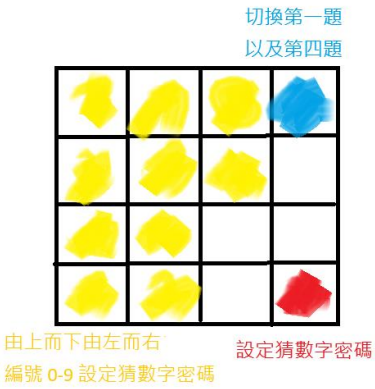
功能說明

選題：結合第一題猜數字以及第四題解碼摩斯密碼



初始設定密碼：

使用 16key 矩陣鍵盤進行答案 0-9 的設定 其中**數字不可重複**

按下 16key 矩陣鍵盤右下角的第 16 個按鈕為**設定密碼**

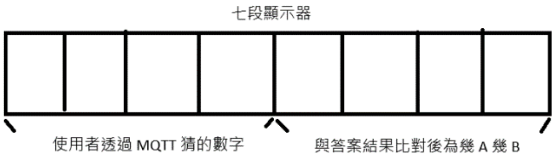
設定的密碼會回傳給 esp32 端顯示

手機端利用 my mqtt app 進行發送使用者猜的數字給 StarStar415/feeds/finalNumber

ESP32 訂閱 StarStar415/feeds/finalNumber 接收我們猜的數字

並透過 UART 將猜的數字傳送給 8051 判斷

七段顯示器顯示：



猜的數字會呈現在上方八個七段顯示器上的 1-4 個

猜的結果會呈現在上方八個七段顯示器上的 5-8 個

判斷結果：

判斷結果（ex: 1A2B）透過 UART 傳送給 ESP32

ESP32 再透過 mqtt 發送使用者猜的數字的結果給 StarStar415/feeds/finalNumberAns

顯示猜題的結果

提示功能：

使用者透過手機發送"*"表示要求提示 會回傳現在已經猜對的數字的位置給使用者，七段顯示器也會顯示正確的位址，錯誤的位址不會呈現數字 以-代替

ex 答案為 1234

使用者猜測 5263 為 1A1B，回傳－2 - - 給使用者呈現 2 為正確的位置

猜對數字：

若使用者猜題成功會回傳 success 字串給 ESP32

並且 8051 端的七段顯示器會呈現一小段動畫

第四題：透過 8051 的 8key 傳送摩斯密碼:

