逢　甲　大　學

資訊工程學系專題報告

8051 智慧鼠迷宮遊戲設計

劉明機(四甲)
學　　生：江佳霖(四甲)
程毓北(四甲)
指導教授：陳啟鏘老師

中華民國九十二年十二月

目錄

摘要

本專題是『8051 智慧鼠走迷宮』 在硬體方面，我們採用的是一台伺服器自走車搭配七個光感測器，感光器能判別迷宮路徑以驅動自走車的方向，迷宮的部分則是自製的迷宮，材料為一般黑色電工膠帶和厚紙版若干片，以便可隨時變更地圖的路徑。硬體設備中，主控制晶片為 AT89C51，AT89C51 是 MCS-51 家族的新成員，除了仍然保有其硬體的向上相容性，還提供更利於以往程式作業的方便性，程式作業介面我們採用 8051 系統開發的 KEIL C 語言，C 語言除了在 8051 系列上較耗費系統資源的缺點較令人詬病，但其開發系統的速度以及容易除錯的功能比較利於程式人員的撰寫。程式作業我們分為三部分：

(1) 控制自走車基本方向功能：向前、迴轉、往左、往右。

(2) 演算法 --Search 路徑

(3) 演算法 –找到最快路徑

伺服器自走車分兩次完成迷宮，第一次是 Search 到達終點之後再置回起點，根據演算法算出最快路徑再作模擬。我們的重點放在如何，完整 Search 路徑，光感測器的紀錄路徑正確性，還有最快路徑分析的演算法。最後，探討 8051 晶片的功能擴充能夠展現出來的效能，以及未來的市場動向。

第一章　導論

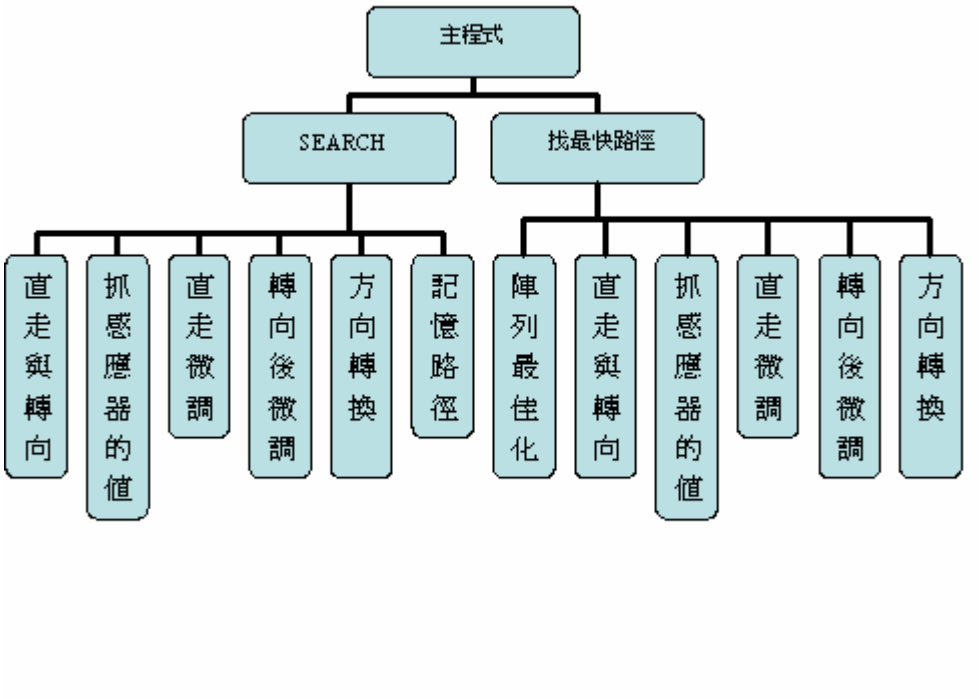
1.1 動機

Micro mouse，的歷史沿革已久，早在 50 年代，就有學者提出『機械思考』的問題，70 年代，已經有研究機械原型設計，正式出版在 IEEE 期刊上，80 年代，單晶片微控制器問世，微電腦控制的機械開始廣泛的應用在各界，而以微鼠(Micro mouse)為名的科技競賽自此也如荼的展開。在亞洲也旋起了一股風潮，微鼠設計每年不斷推陳出新，能夠以更穩更聰明的方法解出迷宮為目標，各大專院校參賽者更以在國際性的賽事上得獎為榮。現在，我們見證一個世代過去了，微鼠的技術已趨近成熟，這也似乎說著一個世代有著一個世代的瓶頸，無法突破，但不變的是還對科技始終來自人性的那份熱忱，有幸能夠搭上這末班的列車，參與這類資訊的研究，我們以此為專題，期望能對機械控制與軟體整合有更一步的認識。

1.2　目的

本專題是智慧鼠走迷宮， 既然是智慧鼠， 就一定要有智慧的功能。我們的目標 ，就是要讓智慧鼠能夠以程式自我控制 ，完成迷宮搜尋、 最快路徑的功能。

1.3 系統的架構與設計



1.4 報告結構

在這個專題中，我們將專題劃分幾個階段性。前面階段中，我們在此收集 8051 的相關資料和研討。後階段中，微鼠的功能測試，軟體撰寫屬於專題實作部份的部分。下列是我們的工作計劃

(1)單晶片微處理器知識相關學習

(2)認識微鼠架構

(3)機械控制功能測試

(4)路徑演算法的整合

第二章 使用之硬體與發展工具及設備

微電腦普遍應用在日常生活中的一些自動化設備中，可以說帶動了整個人類科技的進步。本章主要介紹微電腦組成的基本架構，單晶片微電腦的特點及其應用領域使初學者可以很快地了解一套控制系統是如何構成的。

2.1 8051 單晶片概述

8051 單晶片介紹

單晶片微電腦一般是指將CPU、記憶體、I/O 埠介面、時脈電路等單元，製作在同一個IC上　可以說是一個不帶週邊裝置的微電腦。這些晶片只需要少量的支援電路即可獨立工作，如此就可以大量地減少電路板面積及降低成本，因此頗為適合家電、汽車、工業控制等產品及用途上，所以單晶片微電腦又稱之為微控制器(Microcontroller)。

