微處理機系統與介面技術 LAB 8

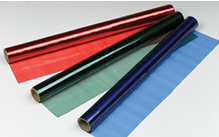
系所：電機 學號 :612415013 姓名：蕭宥羽

**<實驗器材>**

NUC 140 V2.0 開發板

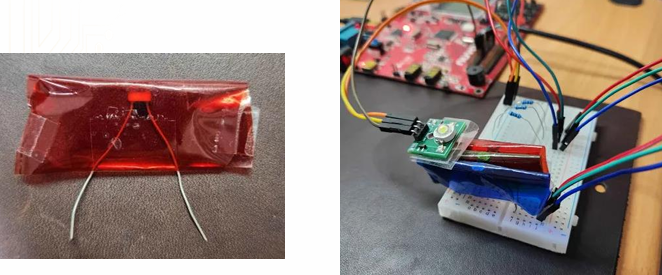


透明紙、光敏電阻、LED光源



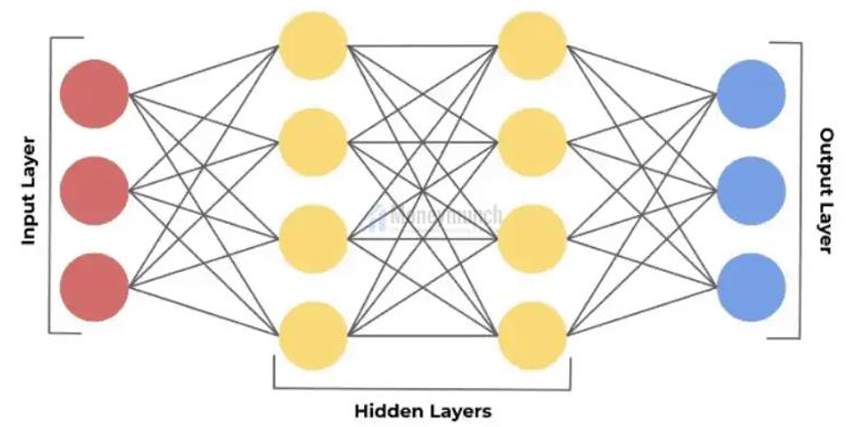
**<實驗過程與方法>**

* **實驗要求 :** 
  + Basic : Use machine learning to predict 4 outputs -red, blue, green and ambient light
  + Bonus : Use machine learning to predict 7 outputs -red, blue, green, magenta, orange, yellow and ambient light
* **實驗環境設置 :** 
  + Cover the photoresistor with cellophane
  + Align three photoresistors side by side
  + Attach the LED module above the three photoresistors
  + Connect the remaining circuitry (VCC, GND, ADC, Control Pin)

