壹、前言

一、研究動機

生活在科技快速更新的年代，誰掌握技術並讓技術扎根而延伸出科技更新就會是該業內新一代的領頭羊。近年來，由於晶片製程不斷的突破瓶頸，讓相關需求的產業更是蓬勃發展。機器人如此，AI 產業更是整合各項應用的綜合產物。而機器人從早期地上走的、爬的、滾的，到目前能在天空飛的、能在水裡面潛航的，都離不開那一塊小小的晶片。無人機技術發展迅速，學校也在我們相關課程應變未來發展的需求下引進「足球無人機」供我們學習體驗。然而受限正式競賽場地設備動輒高達數萬元，對於一般班級推廣而言，硬體門檻太高了。

而為了克服資源不足的先天限制，我們決定好好運用實習課所學，即便有些技術可能是還未碰過，但我們相信靠著我們後天的努力一定有機會去突破並完成我們實際場域所需。本著以最少的資源與最大的熱忱嘗試以「自製取代購置」的精神，將足球機競賽所需的周邊設備納入實作主題。我們希望透過親手設計遊戲規則並建置相關裝置，不僅能解決硬體設備短缺的問題，更能在過程中實踐解決問題的連貫體驗，將理論課所學延伸至實際的競賽情境中。

二、研究目的

(一) 了解超音波感測器工作原理。
(二) 了解顏色感測器工作原理。
(三) 運用超音波與顏色感測器偵測/判斷實際可計分的隊伍。
(四) 運用七段顯示器計數得分狀況。
(五) 運用 Arduino 平台完成韌體功能要求與動作驗證。

貳、文獻探討

一、根據 HC-SR04 數據手冊可以得知它「**是一款常用於機器人和自動化專案中測量距離的感測器。**其**提供 2cm 至 400cm 的非接觸式測量功能，測距精度可達 3mm**」(Manager，2024)。對於測量足球機接近時的測距功能並做出反應的動作要求將會非常實用。

二、根據 TCS3200 規格書的資訊中所提其具備「**高解析度光至頻率的轉換、可程式化顏色全亮程輸出、具備省電與關斷模式功能、可直接與微控制器通訊**」(Gizmo Mechatronix Central，2015)可以得知其對於紅、藍、綠的顏色辨識率在 50KHz 以下之精準度極高，且其顏色數值能直接讓 Arduino 讀取，定能跟超音波感測器搭配同步判斷並做出合理的反應。

三、根據書本專文介紹 Arduino 的部分描述其「**配備一組數位/類比輸入/輸出引腳，可連接到各種擴展板。可用於感測器和執行器間互動的裝置**」(施士文，2022)。如此能

簡易上手的設計平台非常適合用於編程實作平台。

參、研究方法

本次實作是透過任何物體接近超音波距離低或等於 30 公分且經顏色感測器辨識物體確為設定的顏色時，一位元七段顯示器能做出+1 的動作。反之，則一位元七段顯示器無任何動作。

一、研究流程

圖 1 研究流程圖

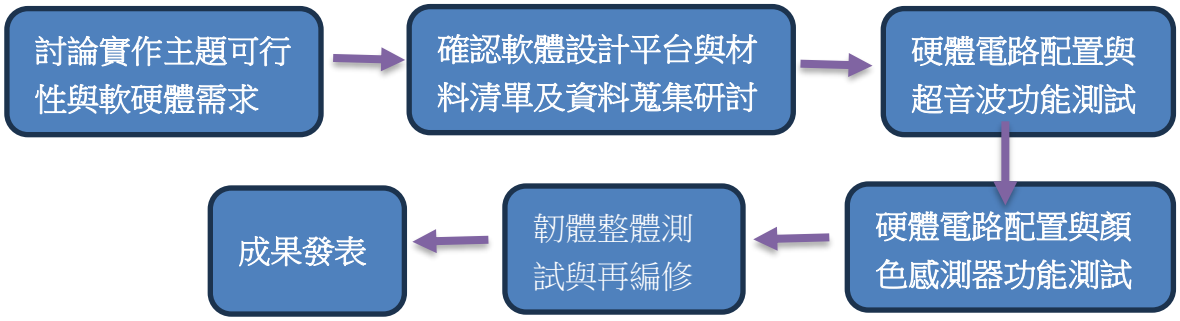

圖 1 研究流程圖資料來源：作者自行繪製

二、材料說明

(一) 人類耳朵能聽到聲波的頻率介於 20Hz~20KHz，因此當聲波頻率 ＞20KHz 時，就被定義為超音波。聲音在空中傳播速度大約是 331+0.6t m/s，且傳播速度會因溫度的差異而有傳播快或慢之別。傳播速度一般與溫度成正比關係，當溫度越高，則傳播速度越快。而市面上較常使用的超音波感測器如下圖所示，其聲波頻率約為 40KHz。

