多功能防盜保險箱

組員: 沈資凱、鄭丞邑、陳泓銘、莊子箴、沈益安| 指導老師: 黃楚喬、陳忠謀| 第三組專題報告

保險箱，能夠存放個人的重要物品，並使用各種加密保護內容物不被盜竊或外洩個資，讓使有者能夠放心的存放自己的重要物品。

我們希望能設計一款有別於傳統的加密保險箱，除了傳統輸入密碼來解鎖，也添加更多的解鎖方式，以及更多功能來提升使用者的體驗，並更進一步的保護內部重要財物。

專題構想

功能講解

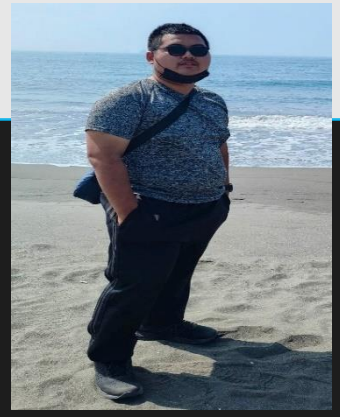
- 解鎖方式: 密碼解鎖，指紋解鎖，磁卡感應解鎖。
- 檢測到箱門開啟，密碼輸入或指紋感應嘗試，傳送通知簡訊到手機提醒。
- 使用SG90伺服馬達鎖定保險箱門。
- OLED顯示器上方顯示鎖定或解除鎖定，下方顯示輸入密碼。
- 保險箱門開啟後，自動點亮照明燈。
- 偵測到輸入密碼錯誤或指紋不符時，透過軟體 LINE 傳送簡訊到手機，以及蜂鳴器作響警示。
- 使用外接供電。

預計進度表

- 3/17 材料到齊，開始分配工作
- 3/24 構圖&接線
- 3/31 程式撰寫
- 4/7 程式撰寫
- 4/14 程式除錯
- 4/21成品檢測
- 4/28 成品展示

工作分配

沈資凱	鄭丞邑	陳泓銘	莊子箴	沈益安
• 組裝 • 拍照攝影 • 採購	• 介紹 • 組裝	• 程式 • 文書	• 文書 • 拍照攝影	• 程式 • 文書





材料表

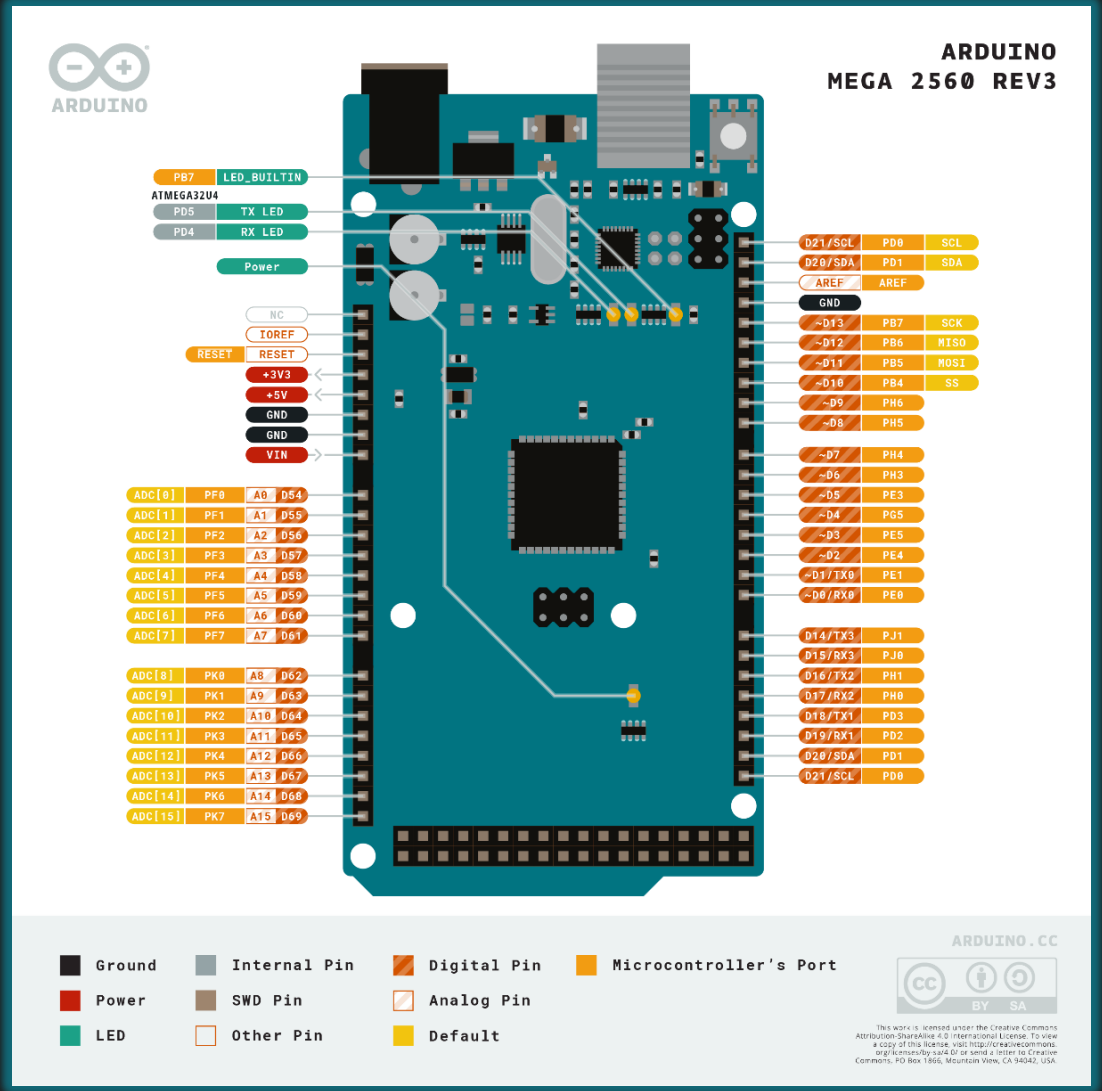
材料 (詳細清單)	價格
5mm 超高亮度LED 黃光	$5
有源蜂鳴器 5v 電磁式蜂鳴器	$5
壓克力鞋盒	$199
12V 3A 變壓器 雙線	$145
MFRC-522 RC522 RFID IC卡感應	$150
二路 2路繼電器模組	$45
NodeMCU V1 相容版 ESP8266 開發板	$175

