <fcntl.h>
#include <errno.h>
#include <unistd.h>
// #include <linux/delay.h>
#include <sys/ioctl.h>

#define DEVICE_BLTEST "/dev/infrared"
#define infrared_OPEN 1
#define infrared_STOP 0
static int infraredfd = -1;

int fd, infret, val;

JNIEXPORT jint JNICALL Java_tw_com_dmatek_buzzer_Linux_1infrared_openinfrared
(JNIEnv * env, jclass this)
{
    fd = open(DEVICE_BLTEST, O_RDONLY);
    return fd;
}

JNIEXPORT jint JNICALL Java_tw_com_dmatek_buzzer_Linux_1infrared_closeinfrared
(JNIEnv * env, jclass this)
{
    printf("EXIT\n");
    close(fd);
    return 0;
}

JNIEXPORT jint JNICALL Java_tw_com_dmatek_buzzer_Linux_1infrared_ioctl_infrared
(JNIEnv * env, jclass this, jint condition_num)
{
    ioctl(fd, condition_num, infrared_OPEN);
    return 0;
}

JNIEXPORT jint JNICALL Java_tw_com_dmatek_buzzer_Linux_1infrared_read_infrared
(JNIEnv * env, jclass this)
{
    infret = read(fd, &val, sizeof(val));
    return val;
}
```

3. Passive Buzzer (無源蜂鳴器)部分

將原本的 C 語言轉成類似函式的功能，提供 Java 層來呼叫，也就是 Java Call C，以下是藉由 javah 指令，從 java 做出的 header(.h)檔案。

```
/* DO NOT EDIT THIS FILE - it is machine generated */
#include <jni.h>
/* Header for class com_ispan_android_piano_linux_passive_buzzer */

#ifdef __cplusplus
extern "C" {
#endif
/*
 * Class: com_ispan_android_piano_linux_passive_buzzer
 * Method: open_passive_buzzer
 * Signature: ()I
 */
JNIEXPORT jint JNICALL Java_com_ispan_android_piano_linux_1passive_1buzzer_open_1passive_1buzzer
(JNIEnv *, jclass);

/*
 * Class: com_ispan_android_piano_linux_passive_buzzer
 * Method: close_passive_buzzer
 * Signature: ()I
 */
JNIEXPORT jint JNICALL Java_com_ispan_android_piano_linux_1passive_1buzzer_close_1passive_1buzzer
(JNIEnv *, jclass);

/*
 * Class: com_ispan_android_piano_linux_passive_buzzer
 * Method: play_run_passive_buzzer
 * Signature: ()I
 */
JNIEXPORT jint JNICALL Java_com_ispan_android_piano_linux_1passive_1buzzer_play_1run_1passive_1buzzer
(JNIEnv *, jclass);

/*
 * Class: com_ispan_android_piano_linux_passive_buzzer
 * Method: play_music_passive_buzzer
 * Signature: ()I
 */
JNIEXPORT jint JNICALL Java_com_ispan_android_piano_linux_1passive_1buzzer_play_1music_1passive_1buzzer
(JNIEnv *, jclass);

/*
 * Class: com_ispan_android_piano_linux_passive_buzzer
 * Method: send_passive_buzzer
 * Signature: (II)I
 */
JNIEXPORT jint JNICALL Java_com_ispan_android_piano_linux_1passive_1buzzer_send_1passive_1buzzer
(JNIEnv *, jclass, jint, jint);
#ifdef __cplusplus
}
#endif
```

而針對做出的.h 檔案，進行函數的實際實作，相關的檔案如下。

