# 106學年度臺中市公私立中小學科學展覽會

# 作品說明書

# 科　　別：生活與應用科學科（一）機電與資訊

# 組　　別：國中組

# 作品名稱：PID循線控制

# 關 鍵 詞： 、 、 （最多3個）

# 編 號：

# 摘要

PID為Proportional–Integral–Derivative（比例、積分、微分）的縮寫，嘗試將PID控制應用到現今熱門的循線車上，挑戰循線感應器「一定要兩個以上」的限制。

# 壹、研究動機

學校買了一批Mbot循線車，它由兩顆紅外線感應器判斷路面明暗（黑白），以此傳送數據供程式使用。但這種方法只能感測到黑黑、黑白、白黑、白白四種狀況，無法準確判斷車子該如何行動。我們希望能做出更精確的循線車，因而開始了這次研究。

# 貳、研究目的

一、利用單感應器控制循線車

二、利用PID優化自製循線車

三、研究調整各項參數對循線能力的影響

# 參、研究設備及器材

# Arduino Mega 2560: 車子主控板

# L298N :馬達驅動板

# TCRT5000: 循線感應器

# LCD I2C :液晶顯示器

# 9V電池: 電源

# 直流減速馬達：移動裝置

# 麵包版：方便連接電路

# 麵包線：電線

# 智高積木：多用途，測量工具，防護工具

# 七段顯示器：顯示數字 \*\*已拆卸\*\*

# 74hc595：移位暫存器，搭配七段顯示器使用

# 剝線箝：剝開電線外皮

# 按鈕：多用途

# Arduino IDE：程式開發、上傳環境

# Git：版本控制

# Scratch：模擬器

# Fritzing：電路圖繪製

# Libre Office：報告編輯

# 車子電路配置圖：

# 

# 肆、研究過程

### 程式邏輯：

我們只使用一個循跡感測器，並且讓車子走在**黑白交界處**。感測器有一個偵測範圍，當範圍內黑白面積相當，接收到的值就會是黑白線的平均值，如果偏向黑線，數值就會比較高，反之則比較低，那車子可以藉由數字的高低來判斷左右修正，即可讓車子走在中央。

