微處理機系統實習　　Lab9

班級：資訊三丁學號：D1009212 姓名：邱柏宇

一、【實驗目的】：

What was your design? What were the concepts you have used for your design?

9.1

這次主要是練習 UART，題目邏輯很像之前第一次上機考第二題，所以取隨機不重複是直接拿之前的程式碼，用陣列去存隨機四位數，每次都先 rand() % 9 + 1 後都會在檢查陣列裡面有沒相同的，有的話就再取一次隨機，如果沒有再放進去陣列，最後再轉為 int 回傳。

```
uint16_t get4rand(void) {
    uint8_t i, j;
    uint8_t rands[4] = {0, 0, 0, 0};
    uint16_t rand_num = 0;
    uint8_t temp = 0;
    uint8_t check = 0;
    for (i = 0; i < 4; i++) {
        while (TRUE) {
            temp = rand() % 9 + 1;
            check = 0;
            for (j = 0; j < 4; j++) {
                if (temp == rands[j]) {
                    temp = rand() % 9 + 1;
                } else {
                    check++;
                }
            }
            if (check == 4) break;
        }
        rands[i] = temp;
        rand_num = rand_num * 10 + temp;
    }
    return rand_num;
}
```

關於幾 A 幾 B 的邏輯就要小改一下，因為七段顯示的陣列是 3210，但是對方輸入數字時是從左邊，所以我在使用 for 去比對時會將七段顯示顯示的密碼倒過來檢查(3 - i)。

Ps. print_c_in_buffer = 原版 printC。

```
void check_num_is_correct(void) {
    uint8_t a = 0, b = 0;
    uint8_t i = 0, j = 0;
    for (i = 0; i < 4; i++) {
        if (rx_data[i] == pwd[3 - i]) {
            a++;
        } else {
            for (j = 0; j < 4; j++) {
                if (rx_data[i] == pwd[3 - j] && i != j) {
                    b++;
                    break;
                }
            }
        }
    }
    print_c_in_buffer(10 * 8, printc_line_index * 16, 8, a + '0', FG_COLOR);
    print_c_in_buffer(11 * 8, printc_line_index * 16, 8, 'A', FG_COLOR);
    print_c_in_buffer(12 * 8, printc_line_index * 16, 8, b + '0', FG_COLOR);
    print_c_in_buffer(13 * 8, printc_line_index * 16, 8, 'B', FG_COLOR);
    show_lcd_buffer();
}
```

這題感覺主要是用 UART，我是每按一個按鍵就傳送一個數字給對方，不過接收時就比較特別，因為我不想要每次都清空 lcd，然後在印出，所以也是跟上面幾 A 幾 B 類似，創建一個變數來當作游標，然後印出時游標位置++，這樣就不用每次都清除。

（還有下一頁）

```
if (c != '0') {
    print_c_in_buffer(8 * printc_index, 16 * (printc_line_index % 4), 8, c, FG_COLOR);
    show_lcd_buffer();
    if (++printc_index == 4) {
        check_num_is_correct();
        printc_index = 0;
        printc_line_index++;
        rx_index = 0;
    }
}
```

9.2

MPU6050 這題就很玄了，我看大家都用 sample code，但我的板子用 1.2 就動不起來，所以我去參考一下官方 1.4.5 版本的 code，發現新版把 MPU6050.h 改了一遍，主要是用了 NVT_I2C.h，主要的差異就是 I2C 改用 Cortex-M 中 NVIC 的中斷，剩下的算式就是把三軸加速度來換算出 pitch 和 roll，下面為了防止球在抖和飄，用>10 當作傾斜超過一定大小才作用給球。

```
while (TRUE) {
    MPU6050_getAcceleration(&ax, &ay, &az);
    pitch = atan(ax / sqrt(pow(ay, 2) + pow(az, 2))) * 180 / PI;
    roll = atan(ay / sqrt(pow(ax, 2) + pow(az, 2))) * 180 / PI;

    dx = 0;
    if (pitch > 10 || pitch < -10) dx = pitch / abs(pitch) * -1;

    dy = 0;
    if (roll > 10 || roll < -10) dy = roll / abs(roll);

    x += dx * vsp;
    y += dy * hsp;
    if(x - r < 0 || x + r > LCD_Xmax) x -= dx * vsp;
    if(y - r < 0 || y + r > LCD_Ymax) y -= dy * hsp;

    draw_circle_in_buffer(x, y, r, FG_COLOR, TRUE);
    show_lcd_buffer();
}
```

二、【遭遇的問題】：

What problems you faced during design and implementation?

MPU6050 的斷線問題，懷疑是杜邦線的接觸不良。

三、【解決方法】：

How did you solve the problems?

嘗試使用 MPU6050_testConnection() 去偵測斷線，如果斷線就重新初始化。

```
/** Verify the I2C connection.
 * Make sure the device is connected and responds as expected.
 * @return True if connection is valid, false otherwise
 */
bool MPU6050_testConnection(void) {
    return MPU6050_getDeviceID() == 0x34;
}
```

四、【未能解決的問題】：

Was there any problem that you were unable to solve? Why was it unsolvable?

上述的斷線問題還是沒有辦法做到偵測和解決，因為實際上是用 MPU6050_getDeviceID()去取得 id，所以斷線時候連接回來，會無法偵測。

```
/** Get Device ID.
 * This register is used to verify the identity of the device (0b110100, 0x34).
 * @return Device ID (6 bits only! should be 0x34)
 * @see MPU6050_RA_WHO_AM_I
 * @see MPU6050_WHO_AM_I_BIT
 * @see MPU6050_WHO_AM_I_LENGTH
 */
uint8_t MPU6050_getDeviceID(void) {
    I2Cdev_readBits(devAddr, MPU6050_RA_WHO_AM_I, MPU6050_WHO_AM_I_BIT, MPU6050_WHO_AM_I_LENGTH, buffer);
    return buffer[0];
}
/** Set Device ID.
```

五、【上課老師的問題】：

I2C 是甚麼？和 SPI 差異在哪？

I2C 的通訊協定是一種同步循址協定，BUS 由 Vdd、SDA、SCL 組成，所有的裝置都在這條 BUS 上面，可以同時擁有多個主和多個從。

I2C 的格式由 S、ADDRESS、W/R、ACK、DATA、ACK、DATA、ACK、P 組成，所以可以用前面的 ADDRESS 來找到傳輸對方的位置在哪。

傳輸格式

```
_ _______ _   _   _______ _ _   _______ _ _   _
S ADDRESS W/R ACK DATA      ACK DATA      ACK P
```

優點：電路簡單，裝置只需要接在 BUS 上面規定好地址彼此之間就可以傳輸，並且有自帶偵錯機制。

缺點：速度慢，不適合長距離的傳輸，不同電壓裝置的連線問題，需要消耗較多資源。

SPI 與 I2C 的不同

SPI 沒有 BUS，是一種平行傳輸。

需要至少四條電路，分別為 SCLK、MOSI、MISO、SS。

SPI 的傳輸速度可以比 I2C 更快，且消耗更少資源。

只能有一個主。