K1 為長音 K2 為短音 K8 表示一個字母完畢 進行解碼

當按下 K8 將剛剛存放的短音長音資料解碼為英文字母

並將英文字母透過 UART 傳送給 ESP32

ESP32 再透過 MQTT 將解碼結果傳送 StarStar415/feeds/finalMorseCode

手機端傳送 Ascii 給 StarStar415/feeds/finalAscii

ESP32 端訂閱 StarStar415/feeds/finalAscii 接收並傳送給 8051 進行喇叭發聲

切換兩題的模式：

使用 16 key 矩陣鍵盤的右上角按鈕切換現在是猜數字還是解碼摩斯密碼

程式碼

ESP32 端的程式碼

```python
from machine import Pin, PWM
from machine import UART
from umqtt.simple import MQTTClient
import utime, xtools

xtools.connect_wifi_led()
# 連結 8051 喇叭
speaker = Pin(15, Pin.OUT)
# 與 8051 UART 連結 , 設定 tx rx 的 GPIO
com = UART(2, 9600, tx=17, rx=16)
com.init(9600)

# 設定自己的 username 和 api key 以及 feed 名稱
ADAFRUIT_IO_USERNAME = "StarStar415"
ADAFRUIT_IO_KEY      = "aio_pipN76hiXflkkpWKBvCazEktssri"
cnt=1

# 保存接收到的 MQTT 消息
received_msgs = {
    "finalNumber": None,
    "finalAsciiCode": None
}
# 訂閱要使用的 FEEDS
def subscribe_to_feeds(client, username):
    feeds = ["finalAscii", "finalNumber"]
    for feed in feeds:
        topic = f"{username}/feeds/{feed}"
        print(topic)
        client.subscribe(topic)

# MQTT 客戶端
client = MQTTClient (
    client_id = xtools.get_id(),
    server = "io.adafruit.com",
    user = ADAFRUIT_IO_USERNAME,
    password = ADAFRUIT_IO_KEY,
    ssl = False,
)

# 解析收到的訊息
def sub_cb(topic, msg):
    topic = topic.decode()
    msg = msg.decode()
    print(msg)
    # 判斷猜數字或是解碼 ASCII
    if "finalNumber" in topic:
        com.write(f"{msg}\r\n") # 傳送訊息給 8051
        received_msgs["finalNumber"] = msg

    elif "finalAscii" in topic:
        msg = msg.upper()
        com.write(f"{msg}\r\n") # 傳送訊息給 8051
        received_msgs["finalAsciiCode"] = msg


client.set_callback(sub_cb)   # 指定回撥函數來接收訊息
client.connect()              # 連線

subscribe_to_feeds(client, ADAFRUIT_IO_USERNAME)

# 接收 8051 回傳的資料
while True:
    client.check_msg()
    if com.any() > 0:
       a = com.readline()
       print(a.decode().strip())
       combined_msg = f"{cnt}: {received_msgs['finalNumber']} -- {a.decode().strip()}"
       # 猜數字成功 顯示成功訊息並且嘗試了幾次，將計數歸零
       if "4A0B" in a.decode().strip():
           if cnt == 1:
                tmp = "try"
           else:
                tmp = "tries"
           combined_msg = f"SUCCESS!! use {cnt} {tmp} Ans:{received_msgs['finalNumber']}  "
           client.publish("StarStar415/feeds/finalNumberAns", combined_msg)
           cnt=1
       # 提示模式 publish 提示格式
       elif "-" in a.decode().strip():
           combined_msg = f"Hint: {a.decode().strip()}"
           client.publish("StarStar415/feeds/finalNumberAns", combined_msg)
       # 摩斯密碼的格式
       elif a.decode().strip() >= 'A' and a.decode().strip() <= 'Z':
           client.publish("StarStar415/feeds/finalMorseCode", a.decode())
       # 其餘皆為正常猜數字模式 並將計數次數增加
       else:
           client.publish("StarStar415/feeds/finalNumberAns", combined_msg)
           cnt += 1

```

程式碼說明：

詳細程式的每個 function 介紹可在副檔程式碼中的註解中得知。

ESP32 端

要連接 8051 的 tx rx 進行資料傳輸，先設定 com = UART(2, 9600, tx=17, rx=16) 進行連結

為了清楚分辨，使用兩個 feeds 區分猜數字以及解碼 ascii，接收手機端的傳送

猜數字: StarStar415/feeds/finalNumber

解碼 ascii : StarStar415/feeds/finalAscii

再透過回傳的訊息判斷訊息格式以及傳送結果為猜數字結果還是摩斯密碼解碼結果，如果是猜數字則格式為（猜的次數):（猜的數字） - （數字的猜測結果 AB)，摩斯密碼則為 Ascii，提示回傳則會有 - 顯示。

8051 端的程式碼

```c
#include<reg52.h>
#include <string.h>
#define MAX 20
#define DataPort P0
#define KeyPort  P1

// 掃描七段顯示器用
sbit LATCH1= P2^2;
sbit LATCH2= P2^3;
sbit SPK   = P2^1; // 喇叭
// 摩斯密碼使用
sbit k1 = P2^4;
sbit k2 = P2^5;
sbit k3 = P2^6;
sbit k8 = P2^7;

// 定義 unsigned char 和 unsigned int
typedef unsigned char byte;
typedef unsigned int  word;

// 環形佇列
byte buf[MAX];
byte head = 0;
byte tail = 0;

// 七段顯示器的位碼以及顯示 0-F 的編碼
byte code dofly_DuanMa[]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f,
                          0x77,0x7c,0x39,0x5e,0x79,0x71};
byte code dofly_WeiMa[]={0xfe,0xfd,0xfb,0xf7,0xef,0xdf,0xbf,0x7f};
byte TempData[10];
word i;

// 摩斯密碼的長音短音表
byte code MorseMap[][5] = {
    "01",    "1000",  "1010",  "100",   "0",     "0010",  "110",   "0000", // A-H
    "00",    "0111",  "101",   "0100",  "11",    "10",    "111",   "0110", // I-P
    "1101",  "010",   "000",   "1",     "001",   "0001",  "011",   "1001", // Q-X
    "1011",  "1100"                                                         // Y-Z
};

// 掃描矩陣鍵盤
byte KeyScan(void);
byte KeyPro(void);
// 設定七段顯示器位碼與編碼
void Display(byte FirstBit,byte Num);


// 初始化 UART
void InitUART  (void){
    SCON  = 0x50;   // SCON: 模式 1, 8-bit UART, 使能接收
    TMOD |= 0x20;   // TMOD: timer 1, mode 2, 8-bit 重裝
    TH1   = 0xFD;   // TH1:  重裝值 9600 串列傳輸速率 晶振 11.0592MHz
    TR1   = 1;      // TR1:  timer 1 打開
    EA    = 1;      //打開總中斷
}