材料表

合計金額：2521元

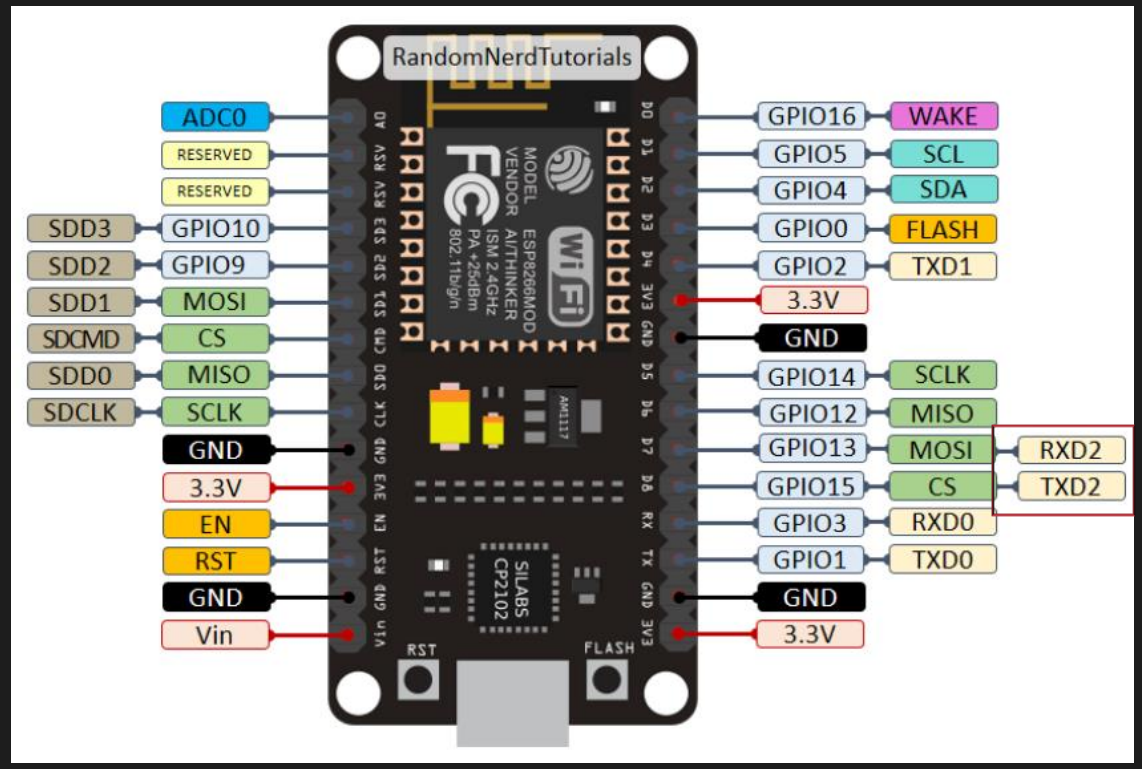
材料	價格
AS608光學指紋感測器	$495
0.96吋 OLED 128x64 低耗電	$185
4×4矩陣薄膜鍵盤	$25
10cm 杜邦線 母對母 10條	$8
Arduino Mega2560 R3 新版	$795
DC5V DS-0420S 電磁閥	$145
MG90S 14g 伺服馬達 90°	$84
麵包板400孔 (長8.5*寬5.5CM)	$60