圖 2 超音波感測器

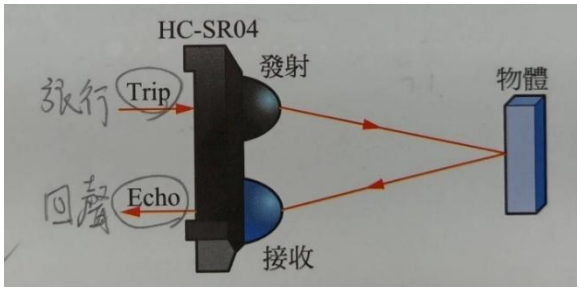

圖 2 超音波感測器圖資料來源：施士文（2022）。Arduino 微電腦應用實習。台北市：台科大圖書股份有限公司。

(二) 顏色感測器可準確與快速地識別顏色並檢測出當前待測物體是紅光或綠光或藍光。它利用光照射到物體讓內部光電二極管檢測反射光波長的強度並測量反射光以找到紅色、綠色和藍色數據值並使用光學傳感可以準確讀取色調並告訴系統它看到的顏色。而市面上較常使用的顏色感測器如下圖所示。

圖 3 顏色感測器

圖 3 顏色感測器圖資料來源：台灣物聯科技。GY-31 TCS3200 顏色感測器 顏色識別模組顏色感應傳感器。

(三) 七段顯示器是常用於電路中顯示數字的電子元件，其內部主要是將 8 只 LED 組合在一起，再將共同的陽極或陰極連到同一個點而形成共陽極或共陰極的七段顯示器。欲使共陽極的七段顯示器能夠顯示需求數字時，需讓該數字對應的腳位於編程時設定為低態或接地，反之若為共陰極的七段顯示器時，則需讓該數字對應的腳位於編程時設定為高態或接正電位。市面上較常使用的一位元七段顯示器如下圖所示。

圖 4 一位元七段顯示器

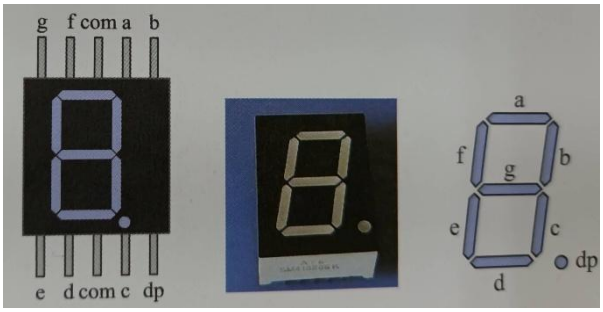

圖 4 一位元七段顯示器圖資料來源：施士文（2022）。Arduino 微電腦應用實習。台北市：台科大圖書股份有限公司。

三、韌體動作要求

(一) 超音波感測器

偵測靠近的物體是否小或等於 30 公分。若有，則送出高態訊息。

(二) 顏色感測器

偵測靠近的物體是否為預先設定的顏色。若是，則送出高態訊息。

(三) 一位元七段顯示器

當超音波感測器與顏色感測器皆送回高態訊息時，啟動一位元七段顯示器做+1的動作。

肆、研究分析與結果

一、各別材料分析

(一) 超音波感測器：負責偵測物體靠近數值並回傳給 Arduino。
(二) 顏色感測器：負責偵測靠近的物體顏色值，並回傳給 Arduino。
(三) 一位元七段顯示器：當 Arduino 條件判斷成立時，負責顯示分數值。

