HUB 8735 辨識手勢控制窗簾

摘要

本專題旨在研發一款智慧窗簾系統，可根據比出的手勢自動控制窗戶的開合，從而實現室內環境的智慧調節。智慧窗簾系統主要由三部分組成：馬達繼電器、HUB 8735 Ultra 和步進馬達控制系統。馬達繼電器可以對馬達進行操作，而 HUB 8735 Ultra 則可以偵測做出的手勢。透過與感測器的搭配，智慧窗簾系統可以自動調節窗戶的開合。

目錄

圖目錄

背景知識介紹

在此章節將會藉由「智慧窗簾系統」所會運用到的材料拆解成獨立的元件或物件作一概括性的介紹:

智慧窗簾系統

能夠節省能源：智慧窗簾可以根據比出的手勢等因素自動調節開啟和關閉，從而節省能源，降低能源消耗和環境污染。

- 提高安全性和隱私：智慧窗簾可以保護居民的隱私和安全，例如在晚上自動關閉窗簾，避免犯罪分子偷窺，同時可以有效避免陽光直射，保護居室內物品。

- 提高舒適度：智慧窗簾可以調節室內的光照和溫度，提高居家的舒適度和品質。

- 增加便利性：智慧窗簾可以透過手勢來控制窗簾的開啟和關閉，從而提高使用者的便利性和自由度。

馬達驅動模組 (L9110)

- L9110 馬達驅動模組可以控制直流馬達的轉動方向和速度，從而實現馬達的正轉、反轉、加速、減速和停止等運動控制。

- L9110 馬達驅動模組通常廣泛運用在機器人、遙控玩具、DIY 電子產品和其他需要控制馬達運動的應用中。

- L9110 馬達驅動模組可以控制馬達的電源開關，從而實現對馬達的電源管理，並防止馬達過熱和電流過載等問題。

- L9110 馬達驅動模組可以檢測馬達的運動狀態，並在發現馬達過載時自動關閉馬達電源，從而實現對馬達的過載保護。

USB Type-C

- USB Type-C 廣泛應用於移動設備、數碼相機、智能手錶、智能家居設備、開發板、音頻設備等各種電子設備中。

- USB Type-C 具有多種通信工具具有的優點，例如:小巧輕便、耐用、易於插拔、傳輸速度快等優點，因此被廣泛使用。

杜邦線

- 杜邦線是由許多細小的單芯導線組合而成,可以按照需要自行拆分重新組合。杜邦線的優點是它非常方便，可靠，易於使用，並被廣泛應用於電子實驗、原型設計、機器製造、電路調試和各種電子設備的製造中。

- 杜邦線常見的主要用途，如下:

  • 連接 Arduino 和面板：杜邦線將 Arduino 面板和面板連接在一起，可以很方便地把線路連接在 Arduino 上，**也就是此專題主要使用的內容**。

  • 連接感測器：將感測器與主機板連接，可以輕鬆讀取感測器的輸出數據，如溫度、濕度、壓力等。

  • 連接電機：使用杜邦線連接電機與控制器，可以實現機器人的動作控制。

  • 連接 LED 燈和顯示器：使用杜邦線連接 LED 燈和顯示器與控制器，可以顯示電路的狀態、顯示文本或圖形。

  • 連接模塊：將各種模塊與控制器連接，可以實現各種功能，如無線通訊、語言識別、圖像處理等。

研究方法及步驟

- **初始概念和規劃**

智慧窗簾是一種透過智慧技術控制窗簾開關的產品，透過 HUB 8735 Ultra 等裝置，感知手勢，並根據預設的開關條件自動開關窗簾。此系統在現代家居中已經變得越來越受歡迎，因可讓家庭變得更加智慧化、節能、環保和舒適。

在本次專題中，計劃設計和製作一種基於手勢感測的智慧窗簾系統，該系統可以透過感知做出的手勢，控制窗簾的開關，以實現節能和舒適的效果。

- **建立系統**

有出初始的概念後，開始尋找關於智慧窗簾的材料及軟體應用，先有效的把智慧窗簾出分為兩大主要方向，第一方向是:以硬體與材料為主;第二方向是:以軟體為主。

為實現智慧窗簾系統，需要建立一個基於手勢感測的自動控制系統。這個系統將包括 HUB 8735 Ultra、控制器和馬達等組件，透過組件協同作用，實現智慧窗簾的開關控制。