```

```
// Delay 時間 function，DelayMs 約為 1ms
void DelayUs2x(unsigned char t){
    while(--t);
}
void DelayMs(unsigned char t){
    while(t--){
        DelayUs2x(245);
        DelayUs2x(245);
    }
}
// 宣告 Delay function
void Delay(unsigned int t){
    while(--t);
}
// 解碼摩斯密碼用
void decoder();
void decodeAscii(byte len);
byte MorseToChar(byte* morse, byte length);
// 喇叭音效
void speak(word t){
    for(i=0;i<t;i++)
    {
        DelayUs2x(200);
        DelayUs2x(200);
        SPK=!SPK;
        Display(0,8);
    }
}

// 現在的模式為猜數字還是解碼摩斯密碼
bit modeFlags = 0;
// 猜數字的答案是否設定
bit setAnsNum = 0;
bit f = 0;

byte ansNum[4]; // 存放猜數字答案
byte decode[4]; // 存放解碼摩斯密碼長音短音
byte guessNum[4]; // 存放 matt 猜數字
byte ascii[10]; // 存放 matt 解碼摩斯密碼
byte cnt=0;
byte ansCnt=0;
byte decodeCnt=0;

void main (void){
    byte num;
    byte j;
    InitUART();
    ES = 1;                 //打開串口中斷
    EA = 1;
    while (1){
        TR0=0;
        Display(0,8);
        num=KeyPro();
        if(num == 12){
        // 切換模式
            if(modeFlags == 1){
                modeFlags = 0;
                setAnsNum = 0;
                ansCnt=0;
                i=0;
                for(j=0;j<8;j++)TempData[j]=0;
                TempData[7]=dofly_DuanMa[modeFlags];
            }
            else {
                modeFlags = 1;
                decodeCnt = 0;
                i=0;
                for(j=0;j<8;j++)TempData[j]=0;
                TempData[7]=dofly_DuanMa[modeFlags];
            }
            DelayMs(100);
        }

        // 設定猜密碼答案
        if(modeFlags == 0 && setAnsNum == 0){
            num=KeyPro(); // 掃描矩陣鍵盤數值
            if(num == 15 && ansCnt>=4){
                // 設定猜數字密碼
                setAnsNum = 1;
                for(j=0;j<4;j++)TempData[j]=0;
                TempData[4]=dofly_DuanMa[ansNum[0]];
                TempData[5]=dofly_DuanMa[ansNum[1]];
                TempData[6]=dofly_DuanMa[ansNum[2]];
                TempData[7]=dofly_DuanMa[ansNum[3]];
                i=0;
            }
```

```
            if(num>=0 && num<=9){
                bit tmpf = 0;
                if(ansCnt>=4) continue;
                // 判斷數字是否重複 發出錯誤音效
                for(j = 0 ; j < ansCnt ; j++){
                    if(ansNum[j] == num) {
                        tmpf = 1;
                        break;
                    }
                }
                if(tmpf == 1) {
                    speak(100);
                    DelayMs(100);
                    speak(100);
                    continue;
                }
                if(ansCnt == 0)
                for(i = 0 ; i < 8 ; i++) TempData[i] = 0;
                TempData[ansCnt]=dofly_DuanMa[num];
                ansNum[ansCnt++] = num;
            }
        }
        else if(modeFlags == 1){
            // 長音
            if(k1 == 0){
                decode[decodeCnt++] = 1;
                speak(300);
                SPK=0;
                Delay(300); // 解彈跳
                while(k1 == 0); // 等放開按鈕
                Delay(300); // 解彈跳
            }
            // 短音
            if(k2 == 0){
                decode[decodeCnt++] = 0;
                speak(100);
                Delay(300); // 解彈跳
                while(k2 == 0); // 等放開按鈕
                Delay(300); // 解彈跳
            }
            // 進行解碼
            if(k8 == 0){
                decoder();
                Delay(300); // 解彈跳
                while(k8 == 0); // 等放開按鈕
                Delay(300); // 解彈跳
            }
        }
    }
}

// 解碼長音短音 並透過 UART 傳送至 ESP32
void decoder(void) {
    byte morseChar = MorseToChar(decode, decodeCnt);
    decodeCnt = 0;

    if (morseChar != 0xFF) {
        SBUF = morseChar;
        while (TI == 0);
        TI = 0;
        SBUF = '\r';
        while (TI == 0);
        TI = 0;
        SBUF = '\n';
        while (TI == 0);
        TI = 0;
    }
}

// 比對長音短音並回傳對應的字元 沒找到回傳 0xFF
byte MorseToChar(byte* morse, byte length) {
    byte i, j, match;
    for ( i = 0; i < sizeof(MorseMap) / sizeof(MorseMap[0]); i++) {
        if (strlen(MorseMap[i]) == length) {
            match = 1;
            for ( j = 0; j < length; j++) {
                if ((MorseMap[i][j] - '0') != morse[j]) {
                    match = 0;