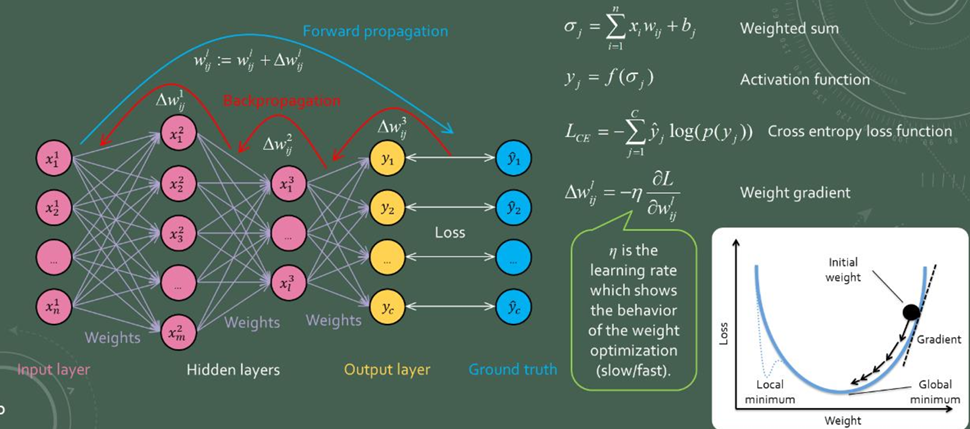


* **什麼是機器學習 :**

機器學習 (Machine Learning) 是一種能夠自動學習並從資料中改進表現的技術。像是這次 lab 中所使用的神經網路模型，透過模擬人類大腦的結構來進行學習。這個模型由輸入層、隱藏層和輸出層組成，輸入層接收資料特徵，隱藏層則負責提取資料中的重要模式與特徵，最後輸出層產生模型的預測結果。藉由不斷學習的過程，模型會根據實際結果與預測值的差異進行優化，透過調整權重和偏差來逐漸減少誤差。隨著訓練次數的增加，模型能更準確地捕捉資料的特徵與規律，達到穩定且可靠的預測效果。

****

* **類神經網路(ANN)的架構 :**

****

ANN一個模仿生物神經的系統，是一種讓電腦具備學習與推理判斷能力的演算法，

**ANN的基本架構**

➊ **節點 (Node)**：人工神經網路中的基本單位，每個節點模仿生物神經元的功能。節點會接收輸入訊號，經過加權計算和激活函數處理後，輸出結果到下一層節點。節點的輸出取決於**輸入值**、**權重**和**激活函數**的類型。

➋ **輸入 (Input)**：輸入是外部資料進入神經網路的起點，例如圖片的像素值、感測器的讀數或其他特徵向量。每個輸入值會被送入輸入層的節點，作為神經網路學習和預測的基礎。

➌ **權重 (Weight)**：權重是節點之間連結的重要參數，表示輸入對下一節點的重要性。每個權重在訓練過程中不斷被調整，以最小化預測誤差。權重的大小直接影響網路對輸入特徵的關注程度。

➍ **連結 (Connection)**：連結是網路中節點與節點之間的通道，負責將輸入、權重和偏差傳遞給下一節點。每條連結都有對應的權重值，並用於計算輸入數值在經過該連結後的輸出。

➎ **隱藏層 (Hidden Layer)**：隱藏層位於輸入層和輸出層之間，是網路的核心部分。隱藏層負責處理和提取輸入資料中的非線性特徵。隱藏層的節點會通過激活函數進一步增強網路表達能力，使其能夠處理複雜的資料模式。

➏ **輸出層 (Output Layer)**：輸出層是網路的最後一層，其節點數量通常與預測的類別數或目標值數量一致。輸出層將隱藏層的處理結果轉換為模型的預測結果，例如分類標籤、回歸值或機率分佈。