- **HUB 8735 Ultra**

使用一個高精度的手勢感測器，將其安裝在窗戶的室內。感測器將感知手勢的變化，並透過數據傳輸模塊將數據傳輸到控制器。採用這種設計可以提高手勢測量的準確性，並減少系統誤差。

- **視訊頭**

視訊頭是整個智慧窗簾系統的核心，它接收從手勢感測器收集到的手勢數據，並根據預定的開關來自動控制窗簾開關。視訊頭可以透過接收手勢來開關條件，控制窗簾的開啟和關閉，以達到節能和舒適的效果。

•步進馬達

為了實現窗簾的開關控制，使用一個小型的馬達，並安裝在窗簾的支幹上。控制器可以透過電路控制馬達的轉動，實現窗簾的開啟和關閉。步進馬達是一種直流無刷電動機，具有如齒輪狀突起（小齒）相鎖合的定子和轉子。它可以透過切換流向定子線圈中的電流，以一定角度逐步轉動。步進馬達的特徵是採用開迴路控制（Open-loop control）處理，不需要運轉量感測器（sensor）或編碼器，且切換電流觸發器的是脈衝信號，不需要位置檢出和速度檢出的回授裝置，所以步進電機可正確地依比例隨脈沖信號而轉動，因此達成精確的位置和速度控制，且穩定性佳。

圖 1 步進馬達

- TB6600 馬達繼電器

馬達繼電器 TB6600 是一種功能強大、可靠性高的電動機控制與保

護裝置。它不僅能有效地控制電動機的啟動和停止，還提供全面的保護功能，防止電動機在運行過程中受到損害。其模塊化設計和易於安裝的特性，使其成為工業和建築電氣系統中不可或缺的重要組件。

圖 2 TB6600 馬達繼電器

- LRS-150-15

LRS-150-15 是一款高效、可靠的電源供應器，適用於多種工業和商業應用。其高效能和多種保護功能確保了穩定的運行，並能有效應對各種潛在的故障和過載情況。電源供應器的設計考慮了現代設備對高效、穩定和安全供電的需求，是各類自動化設備和儀器的理想選擇。