                    break;
                }
            }
            if (match) return 'A' + i;
        }
```

```
    }
    return 0xFF;
}

// 解碼 MQTT傳送給 ESP32 再透過 UART 傳送過來的 ASCII 為摩斯密碼發音呈現
void decodeAscii(byte len) {
    byte morseCode[5];
    byte length, i, j, k;
    for(i = 0 ; i < len  ; i++){
        for (j = 0; j < sizeof(MorseMap) / sizeof(MorseMap[0]); j++) {
            if ((ascii[i] - 'A') == j) {
                strcpy(morseCode, MorseMap[j]);
                length = strlen(morseCode);
                for ( k = 0; k < length; k++) {
                    if (morseCode[k] == '0') {
                        speak(100);
                    } else {
                        speak(300);
                    }
                    SPK = 0;
                    DelayMs(200);
                }
                DelayMs(200);
                DelayMs(200);
            }
        }
    }
}

// 猜數字遊戲
void getGuessNumAns(void) {
    byte a = 0;
    byte b = 0;
    byte i, j;
    byte ansUsed[4] = {0};
    byte guessUsed[4] = {0};

    // 計數 A 的數量
    for (i = 0; i < 4; i++) {
        if ((guessNum[i]-'0') == ansNum[i]) {
            a++;
            ansUsed[i] = 1;
            guessUsed[i] = 1;
        }
    }

    // 計數 B 的數量
    for (i = 0; i < 4; i++) {
        if (guessUsed[i] == 0) {
            for (j = 0; j < 4; j++) {
                if (ansUsed[j] == 0 && guessNum[i]-'0' == ansNum[j]) {
                    b++;
                    ansUsed[j] = 1;
                    break;
                }
            }
        }
    }

    // 七段顯示器顯示幾 A 幾 B
    TempData[4] = dofly_DuanMa[a];
    TempData[5] = dofly_DuanMa[10];
    TempData[6] = dofly_DuanMa[b];
    TempData[7] = dofly_DuanMa[11];

    // 回傳猜數字結果給 ESP32
    SBUF = a + '0';
    while (TI == 0);
    TI = 0;
    SBUF = 'A';
    while (TI == 0);
    TI = 0;
    SBUF = b + '0';
    while (TI == 0);
    TI = 0;
    SBUF = 'B';
    while (TI == 0);
    TI = 0;
    SBUF = '\r';
    while (TI == 0);
    TI = 0;
    SBUF = '\n';
    while (TI == 0);
    TI = 0;
```

```
311
312     // 猜對 顯示 SUCCESS 並發出音效
313     if (a == 4){
314         setAnsNum = 0;
315         for(i=0;i<8;i++)TempData[i]=0;
316         ansCnt=0;
317         // 顯示 success
318         TempData[0] = 0x6D; // S
319         TempData[1] = 0x3E; // U
320         TempData[2] = 0x39; // C
321         TempData[3] = 0x39; // C
322         TempData[4] = 0x79; // E
323         TempData[5] = 0x6D; // S
324         TempData[6] = 0x6D; // S
325         speak(100);
326         DelayMs(100);
327         speak(100);
328         DelayMs(500);
329         speak(100);
330         DelayMs(100);
331         speak(100);
332     }
333 }
334
335 void getHintNumAns(void) {
336     byte a = 0;
337     byte b = 0;
338     byte i;
339     byte ansUsed[4] = {0};
340     byte guessUsed[4] = {0};
341
342     // 提示功能 顯示 A 的位置哪些是正確的
343     for (i = 0; i < 4; i++) {
344         if ((guessNum[i]-'0') == ansNum[i]) {
345             a++;
346             TempData[i] = dofly_DuanMa[guessNum[i]-'0'];
347             SBUF = guessNum[i];
348             while (TI == 0);
349             TI = 0;
350         }
351         else{
352             TempData[i] = 0x40;
353             SBUF = '-';
354             while (TI == 0);
355             TI = 0;
356         }
357     }
358
359     SBUF = '\r';
360     while (TI == 0);
361     TI = 0;
362     SBUF = '\n';
363     while (TI == 0);
364     TI = 0;
365
366     for(i=4;i<8;i++)TempData[i]=0;
367 }
368
369 // 接收訊號
370 void UART_SER (void) interrupt 4 {
371     byte Temp;
372
373     // 如果 RI==1表示有資料送進來
374     if(RI){
375         RI=0;
376         Temp=SBUF;// 從 SBUF 拿取資料存在 buffer 中
377         buf[head] = Temp;
378         // 猜數字模式
379         if(modeFlags == 0 && setAnsNum == 1){
380             // 接收到的字元為換行字元，將 cnt 設為 0 等待下次接收
381             if (Temp == '\n'){
382                 cnt = 0;
383             }
384             else if(Temp == '\r'){
385                 // 如果都沒有錯誤或是提示 則進行猜數字
386                 if(f == 0) getGuessNumAns();
387                 f = 0;
388                 cnt = 0;
389             }
390             // 提示功能
391             else if(Temp == '*'){