```
int melody[] = {
    // HAPPY BIRTHDAY SONG
    // https://github.com/hibit-dev/buzzer/blob/master/src/other/happy_birthday/happy_birthday.ino
    NOTE_C4,4, NOTE_C4,8, NOTE_D4,4, NOTE_C4,8,
    NOTE_F4,4, NOTE_F4,8, NOTE_C4,4, NOTE_C4,8,
    NOTE_D4,4, NOTE_C4,8, NOTE_G4,4, NOTE_F4,8,
    NOTE_C4,4, NOTE_C4,8, NOTE_C5,4, NOTE_A4,4, NOTE_F4,4, NOTE_E4,4,
    NOTE_D4,4, REST,1, NOTE_A5,4, NOTE_A5,8, NOTE_A4,4, NOTE_F4,4,
    NOTE_G4,4, NOTE_F4,8,
};

int notes = sizeof(melody) / sizeof(melody[0]) / 2;

#define DEVICE_BLTEST "/dev/passive_buzzer"
int fd;
JNIEXPORT jint JNICALL Java_com_ispan_android_piano_Linux_ipassive_ibuzzer_open_ipassive_ibuzzer
(JNIEnv* env, jclass this)
{
    fd = open(DEVICE_BLTEST, O_RDONLY);
    return fd;
}

JNIEXPORT jint JNICALL Java_com_ispan_android_piano_Linux_ipassive_ibuzzer_close_ipassive_ibuzzer
(JNIEnv* env, jclass this)
{
    printf("EXIT\n");
    close(fd);
    return 0;
}

JNIEXPORT jint JNICALL Java_com_ispan_android_piano_Linux_ipassive_ibuzzer_send_ipassive_ibuzzer
(JNIEnv* env, jclass this, jint frequency, jint duration)
{
    ioctl(fd, PWM_IOCTL_INPUT_FREQ, frequency, 0);
    return 0;
}

JNIEXPORT jint JNICALL Java_com_ispan_android_piano_Linux_ipassive_ibuzzer_play_irun_ipassive_ibuzzer
(JNIEnv* env, jclass this)
{
    int i, freq = 0;
    for(i=200; i<=6000; i+=50){
        freq = (int)i*1000;
        ioctl(fd, PWM_IOCTL_INPUT_FREQ, freq, 0);
    }

    return 0;
}

JNIEXPORT jint JNICALL Java_com_ispan_android_piano_Linux_ipassive_ibuzzer_play_imusic_ipassive_ibuzzer
(JNIEnv* env, jclass this, jint frequency)
{
    int thisNote;
    for(thisNote = 0; thisNote < notes * 2; thisNote = thisNote + 2) {
        int tempo = 120; // change this to make the song slower or faster
        int wholenote = (60000 * 4) / tempo;
        int noteDuration;
        // calculates the duration of each note
        int divider = melody[thisNote + 1];
        if (divider > 0) {
            // regular note, just proceed
            noteDuration = wholenote / divider;
        } else if (divider < 0) {
            // dotted notes are represented with negative durations!!
            noteDuration = wholenote / abs(divider);
            noteDuration *= 1.5; // increases the duration in half for dotted notes
        }

        //for frequency
        int i=melody[thisNote];
        int freq = (int)i*1000;
        //ioctl(fd, PWM_IOCTL_SET_FREQ_TIME, freq, noteDuration);
        ioctl(fd, PWM_IOCTL_INPUT_FREQ, freq, 0);
        usleep(noteDuration*400);
    }

    return 0;
}
```

接下來需要開啟 VM 來將資料做一個輸出轉檔的動作；首先要進入 jni 的資料夾中，檢查附檔名 Androidmk 檔及原檔是否正確；並下指令(ndk-build)讓原檔進行 Compile 動作，得到 libxxx.so 檔案。

編譯產生 so 檔後，還需要 eclipse 來模擬 app 使用層，並得到 apk 檔案；在要開啟的資料夾中，必須包含環境變數(env)及固定資料夾(src、res 等)，才可以去執行模擬。

(三) Application-MyCombine.apk(第一支 Android App 程式)

1. 主程式

其中使用執行緒做為各動作處理，如下所示:

```
public class BuzzerActivity extends Activity {
    private static final String TAG = BuzzerActivity.class.getSimpleName();
    public int state1=0, state2=0;
    public int buzzer_on = 1;
    public int buzzer_off = 3;
    public int fd = 0;

    private Handler mHandler;
    private HandlerThread mThread;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Log.d(TAG, "find View");
        findViews();
        Log.d(TAG, "Set Listener");
        setListeners();

        fd = Linux_buzzer.openbuzzer();
        if(fd<0){
            Log.d(TAG, "open device fail!");
            Toast.makeText(BuzzerActivity.this, "open device fail!", Toast.LENGTH_SHORT).show();
            //finish();
        }else{
            Toast.makeText(BuzzerActivity.this, "open device success!", Toast.LENGTH_SHORT).show();
            Log.d(TAG, "open device success!");
        }
        fd = Linux_infrared.openinfrared();
        if(fd<0){
            Log.d(TAG, "open infrared device fail!");
            Toast.makeText(BuzzerActivity.this, "open infrared device fail!", Toast.LENGTH_SHORT).show();
            //finish();
        }else{
            Toast.makeText(BuzzerActivity.this, "open infrared device success!", Toast.LENGTH_SHORT).show();
            Log.d(TAG, "open device success!");
        }
        //let Worker standby, and wait for mission
        mThread = new HandlerThread("name");
        //start detection
        mThread.start();
        mHandler = new Handler(mThread.getLooper());
    } // end onCreate

    private Handler mUI_Handler = new Handler(){
        public void haddlerMessage(Message msg){
            super.handleMessage(msg);
            //mmtxt_infrared.setText("read data=", Integer.toString(msg.getData().getInt("read", 0)));
            mmtxt_infrared.setText(String.valueOf(Linux_infrared.read_infrared()));
        }
    };

    private static boolean top = false;
    private Runnable r1 = new Runnable(){
        public void run(){
            while(!top){
                try {
                    Message msg = new Message();

                    int read_value = Linux_infrared.read_infrared();
                    Bundle readBundle = new Bundle();
                    readBundle.putInt("read", read_value);
                    msg.setData(readBundle);
                    mUI_Handler.sendMessage(msg);

                    //Toast.makeText(Buzzer_Control_Activity.this, Integer.toString(msg.getData().getInt("read", 0)), Toast.LENGTH_SHORT).show();

                    if(read_value == 0 && getTitle()=="Warming Sensor Start!!!"){
                        int i;
                        for(i=0; i<5; i++){
                            Linux_buzzer.send(1, buzzer_on);
                            Linux_buzzer.send(2, buzzer_on);
                            Thread.sleep(100);
                            Linux_buzzer.send(1, buzzer_off);
                            Linux_buzzer.send(2, buzzer_off);
                            Thread.sleep(100);
                        }
                        Thread.sleep(2000);
                    } // end try
                } catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            } // end while !top
        } // end run
    }; // end {anon_class_73}
}
```



```

private Button mbutton1;
private Button mbutton2;
private Button mbutton3;
private Button mbutton4;
private Button mbutton5;
private TextView view_welcome;
private Button mmyButton_WARMING;
private Button mmyButton_PARTY;
private Button mmyButton_RESET;
private TextView mmtxt_infrared;
//取得畫面的按鈕物件的控制權
private void findViews(){
    mbutton1 = (Button) findViewById(R.id.myButton1);
    mbutton2 = (Button) findViewById(R.id.myButton1_1);
    mbutton3 = (Button) findViewById(R.id.myButton2);
    mbutton4 = (Button) findViewById(R.id.myButton2_1);
    mbutton5 = (Button) findViewById(R.id.myButton3);
    mmyButton_WARMING = (Button) findViewById(R.id.myButton_WARMING);
    mmyButton_PARTY = (Button) findViewById(R.id.myButton_PARTY);
    mmyButton_RESET = (Button) findViewById(R.id.myButton_reset);
    view_welcome = (TextView) findViewById(R.id.text);
    mmtxt_infrared = (TextView) findViewById(R.id.textView_infrared);
}
//Listen for button clicks
private void setListeners(){
    mbutton1.setOnClickListener(button1_listener);
    mbutton2.setOnClickListener(button2_listener);
    mbutton3.setOnClickListener(button3_listener);
    mbutton4.setOnClickListener(button4_listener);
    mbutton5.setOnClickListener(button5_listener);
    mmyButton_WARMING.setOnClickListener(mmyButton_WARMING_listener);
    mmyButton_PARTY.setOnClickListener(mmyButton_PARTY_listener);
    mmyButton_RESET.setOnClickListener(mmyButton_RESET_listensr);
}