圖 3 LRS-150-15

安裝與實作流程

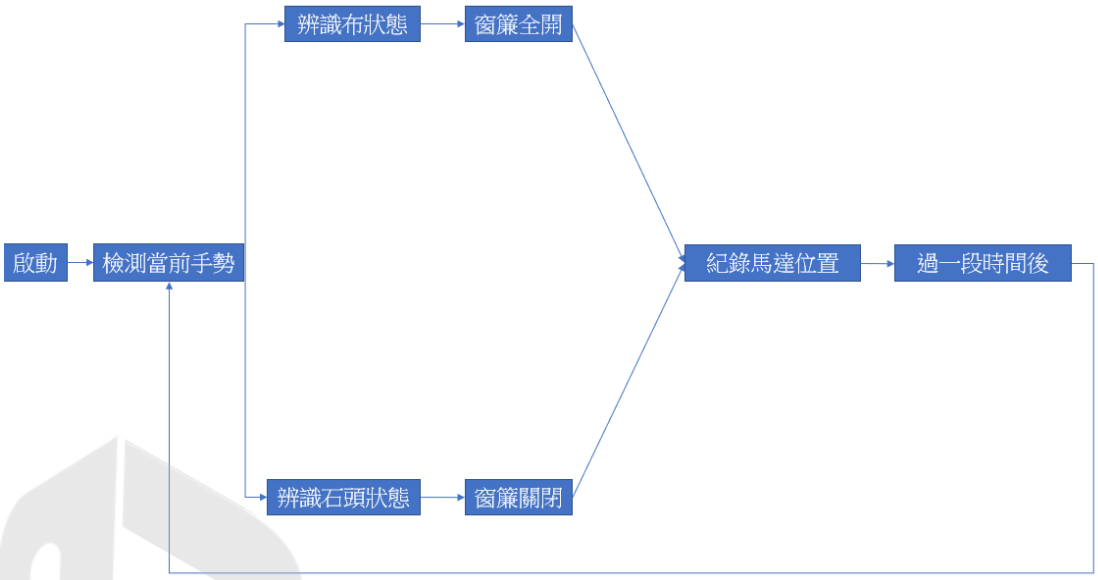

圖 4 流程圖

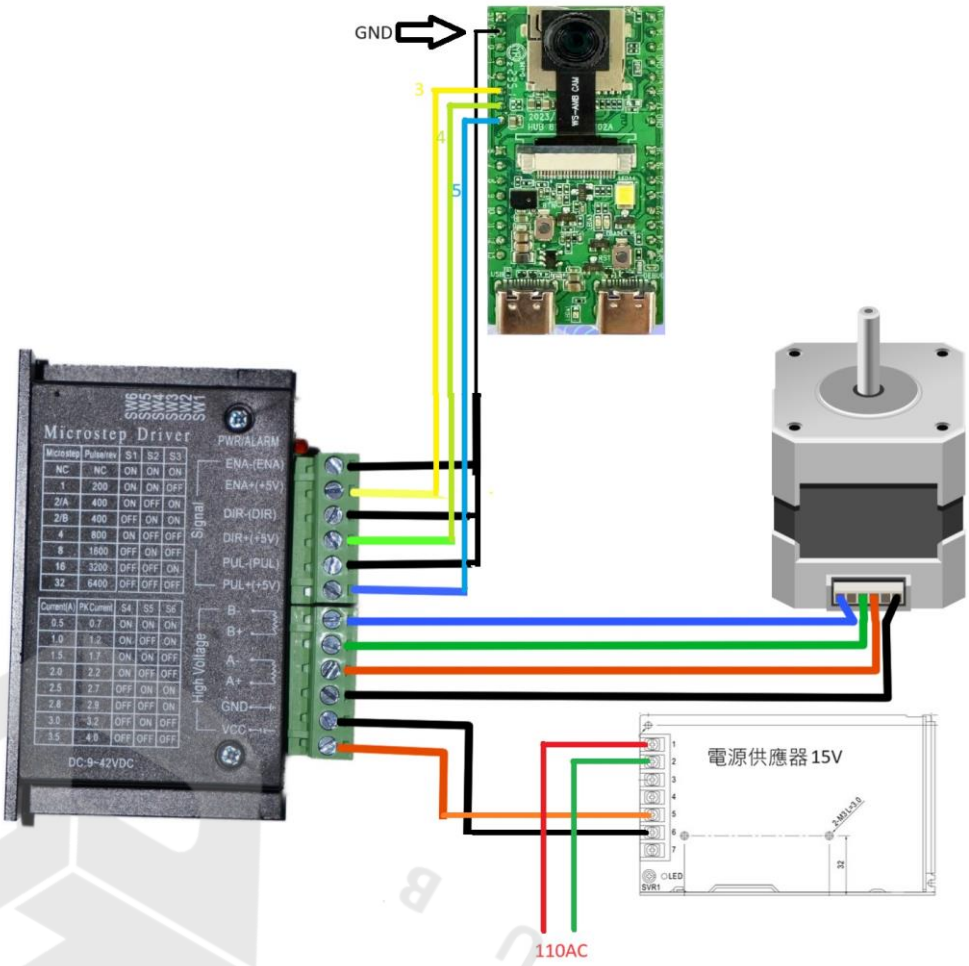

圖 5 電路圖

當 HUB 8735 Ultra 辨識到布手勢的時候，窗簾往下拉，辨識到石頭手勢的時候，窗簾往上拉。

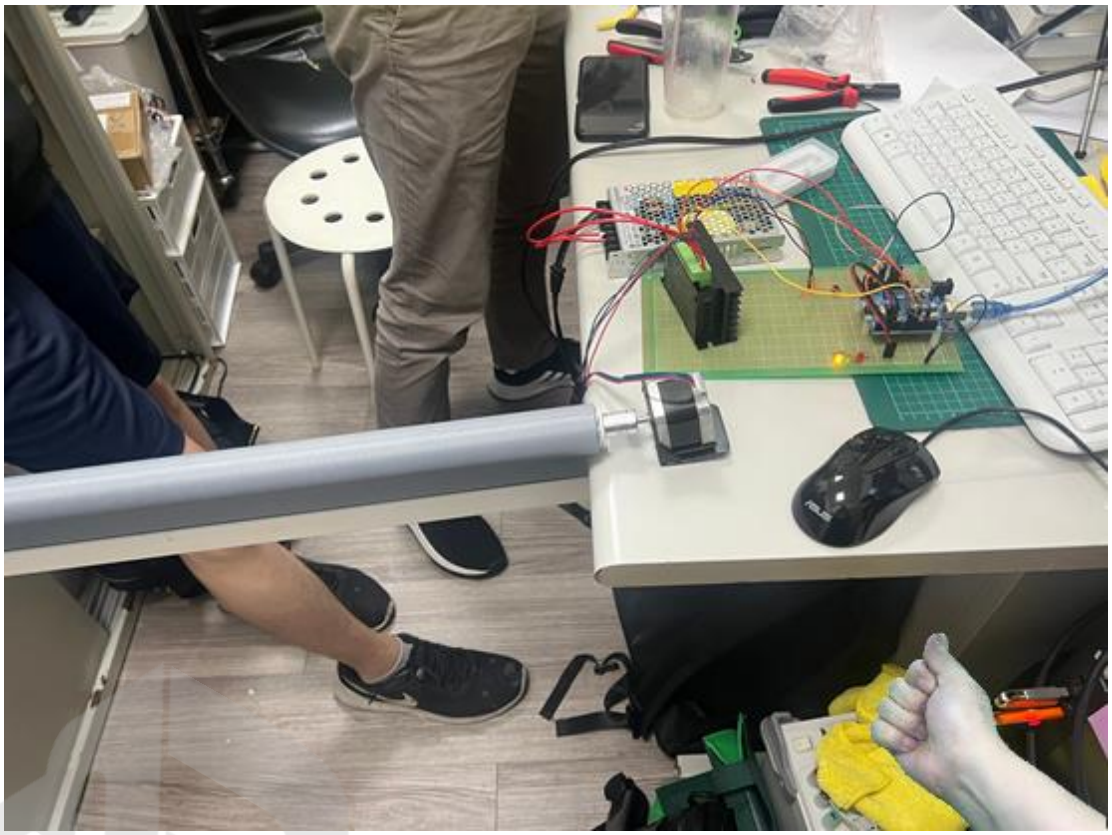

圖 6 當辨識為石頭時

圖 7 當辨識為布時

程式碼

```
#include "WiFi.h"

#include "StreamIO.h"

#include "VideoStream.h"

#include "RTSP.h"

#include "NNObjectDetection.h"

#include "VideoStreamOverlay.h"


#include "ObjectClassList.h"

#include "scissors.h"

#include "paper.h"

#include "rock.h"

#include <StepperMotor.h>



#define CHANNEL   0
```

```
#define CHANNELNN 3


// Lower resolution for NN processing
#define NNWIDTH  576
#define NNHEIGHT 320


VideoSetting config(VIDEO_FHD, 30, VIDEO_H264, 0);
VideoSetting configNN(NNWIDTH, NNHEIGHT, 10, VIDEO_RGB, 0);
NNObjectDetection ObjDet;
RTSP rtsp;
StreamIO videoStreamer(1, 1);
StreamIO videoStreamerNN(1, 1);


char ssid[] = "Max";    // your network SSID (name)
char pass[] = "maxlin910920";       // your network password
int status = WL_IDLE_STATUS;
```

```
IPAddress ip;

int rtsp_portnum;


StepperMotor stepper_motor{ 3, 4, 5, 200, 100, true };


bool detected_paper = false;

bool detected_scissors = false;

bool detected_rock = false;




void setup()

{

    Serial.begin(115200);
```

```
// attempt to connect to Wifi network:
while (status != WL_CONNECTED) {
    Serial.print("Attempting to connect to WPA SSID: ");
    Serial.println(ssid);
    status = WiFi.begin(ssid, pass);

    // wait 2 seconds for connection:
    delay(2000);
}
ip = WiFi.localIP();

// Configure camera video channels with video format
```

```
information

    // Adjust the bitrate based on your WiFi network quality

    config.setBitrate(2 * 1024 * 1024);    // Recommend to use 2Mbps
for RTSP streaming to prevent network congestion

    Camera.configVideoChannel(CHANNEL, config);

    Camera.configVideoChannel(CHANNELNN, configNN);

    Camera.videoInit();


    // Configure RTSP with corresponding video format information