單晶片微電腦的優點如下：

(1) 體積小、成本低

(2) 連接線路簡單

(3) 控制功能強

(4) 穩定度很高

(5) 發展及擴充能力很強

(6) 使用方便、簡易

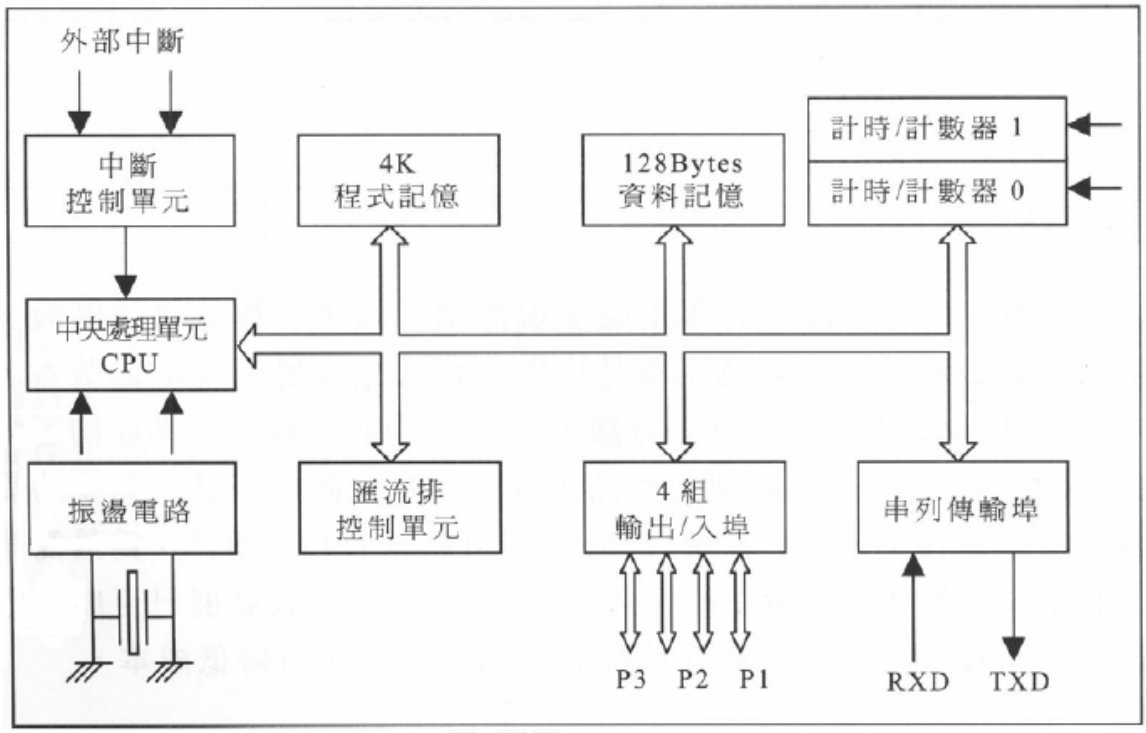


圖2-1-1 MCS-51 單晶片

MCS-51 晶片家族介紹

MCS-51 主要特性如下：

(1) 為8位元CPU。

(2) 內部含震盪及時脈電路。

(3) 32條的輸入/輸出(即I/O 埠)。

(4) 64K位元組的內外部程式記憶體空間。

(5) 128 位元組的內部資料記憶體(8052 為256 位元組)，及128位元

組的特殊暫存器(SFR)。

(6) 64K 位元組的外部資料記憶體空間。

(7) 2 個16 位元的計時/計數器(8052 有3 個)。

(8) 5 種不同的中斷來源(8052 有6 種)，並具有兩種中斷優先次序選擇。

(9) 全雙向的串列埠。

(10) 布林處理。

MCS-51 族系的成員如下圖所示

MCS-51 型號	晶片內記憶體 (位元組)		記憶體 定址範圍 (位元組)		I/O Port		中斷源	計數器	石英震盪器
	ROM EPROM	RAM	ROM EPROM	RAM	並列接腳	串列		位元*(個)	MHz
8051	4k	128	64k	64k	32	UART	5	16*2	2-12
8751	4k	128	64k	64k	32	UART	5	16*2	2-12
8031	-	128	64k	64k	32	UART	5	16*2	2-12
8052AH	8k	256	64k	64k	32	UART	6	16*3	2-12
8752AH	8k	256	64k	64k	32	UART	6	16*3	2-12
8032AH	-	256	64k	64k	32	UART	6	16*3	2-12
80C51	4k	128	64k	64k	32	UART	5	16*2	2-12
80C31	-	128	64k	64k	32	UART	5	16*2	2-12
87C51	4k	128	64k	64k	32	UART	5	16*2	2-12
89C51	4k	128	64k	64k	32	UART	5	16*2	2-12

表2.2.1 MCS-51 成員

8051 是MCS-51 家族中最好的成員， 8031 和8051 的不同處，是8031沒有內部成記憶體(ROM)。8031 提取指令時，必須從外部程式記憶體提供。8751 和8051 具有相同功能，只是8751 的內部程式記憶體採用EPROM。最早期的8051是HMOS-I的技術，且目前採用MOS-II 的版本稱為8051AH。80C51 是採用CHMOS 版本，提供了減少功率模式(power reduced model) 運作為其標準功能版本。8052AH 的接腳與8051類似，其提供了更多的中隊來源及計時/計數器，內部的ROM 與RAM為8051的兩倍。8032AH為無內部程式記憶體的版本，8752AH 為EPROM 的版本，在此都將他們歸入MCS-51家族裡。在部分MCS-51的家族成員中，為了保護設計者的智慧財產權，更設計了保護位元(security bits)，以防止燒錄在單晶片微電腦內部的程式被不法的讀取。

MCS-51的接腳說明

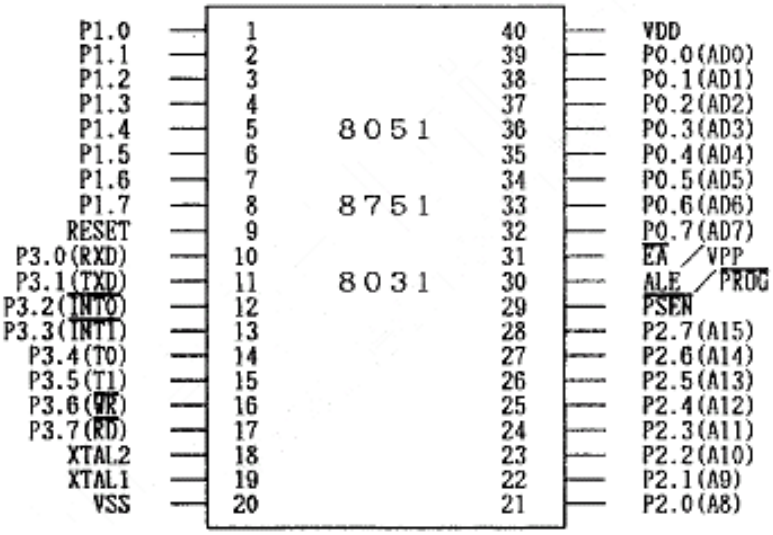


圖2.3.1 MCS-51 接腳圖

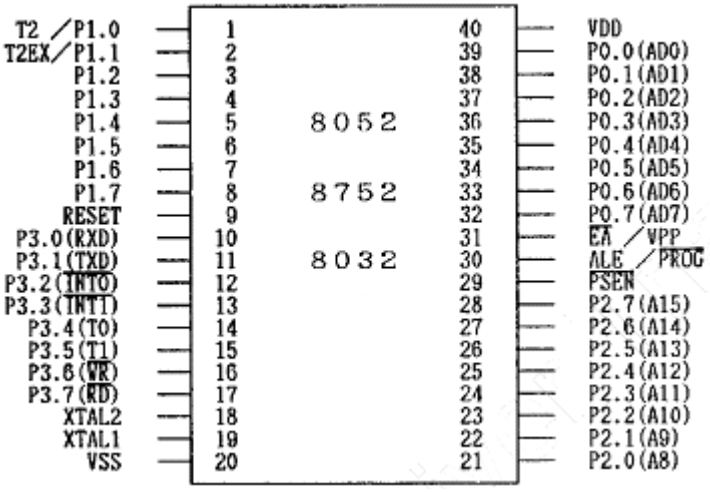
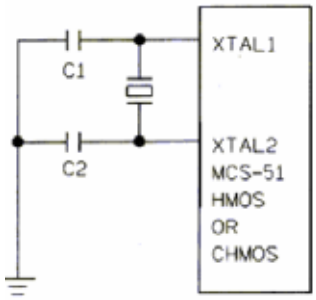


圖2.3.2 MCS-52 接腳圖

接腳編號	接腳名稱	功能說明
1 ~8	P1.0 ~P1.7	P1 為8 位元雙向I/O 埠，內部有提昇電阻。在8051 指當作I/O埠使用；在8052 中P1.1 最為T2EX 接腳。T2 是用來當成計時器2 的外部輸入源；T2XE 則是計時器.2 在”抓取”模式的外部輸入源
9	Reset	重置信號接腳。當此接腳接高電位約兩個機械週期時，將可使 8051/8052 達到重置(Reset)的狀態
10 ~ 17	P3.0 ~P3.7	P3 為8 位元雙向I/O 埠，內部有提昇電阻。其具有三

