112-2學期 弘光科技大學 醫療器材發展與應用系

數位電路與單晶片設計
期末報告

HUB5168+血氧監測裝置

醫材三乙 U110I207 陳俊安

研究動機與目的

動機:

健康意識提升:隨著現代人對健康管理的關注增加，監測個人生命徵象數據成為一個日益重要的趨勢。

疾病管理需求:對於患有慢性心肺疾病的人，如嚴重慢性肺阻塞(COPD)、肺部纖維化、睡眠呼吸中止症、呼吸器依賴、心臟衰竭等，血氧監測顯得略為重要。

血氧機的局限性:價格較低的醫療級血氧機，通常缺乏數據記錄功能，無法滿足長期健康數據追蹤的需求。

目的：

開發上傳數據的閘道器:開發一個簡易操作且具備數據上傳功能的閘道器，使得數據記錄變得方便許多。

建立可查詢的資訊平台:創建一個用戶可以輕易查詢自己的血氧數據的平台，促進健康管理的便利性。

填補市場缺口:解決醫療級的血氧機無記錄功能的缺口，讓價格較低的裝置也能提供長期追蹤功能。

促進慢性心肺疾病管理:使慢性心肺疾病患者能夠隨時掌握自己的健康狀態。

血氧量測上傳架構圖

當閘道器裡的HUB5168+進行通電時將透過藍芽功能與血氧機進行溝通並將數據上傳至MQTT伺服器，再透過資料傳輸程式將我們資料傳輸到資料庫裡

血氧機

Hub5168+

閘道器

MQTT Broker

MQTT

DataBase

資料傳輸程式

Line機器人通知架構圖

MQTT Broker

MQTT伺服器

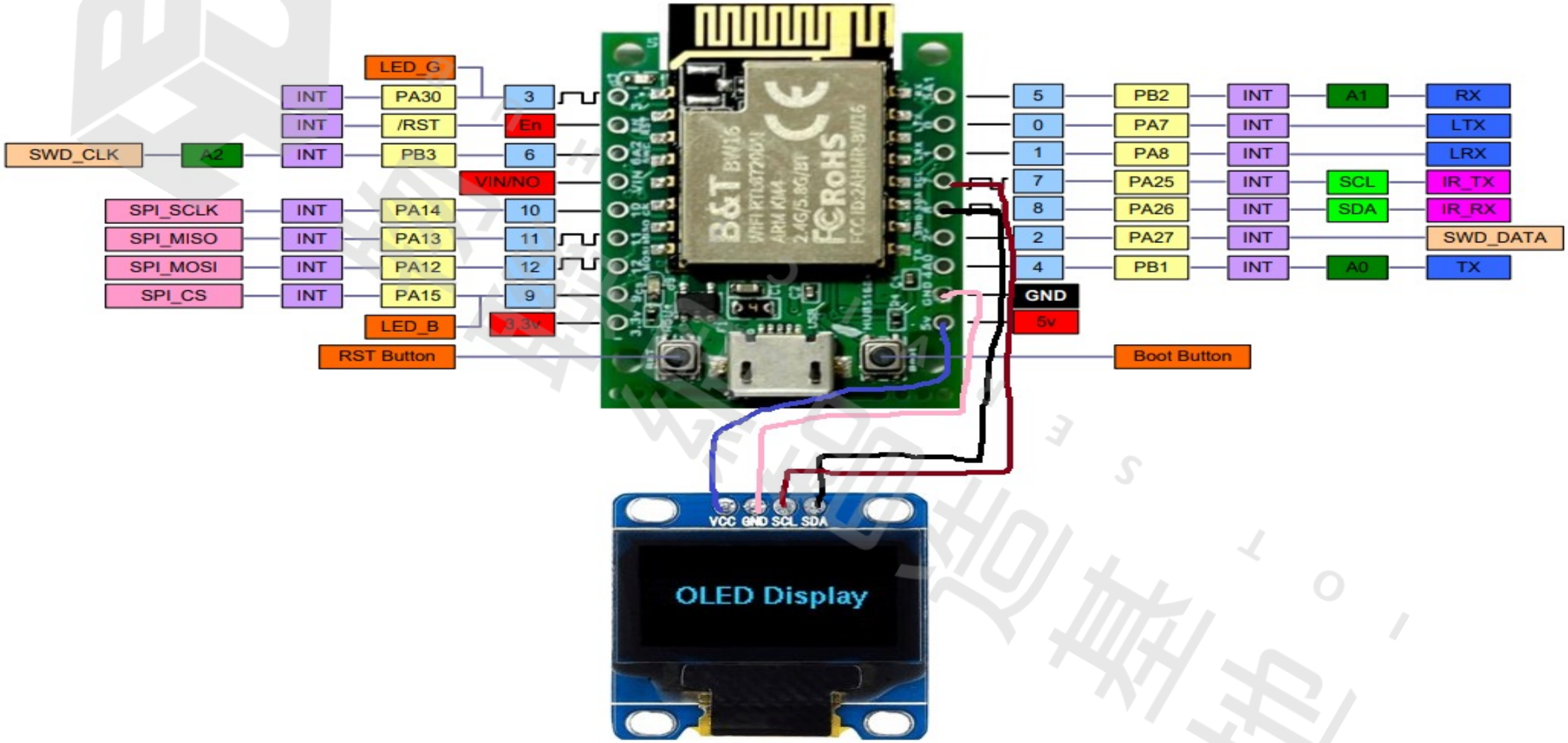
當開啟Line機器人通知程式時，會去訂閱MQTT消息，當使用者測量開始到結束會產生一個統計報告給使用者該報告會顯示幾次低於90%