二、各別材料電路裝配與功能測試結果

1. 超音波感測器搭配一位元七段顯示器做韌體測試物體靠近小或等於 30 公分時之計數功能。

圖 5 測試硬體（顏色感測器）

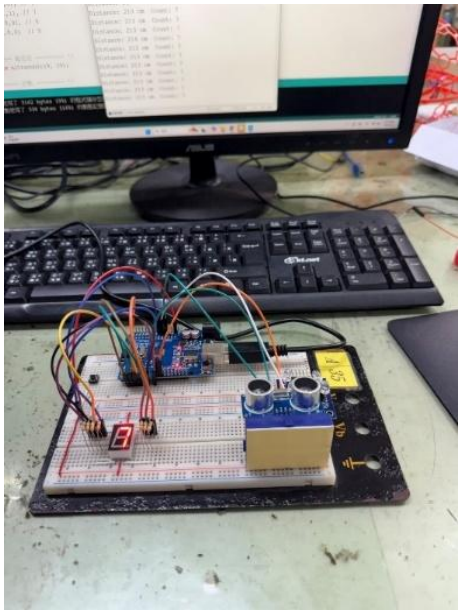

圖 5 測試硬體資料來源：作者自行拍攝

圖 6 測試硬體（顏色感測器）

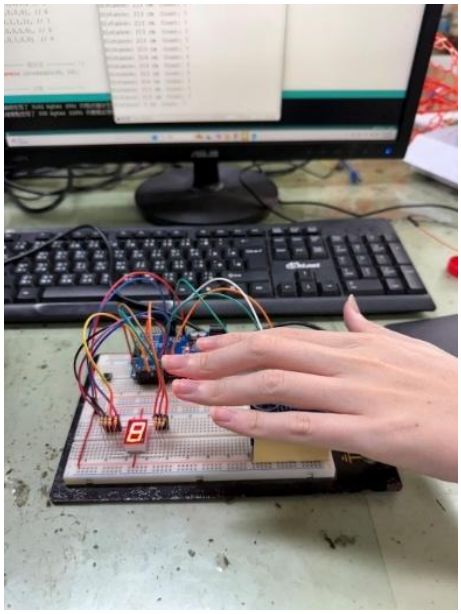

圖 6 測試硬體資料來源：作者自行拍攝

2. 顏色感測器搭配一位元七段顯示器做韌體測試物體靠近時，其顏色值同預設值時之計數功能。

圖 7 測試硬體（一位元七段顯示器）	圖 8 測試硬體（一位元七段顯示器）
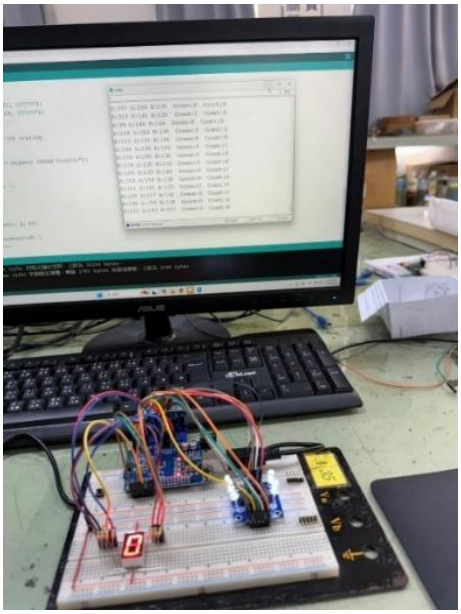	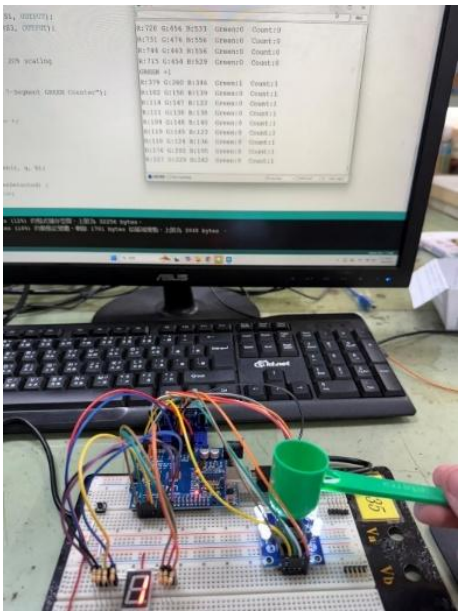
圖 7 測試硬體資料來源：作者自行拍攝	圖 8 測試硬體資料來源：作者自行拍攝