接線圖(mega2560)



元件	元件接腳	Mega2560接腳
OLED	GND	GND
OLED	VCC	3.3V
OLED	SCL	SCL 21
OLED	SDA	SDA 20
RFID	3.3V	3.3V
RFID	RST	D5
RFID	GND	GND
RFID	SDA(SS)	D53
RFID	MOSI	D51
RFID	MISO	D50
RFID	SCK	D52
AS608	黃色TX	RX3(15)
AS609	白色RX	TX3(14)
AS610	紅色VCC	3.3V
AS611	黑色GND	GND
BZ	電源正	A15
BZ	電源負	GND
4x4keyboard	R1	D35
4x4keyboard	R2	D37
4x4keyboard	R3	D39
4x4keyboard	R4	D41
4x4keyboard	C1	D43
4x4keyboard	C2	D45
4x4keyboard	C3	D47
4x4keyboard	C4	D49
SG90	正電(紅)	5V
SG90	負電(棕)	GND
SG90	訊號(黃)	D4
NodeMCU	RX	TX2(16)
NodeMCU	TX	RX2(17)
NodeMCU	GND	GND

接線圖(NodeMCU)



元件	元件接腳	Mega2560接腳
Mega2560	TX2(16)	RX
Mega2560	RX2(17)	TX
Mega2560	GND	GND

程式碼

```
#include <SPI.h>                  //引用RFID/OLED程式庫
#include <Wire.h>                 //引用OLED程式庫
#include <Adafruit_GFX.h>         //引用OLED程式庫
#include <Adafruit_SSD1306.h>     //引用OLED程式庫
#include <MFRC522.h>              //引用RFID程式庫
#include <Adafruit_Fingerprint.h> //引用AS608程式庫
#include <Adafruit_Keypad.h>      //引用Keypad程式庫
#include <Servo.h>                //引用SG90程式庫

#define RST_PIN      5          // 讀卡機的重置腳位
#define SS_PIN       53         // 晶片選擇腳位
#define SCREEN_WIDTH 128        // OLED 寬度像素
#define SCREEN_HEIGHT 64        // OLED 高度像素
#define ROWS 4                  //鍵盤列數(橫的)
#define COLS 4                  //鍵盤行數(直的)
#define OLED_RESET     -1 // Reset pin # (or -1 if sharing Arduino reset pin)
#define BZ_pin A15

int certified_rfid_key[][4]
{
  {67, 30, 154, 170}, {1, 1, 1, 1}
};
String certified_password = "00"; //6位數密碼
String password_entered = "";
int timer0, timer1, timer2;  //0:認證，1:OLED update，2:BZ
bool door_lock_status = false;      //Fasle == 上鎖中   ，True == 已解鎖
bool oled_in_home = true;
byte bz_mode = 0;
int certified_rfid_key[][4]
{
  {67, 30, 154, 170}, {1, 1, 1, 1}
};
String certified_password = "00"; //6位數密碼
String password_entered = "";
int timer0, timer1, timer2;  //0:認證，1:OLED update，2:BZ
bool door_lock_status = false;      //Fasle == 上鎖中   ，True == 已解鎖
bool oled_in_home = true;
byte bz_mode = 0;

char keys[ROWS][COLS] =
{
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};
byte rowPins[ROWS] = {35, 37, 39, 41}; //定義列的腳位
byte colPins[COLS] = {43, 45, 47, 49}; //定義行的腳位
byte InSideLedPins[5] = {22, 23, 24, 25, 26}; //定義行的腳位

// 設定OLED
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

MFRC522 mfrc522(SS_PIN, RST_PIN);  // 建立MFRC522物件

Servo myservo;  // 建立一個舵機對象

Adafruit_Fingerprint finger = Adafruit_Fingerprint(&Serial3);
Adafruit_Keypad customKeypad = Adafruit_Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS);
```

```
void setup() {
  Serial.begin(9600);
  Serial2.begin(9600);
  SPI.begin();
  finger.begin(57600);
  customKeypad.begin();

  TCCR1A = 0;//將整個TCCR1A暫存器設定為0
  TCCR1B = 0;//將整個TCCR1B暫存器設定為0
  TCNT1  = 0;//將計數器值初始化為0
  //設定計數器為1kHZ，即1ms
  OCR1A = 1999;// = (16*10^6)/(1000*8) - 1 (must be <65536)
  TCCR1B |= (1 << WGM12);//開啟CTC模式
  TCCR1B |= (1 << CS11);//設定CS11位為1(8倍預分頻)
  TIMSK1 |= (1 << OCIE1A);

  mfrc522.PCD_Init();   // 初始化MFRC522讀卡機模組

  myservo.attach(4); // 將舵機連接到引腳 9
  for (int i = 0; i < 5; i++)
  {
    pinMode(InSideLedPins[i], OUTPUT);
  }

  pinMode(BZ_pin, OUTPUT);

  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);

  if (finger.verifyPassword())
  {
    Serial.println("Found fingerprint sensor!");
  }
  else
  {
    Serial.println("Did not find fingerprint sensor :(");
    while (1)
    {
      delay(1);
    }
  }
  finger.getTemplateCount();
  if (finger.templateCount == 0)
  {
    Serial.print("Sensor doesn't contain any fingerprint data. Please run the 'enroll' example.");
  }
  else
  {
    //Serial.println("Waiting for valid finger...");
    //Serial.print("Sensor contains "); Serial.print(finger.templateCount); Serial.println(" templates");
  }
  oled_door_lock();
  Serial2.print("!4");
}
```

```
ISR(TIMER1_COMPA_vect) { // timer1中斷2Hz切換引腳13(LED)
  //產生頻率為2Hz / 2 = 1Hz的脈衝波

  if (timer0 >= 150) //每秒執行一次
  {
    timer0 = 0;

  }
  else
  {
    timer0++;
  }
  if (oled_in_home == false)
  {
    if (timer1 >= 3000)
    {
      timer1 = 0;
    }
    else
    {
