# 反應遊戲開發文件

## 目錄

1. 概述
2. 系統需求
3. 硬體設置
4. 遊戲流程
5. 主要功能說明
6. 程式結構
7. 模組詳解
8. 數學運算詳解

## 1. 概述

此遊戲透過玩家對顏色變化的快速反應進行挑戰。玩家需要在顏色為綠色時拍手，並在其他顏色顯示期間保持安靜。遊戲包含多層次的難度設定，並設有高分紀錄系統。

## 2. 系統需求

### 硬體需求

* Arduino 開發板 (如 Arduino Uno)
* RGB LED (紅、綠、藍)
* 壓電蜂鳴器
* 按鈕
* 聲音感測器 (如麥克風模組)
* 顯示三色 LED 的三個輸出腳位
* EEPROM（內建於 Arduino 中）

### 軟體需求

* Arduino IDE
* EEPROM 函式庫

## 3. 硬體設置

1. **RGB LED**：
   * 藍色接至 PIN 9
   * 綠色接至 PIN 10
   * 紅色接至 PIN 11
2. **蜂鳴器**：
   * 接至 PIN 6
3. **按鈕**：
   * 接至 PIN 7，使用內建拉高電阻 (INPUT\_PULLUP)
4. **聲音感測器**：
   * 模擬輸出接至 A0 腳位
5. **LED 指示燈**：
   * 三個 LED 分別接至 PIN 2、3、4

## 4. 遊戲流程

1. **初始化階段**：遊戲開始時啓動當前等級與高分。
2. **遊戲開始**：按下按鈕後進入等待顯示顏色的狀態。
3. **顏色顯示**：隨機顯示紅色、綠色或藍色。
4. **玩家反應**：
   * 綠色：玩家需拍手。
   * 紅色/藍色：玩家需保持安靜。
5. **獎勵與懲罰**：
   * 正確反應可得分，錯誤反應遊戲結束。
6. **升級**：達到特定分數後自動升級，增加遊戲難度。
7. **遊戲結束**：顯示最終分數，更新高分後重置遊戲。

## 5. 主要功能說明

### 5.1 顯示隨機顏色

隨機選擇 RGB 中的紅、綠、藍顏色，並點亮對應的 LED。

void setRandomColor() {

int color = random(0, 3);

switch(color) {

case 0: // 藍色

digitalWrite(rgbPins[0], HIGH);

break;

case 1: // 綠色

digitalWrite(rgbPins[1], HIGH);

break;

case 2: // 紅色

digitalWrite(rgbPins[2], HIGH);

break;

}

}

### 5.2 拍手檢測

根據聲音感測器的輸出值，判斷玩家是否拍手。

bool currentSoundState = analogRead(soundSensorPin) > 500;

if (currentSoundState && !previousSoundState) {

// 檢測到拍手

}

### 5.3 分數與等級

* 每次正確反應可增加分數。
* 累積分數達到升級門檻後進入下一等級。

if (score >= pointsPerLevel \* level) {

level++;

updateLevel();

}

### 5.4 高分儲存

使用 EEPROM 儲存並讀取最高分。

EEPROM.get(highScoreAddress, highScore);

EEPROM.put(highScoreAddress, highScore);

## 6. 程式結構

### 全域變數

* **按鈕與 LED 腳位**：定義所有輸出與輸入腳位。
* **遊戲狀態**：使用枚舉型態記錄當前遊戲狀態。
* **計時變數**：追蹤顯示顏色與反應時間。
* **分數與等級**：記錄玩家分數與當前等級。

### 函式

* setup()：初始化硬體與遊戲參數。
* loop()：主循環，負責遊戲邏輯。
* setRandomColor()：設定隨機顏色。
* giveReward()：依反應時間給予獎勵。
* gameOver()：結束遊戲。
* resetGame()：重置遊戲。

## 7. 模組詳解

### 7.1 按鈕長按檢測

若玩家按住按鈕超過 5 秒，遊戲將重置。

if (currentTime - buttonPressStartTime >= longPressDuration) {

resetGame();

}

### 7.2 遊戲難度調整

隨等級提升，減少反應與顏色顯示時間。

reactionTime = max(baseReactionTime - (level - 1) \* reactionTimeReductionPerLevel, minReactionTime);

## 8. 數學運算詳解

### 8.1 隨機顏色選擇

使用 random(0, 3) 生成隨機數值，對應紅、綠、藍顏色：

* random(0, 3) 回傳 0 到 2 的整數，對應：
  + 0: 藍色
  + 1: 綠色
  + 2: 紅色

### 8.2 分數與等級計算

玩家得分後進行判斷是否達到升級條件：

* 分數公式：
* if (score >= pointsPerLevel \* level) {
* level++;
* }

其中：

* + pointsPerLevel 為每等級所需的分數。
  + level 為當前等級。

### 8.3 反應與顏色顯示時間調整

隨等級升高減少顯示時間與反應時間：

* **反應時間計算公式**：
* reactionTime = baseReactionTime - (level - 1) \* reactionTimeReductionPerLevel;
* reactionTime = max(reactionTime, minReactionTime);
  + 初始反應時間：baseReactionTime
  + 每升一級減少：reactionTimeReductionPerLevel
  + 最小反應時間限制：minReactionTime
* **顏色顯示時間計算公式**：
* currentColorDisplayTime = baseColorDisplayTime - (level - 1) \* colorDisplayReductionPerLevel;
* currentColorDisplayTime = max(currentColorDisplayTime, minColorDisplayTime);
  + 初始顏色顯示時間：baseColorDisplayTime
  + 每升一級減少：colorDisplayReductionPerLevel
  + 最小顏色顯示時間限制：minColorDisplayTime

### 8.4 拍手反應時間

玩家拍手時間與顯示顏色的時間差：

* 公式：
* userReactionTime = clapTime - lastStateChange;
  + clapTime 為檢測到拍手的時間戳記。
  + lastStateChange 為當前顏色顯示開始的時間戳記。

### 8.5 獎勵邏輯

根據玩家的反應時間給予不同等級的獎勵：

* 獎勵條件：
* if (reactionTimeUser < 1000) {
* score += 3;
* } else if (reactionTimeUser < 2000) {
* score += 2;
* } else {
* score += 1;
* }
  + 快速反應：得 3 分。
  + 中速反應：得 2 分。
  + 慢速反應：得 1 分。