		個功能： (1) 可以作為一般I/O 埠使用 (2) 可以作位元定址 (3)可作為特殊功能接腳
18 ~ 19	XTAL2 XTAL1	石英震盪器的兩個連接腳，決定震盪頻率 
20	VSS	接地點 0 V
21 ~ 28	P2.0 ~ P2.7	P2 為8 位元雙向I/O 埠，內部有提昇電阻。其具有二個功能： (1) 當擴充外部記憶體時，作為位置匯流排的高位址線(A8~A15)使用。 (2) 可做一般I/O 埠使用
29	PSEN = 0 (Program Store Enable)	外部程式之致能訊號，只有在存取外部程式時，接腳才會自動地送出訊號
30	ALE	位址栓鎖致能(address latch enable)訊號，其具有三個功能 (1) 使用外部記憶體時：對外部記憶體進行存取時，用來鎖住低位元位址的訊號 (2) 未使用外部記憶體時：提共1/6 的時應的震盪頻率，可作為外部時脈訊號 (3) 在燒錄ROM/EPROM 時：ALE 接腳作為接受燒錄脈波的訊號
31	EA = 0 (External Latch Enable)	外部提取致能，其功能有： (1) 當接低電位時：讀取外部程式記憶體(ROM)，例如使用8031/8032 時 (2) 當接高電位時： (a) 當讀取內部程式記憶體(ROM) (b) 當讀取內部程式記憶體的位址超過0FFFH(4K位元

		組) 時(8052 為超過1FFFH)，自動讀取外部程式記憶體 (3) 當8751/8752 在燒錄過程中，利用此接腳輸入21V的燒錄電壓
32 ~ 39	P0.0 ~ P0.7	P0 為MCS- 51之資料匯流排 可作為8位元之I/O雙向埠 存取外部之低階8位元位址(AD0 ~ AD7)由此 輸出 可用來配合外部程式記憶體的低階8位元程式位址 (PC0~PC7)產生
40	VCC	電源輸入 接 +5V

表 2.3.1 MCS-51 接腳簡引

程式記憶體(ROM)

8051及8071皆具有4K Bytes 的內部程式記憶體，並可在外部再擴充60K Bytes EPROM，如下圖所示，而8031則沒有這些內部程式記憶體。在程式記憶體中所存放的是8051所要執行的程式碼，單晶片會主動到這塊記憶體要執行的指令碼，而8051要讀取程式記體時需激發信號PSEN。如下圖：

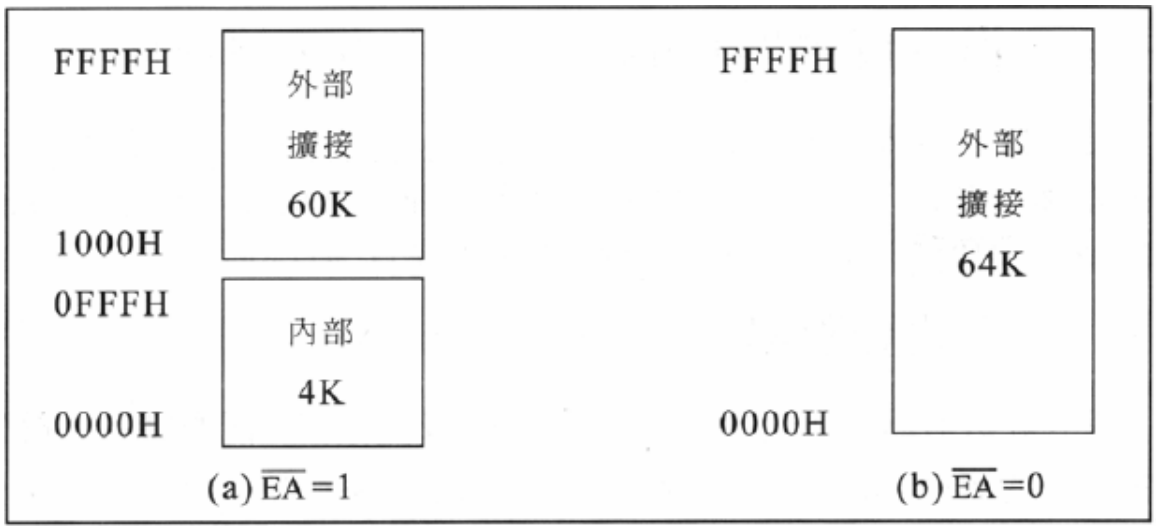


圖 2.4.1 8051程式記憶體結構圖

8051是如何決定程式記憶體的前面4K Bytes要內部或外部程式記憶體去讀取指令呢?這就要靠8051的EA接腳來決定內部程式記憶體是否有效，當EA=0，代表內部程式記憶體無效，8051會將前面4K移到外部；當EA=1，則內部程式記憶體有效。

資料記憶體(RAM)

8051 內部有一塊256 個Byte的位址空間，這塊空間是存放資料記憶體(RAM)和特殊功能暫存器(SFR)的地方，並可在外部擴充64KBytes的資料記憶體。其資料記憶體的結構圖如下

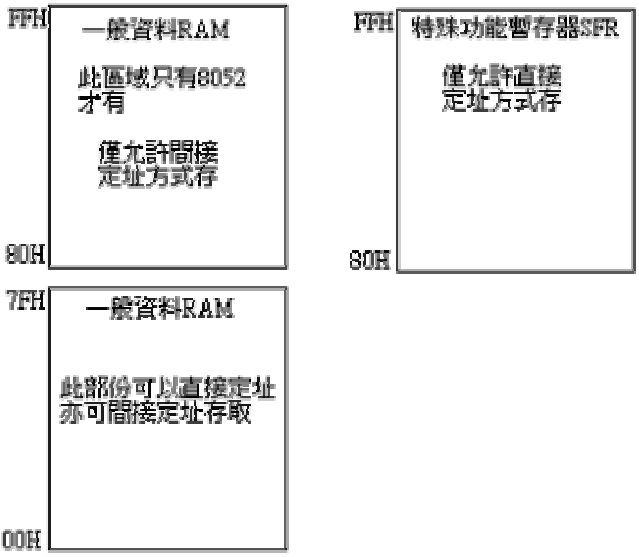


圖 2.5.1 8051 資料記憶體結構圖

8051系列單晶片具有128 Bytes的內部資料記憶體，其中位址編號為00H~7FH。這些內部資料記憶體可供使用者的程式自由存取資料，不過，00H~7FH記憶體的資料可用直接定址法來存取資料，而8052系列的80H~FFH記憶體的資料則必須間接定址法才可以存取。依單晶片的特性又可將這些內部資料記憶體(00H~7FH)分成三個不同的部分

1.暫存器庫(Register Banks)區

2.可位元定址(Bit-addressable)區

3.一般用途區

● 8051 常用暫存器

1. A 累加器 (Accumulator)

A 是一般性暫存器 用來當作指令運算的主要媒介 是一個 8-bit 的暫存器。許多 8051 的指令都需要用到 A 暫存器，例如：

```
MOV   A , #25
ADD   A , #35
```

此動作是 25+35 結果是 60 並存入 A 暫存器

2. Rn 暫存器

Rn 暫存器包括 R0, R1,… R7 等 8 個暫存器 為 8- bit 暫存器是一般輔助暫存器 以上為例 若 R5 內含 35 則視 R5 為一變數

```
MOV A ,#25
ADD A ,R5
```