      timer1++;
    }

  }
  else
  {
    timer1 = 1;
  }
  switch (bz_mode)
  {
    case 0://BZ OFF
      {
        digitalWrite(BZ_pin, LOW);
        break;
      }
    case 1://BZ unlock
      {
        if (timer2 <= 200)
        {
          timer2++;
          digitalWrite(BZ_pin, HIGH);
        }
        else
        {
          timer2 = 0;
          digitalWrite(BZ_pin, LOW);
          bz_mode = 0;
        }
        break;
      }
    case 2://BZ lock
      {
        if (timer2 <= 1000)
        {
          timer2++;
          digitalWrite(BZ_pin, HIGH);
        }
        else
        {
          timer2 = 0;
          digitalWrite(BZ_pin, LOW);
          bz_mode = 0;
        }
        break;
      }
  }
}
```

```
void loop()
{
  if (timer0 == 0)
  {
    safety_verification();
    delay(10);
  }

  //----------------------------------------
  if (timer1 == 0)
  {
    if (door_lock_status  == true)
    {
      oled_door_unlock();
      Serial2.print("!1");
    }
    else if (door_lock_status  == false)
    {
      oled_door_lock();
      Serial2.print("!2");
    }
  }
}

  display.setTextSize(2);          // 設定文字大小
  display.setTextColor(1);         // 1:OLED預設的顏色(這個會依該OLED的顏色來決定)
  display.setCursor(5, 0);         // 設定起始座標
  display.print("Door state");     // 要顯示的字串
  display.setCursor(26, 40);       // 設定起始座標
  display.print("Unlock");        // 要顯示的字串
  display.display();               // 要有這行才會把文字顯示出來
  myservo.write(0);
  inside_led_ctrl(true);
  oled_in_home = true;
}
void oled_door_lock(void)
{
  display.clearDisplay();
  display.setTextSize(2);          // 設定文字大小
  display.setTextColor(1);         // 1:OLED預設的顏色(這個會依該OLED的顏色來決定)
  display.setCursor(5, 0);         // 設定起始座標
  display.print("Door state");     // 要顯示的字串
  display.setCursor(35, 40);       // 設定起始座標
  display.print("lock");          // 要顯示的字串
  display.display();               // 要有這行才會把文字顯示出來
  myservo.write(90);
  inside_led_ctrl(false);
  oled_in_home = true;
}
```

```
void oled_door_fail(void)
{
  display.clearDisplay();
  display.setTextSize(2);          // 設定文字大小
  display.setTextColor(1);         // 1:OLED預設的顏色(這個會依該OLED的顏色來決定)
  display.setCursor(5, 0);         // 設定起始座標
  display.print("Door state");     // 要顯示的字串
  display.setCursor(40, 40);       // 設定起始座標
  display.print("Fail");          // 要顯示的字串
  display.display();               // 要有這行才會把文字顯示出來
  oled_in_home = false;
}
void oled_door_nowpassword(void)
{
  display.clearDisplay();
  display.setTextSize(2);          // 設定文字大小
  display.setTextColor(1);         // 1:OLED預設的顏色(這個會依該OLED的顏色來決定)
  display.setCursor(5, 0);         // 設定起始座標
  display.print("Now passwd");     // 要顯示的字串
  display.setCursor(0, 40);        // 設定起始座標
  display.print(password_entered);         // 要顯示的字串
  display.display();               // 要有這行才會把文字顯示出來
  oled_in_home = true;
}
//檢查卡片與金鑰是否相等(相等return 1，不相等則return 0)
byte cheak_RFID_Card()
{
  byte *id = mfrc522.uid.uidByte;   // 取得卡片的UID
  byte idSize = mfrc522.uid.size;   // 取得UID的長度
  byte IsSame = 0;                  //0:未驗證 1:驗證成功 2:驗證失敗
  for (int j = 0; j < (sizeof(certified_rfid_key) / sizeof(certified_rfid_key[0])); j++)
  {
    for (int i = 0; i < idSize; i++)
    {
      if (id[i] == certified_rfid_key[j][i])
      {
        IsSame = 1;
        if (i == 3)
        {
          return IsSame;
        }
      }
      else
      {
        IsSame = 2;
        break;
      }
    }
  }
  return IsSame;
}
```

```
//檢查卡片與金鑰是否相等(相等return 1，不相等則return 0)
byte cheak_RFID_Card()
{
  byte *id = mfrc522.uid.uidByte;   // 取得卡片的UID
  byte idSize = mfrc522.uid.size;   // 取得UID的長度
  byte IsSame = 0;                  //0:未驗證 1:驗證成功 2:驗證失敗
  for (int j = 0; j < (sizeof(certified_rfid_key) / sizeof(certified_rfid_key[0])); j++)
  {
    for (int i = 0; i < idSize; i++)
    {
      if (id[i] == certified_rfid_key[j][i])
      {
        IsSame = 1;
        if (i == 3)
        {
          return IsSame;
        }
      }
      else
      {
        IsSame = 2;
        break;
      }
    }
  }
  return IsSame;
}
```