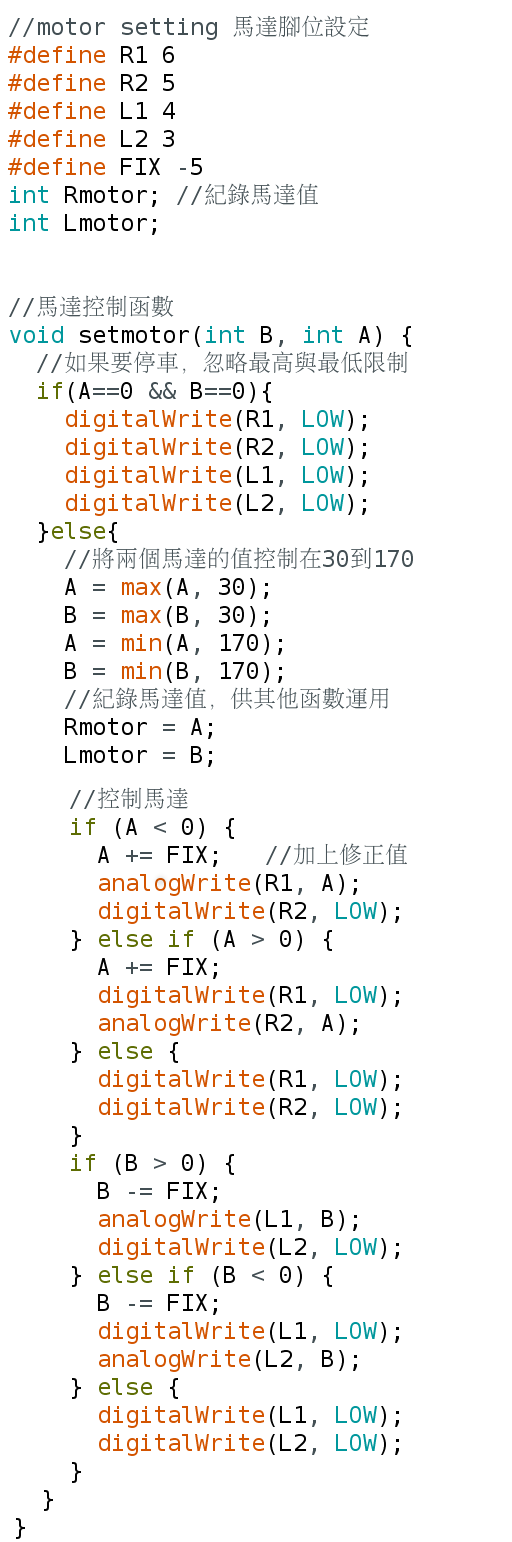
***（補圖）***

一、車體控制程式

**1.馬達驅動：**

將馬達接上馬達驅動版，電源以9V電池驅動，Arduino板接4條電線到驅動版上，每兩條電線控制一顆馬達。當電線一條為高電壓、一條為低電壓，馬達將正轉，反之則反轉，停車則是兩條電壓相當。轉速由高電壓電線的電壓控制，Arduino板的pwn類比輸出為0~255，因此馬達轉速可為-255~255。

此馬達控制函數簡化了這個過程，使用時只要輸入左、右馬達的轉速（包括負數）即可。



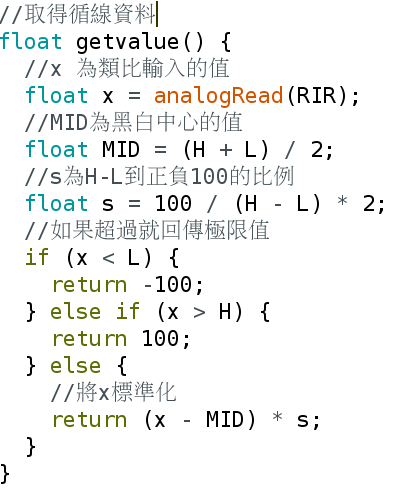
2.紅外線循線感測

紅外線感測器讀到的類比數值範圍在1~5V間，arduino的類比輸入為0~1023。

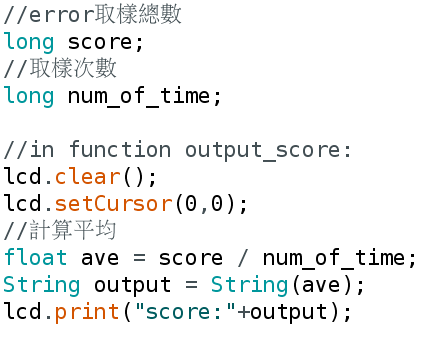
循跡感測器有一對發射和接收的紅外線模組，接收到的量越高則輸出電壓會越低，而黑色吸收紅外線量較白線高，因此黑線數值較高、白線數值較低。

此函數返回值是距離黑白中間的距離，範圍是±100。

由於黑白線的範圍不固定，轉彎的範圍也會因此有所波動，為了固定最大與最小範圍，我們將感測器接收到的訊息**標準化**，將其固定在±100以內，使我們可以預測轉彎的最大與最小值，並且避免感測器讀到錯誤數值而造成車子暴衝。

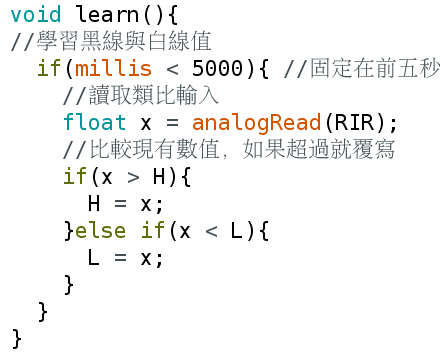
**3.評分機制**

為了量化每次運行的結果，我們使用**error的平均值**來計分。error是感測器讀取並處理後的數值，代表距離黑白交界的差。我們將error的平均值加總，並將平均顯示在LCD上，即可知道此次運行的分數。有了此分數，我們就可以知道運行時車體距離黑白交界的平均誤差。



### 4.自動學習

為了避免場地變化以及光線明暗造成誤差，車子會在每次啟動的前五秒紀錄黑白線的數值，將最黑（偵測數字最大）的作為黑線，最白（偵測數字最小）的作為白線，並在啟動五秒後鎖定這些值，避免後續造成誤差。