3. 韌體整體全功能測試。

圖 9 整體測試	圖 10 整體測試
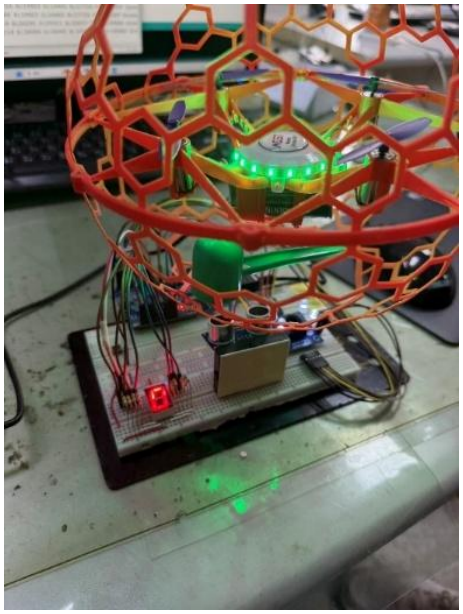	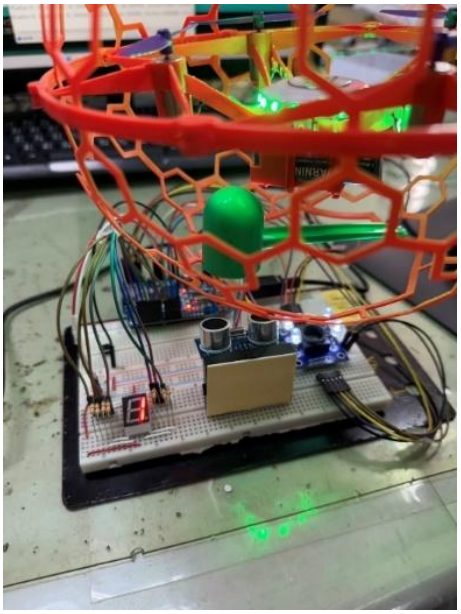
圖 9 整體測試資料來源：作者自行拍攝	圖 10 整體測試資料來源：作者自行拍攝