LineBot通知機器人程式

Webhook URL [redacted] /callback

Line機器人Webhook URL

電路圖


LED_G
INT
PA30
3
INT
/RST
En
SWD_CLK
A2
INT
PB3
6
VIN/NO
SPI_SCLK
INT
PA14
10
SPI_MISO
INT
PA13
11
SPI_MOSI
INT
PA12
12
SPI_CS
INT
PA15
9
LED_B
3.3v
RST Button
5
PB2
INT
A1
RX
0
PA7
INT
LTX
1
PA8
INT
LRX
7
PA25
INT
SCL
IR_TX
8
PA26
INT
SDA
IR_RX
2
PA27
INT
SWD_DATA
4
PB1
INT
A0
TX
GND
5v
Boot Button
VCC GND SCL SDA
OLED Display

程式碼

```
1  #include "BLEDevice.h"
2  #include <PubSubClient.h>
3  #include <WiFi.h>
4  #include <ArduinoJson.h>
5  #include <Adafruit_GFX.h>
6  #include <Adafruit_SSD1306.h>
```

依賴庫

```
16 // 定義MQTT的相關參數
17 #define MQTT_HOST "test.mosquitto.org"
18 #define MQTT_PORT 1883
19 #define MQTT_TOPIC "oximeter/json"
20
21 // WiFi網絡名稱和密碼
22 char ssid []     ="****";//(****字號改為你的WIFI帳號)
23 char password[] = "*****";//(****字號改為你的WIFI密碼)
24
```

設定MQTT伺服器與WIFI

```
32 // 定義BLE服務和特徵UUID
33 #define UART_SERVICE_UUID      "CDEACB80-5235-4C07-8846-93A37EE6B86D"
34 #define CHARACTERISTIC_UUID_RX "CDEACB81-5235-4C07-8846-93A37EE6B86D"
35 #define CHARACTERISTIC_UUID_TX "CDEACB81-5235-4C07-8846-93A37EE6B86D"
```

設定搜尋血氧機藍芽

程式碼

```
65  // BLE通知回調函數，用於處理接收到的數據
66  void notificationCB (BLERemoteCharacteristic* /*chr*/, uint8_t* data, uint16_t len) {
67    byte msg[len + 1];
68    memcpy(msg, data, len);
69
70    if (msg[4] == 4) {
71      heart_bit = msg[6];
72      blood_oxygen = msg[7];
73      PI_index = msg[8];
void setup() {
  // 初始化串口和WiFi連接
  Serial.begin(115200);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
```