一、車體測試

我們組裝了一輛循線車（見圖X）。為了確認這台車能正常運作，我們先進行了一系列的測試：包括車體測量、移動測試及紅外線感測範圍測試，並計算正常狀況的理論值。

（一）、符號說明

在此說明以下使用的符號，如下表：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| a | 旋轉角度 | c | 圓周長（繞轉產生的圓） | d | 弧長（繞轉的移動距離） | r | 繞轉半徑 |
| m | 馬達轉速 | s | 車的移動速度 | w | 兩輪最高點距離 | t | 運轉時間 |

繞轉是指車以等速轉向時，經相同路徑行圓周運動的過程。此外，各項編號皆由左至右，如：（r1,r2） 即代表（左馬達轉速,右馬達轉速）。

（二）、車體測量與理論值計算

由於車體固定，左右車輪的移動路徑在理想狀況下會是同心圓。若馬達轉速∝車輪移動距離，則各項數據關係可以下列方程式表示：

 r為r1或r2

其中a代表車體旋轉角度，d代表左輪移動距離（弧長），c代表左輪繞轉的圓周長，s代表車體移動速率，t代表馬達運轉時間，r代表繞轉圓的半徑；又因為馬達轉速∝車輪移動距離⇒兩馬達的轉速∝兩圓的半徑r1（左）與r2（右），∴ 。右輪通用此式。

|  |  |
| --- | --- |
| C:\Users\ss1051210\AppData\Local\Microsoft\Windows\INetCache\Content.Word\車轉動示意圖.png | w = 13.2（cm）  r1 = 13.2（cm）  r2 = 26.4（cm）  （m1,m2）=（100,200）  d1、d2隨時間（t）改變。  s與m成正比，將在移動測試中計算關係。 |

**圖**1 **轉速為（100,200）時，兩輪會沿固定路徑行進且路徑長比 = 1：2**

（二）、移動測試

我們想知道馬達轉速是否與車輪的移動距離成正比，還有馬達轉速與車體移動速度的關係。每項測試均取偏轉角度5度以內的4次結果。如下表：

# 表1 移動測試結果

|  |  |  |
| --- | --- | --- |
| 起速 | 轉速（馬達運轉一秒） | 移動距離（cm） |
|  | （50,50） |  |
|  | （100,100） |  |
|  | （150,150） |  |
|  | （200,200） |  |

經測試，得出s≒0.4m。

（三）、紅外線感測範圍測試

我們用手機拍下紅外線感應器的照射範圍，再以智高積木作為比例尺計算感測面積。此範圍大約等同一個直徑2cm的圓。如下圖：

圖片

（四）、結論

以上數據可做為後續調整程式之用。

二、簡單循線

在開始PID控制之前，為了測試感測器的性能以及找到轉彎合適的數值，我們先實做了簡單版循線功能。

程式的邏輯很簡單，先找出亮度中間值—也就是黑白交界處，如果大於（代表是黑線）就右轉，反之則左轉，如此一來車子將會沿著黑線的右邊走，並且自動修正位置。但假如只有這樣，車子會一直搖晃，而且前進速度很慢，因為並沒有時間是真正往前走的，因此我們設定一個range變數，只要亮度跟中間值的差在range之內，就設定馬達向前進。這樣子我們就完成了最簡易的循線功能。

目前的車子能夠走直線，和微小的轉彎，但如果轉彎幅度太大，車子就會出軌，而且在直線上會有很多不夠精確的修正，造成車子左右晃動。

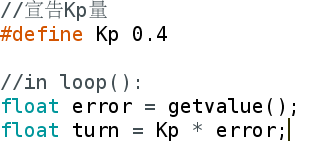
在完成簡單循線之後，我們發現當車子早上在拿出來使用時，跑得結果卻不一樣了，經我們的推測，估計是早上有日照影響，造成感測器的數據整體偏低，才讓車子「走歪」了。

為了避免日夜差異，還有未來到不同測試環境的影響，我們改成讓車子自己「學習」黑白線的參數，我們定義最黑的為黑線，最白的為白線，如果遇到比目前的值更極端的，就複寫目前的值。完成更改後，車子也能在很快的時間內就適應，並且正常的循線，但是如果中途將車子抬起，車子將會讀到非常大的值，因此將車體放回跑道上後，將無法正常的循線。