指紋程式碼

```
//指紋辨識驗證身分
uint8_t getFingerprintID() {
  uint8_t p = finger.getImage();//取得指紋
  switch (p) {
    case FINGERPRINT_OK:
      //Serial.println("Image taken");
      break;
    case FINGERPRINT_NOFINGER://沒指紋
      //Serial.println("No finger detected");
      return 0;
    case FINGERPRINT_PACKETRECIEVEERR://沒有收到指紋封包
      //Serial.println("Communication error");
      return 0;
    case FINGERPRINT_IMAGEFAIL://影像失敗
      //Serial.println("Imaging error");
      return 0;
    default:
      //Serial.println("Unknown error");
      return 0;
  }
  // OK success!
  p = finger.image2Tz();//把指紋轉成AS608能讀到的格式
  switch (p) {
    case FINGERPRINT_OK:
      //Serial.println("Image converted");
      break;
    case FINGERPRINT_IMAGEMESS:
      //Serial.println("Image too messy");
      return 0;
    case FINGERPRINT_PACKETRECIEVEERR:
      //Serial.println("Communication error");
      return 0;
    case FINGERPRINT_FEATUREFAIL:
      //Serial.println("Could not find fingerprint features");
      return 0;
    case FINGERPRINT_INVALIDIMAGE:
      //Serial.println("Could not find fingerprint features");
      return 0;
    default:
      //Serial.println("Unknown error");
      return 0;
  }
  // OK converted!
  p = finger.fingerSearch();//把取得指紋跟資料庫比對
  if (p == FINGERPRINT_OK) {
    //Serial.println("Found a print match!");
  } else if (p == FINGERPRINT_PACKETRECIEVEERR) {
    //Serial.println("Communication error");
    return 0;
  } else if (p == FINGERPRINT_NOTFOUND) {//指紋匹配錯誤:2
    //Serial.println("Did not find a match");
    return 2;
  } else {
    //Serial.println("Unknown error");
    return 0;
  }

  // found a match!
  //Serial.print("Found ID #"); Serial.print(finger.fingerID);
  //Serial.print(" with confidence of "); Serial.println(finger.confidence);
  if (finger.confidence >= 100)
  {
    //return finger.fingerID;
    return 1; //指紋匹配成功:1
  }
  else
  {
    return 2; //指紋匹配錯誤:2
  }
}
```