設定抓取血氧機的資料

連線WIFI

程式碼

```
// 初始化BLE
BLE.init();
BLE.setScanCallback(scanCB);
BLE.beginCentral(1);

// 開始掃描BLE設備
BLE.configScan()->startScan(3000);
BLE.configConnection()->connect(targetDevice, 4000);
delay(1000);
int8_t connID = BLE.configConnection()->getConnId(targetDevice);
if (!BLE.connected(connID)) {
  Serial.println("BLE not connected");
} else {
  // 發現BLE服務
  BLE.configClient();
  client = BLE.addClient(connID);
  client->discoverServices();
  Serial.print(">> 發現連接設備的服務");
  do {
    Serial.print(".");
    delay(1000);
  } while (!(client->discoveryDone()));
  Serial.println();
```

初始化藍芽與藍芽掃描

```
170  // 構建JSON字符串
171  StaticJsonDocument<200> jsonDoc;
172  JsonObject status = jsonDoc.createNestedObject("d");
173  status["id"] = id;
174  status["bpm"] = heart_bit;
175  status["spo2"] = blood_oxygen;
176  status["pi"] = PI_index * 0.1;
177
178  char jsonString[200];
179  serializeJson(jsonDoc, jsonString);
180  Serial.println(jsonString);
181
182  // 發佈MQTT訊息
183  if (heart_bit > 0 && heart_bit < 255) {
184    if (!mqtt.publish(MQTT_TOPIC, jsonString)) {
185      Serial.println("MQTT publish failed");
186    }
187  }
```

將數據發送至MQTT

程式碼

```
189   // 更新OLED顯示器
190   display.clearDisplay();
191   display.setTextSize(2);
192   display.setTextColor(WHITE);
193   display.setCursor(0, 10);
194   display.println("BPM");
195   display.setCursor(60, 10);
196   display.println(heart_bit);
197   display.setCursor(0, 30);
198   display.println("SPO2:");
199   display.setCursor(60, 30);
200   display.println(blood_oxygen);
201   display.setCursor(0, 50);
202   display.println("PI:");
203   display.setCursor(60, 50);
204   display.println(PI_index / 10.0);
205   display.display();
206
207   // 延遲3秒
208   delay(3000);
209 }
```

將數據顯示於OLED

```
// 連接到MQTT伺服器
void connectToMqtt() {
  while (!mqtt.connected()) {
    Serial.print("嘗試 MQTT 連接...");
    if (mqtt.connect(MQTT_DEVICEID)) {
      Serial.println("MQTT 連接");
      mqtt.subscribe(MQTT_TOPIC);
    } else {
      Serial.print("MQTT 連線失敗, rc=");
      Serial.print(mqtt.state());
      Serial.println(" 5 秒後重試");
      delay(5000);
    }
  }
}

// 重新連接到MQTT伺服器
void reconnectMqtt() {
  while (!mqtt.connected()) {
    Serial.print("嘗試 MQTT 重新連接...");
    if (mqtt.connect(MQTT_DEVICEID)) {
      Serial.println("MQTT 重新連接");
      mqtt.subscribe(MQTT_TOPIC);
    } else {
      Serial.print("MQTT 重連失敗, rc=");
      Serial.print(mqtt.state());
      Serial.println(" 5 秒後重試");
      delay(5000);
    }
```

MQTT連線

程式碼 Linebot.py

當從MQTT收到血氧的數據低於95%發送指定訊息

```python
# 你的 Channel Access Token 和 Channel Secret
line_bot_api = LineBotApi('McVDfYNesigDvZ4ZlHDTMhpsf
handler = WebhookHandler('d5af999e4604818f97dfbe1859

# 文件路徑
USER_MAP_FILE = 'user_map.json'

# 儲存ID和LINEID的映射
user_map = {}

# 讀取用戶映射信息
1 个用法
def load_user_map():
    global user_map
    if os.path.exists(USER_MAP_FILE):
        with open(USER_MAP_FILE, 'r') as file:
            user_map = json.load(file)
    else:
        user_map = {}

# 保存用戶信息
3 个用法
def save_user_map():
    with open(USER_MAP_FILE, 'w') as file:
        json.dump(user_map, file)

# 加載用戶資料
load_user_map()
```

```python
# 當收到消息時調用
2 个用法
def on_message(client, userdata, msg):
    try:
        payload = json.loads(msg.payload.decode())
        data = payload['d']
        user_id = str(data['id'])
        spo_value = data['spo2']
        if user_id in user_map:
            line_user_id = user_map[user_id]
            current_time = datetime.now().strftime('%Y-%m-%d %H:%M:%S')

            if spo_value < 90:
                alert_message = '血氧濃度嚴重低於標準值'
                alert_message1 = '請立即採取措施！'
                alert_color = '#FF0000'  # 紅色
                alt_text = '血氧濃度警告'
                warning_text = '警告！'
            elif 90 <= spo_value <= 94:
                alert_message = '血氧濃度略低'
                alert_message1 = '建議注意休息與呼吸'
                alert_color = '#FF8C00'  # 橙色
                alt_text = '血氧濃度提示'
                warning_text = '注意！'
            else:
                return  # 如果血氧正常，不進行任何操作
```