二、PID──P控制

p 為比例，將讀到的error值透過一個比例反應到馬達上，使得「偏離越多，修正越多」。Kp值需要手動調整。

 在原先的設定下，Kp名稱叫做MT （M為質量，T為轉向，轉向的質量之意），後來為了方便分辨就改名為與PID 相稱的Kp，避免讀者誤會。

在做簡單循線之前，我們就已經嘗試直接製作P控制，但是由於對參數範圍沒有概念，車子讀取後修正太多，沒辦法維持在跑道上，為了解決這個問題，我們先實做了簡單循線，透過此抓出轉彎合適的範圍，再由此範圍去反推合適的Kp值。

在此之前，為了確保用來運算的亮度值有固定的範圍，我們進行「標準化」的工程。將最高值H到最小值L間切成200等分，並且減去中間值，即會得到介於±100內的感測數值，其計算如下。

***（放算式）***

固定了感測值範圍後，我們藉由簡單循線的數值去反推合適的Kp值，如同前面程式碼所寫，轉彎的數值為80和0，由於馬達轉速是由SPEED±turn得出，所以SPEED值就是（80 + 0) /2 = 40，為避免太慢造成馬達停擺我們增加10變50，而Kp就是 馬達範圍除上感測值範圍 = (80 – 0) / 200 = 0.4，因此我們將0.4定為Kp值。

在完成P控制之後，車子已經能順暢的行走直線，並且走少部份的曲線，跟簡單循線的差別是馬達的修正是連續性變化，所以不會有斷斷續續之感。同樣的，過大的彎因為馬達角度不足，會在中途離軌，而且因為馬達輸出最低值會到 50 – 0.4 \* 100 = 10，數值很低會讓馬達轉不動，並且在後續加快速度時沒辦法即時開始工作。

三、PID──I控制

I 為積分，記憶過去的偏移量，於下一次修正。

宣告Integral變數，儲存每一次的error量，為了避免被太久以前的數值影響，每次加入時先乘上3/4。在turn值加入Ki乘上Integral。

效果：在較大的轉彎時，由於一直保持是偏移的狀態，Integral值將會變得巨大，讓車子修正更多以成功過彎。但如果Ki設的太大，在直線時車體會抖動（過度修正），所以大約取跟Kp差不多就好。

而完成I後，我們發現車子在轉彎時常常內輪停住，造成即使轉完了車子還是在原地照圈圈，推測此現象是因為轉速太低導致馬達比不過摩擦力而停住，而且在後續加速時無法即時反應。

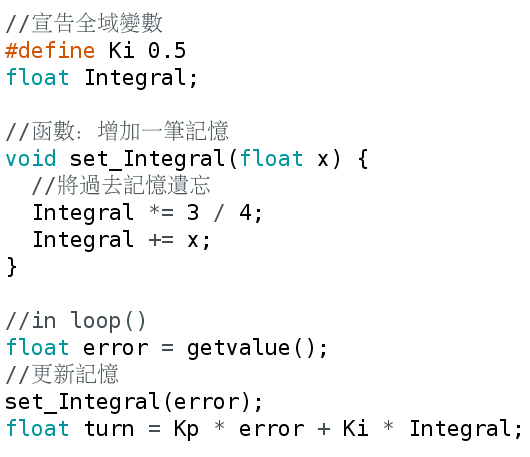
為了避免這樣的情況，我們必須將轉速控制在馬達能隨時反應的範圍，因此我們進行一項實驗，測量馬達所能負擔的最低轉速為多少。

實驗方法：安裝一個七段顯示器，讓它顯示出目前馬達轉速的十位數，寫一個迴圈讓馬達轉速從100一路降到0，觀察車子停住時的轉速為多少，進一步抓出十位數的範圍。接下來縮小範圍，從停下的前一個十位數開始測到停下的十位數，並顯示馬達轉速的個位數。