392                 f = 1;
393                 getHintNumAns();
394             }
395             // 將收到的字元儲存並設定到七段顯示器上顯示
396             else if(Temp >= '0' && Temp <= '9') {
397                 guessNum[cnt] = Temp;
```

```
                TempData[cnt] = dofly_DuanMa[guessNum[cnt]-'0'];
                cnt++;
            }
            // 接收到錯誤字元 顯示 error 並發出音效
            else{
                f=1;
                for(i=0;i<8;i++)TempData[i]=0;
                // 顯示 error
                TempData[0]=0x79;
                TempData[1]=0x50;
                TempData[2]=0x50;
                TempData[3]=0x5c;
                TempData[4]=0x50;
                speak(100);
                DelayMs(100);
                speak(100);
                SPK=0;
            }
        }
        // 摩斯密碼模式
        if(modeFlags == 1){
            // 接收到的字元為換行字元，將 cnt 設為 0 等待下次接收
            if (Temp == '\n'){
                cnt = 0;
            }
            else if(Temp == '\r'){
                // 如果都沒有錯誤或是提示 則進行 ascii 解碼
                if(f==0)decodeAscii(cnt);
                f = 0;
                cnt = 0;
            }
            // 將收到的字元儲存並設定到七段顯示器上顯示
            else if((Temp >= 'A' && Temp <= 'Z') || (Temp >= 'a' && Temp <= 'z')) {
                for(i=0;i<7;i++)TempData[i]=0;
                ascii[cnt] = Temp;
                cnt++;
            }
            // 接收到錯誤字元 顯示 error 並發出音效
            else  {
                f=1;
                for(i=0;i<8;i++)TempData[i]=0;
                // 顯示 error
                TempData[0]=0x79;
                TempData[1]=0x50;
                TempData[2]=0x50;
                TempData[3]=0x5c;
                TempData[4]=0x50;
                speak(100);
                DelayMs(100);
                speak(100);
                SPK=0;
            }
        }
        // 環形佇列
        head++;
        if (head == MAX) head = 0;
    }
}

// 顯示七段顯示器
void Display(byte FirstBit,byte Num){
    static byte i=0;

    DataPort=0;
    LATCH1=1;
    LATCH1=0;

    DataPort=dofly_WeiMa[i+FirstBit];
    LATCH2=1;
    LATCH2=0;

    DataPort=TempData[i];
    LATCH1=1;
    LATCH1=0;

    i++;
    if(i==Num)
    i=0;
}

// 掃描矩陣鍵盤
unsigned char KeyScan(void){
    unsigned char Val;
    KeyPort=0xf0;
    if(KeyPort!=0xf0){
        DelayMs(10);
        if(KeyPort!=0xf0){
            KeyPort=0xfe;
```

```
            if(KeyPort!=0xfe){
                Val=KeyPort&0xf0;
                Val+=0x0e;
                while(KeyPort!=0xfe);
                DelayMs(10);
                while(KeyPort!=0xfe);
                return Val;
            }
            KeyPort=0xfd;
            if(KeyPort!=0xfd){
                Val=KeyPort&0xf0;
                Val+=0x0d;
                while(KeyPort!=0xfd);
                DelayMs(10);
                while(KeyPort!=0xfd);
                return Val;
            }
            KeyPort=0xfb;
            if(KeyPort!=0xfb){
                Val=KeyPort&0xf0;
                Val+=0x0b;
                while(KeyPort!=0xfb);
                DelayMs(10);
                while(KeyPort!=0xfb);
                return Val;
            }
            KeyPort=0xf7;
            if(KeyPort!=0xf7){
                Val=KeyPort&0xf0;
                Val+=0x07;
                while(KeyPort!=0xf7);
                DelayMs(10);
                while(KeyPort!=0xf7);
                return Val;
            }
        }
    }
    return 0xff;
}

// 回傳矩陣鍵盤掃描要用到的數值
byte KeyPro(void){
    switch(KeyScan()){
        case 0x7e:return 0;break; // 0
        case 0x7d:return 1;break; // 1
        case 0x7b:return 2;break; // 2
        case 0x77:return 3;break; // 3
        case 0xbe:return 4;break; // 4
        case 0xbd:return 5;break; // 5
        case 0xbb:return 6;break; // 6
        case 0xb7:return 7;break; // 7
        case 0xde:return 8;break; // 8
        case 0xdd:return 9;break; // 9
        case 0xdb:return 10;break; // A
        case 0xd7:return 11;break; // B
        case 0xee:return 12;break; // C
        case 0xed:return 13;break; // D
        case 0xeb:return 14;break; // E
        case 0xe7:return 15;break; // F
        default:return 0xff;break; // Invalid key
    }
}
```