    rtsp.configVideo(config);

    rtsp.begin();

    rtsp_portnum = rtsp.getPort();



    // Configure object detection with corresponding video format
information

    // Select Neural Network(NN) task and models

    ObjDet.configVideo(configNN);
```

```
    ObjDet.setResultCallback(ODPostProcess);

    ObjDet.modelSelect(OBJECT_DETECTION, DEFAULT_YOLOV4TINY,
NA_MODEL, NA_MODEL);

    ObjDet.begin();



    // Configure StreamIO object to stream data from video channel
to RTSP

    videoStreamer.registerInput(Camera.getStream(CHANNEL));

    videoStreamer.registerOutput(rtsp);

    if (videoStreamer.begin() != 0) {

        Serial.println("StreamIO link start failed");

    }



    // Start data stream from video channel

    Camera.channelBegin(CHANNEL);



    // Configure StreamIO object to stream data from RGB video
```

```
channel to object detection

    videoStreamerNN.registerInput(Camera.getStream(CHANNELNN));

    videoStreamerNN.setStackSize();

    videoStreamerNN.setTaskPriority();

    videoStreamerNN.registerOutput(ObjDet);

    if (videoStreamerNN.begin() != 0) {

        Serial.println("StreamIO link start failed");

    }

    // Start video channel for NN

    Camera.channelBegin(CHANNELNN);



    // Start OSD drawing on RTSP video channel

    OSD.configVideo(CHANNEL, config);

    OSD.begin();

}
```

```
void loop(){

}


// User callback function for post processing of object detection
results
void ODPostProcess(std::vector<ObjectDetectionResult> results)
{
    uint16_t im_h = config.height();
    uint16_t im_w = config.width();

    Serial.print("Network URL for RTSP Streaming: ");
```

```
    Serial.print("rtsp://");

    Serial.print(ip);

    Serial.print(":");

    Serial.println(rtsp_portnum);

    Serial.println(" ");


    printf("Total number of objects detected = %d\r\n",
ObjDet.getResultCount());

    OSD.createBitmap(CHANNEL);


    //bool detected_paper = false;

    //bool detected_scissors = false;

    //bool detected_rock = false;