private Button.OnClickListener mmyButton_WARMING_listener=new Button.OnClickListener(){
    public void onClick(View v){
        try{
            Toast.makeText(BuzzerActivity.this, "mmyButton_WARMING_listener",Toast.LENGTH_SHORT ).show();
            Linux_infrared.iocctl_infrared(0);
            Linux_infrared.iocctl_infrared(0);
            setTitle("Warming Sensor Start!!!");
            top=false;
            mHandler.post(r1);
        }catch(Exception e){
            Toast.makeText(BuzzerActivity.this, "input_error",Toast.LENGTH_SHORT ).show();
        }
    }
};

private Button.OnClickListener mmyButton_PARTY_listener=new Button.OnClickListener(){
    public void onClick(View v){
        try{
            Toast.makeText(BuzzerActivity.this, "mmyButton_PARTY_listener",Toast.LENGTH_SHORT ).show();
            //Linux_infrared_and_led.iocctl_infrared(3);
            Linux_infrared.iocctl_infrared(1);
            Linux_infrared.iocctl_infrared(1);
            setTitle("PARTY Sensor Start!!!");
            top=false;
            mHandler.post(r1);
        }catch(Exception e){
            Toast.makeText(BuzzerActivity.this, "input_error",Toast.LENGTH_SHORT ).show();
        }
        //mHandler.post(r1);
    }
};

```

```

private Button.OnClickListener mmyButton_RESET_listensr=new Button.OnClickListener(){
    public void onClick(View v){
        try{
            Toast.makeText(BuzzerActivity.this, "mmyButton_RESET_listensr",Toast.LENGTH_SHORT ).show();
            Linux_infrared.ioctl_infrared(3);//close all
            setTitle("Sensor Stop!!!");
            top=true;
        }catch(Exception e){
            Toast.makeText(BuzzerActivity.this, "input_error",Toast.LENGTH_SHORT ).show();
        }
    }
};

private Button.OnClickListener button1_listener=new Button.OnClickListener(){
    public void onClick(View v){
        try{
            Toast.makeText(BuzzerActivity.this, "button1_listener",Toast.LENGTH_SHORT ).show();
            Linux_buzzer.send(1,buzzer_on);
        }catch(Exception e){
            Toast.makeText(BuzzerActivity.this, "input_error",Toast.LENGTH_SHORT ).show();
        }
    }
};

private Button.OnClickListener button2_listener=new Button.OnClickListener(){
    public void onClick(View v){
        try{
            Toast.makeText(BuzzerActivity.this, "button2_listener",Toast.LENGTH_SHORT ).show();
            Linux_buzzer.send(1,buzzer_off);
        }catch(Exception e){
            Toast.makeText(BuzzerActivity.this, "input_error",Toast.LENGTH_SHORT ).show();
        }
    }
};

private Button.OnClickListener button3_listener=new Button.OnClickListener(){
    public void onClick(View v){
        try{
            Toast.makeText(BuzzerActivity.this, "button3_listener",Toast.LENGTH_SHORT ).show();
            Linux_buzzer.send(2,buzzer_off);
        }catch(Exception e){
            Toast.makeText(BuzzerActivity.this, "input_error",Toast.LENGTH_SHORT ).show();
        }
    }
};