**ANN的學習過程**

**ANN的學習過程主要透過數據流動和權重優化來實現，分為以下幾個主要階段**

➊ **輸入資料**：將訓練數據輸入到類神經網路中，作為學習的起點

➋ **前向傳播**：資料從輸入層逐層向前傳遞到輸出層，生成模型的預測結果。

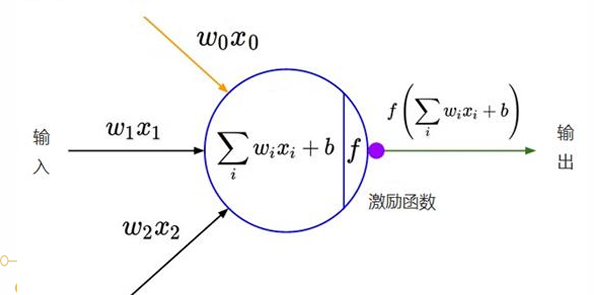
➌ **計算誤差**：根據輸出層的預測結果和實際值之間的差距，計算誤差大小

➍ **反向傳播**：將誤差從輸出層向回傳播至輸入層，計算每個權重對誤差的影響

➎ **更新權重**：根據計算的**誤差**和**梯度**，調整網路中的權重，以減少誤差並改進模型的預測能力

**細節說明**

➊ **前向傳播**

****

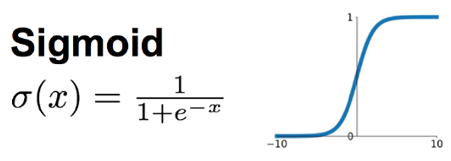
* **Weighted** **Sum**

每個節點接收來自前一層的輸入，這些輸入會被各自的權重() 乘以後相加，再加上偏差項()，這稱為加權求和，數學公式如下：

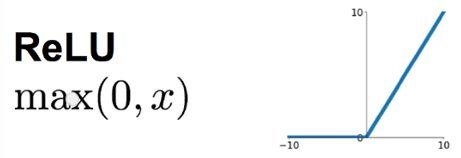
* + - * : 前一層節點的輸出（或直接來自輸入層的數據）。
      * : 每條連接對應的權重，表示輸入對該節點的影響程度。
      * : 偏差項，用於調整節點的輸出，增加模型的靈活性。
        + 避免輸入為零時的模型僵化 : 如果所有輸入 xi=0，沒有偏差項的情況下，節點的輸出總是 z=0。但加上偏差項 b，即使輸入為零，節點仍能輸出非零的值。
* **Activation Function**

加權求和的結果 z 是一個線性組合，若直接輸出，最終的輸出還是一個線性組合，網路將失去處理非線性問題的能力。因此，神經網路在每個節點使用activate function將線性輸出轉換為非線性輸出。常見activate function如下:

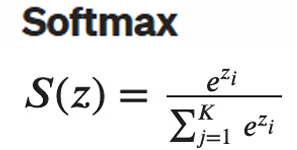
* + - * Sigmoid
        + 輸出範圍在 0 到 1 之間，常用於概率預測。
        + 缺點: 在極端值時，梯度趨近於 0，可能導致訓練緩慢。



* + - * ReLU
        + 當輸入為正時輸出為自身，否則輸出為 0，計算簡單且效率高。
        + 缺點: 可能出現「死亡 ReLU」現象，即某些神經元永遠輸出 0。



* + - * Softmax
        + 將多個輸出轉換為概率分佈，輸出值之和為 1，適用於多分類問題。



➋ **反向傳播**

反向傳播是神經網路訓練的核心過程，其目的是基於預測結果與實際目標之間的誤差來優化網路的參數（權重與偏差）。它主要分為三個步驟：**計算損失函數 (Loss Function)**、**優化 (Optimization Function)** 和 **更新權重 (Weight Update)**，其中**學習率 (Learning Rate)** 在更新權重的過程中扮演重要角色。

* + **損失函數 (Loss Function)**

Loss Function用於量化神經網路的預測結果()與實際值(y)之間的誤差，作為模型學習的指標。

* + - * + Mean Square Error, MSE

適用於回歸問題，計算預測值與實際值差異的平方平均。

* + - * + Mean Absolute Error, MAE

誤差取絕對值後的平均值，更強調偏差的幅度。

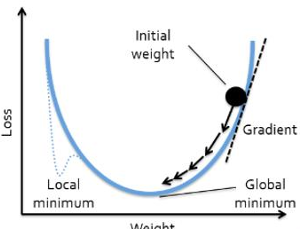
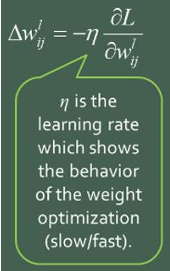
* + - * + Cross Entropy

適用於分類問題，計算預測的概率分佈與實際標籤之間的差異。

其中 是模型對第 c 類的預測概率， 是實際標籤。

* + **優化函數 (Optimization Function)**

Optimization Function負責通過損失函數的梯度來更新網路的參數，以最小化損失值。常見的優化方法包括：

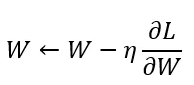
 

* + - * + 梯度下降法 (Gradient Descent, GD) - conceptual

如上圖所示，Gradient Descent會逐步沿著損失函數的梯度下降方向更新權重，直到達到局部最小值。數學式如下:

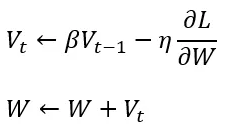
* ：表示權重 的更新量（每次更新的改變值）。
* η：學習率，決定更新步伐的大小。
* ​：損失函數 對權重 的梯度。
  + - * + 隨機梯度下降法 (Stochastic Gradient Descent, SGD)

SGD 是梯度下降的一種變體，它通過隨機選取一個訓練樣本來計算損失函數的梯度，並根據該梯度更新參數（權重）。這種方法避免了使用整個數據集計算梯度的高成本，從而加快了更新速度。然而，由於每次僅基於一個樣本，更新方向可能不穩定，會出現震盪的情況。



* + - * + Momentum

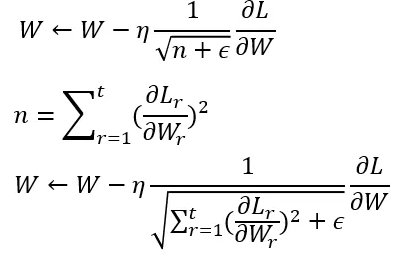
Momentum 是「運動量」的意思，此優化器為模擬物理動量的概念，在同方向的維度上學習速度會變快，方向改變的時候學習速度會變慢。



此方法引入了一個 Vt的參數，可視為「方向速度」，它與上一次的更新相關。當上一次梯度方向與本次相同時，|Vt| 會增大，代表學習速度變快；若方向不同，|Vt|會減小，更新幅度也會降低。動量因子 β（通常設定為 0.9）類似於空氣阻力或摩擦力，控制動量的累積與衰減。

* + - * + AdaGrad

AdaGrad是一種自適應學習率的優化方法，它為每個參數計算不同的學習率，根據歷史梯度信息調整更新幅度。對於Optimizer來說，learning rate相當的重要，太小會花費太多時間學習，太大有可能會造成發散

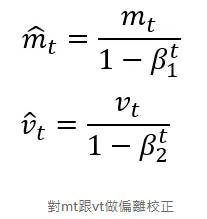
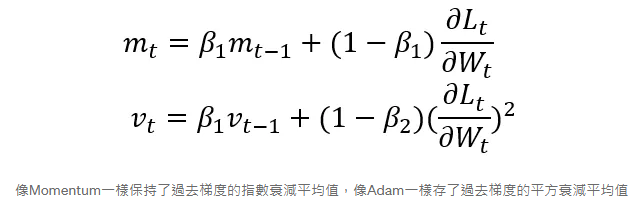


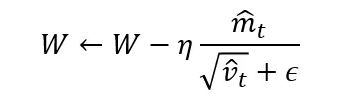
在AdaGrad中，η 乘上 1/√(n+ϵ) 再做參數更新，出現了一個n的參數，n為前面所有梯度值的平方和，利用前面學習的梯度值平方和來調整learning rate，ϵ 為平滑值，加上 ϵ 的原因是為了不讓分母為0，ϵ 一般值為1e-8

* + - * + Adam 優化器

Adam結合了 Momentum 和 AdaGrad 的優點

1. 使用 Momentum 追蹤一階動量（梯度均值），加速收斂。
2. 使用 AdaGrad 追蹤二階動量（梯度平方均值），自適應調整學習率。





Momentum 提供了方向穩定性，AdaGrad 提供了自適應學習率，而 Adam 結合二者，能夠在多數情況下提供高效和穩定的優化效果。

* + 學習率 (Learning Rate)

決定了梯度下降時每次更新權重的步伐大小。若學習率過大，模型可能跳過最佳解或不穩定；若學習率過小，訓練會變得緩慢甚至停滯。選擇合適的學習率能讓模型穩定且快速地收斂到最佳解。

