模擬期中考（Camera 整合單元）

Q1. 指令題 (Camera 基礎操作)

請寫出一行指令使用 `raspistill` 拍攝一張解析度為 **1280x720** 的 PNG 照片，設定為**不顯示預覽**，並在**5 秒後拍照**，儲存為 `midterm.png`。

Q2. Debug 題 (OpenCV 人臉偵測)

請問以下程式中有哪幾個潛在的錯誤？請指出 **錯誤行號與修正建議**。

python Copy Edit

```python
import cv2
imagePath = "face.jpg"
cascadePath = "haarcascade_frontalface_default.xml"
faceCascade = cv2.CascadeClassifier(cascadePath)
image = cv2.imread(imagePath)
gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
faces = faceCascade.detectMultiScale(gray, scaleFactor=1.2, minSize=(40, 40), flags=0)
print("Found {0} faces!".format(len(faces)))
for (x, y, w, h) in faces:
    cv2.rectangle(image, (x, y), (x+w, y+h), (255, 0, 0), 2)
    cv2.imshow("Faces", image)
```

65-fdac-8009-aaff-3e93bc9db238

Copy Edit Share

```python
print("Found {0} faces!".format(len(faces)))
for (x, y, w, h) in faces:
    cv2.rectangle(image, (x, y), (x+w, y+h), (255, 0, 0), 2)
    cv2.imshow("Faces", image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Q3. 概念題 (Haar-Like Feature & Integral Image)

請簡述 **Integral Image 如何幫助 Haar-like 特徵運算加速**？請配合一張 2x2 小區塊示意說明運算公式。

Q4. 應用題 (臉部疲勞偵測)

你打算設計一個**疲勞偵測系統**，請根據課堂上 landmark 檢測的知識，描述你的偵測邏輯（例如：眼睛開合程度），需要提到你會使用哪個模型或偵測器。

Q5. TensorFlow Lite / YOLO 題

請說明如何使用 YOLOv8 推論一張圖片並**儲存已標記的圖像**（也就是包含框與類別的結果圖），請寫出完整 `model.predict(...)` 函數，並標出你加的參數。

9-aaff-3e93bc9db238

Q1. 指令題 (Camera 基礎操作)

> 指令如下：

bash Copy Edit

```
raspistill -n -t 5000 -o midterm.png -e png -w 1280 -h 720
```

- `-n`：不顯示預覽
- `-t 5000`：延遲 5 秒（5000 毫秒）再拍照
- `-o midterm.png`：輸出檔案名稱
- `-e png`：輸出為 png 格式
- `-w`、`-h`：設定解析度為 1280x720

Q2. Debug 題 (OpenCV 人臉偵測)

> ❗ 錯誤與修正說明：

- **第6行：**

python Copy Edit

```
gray = cv2.cvtColor(image, cv2.COLOR_RGB2GR
```

f-3e93bc9db238

- **第6行：**

python Copy Edit

```
gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
```

🔧 修正為：

python Copy Edit

```
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

原因： OpenCV 使用的是 BGR 色彩空間，而不是 RGB。

- **第7行：**

python Copy Edit

```
faces = faceCascade.detectMultiScale(gray, scaleFactor=1.2, minSize=(40, 40), flags=0)
```

✅ 可行，但建議 `minNeighbors` 參數應加入，例如：

python Copy Edit

```
faces = faceCascade.detectMultiScale(gray, scaleFactor=1.2, minNeighbors=5, minSize=(40, 40))
```

- **第11行：**

```python
cv2.imshow("Faces", image)
```

🔧 建議移出 `for` 迴圈，改放在畫完所有方框後再顯示圖片，避免重複開啟視窗：

```python
...
for (x, y, w, h) in faces:
    cv2.rectangle(image, (x, y), (x+w, y+h), (255, 0, 0), 2)
cv2.imshow("Faces", image)
```

Q3. 概念題 (Haar-Like Feature & Integral Image)

> 🧠 **解答如下：**

Integral Image 是一種用來快速計算矩形區域像素和的資料結構。
在進行 Haar-like 特徵匹配時，可以**快速算出黑白區域的平均值差異**，加速分類器的運作。

計算公式（以左上 A、右下 C 的矩形區為例）：

```text
Sum = C - B - D + A
```

舉例圖：

```css
A B
D C
```

- 可以在 O(1) 時間內快速取得矩形區域的像素總和。
- 這對於滑動視窗比對 Haar 特徵時非常有效。

Q4. 應用題 (臉部疲勞偵測)