private Button.OnClickListener button4_listener=new Button.OnClickListener(){
    public void onClick(View v){
        try{
            Toast.makeText(BuzzerActivity.this, "button4_listener",Toast.LENGTH_SHORT ).show();
            Linux_buzzer.send(2,buzzer_on);
        }catch(Exception e){
            Toast.makeText(BuzzerActivity.this, "input_error",Toast.LENGTH_SHORT ).show();
        }
    }
};

private Button.OnClickListener button5_listener=new Button.OnClickListener(){
    public void onClick(View v){
        try{
            Toast.makeText(BuzzerActivity.this, "button5_listener",Toast.LENGTH_SHORT ).show();
            Linux_buzzer.closebuzzer();
            top=true;
            if(mThreadHandler != null){
                mThreadHandler.removeCallbacks(r1);
            }
            if(mThread != null){
                mThread.quit();
            }
            finish();
        }catch(Exception e){
            Toast.makeText(BuzzerActivity.this, "input_error",Toast.LENGTH_SHORT ).show();
        }
    }
};

```

```

private void openOptionDialog(){
    Toast.makeText(BuzzerActivity.this, "about Buzzer", Toast.LENGTH_SHORT ).show();
    //to create warming box
    new AlertDialog.Builder(BuzzerActivity.this)
        .setTitle(R.string.about_title)
        .setMessage(R.string.about_msg)
        .setPositiveButton("Confirm", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialoginterface,int i){}
        })
        .setNegativeButton(R.string.homepage_label, new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                // TODO Auto-generated method stub
                //Homepage
                Uri uri = Uri.parse("http://tw.yahoo.com/");
                Intent intent = new Intent(Intent.ACTION_VIEW,uri);
                startActivity(intent);
            }
        })
        .show();
} // « end openOptionDialog »

protected static final int MENU_ABOUT = Menu.FIRST;
protected static final int MENU_QUIT = Menu.FIRST+1;
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // TODO Auto-generated method stub
    menu.add(0,MENU_ABOUT,0,"關於").setIcon(android.R.drawable.ic_menu_help);
    menu.add(0,MENU_QUIT,0,"結束").setIcon(android.R.drawable.ic_menu_close_clear_cancel);
    return super.onCreateOptionsMenu(menu);
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // TODO Auto-generated method stub
    switch(item.getItemId()){
        case MENU_ABOUT:
            openOptionDialog();
        case MENU_QUIT:
            finish();
            break;
    }
    return super.onOptionsItemSelected(item);
}
} // « end BuzzerActivity »

```

2. 呼叫程式

透過 Linuxc.java 去呼叫 JNI 相關函式

```

package tw.com.dmatek.buzzer;

import android.util.Log;

public class Linux_buzzer {
    static{
        try{
            Log.i("JNI", "Try to load libbuzzer.so");

            System.loadLibrary("buzzer");
        }catch(UnsatisfiedLinkError ule){
            Log.i("JNI", "Warning: Could not load libbuzzer.so");
        }
    }

    public static native int openbuzzer();
    public static native int closebuzzer();
    public static native int send(int buzzer_num, int on_off);
}

```

透過 Linuxc.java 去呼叫 JNI 相關函式

```
package tw.com.dmatek.buzzer;

import android.util.Log;

public class Linux_infrared {
    static{
        try{
            Log.i("JNI", "Try to load libinfrared.so");