鍵盤程式碼

```
byte keypad_enter() //讀按鍵內容
{
  customKeypad.tick();

  while (customKeypad.available())
  {
    keypadEvent e = customKeypad.read();
    if (e.bit.EVENT == KEY_JUST_PRESSED)
    {
      switch ((char)e.bit.KEY)
      {
        case 'A':
          {
            break;
          }
        case 'B':
          {
            break;
          }
        case 'C':
          {
            break;
          }
        case 'D':
          {
            break;
          }
        case '*':
          {
            password_entered = "";
            Serial.println("已清除密碼");
            oled_door_nowpassword();
            oled_in_home = false;
            break;
          }
        case '#':
          {
            Serial.println("確認");
            keypad_check();
            break;
          }
        default:
          {
            password_entered = password_entered + ((char)e.bit.KEY);
            Serial.println(password_entered);
            oled_door_nowpassword();
            break;
          }
      }
    }
  }
  return 0;
}
```

```
void keypad_check() //檢查鍵盤
{
  if (door_lock_status == true)
  {
    door_lock_status = false;
    oled_door_lock();
    Serial2.print("!2");
    bz_mode = 1;
  }
  else
  {
    if (password_entered == certified_password)//判斷開門密碼是否正確
    {
      door_lock_status = !door_lock_status ;
      if (door_lock_status  == true)
      {
        Serial.println("門鎖:解鎖");
        oled_door_unlock();
        bz_mode = 1;
        Serial2.print("!1");
      }
      else if (door_lock_status  == false)
      {
        Serial.println("門鎖:上鎖");
        oled_door_lock();
        bz_mode = 1;
        Serial2.print("!2");
      }
      password_entered = ""; //清空密碼變數
    }
    else
    {
      password_entered = ""; //密碼錯誤
      oled_door_fail();
      bz_mode = 2;
      Serial2.print("!3");
    }
  }

}
```

(驗證程式碼)

```
void safety_verification() //安全驗證
{
  uint8_t p = finger.getImage();//抓指紋
  customKeypad.tick();//開始偵測使用者的按鍵狀態
  if (mfrc522.PICC_IsNewCardPresent() && mfrc522.PICC_ReadCardSerial())//讀卡
  {
    if (cheak_RFID_Card() == 1)//驗證成功
    {
      door_lock_status = !door_lock_status;//門鎖狀態改變
      if (door_lock_status)//開門
      {
        oled_door_unlock();
        bz_mode = 1;
        Serial2.print("!1");//wifi顯示開門
      }
      else if (door_lock_status  == false)//關門
      {
        oled_door_lock();
        bz_mode = 1;
        Serial2.print("!2");//wifi顯示關門
      }
    }
    else if (cheak_RFID_Card() == 2)
    {
      oled_door_fail();
      bz_mode = 2;
      Serial2.print("!3");//嘗試(錯誤)
    }
    mfrc522.PICC_HaltA();  // 讓卡片進入停止模式
  }
  else if ( p == FINGERPRINT_OK)
  {
    if (getFingerprintID() == 1) //指紋匹配成功
    {
      door_lock_status = !door_lock_status;
      if (door_lock_status) //開門
      {
        oled_door_unlock();
        Serial2.print("!1");
        bz_mode = 1;
      }
      else if (door_lock_status  == false) //關門
      {
        oled_door_lock();
        Serial2.print("!2");
        bz_mode = 1;
      }
    }
    else if (getFingerprintID() == 2)//指紋匹配錯誤
    {
      oled_door_fail();
      Serial2.print("!3");//嘗試(錯誤)
      bz_mode = 2;
    }
  }
  else if (customKeypad.available()) //判斷有否按鍵
  {
    keypad_enter(); //進入鍵盤模式
  }
}
```

(LED程式碼)

```
void inside_led_ctrl(bool light)
{
  if (light == true)
  {
    for (int i = 0; i < 5; i++)
    {
      digitalWrite(InSideLedPins[i], HIGH); //5顆LED全亮
    }
  }
  else if (light == false)
  {
    for (int i = 0; i < 5; i++)
    {
      digitalWrite(InSideLedPins[i], LOW); //5顆LED全滅
    }
  }

}
```

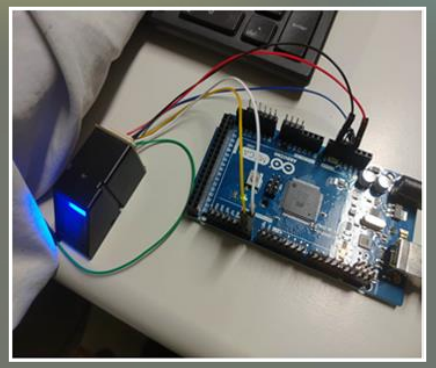
NodeMCU程式碼(1)