> ⚬ **偵測邏輯如下：**

- 使用 **dlib** 的 `shape_predictor_68_face_landmarks.dat` 取得臉部 68 個標記點。
- 特別觀察 **眼睛的6個點**（左眼點 36~~41~~、右眼點 42~~47~~）。
- 計算每隻眼睛的 **Eye Aspect Ratio (EAR)**：

```text
EAR = (||p2 - p6|| + ||p3 - p5||) / (2 * ||p1 - p4||)
```

- 當 EAR 小於某個閾值（例如 0.2）且持續超過一定時間（例如 10 幀），就判定為閉眼。

🔧 可搭配 OpenCV 顯示結果框、畫出 landmark 點。

Q5. TensorFlow Lite / YOLO 題

> 🦾 **YOLOv8 的推論並儲存圖片寫法：**

```python
model.predict(source="dog.jpg", save=True, save_txt=False, conf=0.5, verbose=True)
```

✅ 關鍵參數：

- `source="dog.jpg"`：輸入圖片
- `save=True`：儲存帶有標記框的圖片（自動儲存為 runs/predict/...）
- `conf=0.5`：設定信心值門檻
- `save_txt=False`：不要另存文字檔（可以改成 `True` 如果想要邊框坐標）
- `verbose=True`：顯示詳細推論過程

Q1. 實作題：音訊錄製與播放

請寫出完整的指令與 Python 函數，使 Raspberry Pi 錄製 3 秒的音訊為 `test.wav`，並播放出來（使用 `aplay` 與 Python 方式各一種）。

Q2. Debug 題：使用 SpeechRecognition 做 STT

以下程式有錯，請指出哪裡會出錯，並說明如何修正：

```python
import speech_recognition as sr
r = sr.Recognizer()
with sr.Microphone() as src:
    audio = r.listen(source)
print(r.recognize_google(audio))
```

Q3. 概念題：MFCC 是什麼？為何適合語音處理？

請簡要說明 MFCC（Mel-Frequency Cepstral Coefficients）的提取流程，並解釋它為何適合語音辨識用途。

Q4. 應用題：你要如何讓 OpenAI 的 TTS 說中文？

請說明如何用 OpenAI 的 `gpt-4o-mini-tts` 模型講中文，並指出程式中要調整的部分是什麼（語音或輸入文字？）。

Q5. 統整題：請描述如何實作語音控制距離量測並唸出結果的流程（Quiz 2 題）

你要讓 Raspberry Pi 執行一個語音指令（例如 "measure distance"），接著量測距離（使用 HC-SR04），最後唸出結果。請列出實作步驟與會用到的函式或工具。

Q1. 實作題：音訊錄製與播放

🔹 **(1) 使用指令錄製與播放：**

bash Copy Edit

```bash
arecord -f cd -d 3 test.wav   # 錄製3秒音訊
aplay test.wav                # 播放錄音
```

🔹 **(2) 使用 Python 播放：**

python Copy Edit

```python
import wave
import pyaudio

def play_wav(file_path):
    CHUNK = 1024
    wf = wave.open(file_path, 'rb')
    p = pyaudio.PyAudio()
    stream = p.open(format=p.get_format_from_width(wf.getsampwidth()),
                    channels=wf.getnchannels(),
                    rate=wf.getframerate(),
                    output=True)
```

```
                    channels=wf.getnchannels(),
                    rate=wf.getframerate(),
                    output=True)
    data = wf.readframes(CHUNK)
    while data:
        stream.write(data)
        data = wf.readframes(CHUNK)
    stream.stop_stream()
    stream.close()
    p.terminate()

play_wav("test.wav")
```

Q2. Debug 題：SpeechRecognition 使用錯誤

🔎 原始錯誤程式：

python Copy Edit

```
with sr.Microphone() as src:
    audio = r.listen(source)  # 🔴 錯誤變數名
```

✅ **修正後：**

✅ **修正後：**

python Copy Edit

```
with sr.Microphone() as source:
    audio = r.listen(source)
```

✅ 補充：加上噪音校正建議更完整：

python Copy Edit

```
with sr.Microphone() as source:
    r.adjust_for_ambient_noise(source, duration=1)
    audio = r.listen(source)
```