實驗結果：車子都在轉速20時停下，細分測量30 到 20的範圍，車子在轉速25 到28的時候會停下來。

因此我們在馬達控制中加入最低轉速門檻，假如輸入轉速低於30就設定轉速為30。加入此設定後就大大減少了馬達停滯不前的問題，轉彎也更為穩定。（放程式轉速控制）

現在雖然已經能轉各種大小彎，但是出軌問題仍然嚴重，車子經常在直線變轉彎就跑出去，而且大彎的修正也經常過猛，明明已經結束了，車子卻不受控的繼續衝，造成出軌。



四、PID──D控制

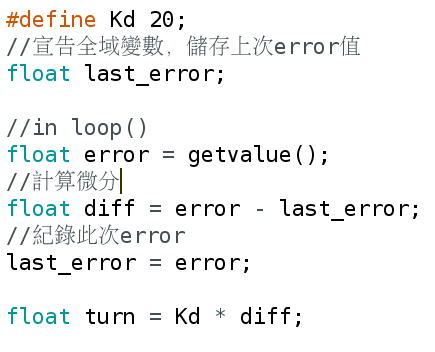
在實做D之前，我們發現量化每次運行結果的重要性，由於我們要講求的是精確度，所以用速度來評分有點不太適合，經過討論後，我們決定使用感測器的平均誤差值作為評分標準。

在呼叫getvalue()後，函數會回傳介於±100數值，其意義就是距離黑白線的距離，將這些error的絕對值加起來取平均，就代表著「運行這一次車子平均離完美循線有多遠？」，數字越低，代表距離黑白中線越近，也就代表跑得越好、越精確。

得到了運行分數，就得要想個方法讓人知道，車子是離線自走，沒辦法把數字傳回電腦，使用七段顯示器太大費周章，只有四位數就要花數十條電線，不太值得，最後選擇使用LCD I2C來顯示訊息，由於I2C已簡化了腳位，只需要4跟線、2個腳位就可以使用。軟體驅動方面，使用了arduino 套件LiquidCrystal\_I2C來驅動，使用起來非常簡單。

顯示器做好了，接下來就是要計算並印出分數了。我們原先是將每個loop的error加到float score中，每個loop將int num\_of time ++; 但是顯示出來的值非常奇怪，照理說無論如何我們的值都會在100以內，但是大概啟動2秒之後，就會有112、109這樣奇怪的數字跑出來，到後面甚至有負數，兩個正數相除竟然出現負數！為了抓出錯誤，我們將num\_of\_time顯示出來，並重新啟動。我們觀察到每一秒num\_of\_time大約增加了1000，而大概過了十幾秒，數字竟然突然變成了負數！很顯然的int不夠用，數字溢位了。如果連數字較小的num\_of\_time都溢位了，那score更不用說了。

為了解決溢位，我們將score和num\_of\_time都改成long

 D為微分，也就是趨勢，透過過去和現在的差來推算未來，跟P和I不同的是，D可以不被感測器到線的距離影響判斷，例如說車子從a點(80)出發，到了b點(0)，P因為已經到了中間了所以讓它直走，但是如果照這個趨勢下去，下一點c一定會變成-80。這時候如果加入D運算，b – a = -80，這樣子對車子來說，還需要再向右修正，因此可以避免掉衝到-80的不好結果。

實際將D運算加入程式的效果十分顯著，一開始不清楚Kd應該的範圍，就保守的將它設的跟Kp一樣，但放進去似乎沒什麼差異，為了看出他的效果，越試越大，效果才慢慢有出來，最終Kd值我們把它放大了50倍（0.4 \* 50 = 20)，車子可以說是「完全沿著線移動」，移動的分常平順，且輕輕鬆鬆就達到過去跑不到的路線。

五、PID與數位感應車循線效果比較

# 伍、研究結果

順序：一、（一）、1、（1）

# 陸、討論

# 柒、結論

# 捌、參考資料及其他

一、A PID Controller For Lego Mindstorms Robots <http://www.inpharmix.com/jps/PID_Controller_For_Lego_Mindstorms_Robots.html>

二、直流馬達定位控制-最佳 PID 控制器之設計

<http://ir.lib.ntust.edu.tw/retrieve/50876/Optimal+PID+Controller+Design+for+DC+Motor+Positioning+Control.pdf>

三、HOW TO USE IR SENSOR USING OP AMP AND ARDUINO

http://www.instructables.com/id/How-to-use-IR-Sensor-using-Op-amp-and-Arduino/