醫療科技化及臨床照護自動化

成為病床患者的手腳---餵藥機器人

一、專題起點與靈感

我的專題靈感源自於醫療照護領域的實際需求，特別是針對行動不便者或記憶力受限者的服藥困難。傳統服藥過程容易因疏忽或身體限制導致問題，因此我決定設計一個餵藥機器人，能自動偵測使用者的嘴巴位置，並精準餵藥。這個專題結合了影像辨識、機械控制和通訊技術，既具備技術挑戰，又有實用價值。我希望這個機器人能應用於家庭或醫療機構，改善照護效率。

二、設計與規劃

我設計了一個由兩部分組成的系統，分工明確：

- **樹莓派（Raspberry Pi）**：
  使用 USB 攝影機與 OpenCV 進行 AI 臉部與嘴巴偵測。
- 計算嘴巴座標 (mouth_x, mouth_y)，並透過 UART 傳給 Arduino。
- 顯示即時影像，方便測試與調整。
- **Arduino**：
  接收座標，控制 MG995 伺服馬達移動機械手臂。
- 操作另一伺服馬達驅動藥盤，釋放藥物。

硬體清單（從相關資源或平台獲取）：

- 樹莓派（主控板）
- USB 攝影機（影像輸入）
- MG995 伺服馬達（機械手臂）
- 標準伺服馬達（藥盤旋轉）

- Arduino（動作控制）
- 透明藥盤與導管（藥物傳輸）
- 全向輪（mecanum wheels，增加移動靈活性）
- 跳線與 UART 通訊線

軟體工具：

- Python 3（搭配 OpenCV）
- Arduino IDE（編寫 C++ 程式）

我手繪了草圖，規劃機械手臂、藥盤與全向輪的結構，確保軟硬體協作順暢。

三、製作過程

製作過程分為軟體開發與硬體組裝兩個階段，並記錄可能遇到的實際問題與解決方案。

1. 軟體開發

樹莓派程式（Python）與 AI 人臉辨識

我使用 OpenCV 的 Haar Cascade 進行人臉與嘴巴偵測，開發過程中遇到以下問題，並逐步解決：

問題 1：程式碼輸入錯誤（初步測試程式）

在測試 USB 攝影機與 OpenCV 的基本功能時，程式運行時出現錯誤，原始程式如下：

import cv2

import numpy as np

import PIL import Image # 錯誤：import 語法錯誤，應為 from PIL import Image

```
import os
detector = cv2.CascadeClassifier('haarcascade_frontalface_default.xml') # 錯誤：路徑未正確指定
cap=cv2.VideoCapture(0)

while True:
  ret,frame=cap.read()
  cv2.imshow('Image',frame )
  img = frame.convert('L') # 錯誤：OpenCV 陣列無法使用 PIL 的方法
  cv2.imshow('Image2',img )
  if cv2.waitKey(1) & 0xFF == ord('q'):
    break

cap.release()
cv2.destroyAllWindows()
```

原因：對 Python 的語法和 OpenCV 的使用不熟悉，特別是 PIL 的正確 import 方式和 cv2.cvtColor() 的使用。 **解決方案**：透過查閱 OpenCV 官方文件（https://docs.opencv.org/）與技術資源，我修正程式：

```
import cv2
import numpy as np

cap = cv2.VideoCapture(0)
if not cap.isOpened():
  print("無法開啟攝影機，請檢查連接或驅動程式")
  exit()

while True:
```

```
    ret, frame = cap.read()
    if not ret:
        print("攝影機讀取失敗，結束程式")
        break

    cv2.imshow('Image', frame)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)  # 修正為 OpenCV 的灰階轉換
    cv2.imshow('Image2', gray)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

問題 2：人臉偵測不穩定（實時人臉偵測程式）
在實現實時人臉偵測時，程式偶爾無法偵測到人臉，或誤將背景識別為人臉。原程式如下：

```
import cv2

face_cascade = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")  # 錯誤：未使用正確路徑
cap = cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()
    if not ret:
        break
```

```
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=5, minSize=(30, 30))