程式碼說明：

詳細程式的每個 function 介紹可在副檔程式碼中的註解中得知。

8051 端

程式主要控制 UART 接收資料，並將其顯示在七段顯示器上，進行猜數字以及解碼摩斯密碼的功能，當接收到錯誤字元會控制 8051 喇叭發出音效並在七段顯示器中顯示 error，猜數字結果呈現在 8051 的七段顯示器上，並且透過 UART 回傳至 ESP32 處理，再透過 MQTT 將結果回傳至手機查看，摩斯密碼解碼也是透過 UART 回傳至 ESP32 處理，再透過 MQTT 將結果回傳至手機，如果設定數字設定重複的數字也會發出錯誤音效。

程式碼附在副檔
ESP32 端: final_esp32.py
8051 端 : final_8051.c

程式運作模式大致上為以下解釋:

mode1: 猜數字

步驟：
1. 透過 16 key 矩陣鍵盤設定密碼
2. my mqtt app 發送猜的數字
3. esp32 將訂閱接收到的數字傳送給 8051
4. 8051 進行猜數字的判斷並且顯示在七段顯示器 並回傳結果給 eps32
5. esp32 傳送結果給指定的 feesd 並且手機端訂閱該 feeds 接收猜題結果

mode2: 解碼摩斯密碼

(8051 透過 ESP32 並將解碼結果傳到手機)
1. 透過 8051 的 8key 傳送摩斯密碼
2. K1 為長音 K2 為短音 K8 表示一個字母完畢。speaker 同時發出長音短音表示現在按下的按鈕為長音還是短音
3. 8051 進行解碼
4. 將解碼資料傳送給 ESP32
5. ESP32 利用 MQTT 發送給手機端