LineDevelopers

使用LineDevelopers創建一個官方帳號，到Basic settings畫面將Channel secret的代碼貼至python程式碼裡的handler = WebhookHandler()

TOP > 血氧數據通知 > 血氧數據平台 > **Basic settings**

Edit

App types Bot

Permissions PROFILE

Channel secret [redacted]

Assertion Signing Key Register a public key

Your user ID Uba0d7db317217e990a1d1b97a94af9d0

LineDevelopers

TOP > 血氧數據通知 > 血氧數據平台 > **Messaging API**

group chats

Auto-reply messages Disabled

Greeting messages Enabled

Channel access token

Channel access token (long-lived)

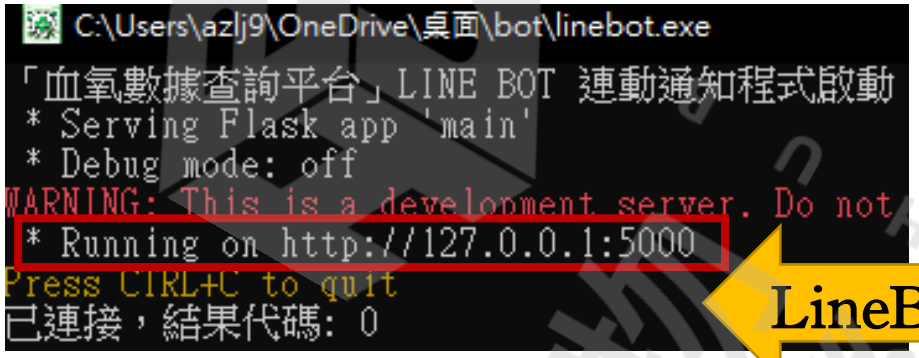
[redacted]

使用LineDevelopers選擇一個官方帳號，到Messaging API畫面將Channel access token的代碼貼至python程式碼裡的line_bot_api = LineBotApi()