此結果 A 仍是 60

3. B 暫存器

B 暫存器為 8-bit 之暫存器 主要用於乘法與除法的指令

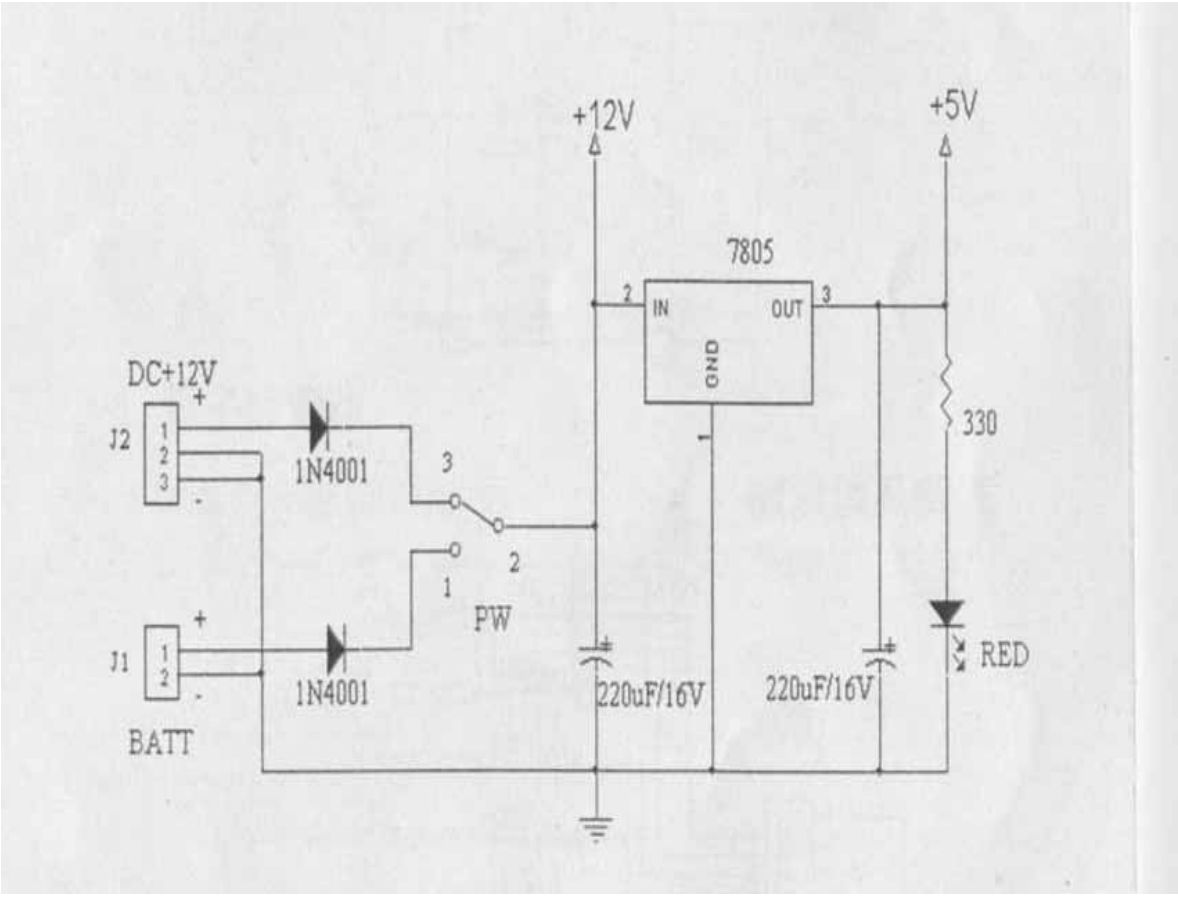
4. 程式狀態字(PSW)