**資料處理**

會將資料分為**training data**以及**testing data** (正常會分8:2 or 7:3)

* **Training Data**：用於訓練模型，讓模型學習數據特徵與輸出之間的關係。
* **Testing Data**：用於評估模型的性能，確保模型能在未見過的數據上進行準確的預測。

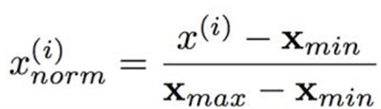
通常會做**Normalization / Standardization (正規化與標準化)**的動作

* **Normalization**

將數據縮放到 [0, 1] 或 [-1, 1] 的範圍內，使每個特徵的數值保持在統一的比例範圍內。

* + - * 使用場景 : 1. 特徵值範圍分布相差很大 2. 模型對範圍敏感
      * 如果不做會發生什麼事情 !

1. 特徵值較大的數據主導了模型，其他特徵可能被忽略。
2. 斂速度慢: 對於基於梯度下降的模型，數值差異大會導致梯度更新幅度不一致 ，使模型訓練變得不穩定或收斂過慢。

****

* **Standardization**

將數據調整為均值為 0、標準差為 1。數據分佈呈標準正態分佈，數據點相對於均值的位置（z-score）可以更清晰地比較。

* + - * 使用場景 :

1. 數據分佈無上下界
2. 對梯度敏感的模型

使用基於梯度優化的模型（如線性回歸、邏輯回歸、神經網絡），標準化使得每個特徵的梯度更新幅度更均勻，有助於加速收斂。

* + - * 如果不做會發生什麼事情 !

1. 梯度更新幅度不一致：

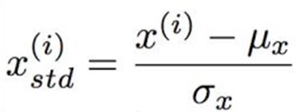
特徵值的範圍差異會導致梯度方向失衡，使模型在優化過程中震盪或過於緩慢。

1. 模型訓練不穩定 :

特別是對於需要多層運算的神經網絡，沒有標準化的輸入可能導致激活函數進入飽和區域（梯度消失）。

1. 影響模型泛化能力：

特徵值尺度差異大會增加模型的偏倚，降低其在測試數據上的表現。



<Mian function code>

**這邊我會先介紹watch dog timer的一些重要function後續再說明程式流程**

1. SYS\_UnlockReg()

解鎖受保護的寄存器，允許對系統控制暫存器進行修改，NUC140對重要register（如 **WTCR**）實施寫保護機制，防止意外修改。所以我們要用SYS\_UnlockReg()解鎖保護暫存器，使其暫時允許修改。

1. WDT\_Open(WDT\_TIMEOUT\_2POW14, WDT\_RESET\_DELAY\_1026CLK, TRUE, FALSE);

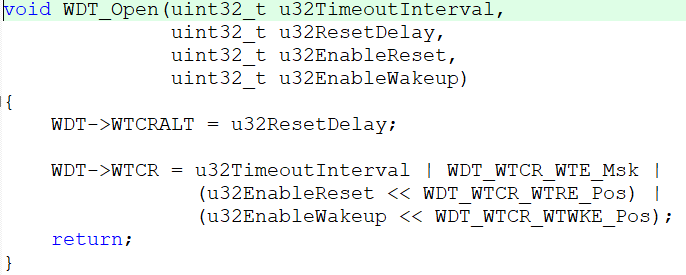
* **u32TimeoutInterval** :設置看門狗定時器的溢出時間間隔，寫入WTIS暫存器

WDT\_TIMEOUT\_2POW14: 設置看門狗的超時時間為 2^14 個時鐘週期

* **u32ResetDelay**: 配置看門狗重置前的延遲時間

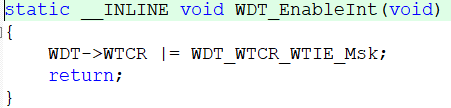
WDT\_RESET\_DELAY\_1026CLK: 設置超時後延遲 1024 個時鐘週期再觸發重置