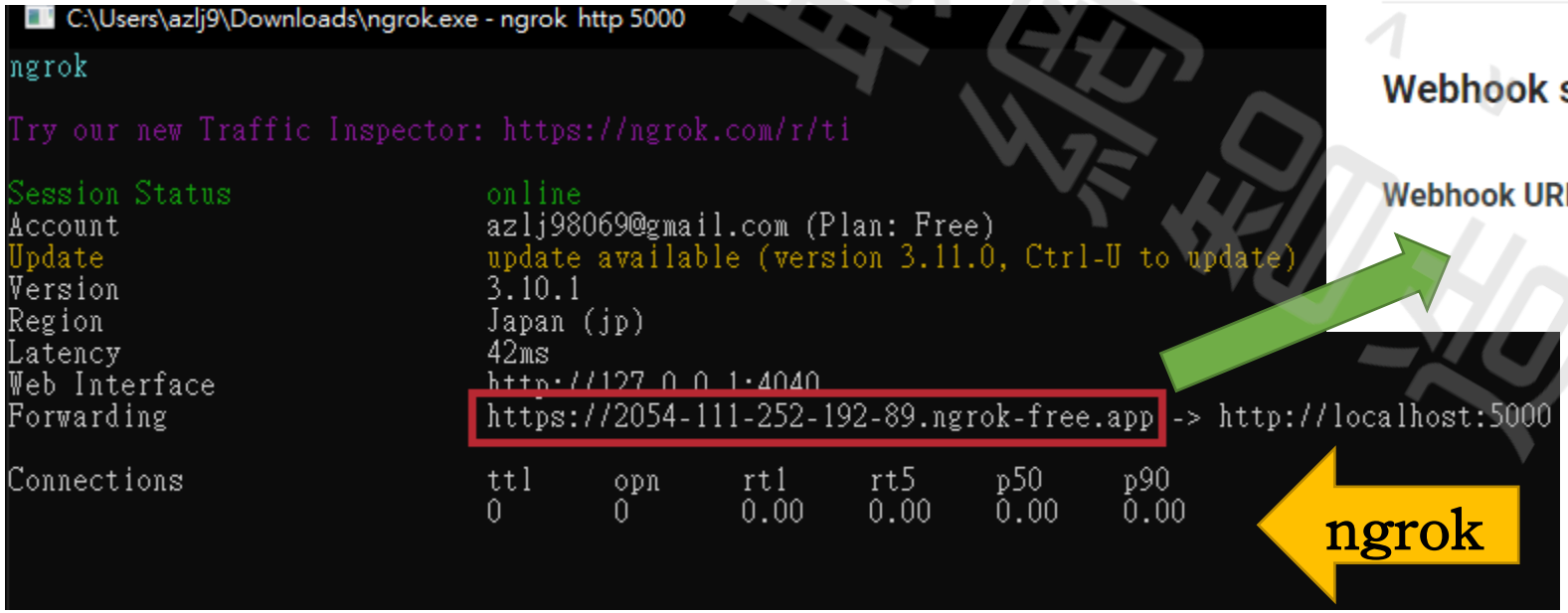
```
# 你的 Channel Access Token 和 Channel Secret
line_bot_api = LineBotApi([redacted]
```

Webhook settings

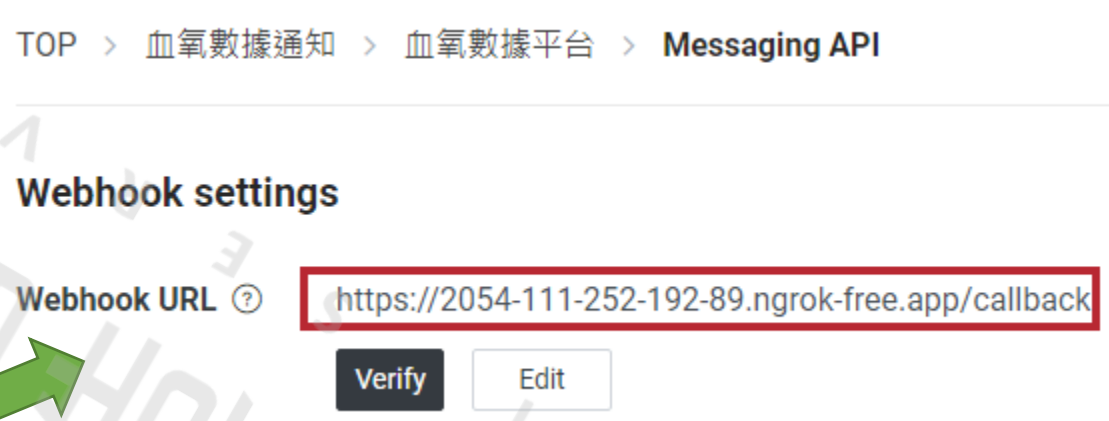
由於我們使用自己的電腦當作伺服器但我們的IP沒有對外公開，那我們使用ngrok來將IP進行公開，將我們得到的IP貼到Messaging API裡的Webhook URL裡，後面需要添加/callback才能與我們的程式進行溝通