            System.loadLibrary("infrared");
        }catch(UnsatisfiedLinkError ule){
            Log.i("JNI", "Warning: Could not load libinfrared.so");
        }
    }

    public static native int openinfrared();
    public static native int closeinfrared();
    public static native int ioctl_infrared(int condition_num);
    public static native int read_infrared();
}
```

最後，從 VMware 編譯得到的 so 檔及 apk 檔，接下來就要放入 Samsung Exynos 4412 中；首先，要先在 4412 建立網路與 tftp32 做連接，測試成功，已可以取得封包。建立完連線後，必須把剛剛產出的 apk 檔及 libxxx.so 檔放入 Android 層相對應的資料夾下，使其載入使用 busybox 指令並下參數將 apk 檔案需放入/app 下，而 so 檔案則放入/lib 下，讀取 100%時表示已完成傳送。

(四) Application-MyPiano.apk(第二支 Android App 程式)

1. 主程式

```
public class MyPianoActivity extends Activity {
    private static final String TAG=MyPianoActivity.class.getSimpleName();
    public int fd=0;
    public int freq;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Log.d(TAG,"findView");
        findViews();
        Log.d(TAG,"Set Listener");
        setListeners();

        fd = Linux_passive_buzzer.open_passive_buzzer();
        if(fd<0){
            Log.d(TAG,"open device fall!!");
            Toast.makeText(MyPianoActivity.this, "open device fall!!", Toast.LENGTH_SHORT).show();
        }else{
            Log.d(TAG,"open device success!!");
            Toast.makeText(MyPianoActivity.this, "open device success!!", Toast.LENGTH_SHORT).show();
        }
    }

    private Button mButton_run;
    private Button mButton_music;
    private Button mButton_exit;
    private View view_fa_minus;
    private View view_so_minus;
    private View view_la_minus;
    private View view_si_minus;
    private View view_do;
    private View view_re;
    private View view_mi;
    private View view_fa;
    private View view_so;
    private View view_la;
    private View view_si;

    private void findViews(){
        mButton_run = (Button)findViewById(R.id.button_run);
        mButton_music = (Button)findViewById(R.id.button_music);
        mButton_exit = (Button)findViewById(R.id.button_exit);
        view_fa_minus = (View)findViewById(R.id.button_fa_minus);
        view_so_minus = (View)findViewById(R.id.button_so_minus);
        view_la_minus = (View)findViewById(R.id.button_la_minus);
        view_si_minus = (View)findViewById(R.id.button_si_minus);
        view_do = (View)findViewById(R.id.button_do);
        view_re = (View)findViewById(R.id.button_re);
        view_mi = (View)findViewById(R.id.button_mi);
        view_fa = (View)findViewById(R.id.button_fa);
        view_so = (View)findViewById(R.id.button_so);
        view_la = (View)findViewById(R.id.button_la);
        view_si = (View)findViewById(R.id.button_si);
    }

    private void setListeners(){
        Log.d(TAG,"Set Listener");
        mButton_run.setOnClickListener(mButton_run_listener);
        mButton_music.setOnClickListener(mButton_music_listener);
        mButton_exit.setOnClickListener(mButton_exit_listener);
        view_fa_minus.setOnClickListener(view_fa_minus_listener);
        view_so_minus.setOnClickListener(view_so_minus_listener);
        view_la_minus.setOnClickListener(view_la_minus_listener);
        view_si_minus.setOnClickListener(view_si_minus_listener);
        view_do.setOnClickListener(view_do_listener);
        view_re.setOnClickListener(view_re_listener);
        view_mi.setOnClickListener(view_mi_listener);
        view_fa.setOnClickListener(view_fa_listener);
        view_so.setOnClickListener(view_so_listener);
        view_la.setOnClickListener(view_la_listener);
        view_si.setOnClickListener(view_si_listener);
    }
}
```