    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0, 255), 2)

    cv2.imshow("Faces Detected", frame)
    key = cv2.waitKey(1)
    if key == 27:  # 錯誤：退出按鍵不一致，應為 'q'
        break

cap.release()
cv2.destroyAllWindows()
```

原因：對 OpenCV 的參數設定不熟悉，且退出按鍵（ESC）與專題需求（'q'）不一致，光線不足也影響偵測。

解決方案：透過上網查詢（Stack Overflow，https://stackoverflow.com/questions/tagged/opencv）與技術討論，我修正路徑為 cv2.data.haarcascades + 'haarcascade_frontalface_default.xml'，調整參數（scaleFactor=1.1, minNeighbors=3, minSize=(50, 50)），並統一退出按鍵為 'q'：

```
import cv2

face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
cap = cv2.VideoCapture(0)

while True:
```

```
    ret, frame = cap.read()
    if not ret:
        break

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=3, minSize=(50, 50))

    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0, 255), 2)

    cv2.imshow("Faces Detected", frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

問題 3：嘴巴偵測失敗與 UART 傳輸錯誤（最終專題程式）

在整合人臉與嘴巴偵測並傳送座標給 Arduino 時，嘴巴偵測不穩定，且 UART 傳輸偶爾失敗。原程式如下：

```
import cv2
import serial
import time

ser = serial.Serial('/dev/ttyUSB0', 9600, timeout=1)  # 錯誤：路徑可能不正確
time.sleep(2)
```

```
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')

mouth_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_mcs_mouth.xml')

cap = cv2.VideoCapture(0)



while True:

  ret, frame = cap.read()

  if not ret:

    break



  gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

  faces = face_cascade.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=5)



  for (x, y, w, h) in faces:

    roi_gray = gray[y:y+h, x:x+w]

    mouths = mouth_cascade.detectMultiScale(roi_gray, scaleFactor=1.5, minNeighbors=5)  # 錯誤：參數過於嚴格

    for (mx, my, mw, mh) in mouths:

      mouth_x = x + mx + mw // 2

      mouth_y = y + my + mh // 2

      cv2.circle(frame, (mouth_x, mouth_y), 5, (0, 0, 255), -1)

      print(f"嘴巴座標: {mouth_x}, {mouth_y}")

      ser.write(f"{mouth_x},{mouth_y}\n".encode())  # 錯誤：無錯誤處理

      time.sleep(0.5)
```

```
        break


    cv2.imshow("Feed", frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break


cap.release()
cv2.destroyAllWindows()
ser.close()
```

原因：對 OpenCV 參數設定不熟悉，scaleFactor=1.5 和 minNeighbors=5 過於嚴格；UART 路徑 /dev/ttyUSB0 可能錯誤，且未加入錯誤處理。 **解決方案**：透過技術資源（OpenCV 論壇，https://forum.opencv.org/）與實測，我調整參數為 scaleFactor=1.2 和 minNeighbors=3，限制嘴巴在人臉範圍內，並檢查 UART 路徑（發現應為 /dev/ttyUSB1）。修正後程式：

```
import cv2
import serial
import time


try:
    ser = serial.Serial('/dev/ttyUSB1', 9600, timeout=1)  # 修正路徑
    time.sleep(2)
except serial.SerialException:
    print("無法連接 Arduino，請檢查 UART 連接")
    exit()


face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')
```

```
mouth_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_mcs_mouth.xml')

cap = cv2.VideoCapture(0)


while True:
    ret, frame = cap.read()
    if not ret:
        print("攝影機讀取失敗，結束程式")
        break


    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=5)


    for (x, y, w, h) in faces:
        roi_gray = gray[y:y+h, x:x+w]
        mouths = mouth_cascade.detectMultiScale(roi_gray, scaleFactor=1.2, minNeighbors=3)
        for (mx, my, mw, mh) in mouths:
            mouth_x = x + mx + mw // 2
            mouth_y = y + my + mh // 2
            if x <= mouth_x <= x + w and y <= mouth_y <= y + h:  # 限制在人臉範圍
                cv2.circle(frame, (mouth_x, mouth_y), 5, (0, 0, 255), -1)
                print(f"嘴巴座標: {mouth_x}, {mouth_y}")
                try:
                    ser.write(f"{mouth_x},{mouth_y}\n".encode())
                except Exception as e:
                    print(f"傳輸錯誤: {e}")
```

```
            time.sleep(0.3) # 縮短延遲
            break

    cv2.imshow("Feed", frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
ser.close()
```