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
const char* ssid = "wifi01"; //連網路的名稱
const char* password = "a1234567"; //連網路的密碼
const int httpPort = 80; //網頁端口
const String DomainName = "maker.ifttt.com"; //域名
String key = "cxuJ_FvzSa7BcmJPQXRNuP"; //身分
String input = "";
char a;

void sendmsg(String value1 = "", String value2 = "", String value3 = "")
{
  WiFiClient client;

  client.connect(DomainName, httpPort);

  client.print(String("GET ") + "/trigger/safe/with/key/" + key + "?" + "value1=" + value1 + "&&value2=" + value2 + "&&value3=" + value3 + " HTTP/1.1\r\n" + "Host: maker.ifttt.com\r\n" + "Connection: close\r\n\r\n");

  while (client.available())
  {
    String line = client.readStringUntil('\r');
    Serial.println(line);
  }
  delay(5000);
}
```

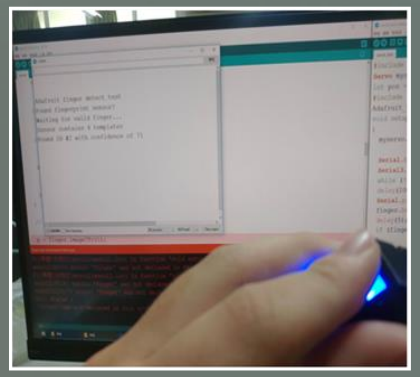
NodeMCU程式碼(2)