三、電路配置

各項材料配置於麵包板上。

圖 11 電路裝配圖

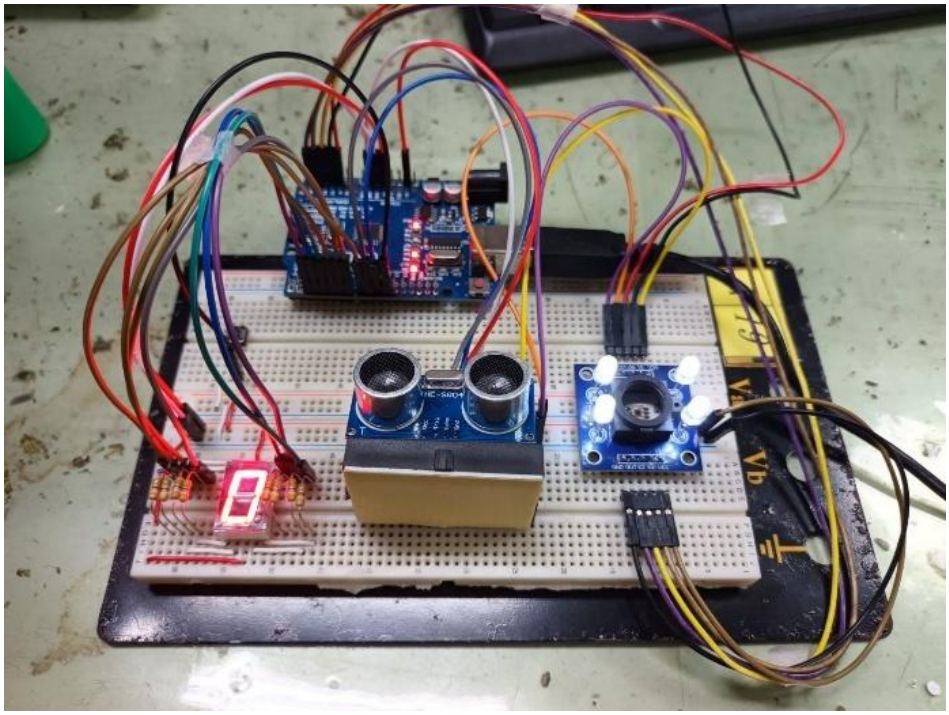

圖 11 電路裝配圖資料來源：作者自行拍攝

四、程式設計

圖 12 程式設計

```
#include <Ultrasonic.h>

/* ========= 七段顯示器（共陽極） ========= */
int segPins[7] = {2, 3, 4, 5, 6, 7, 8};
byte segNumber[10][7] = {
  {0,0,0,0,0,0,1},{1,0,0,1,1,1,1},{0,0,1,0,0,1,0},
  {0,0,0,0,1,1,0},{1,0,0,1,1,0,0},{0,1,0,0,1,0,0},
  {0,1,0,0,0,0,0},{0,0,0,1,1,1,1},
  {0,0,0,0,0,0,0},{0,0,0,0,1,0,0}
};
```

圖 12 程式設計圖資料來源：作者自行拍攝

圖 13 程式設計

```
/* ========= 超音波 ========= */
Ultrasonic ultrasonic(9, 10);

/* ========= TCS3200 ========= */
#define S0 A1
#define S1 A2
#define S2 A3
#define S3 A4
#define OUT A5

/* ========= 按鈕 ========= */
#define SW 11
int count = 0;
```

圖 13 程式設計圖資料來源：作者自行拍攝

圖 14 程式設計

```
/* ========== 顏色數值 ========== */
unsigned long lastRed = 0, lastGreen = 0, lastBlue = 0;

/* ========== 狀態機 ========== */
enum State { IDLE, COUNTING };
State currentState = IDLE;

/* ========== 顏色穩定判斷 ========== */
int greenStableCount = 0;    // 連續兩次綠色計數
const int GREEN_THRESHOLD = 2;

/* ========== 七段顯示 ========== */
void displayNumber(int num) {
  for (int i = 0; i < 7; i++) {
    digitalWrite(segPins[i], segNumber[num][i]);
  }
}

/* ========== 讀取顏色 ========== */
unsigned long readColor() {
  unsigned long t = pulseIn(OUT, LOW, 100000);
  if (t == 0) return 30000;
  return t;
}

void readRGB() {
  long r = 0, g = 0, b = 0;
  for (int i = 0; i < 10; i++) { // 10 次平均
    digitalWrite(S2, LOW);  digitalWrite(S3, LOW);   r += readColor(); // R
    delay(2);
    digitalWrite(S2, HIGH); digitalWrite(S3, HIGH);  g += readColor(); // G
    delay(2);
    digitalWrite(S2, LOW);  digitalWrite(S3, HIGH);  b += readColor(); // B
    delay(2);
  }
  lastRed = r / 10;
  lastGreen = g / 10;
  lastBlue = b / 10;
}
```

圖 14 程式設計圖資料來源：作者自行拍攝

圖 15 程式設計

```
/* ========== 顏色判斷 ========== */
String detectColor() {
  if (lastRed > 30000 || lastGreen > 30000 || lastBlue > 30000) return "OUT";
  if (lastRed < 200 && lastGreen < 200 && lastBlue < 200) return "OUT";
  unsigned long minVal = min(lastRed, min(lastGreen, lastBlue));
  unsigned long diffGreen = 800;
  unsigned long diffRed   = 400;
  unsigned long diffBlue  = 400;
  if (minVal == lastGreen &&
      lastGreen + diffGreen < lastRed &&
      lastGreen + diffGreen < lastBlue)
    return "GREEN";
```

圖 15 程式設計圖資料來源：作者自行拍攝

圖 16 程式設計

```
  if (minVal == lastRed &&
      lastRed + diffRed < lastGreen &&
      lastRed + diffRed < lastBlue)
    return "RED";

  if (minVal == lastBlue &&
      lastBlue + diffBlue < lastRed &&
      lastBlue + diffBlue < lastGreen)
    return "BLUE";
  return "OTHER";
}

/* ========= setup ========= */
void setup() {
  Serial.begin(9600);

  for (int i = 0; i < 7; i++) {
    pinMode(segPins[i], OUTPUT);
    digitalWrite(segPins[i], HIGH);
  }
  pinMode(S0, OUTPUT); pinMode(S1, OUTPUT);
  pinMode(S2, OUTPUT); pinMode(S3, OUTPUT);
  pinMode(OUT, INPUT);

  digitalWrite(S0, HIGH);
  digitalWrite(S1, LOW);

  pinMode(SW, INPUT_PULLUP);

  displayNumber(count);
}

/* ========= loop ========= */
void loop() {
  // ---- Reset ----
  if (digitalRead(SW) == LOW) {
    count = 0;
    greenStableCount = 0;
    currentState = IDLE;
    displayNumber(count);
    Serial.println("RESET");
    delay(300);
  }
```