Arduino 程式（C++）

開發 Arduino 程式時，我也遇到以下問題：

問題：程式碼輸入錯誤與馬達不動作

初期編寫時，我誤將 Servo 庫的 attach 接腳寫錯（例如寫成 11 而非 9），導致伺服馬達不動作。原程式如下：

```
#include <Servo.h>

Servo servoArm;
Servo servoDisk;

int mouth_x, mouth_y;
String inputString = "";

void setup() {
```

```
  Serial.begin(9600);
  servoArm.attach(11); # 錯誤：應為 9
  servoDisk.attach(10);
  servoArm.write(90);
  servoDisk.write(0);
}

void loop() {
  while (Serial.available()) {
    char c = Serial.read();
    if (c == '\n') {
      int commaIndex = inputString.indexOf(',');
      if (commaIndex > 0) {
        mouth_x = inputString.substring(0, commaIndex).toInt();
        mouth_y = inputString.substring(commaIndex + 1).toInt();
        Serial.print("收到嘴巴座標: ");
        Serial.print(mouth_x);
        Serial.print(", ");
        Serial.println(mouth_y);
        moveArmToMouth();
        dispensePill();
      }
      inputString = "";
    } else {
      inputString += c;
```

```
      }
    }
}

void moveArmToMouth() {
  Serial.println("移動機械手臂...");
  servoArm.write(150);
  delay(1000);
}

void dispensePill() {
  Serial.println("投放藥物...");
  servoDisk.write(90);
  delay(500);
  servoDisk.write(0);
}
```

原因：對 Arduino 的接腳配置不熟悉，且未仔細檢查硬體連接圖。 **解決方案**：透過技術資源與實測，修正 servoArm.attach(9)。同時發現馬達不動作可能是供電不足，我改用獨立 5V 電源，並測試不同延遲與角度。修正後程式：

```
#include <Servo.h>

Servo servoArm;
Servo servoDisk;

int mouth_x, mouth_y;
String inputString = "";
```

```
void setup() {
  Serial.begin(9600);
  servoArm.attach(9);  # 修正接腳
  servoDisk.attach(10);
  servoArm.write(90);
  servoDisk.write(0);
}

void loop() {
  while (Serial.available()) {
    char c = Serial.read();
    if (c == '\n') {
      int commaIndex = inputString.indexOf(',');
      if (commaIndex > 0) {
        mouth_x = inputString.substring(0, commaIndex).toInt();
        mouth_y = inputString.substring(commaIndex + 1).toInt();
        Serial.print("收到嘴巴座標: ");
        Serial.print(mouth_x);
        Serial.print(", ");
        Serial.println(mouth_y);
        moveArmToMouth();
        dispensePill();
      } else {
        Serial.println("資料格式錯誤，忽略此數據");
```

```
        }
        inputString = "";
      } else {
        inputString += c;
      }
    }
}

void moveArmToMouth() {
  Serial.println("移動機械手臂...");
  servoArm.write(145); # 測試後的最佳角度
  delay(800);         # 優化延遲
}

void dispensePill() {
  Serial.println("投放藥物...");
  servoDisk.write(100); # 測試後的最佳角度
  delay(600);          # 延長釋放時間
  servoDisk.write(0);
  delay(500);
}
```

2. 硬體組裝

我組裝樹莓派、Arduino、MG995 伺服馬達、藥盤與全向輪時，遇到以下問題：

問題：全向輪無法正常移動

組裝後，全向輪無法順利轉動，導致機器人無法移動。

原因：直流馬達的正負極接反，且未檢查電源供應是否足夠。

解決方案：檢查接線並更正正負極，確保直流馬達使用 12V 電源，並測試全向輪的獨立移動功能。

問題：藥盤藥物卡住

測試時，藥盤旋轉後藥物偶爾卡住，未順利釋放。

原因：藥盤角度或導管設計不合適，導致摩擦過大。

解決方案：調整 servoDisk.write(100) 的角度，並傾斜導管 15 度，確保藥物順利掉落。

四、測試與改進

我進行了多次測試，觀察機器人在不同光線、距離和使用者位置下的表現，並記錄以下結果：

- **成功之處**：
  機器人能正確偵測臉部與嘴巴，顯示即時影像並標記目標，機械手臂與藥盤動作順利完成餵藥。
- 從照片中的展示場景（展覽看板與實際操作）可以看到系統穩定運行，獲得正面回饋。