```

private Button.OnClickListener mButton_run_listener=new Button.OnClickListener(){
    public void onClick(View v){
        try{
            Toast.makeText(MyPianoActivity.this, "氣氛音樂",Toast.LENGTH_SHORT ).show();
            Linux_passive_buzzer.play_run_passive_buzzer();
        }catch(Exception e){
            Toast.makeText(MyPianoActivity.this, "mButton_music_listener_error",Toast.LENGTH_SHORT ).show();
        }
    }
};

private Button.OnClickListener mButton_music_listener=new Button.OnClickListener(){
    public void onClick(View v){
        try{
            Toast.makeText(MyPianoActivity.this, "播放歌曲",Toast.LENGTH_SHORT ).show();
            Linux_passive_buzzer.play_music_passive_buzzer();
        }catch(Exception e){
            Toast.makeText(MyPianoActivity.this, "mButton_music_listener_error",Toast.LENGTH_SHORT ).show();
        }
    }
};

private Button.OnClickListener mButton_exit_listener=new Button.OnClickListener(){
    public void onClick(View v){
        try{
            Toast.makeText(MyPianoActivity.this, "EXIT",Toast.LENGTH_SHORT ).show();
            Linux_passive_buzzer.close_passive_buzzer();
            finish();
        }catch(Exception e){
            Toast.makeText(MyPianoActivity.this, "mButton_music_listener_error",Toast.LENGTH_SHORT ).show();
        }
    }
};

private View.OnClickListener view_fa_minus_listener=new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        try{
            Toast.makeText(MyPianoActivity.this, "-Fa key",Toast.LENGTH_SHORT ).show();
            freq = 174600; //174.6KHz
            Linux_passive_buzzer.send_passive_buzzer(freq, 1);
        }catch(Exception e){
            Toast.makeText(MyPianoActivity.this, "button_fa_minus_error",Toast.LENGTH_SHORT ).show();
        }
    }
};

```

(部分程式)

2. 呼叫程式

透過 Linuxc.java 去呼叫 JNI 相關函式

```

package com.ispan.android.piano;

import android.util.Log;

public class Linux_passive_buzzer {
    static{
        try{
            Log.i("JNI", "Try to load libpassivebuzzer.so");
            System.loadLibrary("passivebuzzer");
        }catch(UnsatisfiedLinkError ule){
            Log.i("JNI", "Warming: Could not load libpassivebuzzer.so");
        }
    }

    public static native int open_passive_buzzer();
    public static native int close_passive_buzzer();
    public static native int play_run_passive_buzzer();
    public static native int play_music_passive_buzzer();
    public static native int send_passive_buzzer(int frequency, int duration);
}

```


3. 4412 開發板執行

最後，從 VMware 編譯得到的 so 檔及 apk 檔，接下來就要放入 Samsung Exynos 4412 中

- (1). 首先，要先在 4412 建立網路連接，測試成功，已可以取得封包。
- (2). 建立完連線後，必須把剛剛產出的 apk 檔及 libxxx.so 檔放入 Android 層相對應的資料夾下(/system/app)及(/system/libs)，而驅動程式所屬位置則在/system/bin；使其載入使用 busybox 指令，讀取 100%時表示已完成傳送。
- (3). 本次採手動配置驅動程式並開啟權限，所以在驅動程式所在目錄下執行

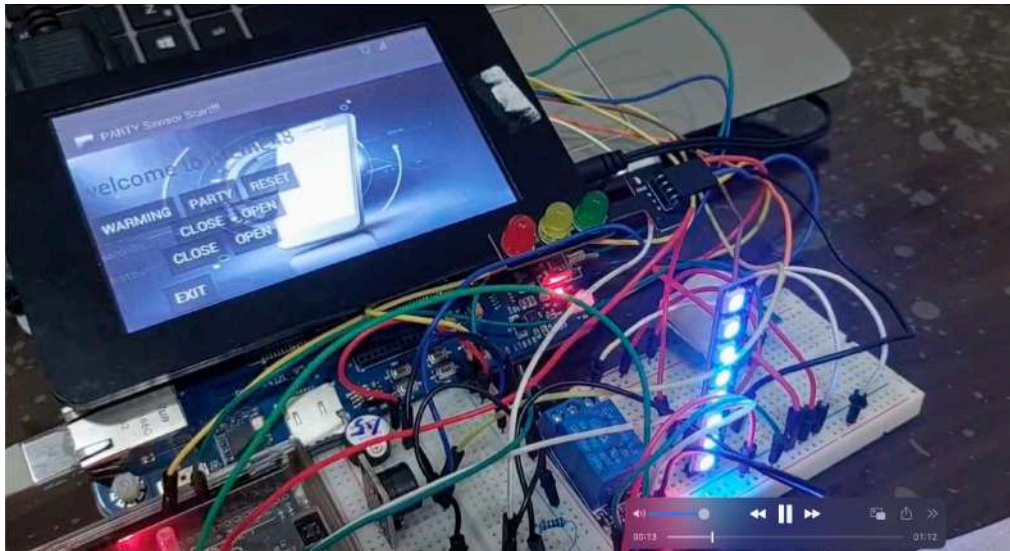
```
#insmod buzzer.ko
#insmod passivebuzzer.ko
#insmod indrared.ko
```
- (4). 安裝驅動程式成功後尚需至設備目錄/dev/下開啟權限