(手機傳到 ESP32)
1. 手機端傳送 Ascii 給 StarStar415/feeds/finalAscii
2. ESP32 訂閱 StarStar415/feeds/finalAscii 接收訊息
3. 將解碼資料傳送給 8051
4. 8051 透過解碼資料進行 speaker 的播放

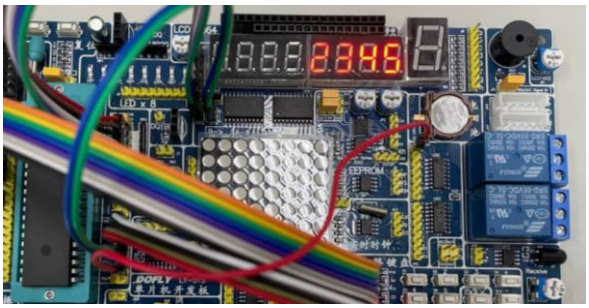
結果

詳細 demo 結果如上課錄製的 demo 影片：

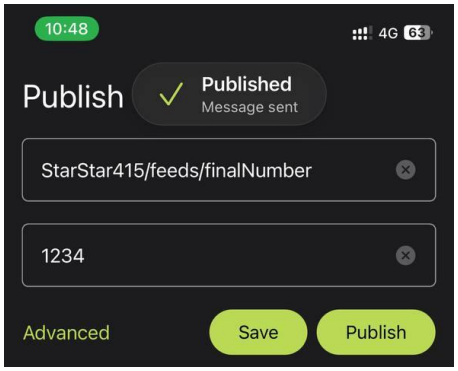
影片有口頭解釋 demo 的功能，以下說明主要控制與達成結果。

猜數字

1. 透過 16 key 矩陣鍵盤設定密碼: 2345



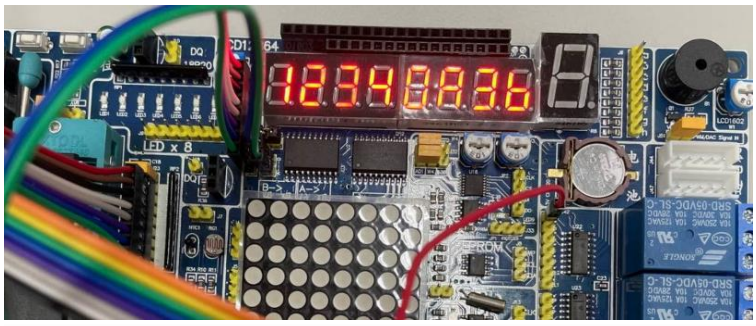
2. my mqtt app 發送猜的數字 1234



3. esp32 將訂閱接收到的數字傳送給 8051: 接收到 1234

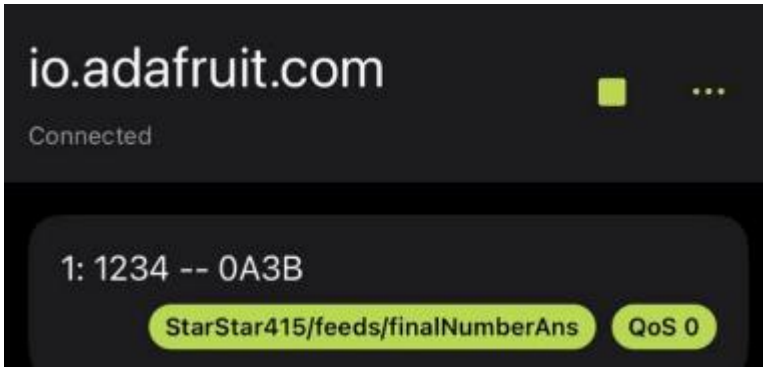
```
MPY: soft reboot
network config: ('172.20.10.3', '255.255.255.240', '172.20.10.1', '172.20.10.1')
StarStar415/feeds/finalAscii
StarStar415/feeds/finalNumber
1234
```