1. **AI 人臉辨識的挑戰與改進**：
   光線變化：在昏暗環境下，Haar Cascade 偵測失敗率高。我調整環境光線並優化參數（scaleFactor=1.1, minNeighbors=3），但仍考慮升級至深度學習模型（如 MTCNN，https://github.com/ipazc/mtcnn）。
2. **角度與遮擋**：側臉時，嘴巴偵測失敗。我限制嘴巴在人臉範圍內，穩定性提升。
3. **硬體與動作的挑戰與改進**：
   伺服馬達角度校正：手臂與藥盤角度需多次測試，最終定為 145 度和 100 度。
4. **傳輸延遲**：縮短 time.sleep(0.3) 後，動作更連貫，但偶爾有資料丟失，我計畫加入緩衝機制。
5. **全向輪與藥物釋放問題**：修正接線與角度後，系統穩定性提升。

五、心得與反思

這個專題讓我學會如何整合 AI 影像辨識（OpenCV Haar Cascade）、硬體控制（Arduino）和通訊協作（UART）的技術。從設計草圖到硬體組裝，再到程式 debug，我深刻體會到軟硬體協作的複雜性與樂趣。最大的挑戰是基礎知識不足（例如程式語法錯誤、硬體接線問題）和實作經驗有限（例如全向輪接反、藥物卡住），但這些問題也成為我成長的契機。

透過技術資源與實測，我逐漸克服這些困難，學會查閱技術文件（OpenCV 官方文件、Stack Overflow）、測試與優化參數，並改善硬體設計。雖然 Haar Cascade 效率高，但其限制讓我意識到未來需要學習深度學習技術（如 MTCNN），提升系統魯棒性。

從照片中看到機器人在展覽中的表現，我感到非常有成就感，但也意識到仍有改進空間。例如，線路管理需要更整齊，AI 辨識應更穩健地應對複雜環境，硬體動作需更精準。展示時使用者的回饋（光線影響、動作速度等）也提醒我，未來需要考慮更多實際場景的需求。

未來優化方向：

1. 升級 AI 模型至深度學習方案（如 MTCNN，https://github.com/ipazc/mtcnn）。
2. 加入距離感測器（如超音波或雷射測距）校正手臂位置，提升精準度。
3. 優化藥盤結構，確保藥物順利釋放，可能增加傾斜設計或減小摩擦。
4. 改善線路管理，提升系統的耐用性與美觀度。
5. 進一步開發全向輪的控制程式，實現更靈活的移動策略。
6. 增加錯誤提示功能（如 LED 閃爍或蜂鳴器），提醒使用者調整位置或系統異常。

與電機系申請的相關連結

這個專題不僅展示了我的電機工程興趣，還展現了我的學習能力與成長潛力。我相信，貴系在控制系統、機器人技術與感測領域的先進研究將幫助我進一步提升技能，開發醫療行動解決方案。照片中的機器人證明了我的實作能力與創新精神，例如解決程式碼輸入錯誤、優化 Haar Cascade 參數與修正硬體

接線等。我期待在電機系學習期間，深入研究醫療機器人、人工智能與行動控制，實現技術與人道價值的結合。

照片中的技術細節與應用價值

- **主板與電路板**：照片中的綠色電路板顯示了樹莓派與 Arduino 的整合，線路整齊但需要進一步束縛，這反映了電路設計中的學習過程。這種設計與電機工程中的穩定性與可靠性的要求一致，特別是在醫療設備中至關重要。
- **全向輪**：mecanum 輪的設計體現了電機工程在機器人移動中的創新，適合醫療場景中的靈活操作。我最初接線錯誤，但透過學習修正後，系統穩定性提升。
- **機械手臂與藥盤**：MG995 伺服馬達與透明藥盤的結構展示了精準控制，這與醫療行動中的自動化需求高度相關。我因經驗不足導致馬達抖動，後來通過供電與角度調整解決。
- **USB 攝影機與 AI**：攝影機與 OpenCV 的結合實現了人臉與嘴巴偵測，這是醫療監控的核心技術。我因參數設定不熟悉導致偵測失敗，但透過測試與學習逐步優化。