    if (ObjDet.getResultCount() > 0) {

        for (int i = 0; i < ObjDet.getResultCount(); i++) {
```

```
            int obj_type = results[i].type();

            if (itemList[obj_type].filter) {    // check if item
should be ignored



                ObjectDetectionResult item = results[i];

                // Result coordinates are floats ranging from 0.00
to 1.00

                // Multiply with RTSP resolution to get coordinates
in pixels

                int xmin = (int)(item.xMin() * im_w);

                int xmax = (int)(item.xMax() * im_w);

                int ymin = (int)(item.yMin() * im_h);

                int ymax = (int)(item.yMax() * im_h);



                // Draw boundary box

                printf("Item %d %s:\t%d %d %d %d\n\r", i,
itemList[obj_type].objectName, xmin, xmax, ymin, ymax);

                OSD.drawRect(CHANNEL, xmin, ymin, xmax, ymax, 3,
OSD_COLOR_WHITE);
```

```
                // Print identification text

                char text_str[20];

                snprintf(text_str, sizeof(text_str), "%s %d",
itemList[obj_type].objectName, item.score());

                OSD.drawText(CHANNEL, xmin, ymin -
OSD.getTextHeight(CHANNEL), text_str, OSD_COLOR_CYAN);



                // Set detection flags based on object type

                if (strcmp(itemList[obj_type].objectName, "paper")
== 0) {

                    detected_paper = true;

                    stepper_motor.rotate(20, true, 100);  // 正轉

                } else if (strcmp(itemList[obj_type].objectName,
"scissors") == 0) {

                    detected_scissors = true;

                    //stepper_motor.rotate(20, false, 100);  // 正
轉

                } else if (strcmp(itemList[obj_type].objectName,
```

```
"rock") == 0) {
                    detected_rock = true;
                    stepper_motor.rotate(20, false, 100);  // 反轉
                }
            }
        }
    }
    OSD.update(CHANNEL);

    // Control motor based on detection results

}
```