Q3. 概念題：MFCC 是什麼？為何適合語音處理？

> 🔬 **MFCC（Mel-Frequency Cepstral Coefficients）簡述：**

1. 將語音信號分成短時間窗（Short-Time Fourier Transform）
2. 映射到 **Mel scale**（符合人耳感知）
3. 對 Mel 頻譜取對數（強調音量變化）
4. 再做離散餘弦轉換（DCT）
5. 前幾個係數就是 MFCC（保留語音主要特徵）

> ✅ **為何適合語音處理？**

- 因為 MFCC 模擬了人耳對音高變化的感知特性
- 它將語音轉成低維度、高語意資訊的向量
- 被廣泛用於語音辨識、語者辨識、語音情緒分析等任務

Q4. 應用題：如何讓 OpenAI TTS 說中文？

> ✅ **做法：直接改變輸入文字為中文即可。**

python Copy Edit

```python
response = client.audio.speech.create(
    model="gpt-4o-mini-tts",
    voice="onyx",  # 中文效果較佳的語者
    input="今天天氣真好，我們一起出門吧！",
    instructions="以自然、清晰的中文語調說話"
)
```

- **不需要改語言參數或 API**
- 輸入內容是中文就會使用相對應的語音模型處理

Q5. 統整題：語音控制距離量測並唸出結果

> 🎯 **完整流程如下：**

1. **錄製或語音輸入：**
   - 使用 `speech_recognition` 偵測語音指令，例如 "measure distance"
   - 可用 `Microphone()` 或錄好的 `WAV` 檔輸入
2. **辨識語音內容：**
   - `r.recognize_google(audio)` 或 `r.recognize_openai(audio)`
3. **呼叫 HC-SR04 感測器：**
   - 透過 `GPIO` 模組觸發與接收回音
   - 計算距離：

```python
distance = (duration * 34300) / 2
```

4. **產生語音回應：**
   - 使用 gTTS：

   - 使用 gTTS：


```python
from gtts import gTTS
tts = gTTS(text=f"距離是 {distance:.1f} 公分", lang='zh')
tts.save("distance.mp3")
os.system("play distance.mp3 > /dev/null 2>&1")
```

🎁 小補充：

- 若題目要求輸入為 MP3 可轉 WAV：

```bash
ffmpeg -i input.mp3 output.wav
```

Q1. 公式題：氣壓換算成高度

請寫出將氣壓值（以 Pa 為單位）轉換為海拔高度的公式，並說明各參數的意義。若壓力為 95000 Pa，請計算大約高度。

Q2. 概念題：加速度計校正的流程為何？為何需要校正？

請簡述加速度計在靜止時的理論值與實際偏差，以及如何取得 offset 並校正資料。

Q3. 應用題：請寫出計算距離的 Python 函數

已知加速度已完成校正，請根據：

$$s = v_0 t + \frac{1}{2} a t^2$$

寫一個函數 `calc_distance(acc, t)` 計算位移，假設初速為 0。

Q4. 應用題：如何以磁力計計算 Heading？請寫出公式與應用場景

請說明如何由 X/Y 軸磁力值計算方位角，並列出一個你能想像的應用情境。

Q5. 程式題：實作跌倒偵測演算法 + 通知 (LED)

請寫出一段簡單 Python 程式：

- 根據總加速度 total_acc = √(x² + y² + z²)
- 若 total_acc 超過某閾值（例如 2G），則開啟 LED（GPIO 控制）

Q1. 公式題：氣壓換算成高度

公式：

$$\text{altitude} = \left(1 - \left(\frac{\text{pressure}}{\text{sea\_level\_pressure}}\right)^{\frac{1}{5.255}}\right) \times 44330$$

參數說明：

- `pressure`：當前氣壓值（單位 Pa）
- `sea_level_pressure`：海平面標準氣壓（預設值為 101325 Pa）
- `altitude`：換算後的高度（單位為公尺）

範例計算：

python Copy Edit

```python
pressure = 95000
sea_level = 101325
altitude = (1 - (pressure / sea_level)**(1/5.255)) * 44330
print(round(altitude, 2))  # 輸出：約 524.25 公尺
```