* **u32EnableReset** : 設置是否啟用看門狗超時後的系統重置功能
* **u32EnableWakeup** : 設置看門狗是否能在低功耗模式下喚醒 CPU



1. WDT\_EnableInt();

* 啟用看門狗中斷功能
* 設置 WTCR 中的中斷使能位（如 WTIE），當看門狗超時時，觸發中斷請求（IRQ）



1. NVIC\_EnableIRQ(WDT\_IRQn); -啟用看門狗的中斷服務
2. WDT\_IRQHandler

* **if(WDT\_GET\_TIMEOUT\_INT\_FLAG() == 1)**

WDT\_GET\_TIMEOUT\_INT\_FLAG() 用於檢查看門狗的超時中斷旗標（WTIF）。如果返回值為 1，表示看門狗超時中斷已觸發，需要處理。



* **WDT\_CLEAR\_TIMEOUT\_INT\_FLAG();**  -清除看門狗的超時中斷旗標
* **WDT\_RESET\_COUNTER(); -重置看門狗計數器(餵狗)**
  + **WDT->WTCR & ~(WDT\_WTCR\_WTIF\_Msk | WDT\_WTCR\_WTWKF\_Msk | WDT\_WTCR\_WTRF\_Msk)**

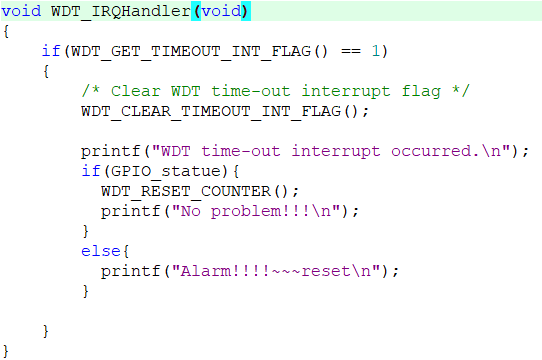
使用位元運算清除以下旗標（設為 0）

* WDT\_WTCR\_WTIF\_Msk（Timeout Interrupt Flag）：表示超時中斷已觸發
* WDT\_WTCR\_WTWKF\_Msk（Wakeup Flag）：表示看門狗喚醒事件已觸發
* WDT\_WTCR\_WTRF\_Msk（Reset Flag）：表示看門狗觸發了系統重置
* **| WDT\_WTCR\_WTR\_Msk**
* 重置計數器位（Watchdog Timer Reset）
* 將該位設置為 1，指示硬體重置看門狗計數器