```
void loop()
{
  if (Serial.available())
  {
    input = "";
    //input = Serial.readString();
    //Serial.print("接收到的字符串为: ");
    //Serial.println(input);
    while (Serial.available() > 0)
    {
      a = Serial.read(); // 逐个字符读取
      input += a;
      delay(10);
    }
    Serial.print(input);
  }
  if (input == "!1") //開門
  {
    sendmsg("%e4%bf%9d%e9%9a%aa%e7%ae%b1%e8%a8%8a%e6%81%af", "%e9%96%80%e5%b7%b2%e8%a2%ab%e9%96%8b%e5%95%9f", "%e5%8d%b1%e9%9a%aa%e7%a8%8b%e5%ba%a6%3a20");
    input = "";
  }
  else if (input == "!2") //上鎖
  {
    sendmsg("%e4%bf%9d%e9%9a%aa%e7%ae%b1%e8%a8%8a%e6%81%af", "%e9%96%80%e5%b7%b2%e4%b8%8a%e9%8e%96", "%e5%8d%b1%e9%9a%aa%e7%a8%8b%e5%ba%a6%3a0");
    input = "";
  }
  else if (input == "!3") //嘗試(錯誤)
  {
    sendmsg("%e4%bf%9d%e9%9a%aa%e7%ae%b1%e8%a8%8a%e6%81%af", "%e6%9c%89%e4%ba%ba%e6%ad%a3%e5%9c%a8%e5%98%97%e8%a9%a6%e9%96%8b%e9%96%80", "%e5%8d%b1%e9%9a%aa%e7%a8%8b%e5%ba%a6%3a100");
    input = "";
  }
  else if (input == "!4") //開機
  {
    sendmsg("%e4%bf%9d%e9%9a%aa%e7%ae%b1%e8%a8%8a%e6%81%af", "Atmega2560%e5%b7%b2%e9%96%8b%e6%a9%9f", "%e5%8d%b1%e9%9a%aa%e7%a8%8b%e5%ba%a6%3a0");
    input = "";
  }
}
```

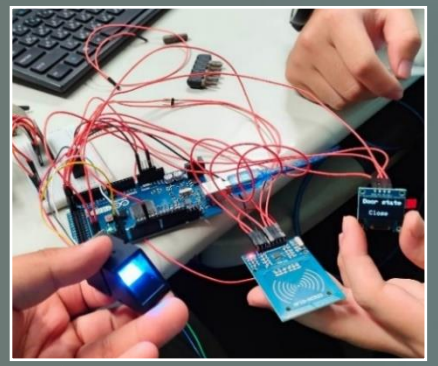
成品展示照



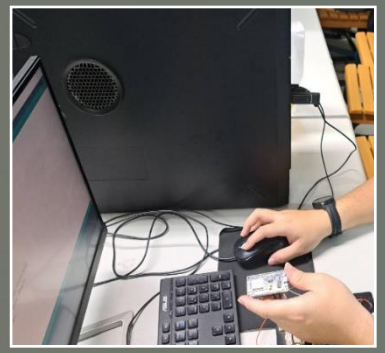
AS608接線



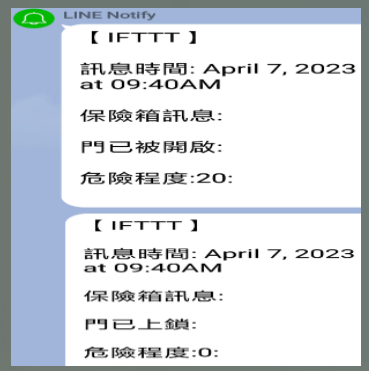
指紋輸入



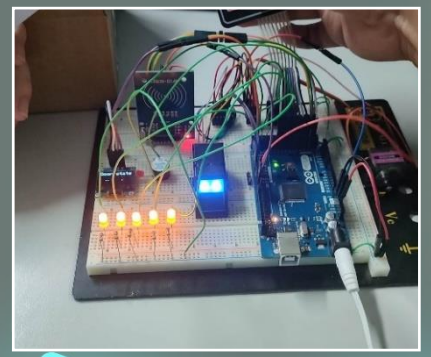
合併OLED和磁扣功能



ESP32

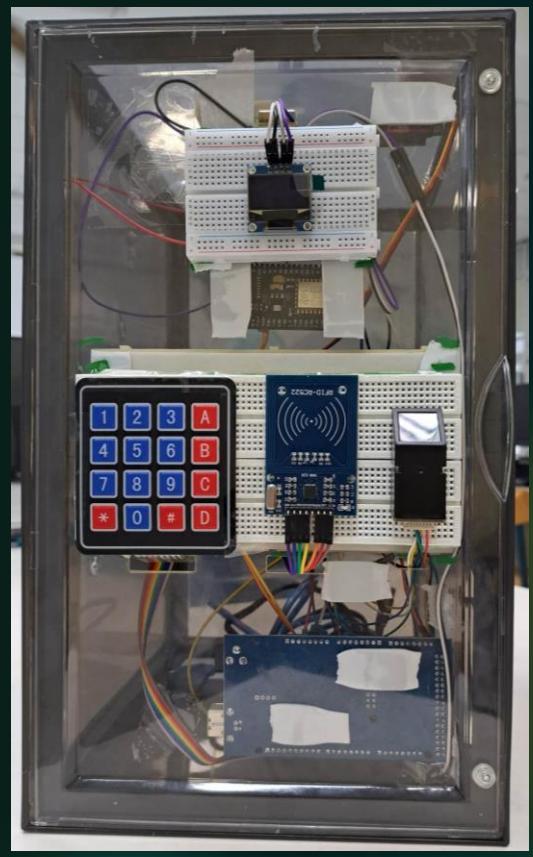


訊息通知



合併測試

成品



組装

製作問題和檢討

接線錯誤	沒有照著元件所要求的接線圖進行接線，造成元件無法正常跑動，解決方法是過網路上的元件資訊進行接圖，並反覆確認接線是否正確。
感測元件的嵌入	要把感測元件貼合保險箱就要將保險箱鑽孔，並把排線拉入保險箱內部，可是在鑽孔的時候並沒有適合的工具和經驗將壓克力板切出剛好的切割面，所幸也借到了雷切機和人員幫助便能將壓克力板切出配給線路的孔洞。
程式	這次專題設計的功能較多所以需要很多智能腳位因此使用Mega2560 ，使用板子上的序列阜(RX,TX)， 因為UNO板上只配置1組，所以使用了Serial port的程式庫，再來就是合併程式，因為要先把每個元件個別寫出再測試，所以並沒有構想好合併時的改寫導致花費了不少時間，也詢問老師同學等做法。

心得感想

- 團隊合作提升效率：

專題的製作是由五個組員為一組完成，每個人都會分配到自己的項目。大部分的感測元件都是有在學校學到的東西，所以能夠快速上手，而一些沒有學過的元件則是透過網路上的資料和老師、同學討論，在一切準備完成之後，就開始動工了。

- 硬體需精準切割提升正確率：

程式和作品的外觀也隨著時間的經過，慢慢的調整成型，不過因為在硬體部分的技術有些不成熟，因為沒有使用合適的工具而沒能乾淨的做出切割面將感測元件崁入

- 線材的操作需再細部確認：

線材的部分過於雜亂，導致不美觀的同時還造成除錯和整線的難度提升，而在測試時也有發生因為接線方式錯誤導致的元件功能錯誤，雖然在之後有將這些接線造成的錯誤排除，但這也讓我們更清楚這些都是往後必須加強的地方。

引用資料

- as608指紋辨識
  https://blog.csdn.net/qq_44629109/article/details/108582138
- IFTTT
  https://ithelp.ithome.com.tw/articles/10272593
- Mega2560
  https://michiel.vanderwulp.be/domotica/Modules/ArduinoMega/
- RFID
  https://lastminuteengineers.com/how-rfid-works-rc522-arduino-tutorial/
- 4x4 keypad
  https://blog.jmaker.com.tw/arduino-keypad-4x4/
- OLED
  https://shop.mirotek.com.tw/arduino/arduino-ssd1306/