4. 8051 進行猜數字的判斷並且顯示在七段顯示器 並回傳結果給 eps32 : 1234 0A3B



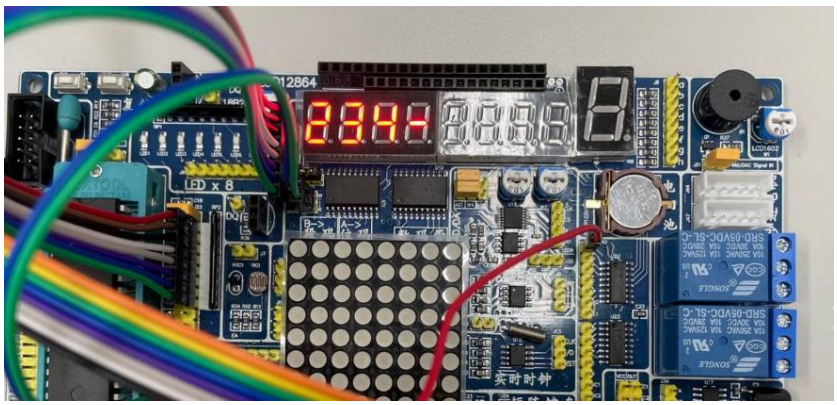
```
MPY: soft reboot
network config: ('172.20.10.3', '255.255.255.240', '172.20.10.1', '172.20.10.1')
StarStar415/feeds/finalAscii
StarStar415/feeds/finalNumber
1234
0A3B
```

5. esp32 傳送結果給指定的 feesd 並且手機端訂閱該 feeds 接收猜題結果

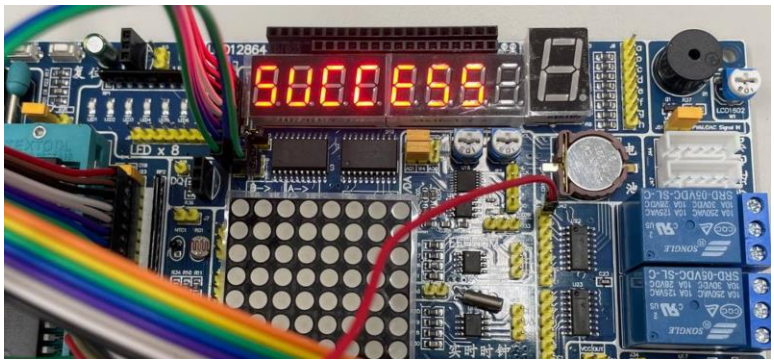


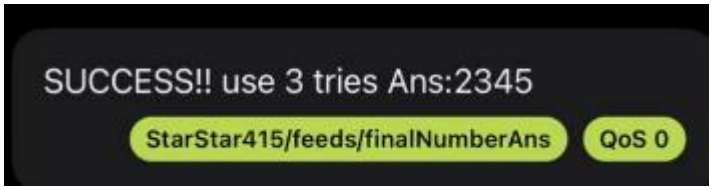
提示功能：傳送 *

因為前一次猜測為 2346 回傳 3A0B 想要知道 A 的位置 使用提示功能呈現 234-

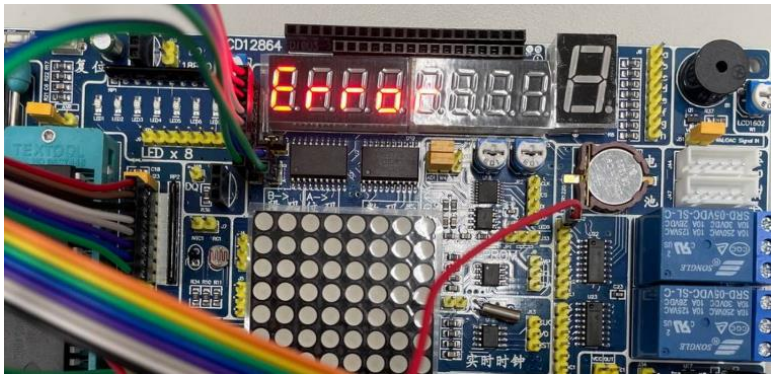


成功畫面: 呈現 SUCCESS 並且手機端也會顯示成功與猜了多少次





錯誤輸入顯示 error



摩斯密碼解碼