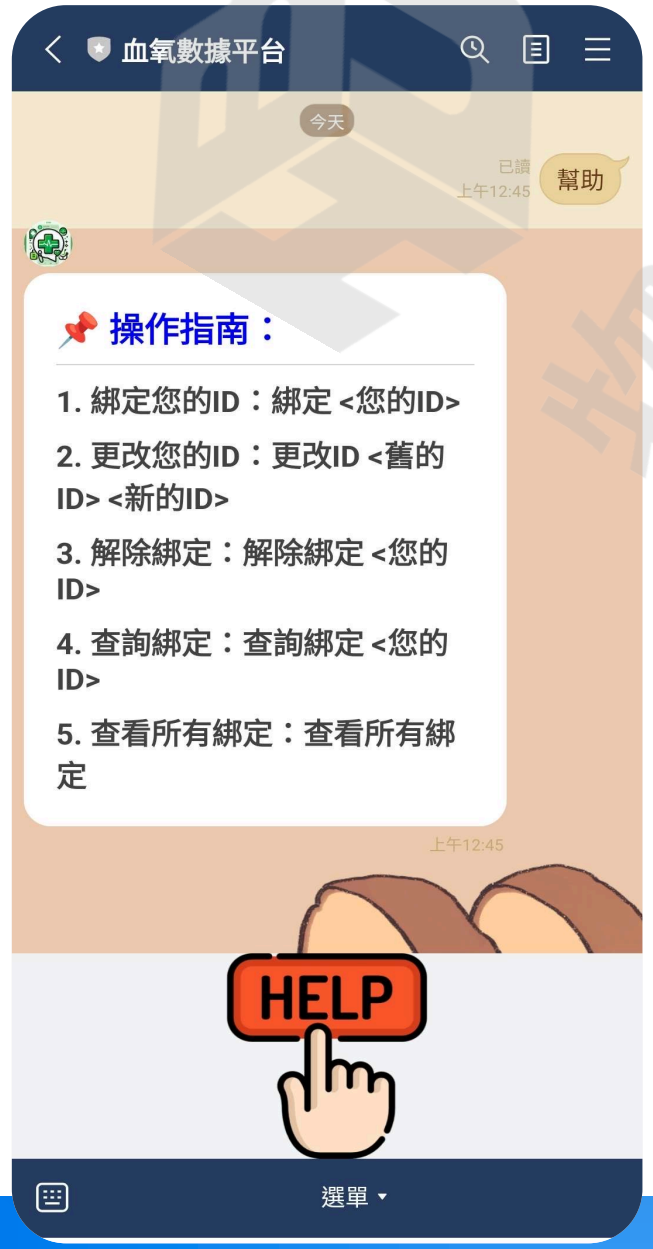
LineBot程式



ngrok



LineBot



當使用者輸入幫助等關鍵字後會跳出使用指南訊息:

1.**綁定ID**:在聊天室輸入「綁定 123」（其中123為您的ID號碼），以進行ID綁定。

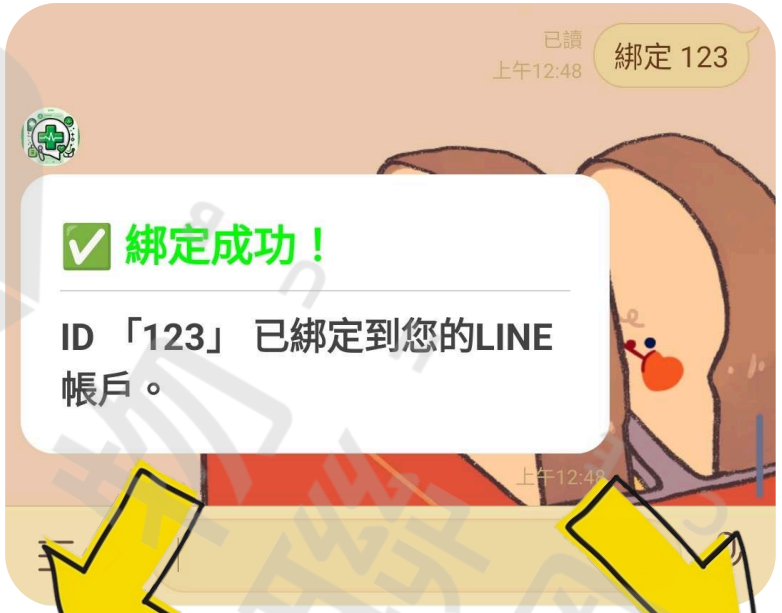
2.**更改ID:**若需更改已綁定的ID，請輸入「更改ID 123 456」（123為原ID號碼，456為新ID號碼）。