程式狀態字暫存器主要紀錄 CPU 執行算術/邏輯運算後進位、溢位、輔助進位的狀態值、暫存器庫的選擇及同位位元，可位元定址。程式狀態字暫存器的位元格式及功能。

2.2 智慧鼠硬體介紹

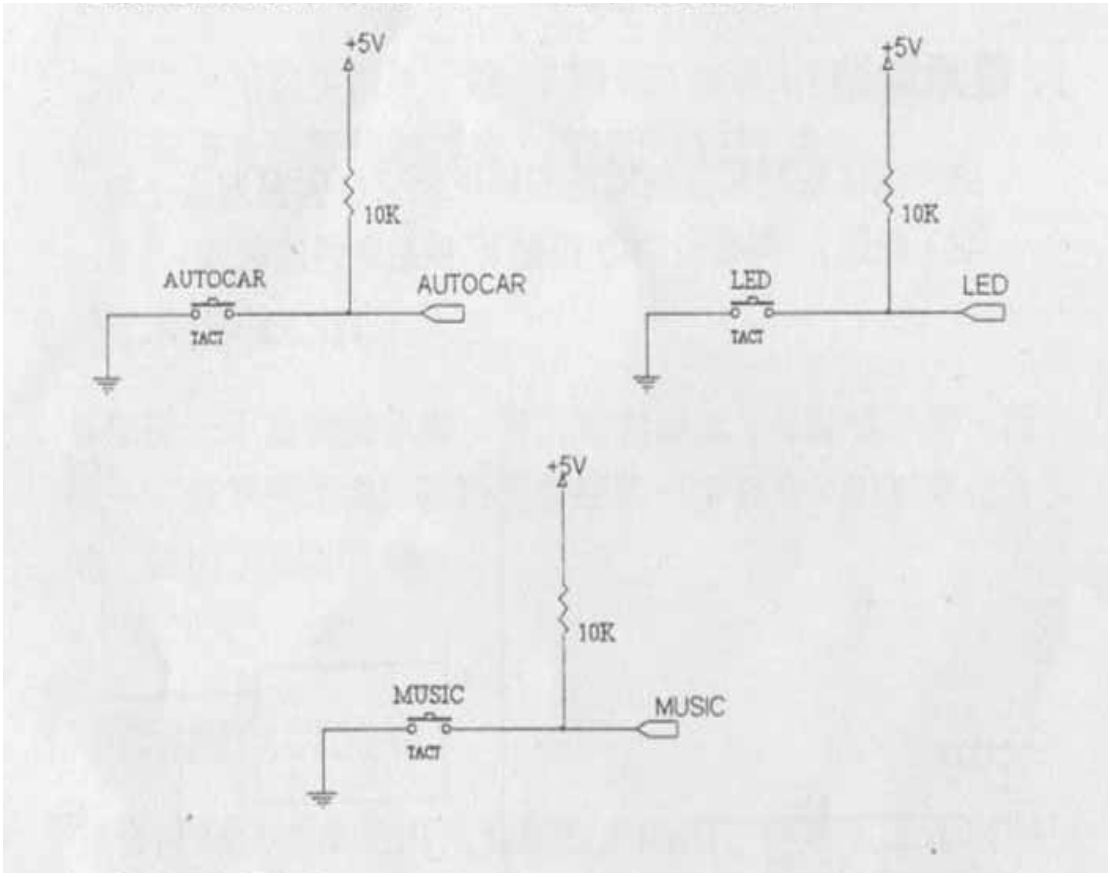
1. 電源電路

由一個搖頭開關控制電源的提供是來自變壓器或者是電池，再經由7805 穩壓 IC 供給電路的電源



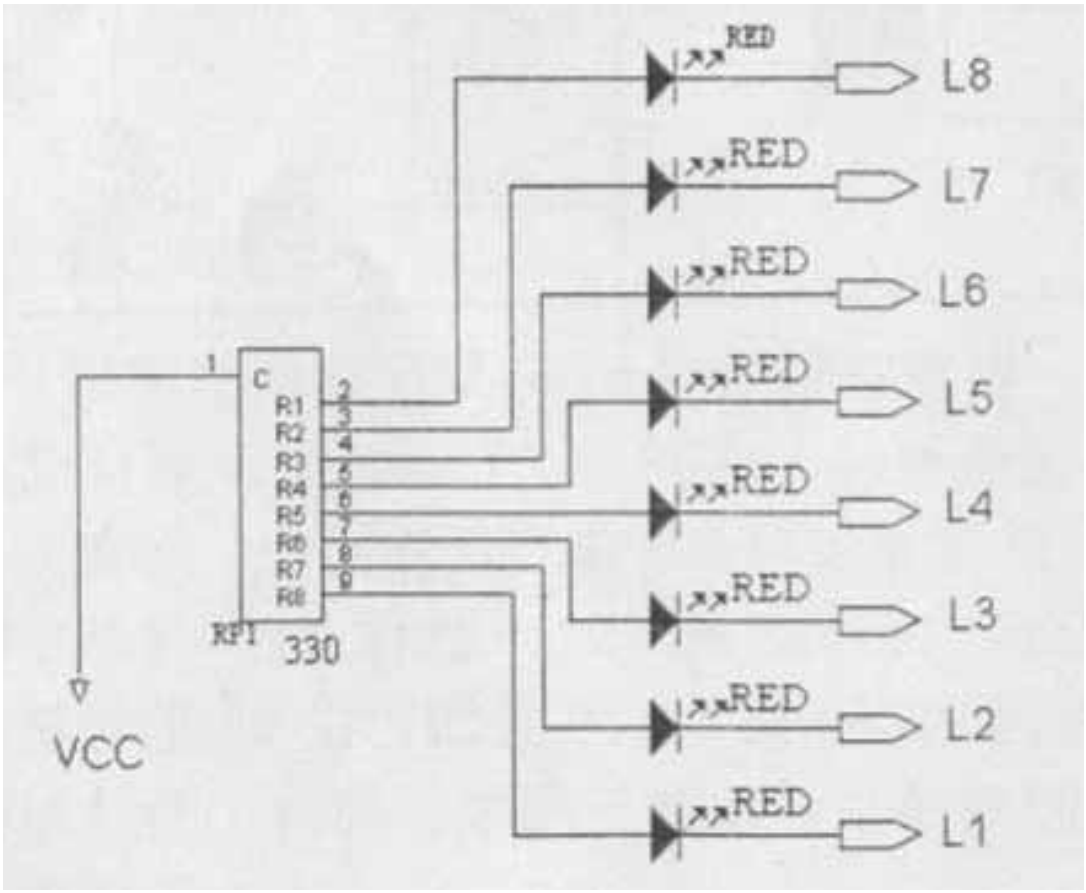
2. 功能展示電路

當按鈕沒有按下時為高電位，按下時為低電位



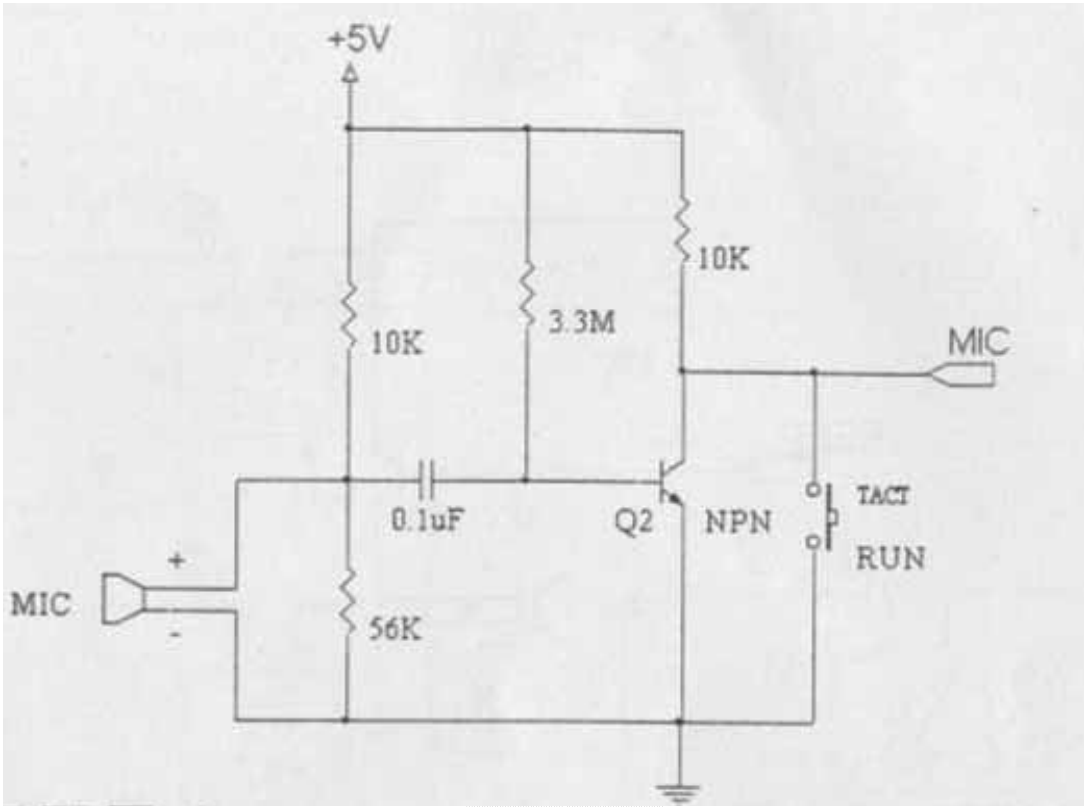
3. 跑馬燈電路

跑馬燈是使用多工器來選擇與驅動 LED。



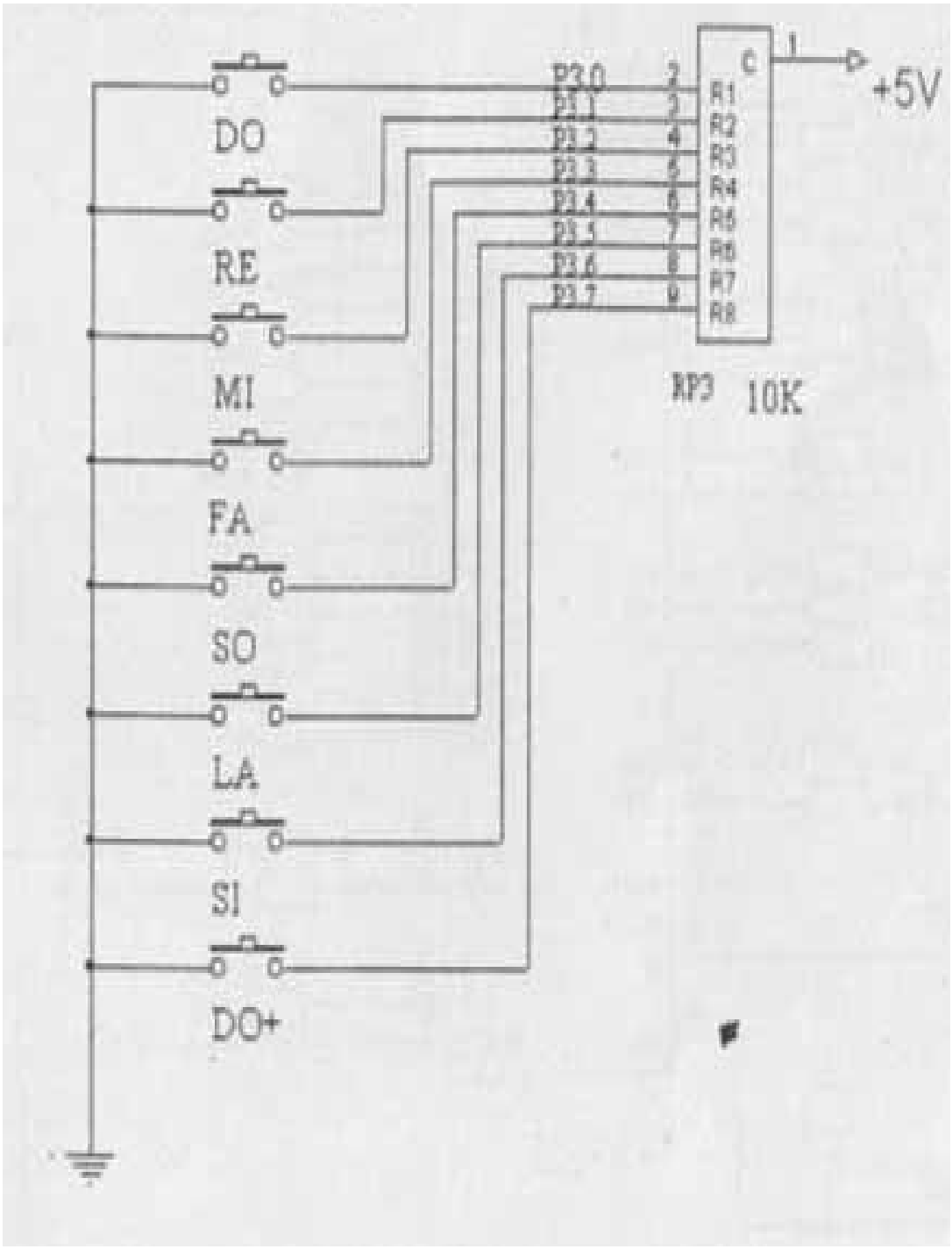
4. 聲控電路

經由電容式麥克風驅動 Q2(9013)導通，使輸出為低電位