未來展望與結論

物聯網已經是未來不變的趨勢了，在未來的日常中，不管是在室外或室內人類在使用的物品都有機會透過自動化系統來控制，包括智慧窗簾也不例外。

智慧窗簾是一種透過物聯網連接的自動化窗系統，可以透過手勢控制。隨著科技的不斷前進，未來的趨勢可能包含以下幾個方面：

- 更多智能化：未來的智慧窗簾可能會聚集更多的感測器和智能演算法，能夠自主感知環境和用戶的需求，實現更多智能化的控制。

- 更多環保節能：智能窗可以透過手勢調節窗簾來開啟合適的溫度調節室內光線和溫度，實現節能環保的效果。

- 更普及化：隨著智能家居市場的不斷擴大和成熟，智慧窗簾的價格也會逐漸下降，更多的家庭和辦公場所會使用智能窗。

- 客製化：未來的智慧窗簾可據用戶的喜好和習慣進行自適應調節，現實中增加一個客製化的體驗。

- 更加安全可靠：智慧窗簾的安全性和可靠也是未來的重要趨勢之一，未來的智慧窗簾可採用更加安全可靠的通訊系統控制，用於保護客戶的安全和隱私。

在這項專題研究中，成功地開發了一個基於手勢感測的智慧窗簾系統。此系統能夠依據手勢的變化自動啟動和關閉窗簾，用來提供使用者更舒適的室內環境。

研究工作主要分為三個階段：需求分析、系統設計和實施。在需求分析階段，仔細查找需要市場需求的研究資料，了解使用者對於智慧窗簾的需求和期望，並明確定義系統的目標。第二階段，在系統設計階段，設計了一個適合的硬體和軟體架構，以實現手勢偵測和窗簾控制功能。最後階段，在實施階段，成功地製作

了一個原型系統並進行了測試和評估。

參考文獻

- 大連理工大學 電工電子實驗中心

  https://labsci.nankai.edu.cn/fileup/PDF/17416.pdf

- 小米科技 (參考其中設計模式及篇章

  https://www.kocpc.com.tw/archives/429952

- lazytomatolab

https://www.lazytomatolab.com/as-09/

- Robotics for beginners

https://full-skills.com/fr/arduino-uno-projects/arduino-limit-switches/

- 連猴子都能懂的程式設計入門

https://chaoyenwu.gitbooks.io/the-arduino-book-for-students/content/ch1.html

- Arduino & 樹莓派的專家 傑森創工

https://blog.jmaker.com.tw/arduino-tutorials-3/

- projecthub.arduino.cc

https://projecthub.arduino.cc/adamsstephen/controlling-an-l9100-motor-driver-board-using-arduino-3c3b16

- Device plus Arduino 基礎： 控制馬達

https://micro.rohm.com/tw/deviceplus/how-tos/arduino-guide/the-basics-of-arduino-electronics-controlling-the-motor/

- Just Do it - Arduino&麵包板&杜邦線

https://sites.google.com/cies.tn.edu.tw/codingrobot/arduino%E7%9B%B8%E9%97%9C/arduino%E9%BA%B5%E5%8C%85%E6%9D%BF%E6%9D%9C%E9%82%A6%E7%B7%9A

- Meduim Arduino 基本語法

https://medium.com/jeasee%E9%9A%A8%E7%AD%86/arduino-%E5%9F%BA%E6%9C%AC%E8%AA%9E%E6%B3%95-a6a580e1650b

- Ray 的 Arduino 教學網站

https://sites.google.com/view/rayarduino/home-%E6%9B%B4%E5%A4%9A-arduino-%E7%B7%B4%E7%BF%92

- HUB-8735 ULTRA

https://www.ideas-hatch.com/news_detail.jsp?id=437