3.**解除綁定**:若不再使用此機器人，可通過輸入「解除綁定 123」（123為您的ID號碼）來解除綁定。

4.**查詢綁定**:可通過輸入「查詢綁定 123」（123為您的ID號碼）來確認是否已正確綁定。

5.**查看所有綁定**:由於機器人支持綁定多個ID，若想查看所有綁定的ID或解除特定ID，可透過聊天室輸入「查看所有綁定」。

LineBot



當使用者綁定ID之後，當量測時低於指定數值時會透過訊息進行通知

LineBot功能


已讀
上午12:52
查詢綁定 207
🔍 綁定查詢
ID 「207」 已綁定到您的LINE帳戶。
上午12:52


上午12:50
已讀
上午12:51
更改ID 123 207
✅ 更改ID成功！
ID 「123」 已更改為 「207」。
上午12:51


已讀
上午12:54
查看所有綁定
已綁定的ID：
207, 123
上午12:54
選單


已讀
上午12:54
解除綁定 123
✅ 解除綁定成功！
ID 「123」 已解除綁定。
上午12:54
選單

程式碼

MQTTtoMysql.py

程式訂閱MQTT broker將數據上傳至資料庫裡

```python
# 當接收到來自MQTT的消息時的回調函數
1 个用法
def on_message(client, userdata, msg):
    try:
        payload = msg.payload.decode("utf-8")
        data = json.loads(payload)
        d = data.get("d", {})
        device_id = d.get("id")
        bpm = d.get("bpm")
        spo2 = d.get("spo2")
        pi = d.get("pi")
        store_data(db_config, device_id, bpm, spo2, pi)
    except Exception as e:
        print(f"Error processing message: {e}")
```

```python
# 將接收到的數據存入資料庫
1 个用法
def store_data(db_config, device_id, bpm, spo2, pi):
    try:
        conn = mysql.connector.connect(
            host=db_config["host"],
            user=db_config["user"],
            password=db_config["password"],
            database=db_config["name"]
        )
        cursor = conn.cursor()
        cursor.execute( operation: '''
            INSERT INTO spo2_data (device_id, bpm, spo2, pi) VALUES (%s, %s, %s, %s)
        ''', params: (device_id, bpm, spo2, pi))
        conn.commit()
        cursor.close()
        conn.close()
    except mysql.connector.Error as err:
        print(f"Error storing data: {err}")
```

MQTT上傳數據至資料庫

當開啟程式時，會先讀取config.json檔案，可以透過config檔案來進行MQTT Broker與資料庫設置，當資料庫沒有資料表時程式會自動新增

config.json

C: > Users > azlj9 > PycharmProjects > pythonProject4 > {} config.json > ...

```
{
    "mqtt": {
        "broker": "test.mosquitto.org",
        "port": 1883,
        "topic": "oximeter/json"
    },
    "database": {
        "host": "127.0.0.1",
        "user": "*****",
        "password": "*****",
        "name": "99"
    }
}
```

Config設置

資料庫數據

	id	device_id	timestamp	bpm	spo2	pi
編輯 複製 刪除	15	207	2024-06-18 21:22:22	80	98	11.8
編輯 複製 刪除	16	207	2024-06-18 21:22:25	80	98	11.8
編輯 複製 刪除	17	207	2024-06-18 21:22:29	78	98	11.4
編輯 複製 刪除	18	207	2024-06-18 21:22:32	78	98	11.4
編輯 複製 刪除	19	207	2024-06-18 21:22:35	79	98	10.6
編輯 複製 刪除	20	207	2024-06-18 21:22:38	79	98	10.6
編輯 複製 刪除	21	207	2024-06-18 21:22:47	81	98	11.2
編輯 複製 刪除	22	207	2024-06-18 21:22:51	80	98	11.6
編輯 複製 刪除	23	207	2024-06-18 21:22:54	79	98	11.4
編輯 複製 刪除	24	207	2024-06-18 21:22:57	79	98	11.4
編輯 複製 刪除	25	207	2024-06-18 21:23:00	79	98	10.5
編輯 複製 刪除	26	207	2024-06-18 21:23:03	79	98	10.5
編輯 複製 刪除	27	207	2024-06-18 21:23:06	76	98	7.4
編輯 複製 刪除	28	207	2024-06-18 21:23:09	76	98	7.4
編輯 複製 刪除	29	207	2024-06-18 21:23:13	77	98	5.8
編輯 複製 刪除	30	207	2024-06-18 21:23:16	77	98	5.8

Thanks!