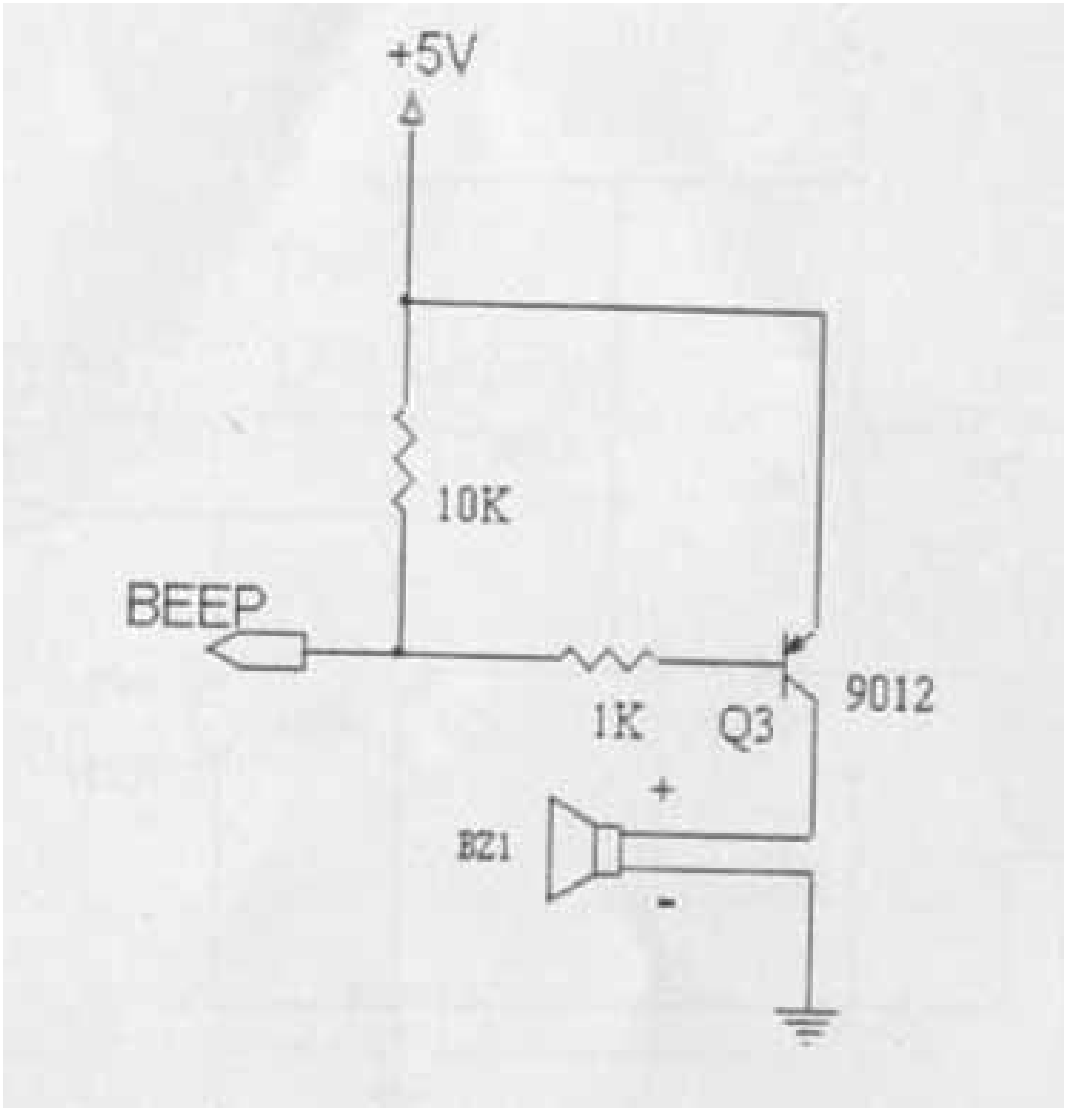
5. 電子琴電路

與功能展示電路的原理相同



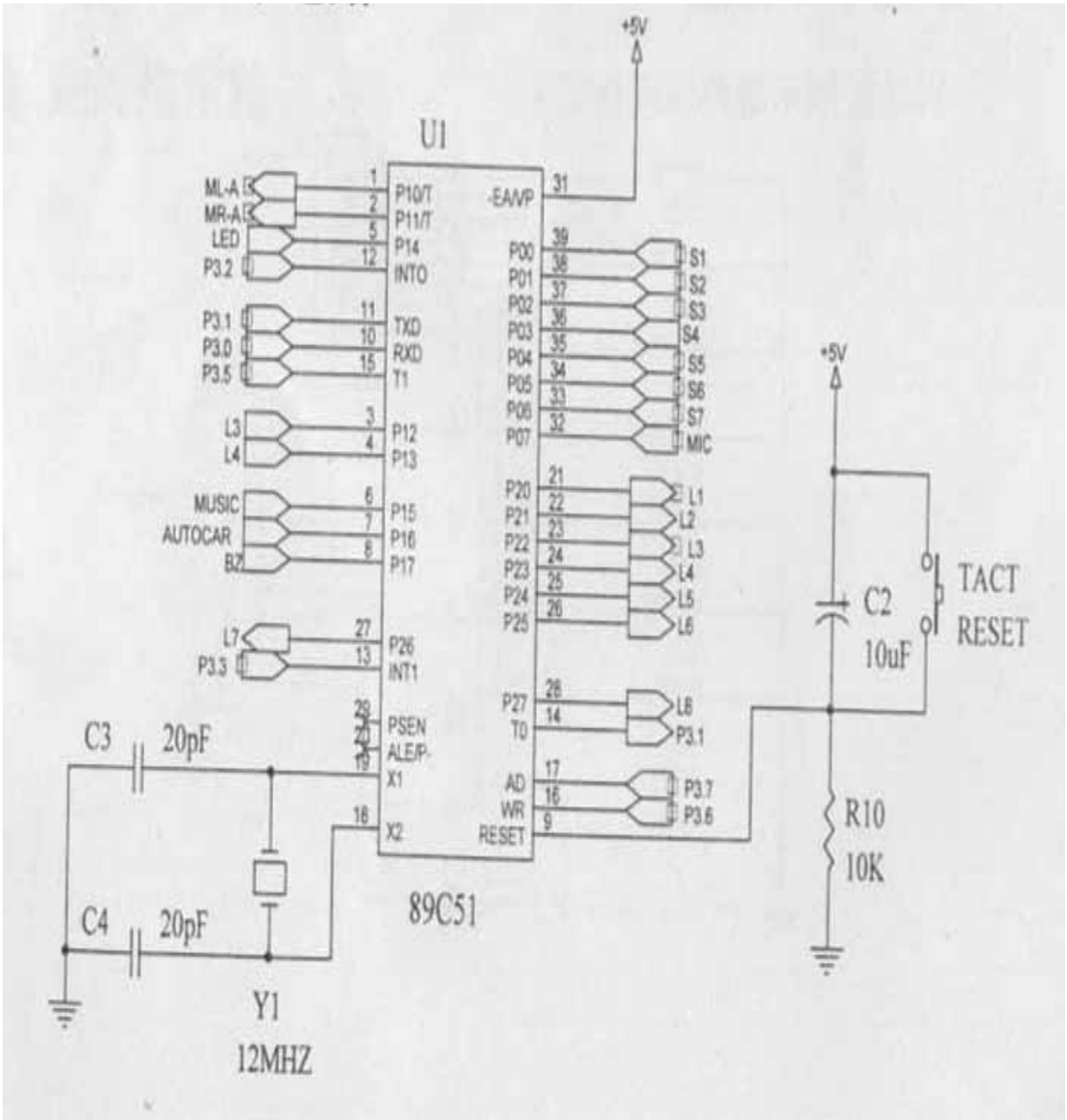
6. 喇叭電路

利用一顆電晶體來放大電流以驅動喇叭發聲



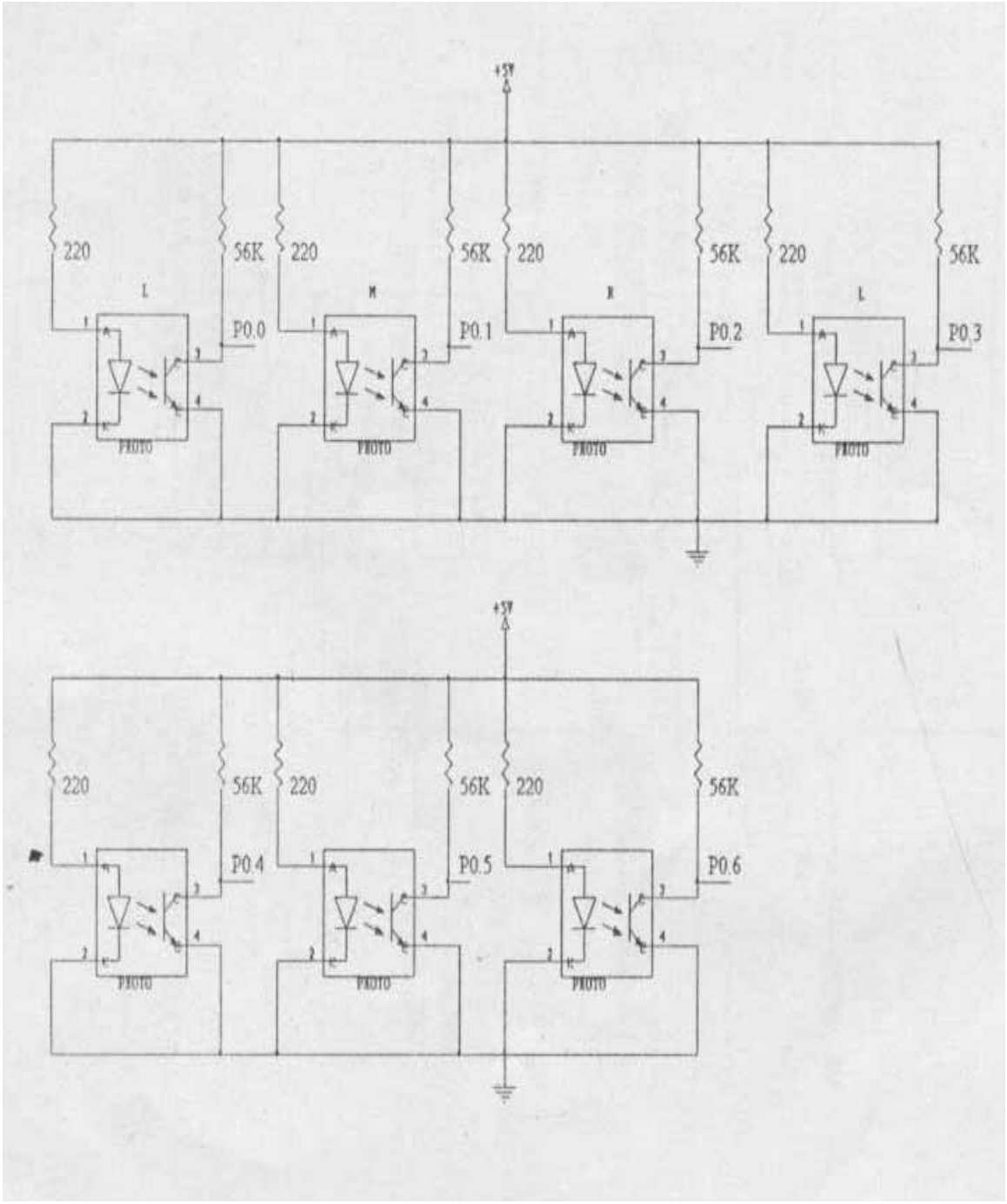
7. 89C51 基本電路

89C51 內部記憶體為 128Byte，Flash Memory 為 4K byte，石英晶體為 12MHz。



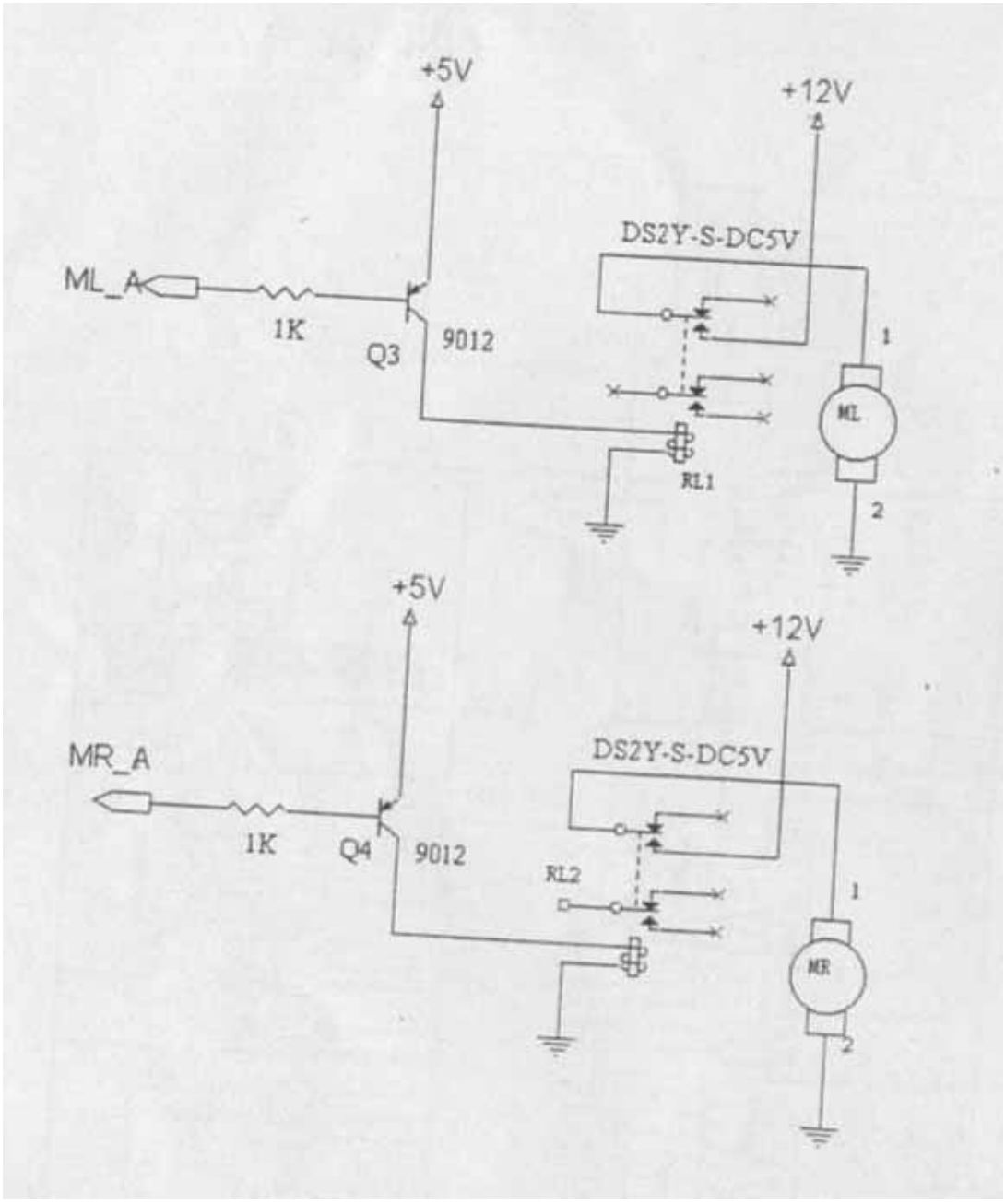
8. 感測器電路

當光感測器有感測到時，會由第三腳輸出低電位，沒有感測到時，會由第三腳輸出高電位



9. 馬達電路

當 89C51 送出低電位時，會使電晶體(Q3 或 Q4)導通，繼電器激磁，以推動馬達運轉



2.3 燒錄器介紹

WINICE 51/52-E 之介紹

硬體模擬功能：

- 可模擬之 CPU：80(C)31/32,80(C)51/52,87(C)51/52;及提供內建式 5V 89C51/52 燒錄功能。
- 模擬頻率可達 16MHz(以軟體切換內外部 CLOCK)。
- 提供 128K bytes 模擬記憶體(program 64K,data 64K)。
- 採並聯傳輸(PRINTER PORTT)界面。

硬體中斷功能：

- 提供 64K 硬體執行中斷點及硬體匯流排中斷點可設定:Address、Data bus、CPU Status 及 4 External Trigger。

即時追蹤能力(Real-time Trace)：

- 可即時記錄 8K*32bits 深度,包括 16 bit address、8 bit data、4bit status 及 4 bit external probes。
- 追蹤深度可任調為 Forward、Backward 及 About Trace 及.提供追蹤設定點及 Qualify Trace。