↓

Q2. 概念題：加速度計校正流程

為何需要校正？

- 加速度計在靜止時應只量到重力，加速度向量應為 (0, 0, ±1) G。
- 實際上因為偏移（offset/drift）會出現偏差，因此需要校正。

校正步驟：

1. 將 IMU 靜置在桌面，保持不動。
2. 持續收集 X、Y、Z 軸數據，例如 1000 筆。
3. 分別計算每軸平均值，作為偏移量。
4. 校正方式：

python Copy Edit

```python
acc_x_calibrated = acc_x_raw - offset_x
```

Q3. Python 應用題：計算距離

> ✅ 根據公式 $s = \frac{1}{2}at^2$，假設 $v_0 = 0$

python Copy Edit

```python
def calc_distance(acc, t):
    """
    acc: 已校正的加速度（單位 m/s^2）
    t: 時間（單位秒）
    回傳距離（單位公尺）
    """
    return 0.5 * acc * t**2

# 範例使用
print(calc_distance(2.5, 3))  # 輸出：11.25 公尺
```

Q4. Heading 計算與應用場景

> **公式：**

$$\theta = \arctan 2(mag\_y, mag\_x)$$

> **補充：**

- 角度可使用 `math.degrees()` 轉換為角度
- 若使用地磁角修正：

$$\theta += declination$$

> **應用場景：**

- **電子指南針**：用於導航應用中，依據磁場方向顯示方位。
- **無人機航向控制**：結合 GPS 提供準確飛行方向。

Q5. 程式題：跌倒偵測並啟動 LED 通知

✅ 使用總加速度超過閾值來偵測跌倒（閾值可設為 2.5 G）

python Copy Edit

```python
import RPi.GPIO as GPIO
import numpy as np
import time

# GPIO 設定
LED_PIN = 18
GPIO.setmode(GPIO.BCM)
GPIO.setup(LED_PIN, GPIO.OUT)

# 模擬讀入加速度值（實際應從 IMU 讀值）
acc_x = 1.5
acc_y = 2.0
acc_z = 1.8

# 計算總加速度
total_acc = np.sqrt(acc_x**2 + acc_y**2 + acc_z**2)
print(f"Total Acc: {total_acc:.2f} G")
```

```python
print(f"Total Acc: {total_acc:.2f} G")

# 閾值判斷
if total_acc > 2.5:
    print("⚠️ 偵測到可能跌倒！亮燈通知")
    GPIO.output(LED_PIN, GPIO.HIGH)
    time.sleep(5)
    GPIO.output(LED_PIN, GPIO.LOW)

GPIO.cleanup()
```

✅ 使用 numpy 的 `sqrt()` 計算加速度模長 ✅ GPIO 控制 LED 作為警示裝置

👋 如需延伸題（例如 sensor fusion、IMU + 音訊等跨模組整合），我可以再幫你出一套更高階的模擬題。需要嗎？

Q1. 實作題：請寫出一段 Python 程式碼，讓 LED 閃爍

要求：

- 每秒閃爍一次
- 使用 `GPIO.BOARD` 模式
- 開始時印出 "LED on"，熄滅時印出 "LED off"

Q2. 概念題：為什麼超音波模組的 Echo 腳位需要降壓？如何降壓？

Q3. Python 實作題：請寫一段程式碼計算超音波距離

條件：

- TRIG 持續 10μs 高電位
- Echo 高電位時間為 `duration`（單位：秒）
- 假設音速為 343 m/s，請給出距離公式與對應程式

Q4. 整合應用題（Quiz 1）：根據超音波距離改變 LED 閃爍速度

規則：

- < 50cm：快速閃爍
- 50–100cm：慢速閃爍
- > 100cm：不閃爍（LED 關閉）

Q5. IMU Flat 檢測（Quiz 2）：使用加速度數據判斷是否水平

說明：

- 如果加速度軸向為 (0, 0, ±1)，視為平放
- 判定條件為：Roll 與 Pitch 角度都接近 0（±10°）
- 平放 → LED 開，否則關

Q1. 實作題：LED 閃爍程式碼

python Copy Edit

```python
import RPi.GPIO as GPIO
import time

LED_PIN = 12  # 實體腳位 12
GPIO.setmode(GPIO.BOARD)
GPIO.setup(LED_PIN, GPIO.OUT)

try:
    while True:
        print("LED on")
        GPIO.output(LED_PIN, GPIO.HIGH)
        time.sleep(1)
        print("LED off")
        GPIO.output(LED_PIN, GPIO.LOW)
        time.sleep(1)
except KeyboardInterrupt:
    print("Program interrupted.")
finally:
    GPIO.output(LED_PIN, GPIO.LOW)
    GPIO.cleanup()
```