```
#chmod 777 buzzer
#chmod 777 passivebuzzer
#chmod 777 indrared
```
- (5). 點選 APP 就可以正式使用了~~~

十、 測試結果

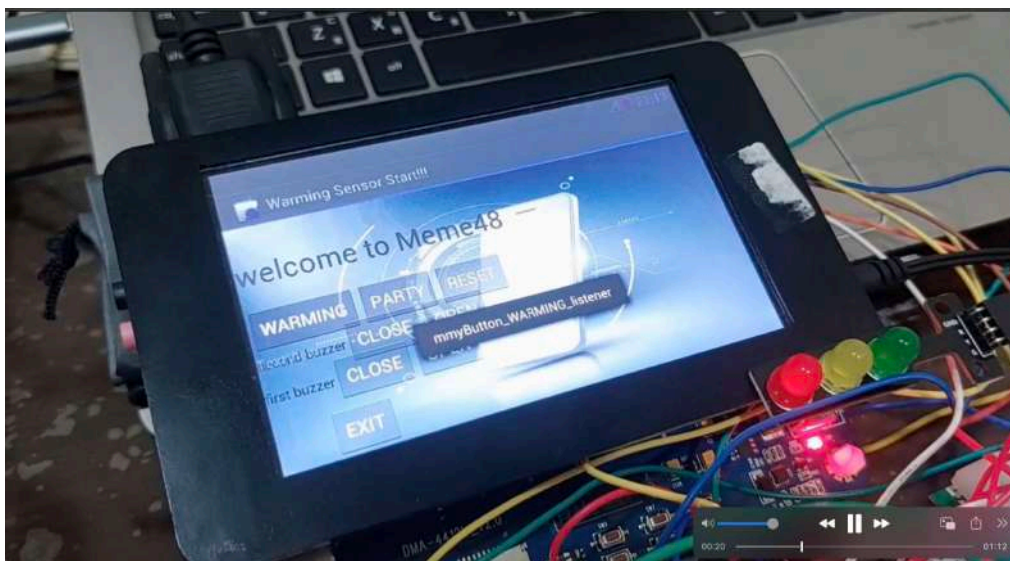
- 啟動燈光氣氛模式，並利用紅外線偵測是否有人：

當紅外線偵測到人時，送訊號至 WS2812B LED 燈條，開啟 LED 燈光調節氣氛。



- 啟動哨兵警衛氣氛模式，並利用紅外線偵測是否有人：

當紅外線偵測到人時，送訊號至 LED 及有源蜂鳴器，開啟 LED 閃爍以及蜂鳴器警告。



- 影訊娛樂模式：

在音樂娛樂的部分，包含三大功能，帶您進入音訊娛樂的殿堂

- (1). 可以當成鋼琴自彈自唱
- (2). 播放預設歌曲音樂
- (3). 播放預設氣氛頻率音樂