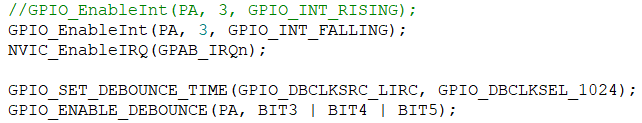
* **GPIO\_statue? 重置看門狗計數器 : 不動作**

根據目前狀態判斷是否要讓WDT重啟系統

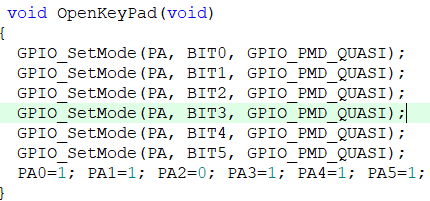


**這邊我接著介紹GPIO的一些重要function**

1. 啟用GPIO中斷 -PA3 -下沿觸發 -防抖動

****

1. GPIO PA3設為雙通道模式這樣PA3平時就會是high， PA2設為接地，當第一顆按鍵被按下時 PA3就會是low，這樣看PA3狀態就可以知道第一顆案件是否被按下

****

1. GPAB\_IRQHandler

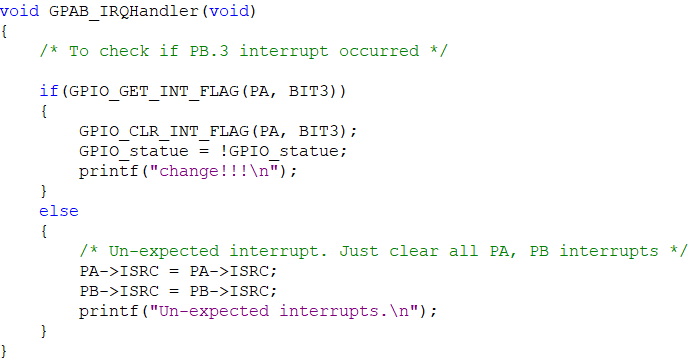
* if(GPIO\_GET\_INT\_FLAG(PA, BIT3))

檢查 PA3 引腳是否設置了中斷旗標

* GPIO\_CLR\_INT\_FLAG(PA, BIT3);

清除 PA3 引腳的中斷旗標，表示這次中斷已被處理，防止重複觸發。

* 如果按鈕被按下就反轉狀態 GPIO\_statue = !GPIO\_statue;

****

**整體程式流程**

1. **按鍵與狀態切換**：

* 按下按鍵觸發 GPIO 中斷，反轉 GPIO\_statue。
* GPIO\_statue 的值模擬系統的正常或異常狀態。

1. **看門狗監控**：

* 若 GPIO\_statue = TRUE，系統正常，看門狗被重置。
* 若 GPIO\_statue = FALSE，看門狗進入超時並觸發中斷。

1. **看門狗的自動恢復**：

* 如果異常狀態持續，看門狗會觸發系統重置，恢復正常運行。

**這樣就可以達成Let emergency alarm can rese、Use WDT reset the emergency alarm when the state is not safe這樣的要求了**

<過程中遇到的困難>

在這次 Lab 的實作中，困難點在於對看門狗的理解不夠深入。一開始雖然知道它的功能是防止系統卡死，但在實際配置暫存器時，像超時設定（WTIS）、重置延遲（WTR）等細節完全不熟悉，這邊花了許多時間去了解，花費不少精力。

<心得與收穫>

在這次的 Watchdog 實驗中，看似簡單的系統監控功能，實際實作過程中卻讓我深刻體會到，要成功實現 Watchdog 的功能，需要對其運作原理及硬體配置有一定的掌握。例如，超時條件的設定（WTIS）、重置延遲的配置（WTR）、以及中斷處理邏輯等，每一個步驟都需要細緻地設計，才能讓 Watchdog 在系統中穩定地發揮其監控作用。

在實驗過程中，我學習到如何有效配置 Watchdog 的參數，包括選擇合適的超時時間、啟用中斷功能，以及正確處理中斷旗標以確保系統能正常運行。我理解了 Watchdog 中斷的核心作用，並掌握了如何在中斷服務例程中結合系統狀態（例如 GPIO 的輸入）來決定是否進行「餵狗」，從而避免系統被誤重置。此外，我還學會了如何處理 Watchdog 與其他模組（如 GPIO 按鍵中斷）的協作，讓系統狀態能夠準確反映實際情況，進一步提升系統的穩定性和可靠性。

同時，我也體會到在 Watchdog 的實作過程中，暫存器配置和中斷管理是非常關鍵的部分。從超時條件的設定，到中斷觸發後清除旗標，直到決定是否重置計數器，每一個操作都需要精確執行，否則可能會導致系統無法正確進行自動監控或頻繁地重置。這些細節讓我理解到，Watchdog 不僅僅是一個簡單的防護機制，更是系統可靠性設計中的核心模組之一。

這次實驗給了我寶貴的經驗，使我對嵌入式系統中的 Watchdog 模組有了更深刻的認識，也學會了如何在硬體與軟體之間進行協調，實現穩定、可靠的系統監控功能。這些經驗不僅加強了我對 Watchdog 的理解，也增強了我在嵌入式系統開發中的實作能力，尤其是在面對系統穩定性與錯誤恢復需求時，能夠設計出更高效的解決方案。