Q2. 概念題：Echo 腳位為何需要降壓？

> **原因：**

- HC-SR04 的 Echo 腳位會輸出 **5V 訊號**
- Raspberry Pi GPIO 腳位僅能承受 **最大 3.3V**
- 直接接收 5V 會 **損壞 GPIO**，甚至燒毀晶片

> **降壓方式（分壓器）：**

- 使用兩個電阻串聯分壓：
  - 上接 1KΩ（R1）
  - 下接 2KΩ（R2）
- 接點輸出約為：

$$V_{out} = \frac{R2}{R1 + R2} \times 5V = \frac{2}{3} \times 5V \approx 3.3V$$

- 安全地將 Echo 輸出降至可接受範圍

Q3. 超音波距離計算程式

> **距離計算公式：**

$$\text{distance} = \frac{\text{duration} \times \text{speed}}{2}$$

> **範例程式：**

python Copy Edit

```python
def calculate_distance(duration):
    speed = 343  # m/s
    distance = (duration * speed) / 2
    return distance * 100  # 轉成公分

# 測試
print(f"距離: {calculate_distance(0.01):.2f} cm")
```

> 若 Echo 高電位時間為 10ms（0.01 秒），結果會是：

ini Copy Edit

```ini
distance = (0.01 * 343) / 2 = 1.715 m = 171.5 cm
```

Q4. Quiz 1：LED 閃爍速度依距離變化

python Copy Edit

```python
import RPi.GPIO as GPIO
import time

TRIG = 16
ECHO = 18
LED = 12

GPIO.setmode(GPIO.BOARD)
GPIO.setup(TRIG, GPIO.OUT)
GPIO.setup(ECHO, GPIO.IN)
GPIO.setup(LED, GPIO.OUT)

def get_distance():
    GPIO.output(TRIG, False)
    time.sleep(0.01)
    GPIO.output(TRIG, True)
    time.sleep(0.00001)
    GPIO.output(TRIG, False)

    while GPIO.input(ECHO) == 0:
        pulse_start = time.time()
    while GPIO.input(ECHO) == 1:
```

Copy Edit

```
    while GPIO.input(ECHO) == 0:
        pulse_start = time.time()
    while GPIO.input(ECHO) == 1:
        pulse_end = time.time()

    duration = pulse_end - pulse_start
    distance = (duration * 343) / 2 * 100  # cm
    return distance

try:
    while True:
        d = get_distance()
        print(f"距離: {d:.2f} cm")
        if d < 50:
            GPIO.output(LED, GPIO.HIGH)
            time.sleep(0.2)
            GPIO.output(LED, GPIO.LOW)
            time.sleep(0.2)
        elif d < 100:
            GPIO.output(LED, GPIO.HIGH)
            time.sleep(1)
            GPIO.output(LED, GPIO.LOW)
            time.sleep(1)
        else:
            GPIO.output(LED, GPIO.LOW)
except KeyboardInterrupt:
    print("終止程式")
finally:
    GPIO.cleanup()
```

Q5. Quiz 2：IMU 偵測是否平放，控制 LED

> 假設你可以從 IMU 取得 roll/pitch 角度（用 dmp 或自行計算）：

python Copy Edit

```python
import RPi.GPIO as GPIO
import time
import random  # 模擬 roll pitch 讀值

LED = 12
GPIO.setmode(GPIO.BOARD)
GPIO.setup(LED, GPIO.OUT)

def is_flat(roll, pitch):
    return abs(roll) < 10 and abs(pitch) < 10

try:
    while True:
        # ⚠️這裡應該是你從 IMU 實際取得的角度值
        roll = random.uniform(-15, 15)   # 假設 roll 值
        pitch = random.uniform(-15, 15)  # 假設 pitch 值
        print(f"Roll: {roll:.1f}°, Pitch: {pitch:.1f}°")

        if is_flat(roll, pitch):
            GPIO.output(LED, GPIO.HIGH)
        else:
            GPIO.output(LED, GPIO.LOW)
        time.sleep(0.5)
except KeyboardInterrupt:
    print("結束監控")
finally:
    GPIO.cleanup()
```