圖 16 程式設計圖資料來源：作者自行拍攝

圖 17 程式設計

```
// ---- 先偵測距離 ----
long distance = ultrasonic.read();
bool inRange = (distance <= 30);

String color = "OUT";

// ---- 距離在範圍內才讀顏色 ----
if (inRange) {
  readRGB();
  color = detectColor();
} else {
  greenStableCount = 0; // 不在範圍重置
}
```

圖 17 程式設計圖資料來源：作者自行拍攝

圖 18 程式設計

```
// ---- 狀態機 ----
switch(currentState) {
  case IDLE:
    if(color == "GREEN") {
      greenStableCount++;
      if(greenStableCount >= GREEN_THRESHOLD) {
        if(count < 9) count++;
        displayNumber(count);
        Serial.println("GREEN +1");
        currentState = COUNTING;
      }
    } else {
      greenStableCount = 0;
    }
    break;

  case COUNTING:
    // 綠色移開或距離超出範圍 → 回到 IDLE
    if(color != "GREEN" || !inRange) {
      greenStableCount = 0;
      currentState = IDLE;
    }
    break;
}

// ---- Debug ----
Serial.print("State:");
Serial.print(currentState);
Serial.print(" Dist:");
Serial.print(distance);
Serial.print(" R:");
Serial.print(lastRed);
Serial.print(" G:");
Serial.print(lastGreen);
Serial.print(" B:");
Serial.print(lastBlue);
Serial.print(" Color:");
Serial.print(color);
Serial.print(" GreenCount:");
Serial.print(greenStableCount);
Serial.print(" Count:");
Serial.println(count);

delay(300);
}
```

圖 18 程式設計圖資料來源：作者自行拍攝

伍、研究結論與建議

一、結論

實驗使用 TCS3200 顏色感測器、HC-SR04 超音波感測器搭配 Arduino 進行綠色物體之辨識，並以七段顯示器顯示計數結果。實驗過程中，將不同顏色物體依序放置於感測器前方測試。初步測試完成後，再用球型無人機進行測試並同時觀察數據，若條件成立，則觀察七段顯示器是否有加 1。此外，本程式綠色判斷方式採用「比例式判斷法」，其中綠色頻率值為三色中最小，並比較紅色與藍色相對於綠色的比例關係。相較於單純使用固定數值差距，此方法能降低環境亮度改變對判斷結果的影響，使系統在不同光線條件下仍能維持一定的穩定度。

二、建議

(一) 增加超音波感測器可感測之距離限制以提升穩定度

當物體距離感測器過遠時，容易因反射光不足而影響判斷結果。

(二) 加入啟動時的自動校正機制

由於環境光源與感測器個體差異會影響量測數值，建議啟動時進行白色與黑色校正，將量測結果轉換為相對比例值，降低環境光變化所造成的誤判問題。

(三) 增加多次取樣與穩定判斷

目前以單次顏色判斷作為加 1 依據，建議未來可要求連續多次判斷皆為綠色時，才進行計數，以避免瞬間雜訊造成誤觸發。

陸、參考文獻

1. HC-SR04 Datasheet – Ultrasonic Sensor。HC-SR04 數據手冊–超音波感測器。https://datasheetgo.com/hc-sr04-ultrasonic-sensor/ 。
2. Gizmo Mechatronix Central。GY31 COLOR module。https://usermanual.wiki/Document/GY31ColorModule.583621932.pdf。
3. 施士文（2022）。Arduino 微電腦應用實習。台北市：台科大圖書股份有限公司。
4. 台灣物聯科技。GY-31 TCS3200 顏色感測器 顏色識別模組 顏色感應傳感器。https://www.taiwaniot.com.tw/product/tcs3200%E6%99%B6%E7%89%87-%E9%A1%8F%E8%89%B2%E6%84%9F%E6%B8%AC%E6%A8%A1%E7%B5%84/。