- 追蹤後之結果可以針對上述 32 bits 中任擇 16 channels 作波形顯示(Waveform Display)。

軟體模擬功能：

- 提供下送(Download)檔案,如 HEX,BIN,OMF 等 12 種格式。
- 具Batch及Log處理能力及可作線上(On Line)全螢幕行組譯及反組譯。
- 具符號除錯(Symbolic Debug)能力。

高階除錯功能：

- 可擴充 MHLD(Microtime High Level Debugger),它提供使用者一個高階除錯環境(可針對 C)。
- 提供三種程式顯示模式 a.純高階碼 b.高階與組合語言混合 c.純組合語言。

出廠配備：

- WINICE51/52-E 主機*1・Parallel cable*1・ICE cable * 1・S/W driver*1・Manual*1・AC power core*1
- External line*1・Remote control probes*1

系統需求：

- IBM PC XT/AT 386,486,pentium(or Compatible)以上或 Notebook
- 至少一部 3 1/2"軟式磁碟機,350K 記憶體以上
- Printer port*1 ・DOS 5.0 版本以上或 Windows95/98
- 尺寸/重量:長:25CM,寬:17CM,高:5CM,重:2.0KG

WINICE 主機之安裝

WINICE 51/52主機之裝設請依下列步驟實施之。

步驟1 自包裝箱中將ICE主機取出，置於一理想之場所。在此建議ICE以外之物體與機殼保持至少5公分之距離，同時機殼底部不應以軟的材料作墊子，以免妨礙機器通風散熱。

步驟2 連接Parallel Cable：

如圖2.2或圖2.3所示將1.5公尺長的Parallel Cable其中25 PIN公頭端連到PC背板上任一 Printer Port之位置，再將25 PIN母頭端連接到ICE主機背板上標示Parallel Port的接頭上。連接完成後請將兩端接頭上之固定螺絲鎖上，以避免發生接觸不良之狀況。

步驟3 將ICE主機接上電源：

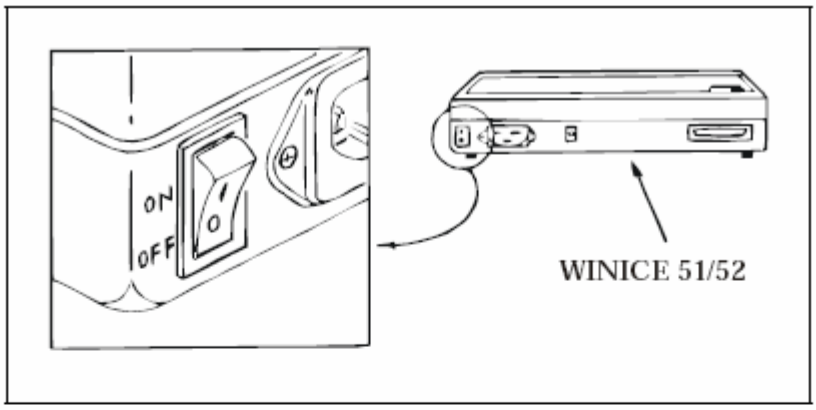
因為主機有所謂內建電源與外加電源之選擇，在此將分二種狀

況來描述：

電源內建型：(標準出貨)

請先確定ICE主機電源是在"OFF"之狀態，如圖2.1所示。

圖2.1
電源開關

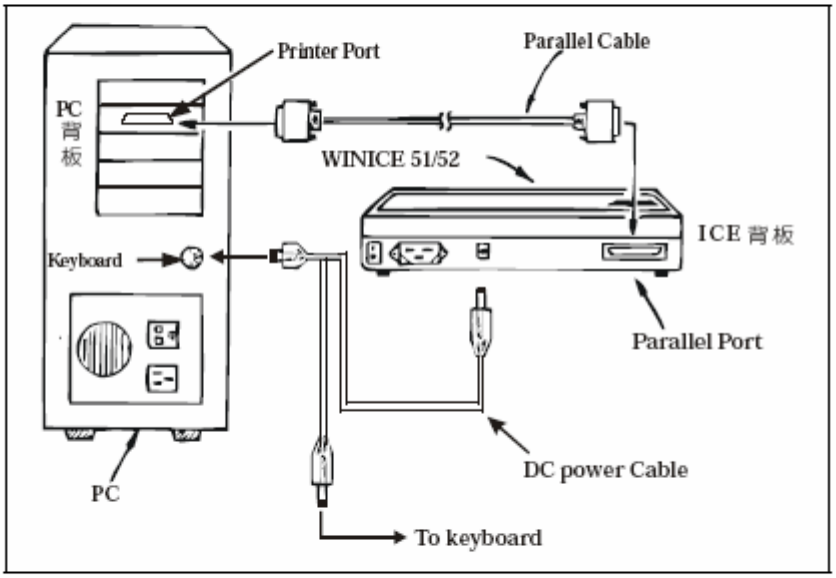


再將市電電源線母頭端插入ICE主機背板上標示～LINE之插座內，而將3 PIN之公頭端插入市電之插座上。Parallel Cable 與電源之接法，請參考圖2.2。

☞ 接入市電之前請先確認市電規格是否為90-260V，50-70HZ單相之AC電源。

Windows版

圖2.2
電源內建型ICE主機之
Parallel Cable 與電源
Cable 之安裝



❑ 電源外加型：(選項配備)
請先確定ICE主機電源是在"OFF"之狀態，參考圖2.1
DC電源線如圖2.3所示，是一個三端的Keyboard延伸線，在連線時請將Keyboard插頭公頭端與PC背板上之Keyboard插座連接，Keyboard插頭母頭端與Keyboard端連接，最後再將DC之插頭端與機殼背板上標示"+5V"處之插座連接。

在做以上DC電源連接動作時請確認PC是在關電的情況下進行，否則很可能因作業不慎造成PC內電源短路，導致不可預知之後果。

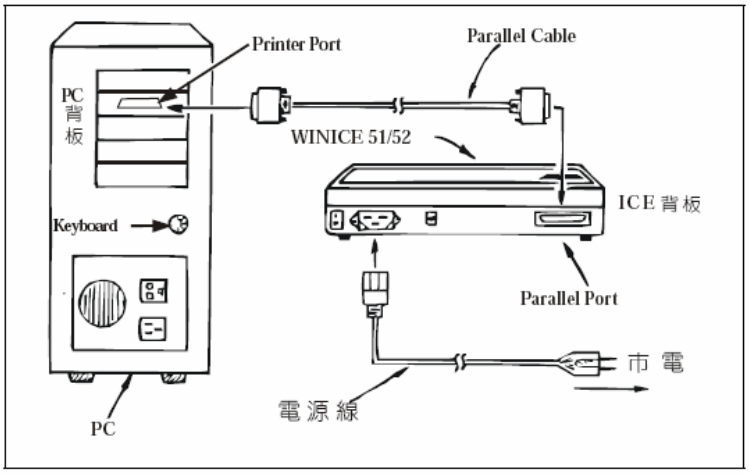


圖2.3
電源外加型 ICE 主機之 Parallel Cable 與電源 Cable之安裝

〔2 - 4〕WINICE

在完成上述Parallel Cable及電源線之裝設後，才可以將主控電腦之電源打開。在打開主控電腦之電源後請觀察電腦之開機動作是否正常。若有不正常之現象請立即關閉電腦之電源，之後再將ICE主機與主控電腦分離。在分離的情況下，重新對主控電腦作開機測試，若是電腦開機正常，則表示ICE主機可能有不正常耗電之狀況，請就近通知新華之維修單位加以處理。完成步驟1至步驟3之動作後，就已經大致上完成了主機之安裝動作。唯採用電源外加型ICE主機的用戶須特別注意前述測試之動作，以免對主控電腦造成損壞而不自知。現在可以打開ICE主機背板上之電源開關，同時觀察面板上之紅色電源指示燈是否亮著。如果亮著的話，在完成"軟體安裝"步驟後應可以執行ICE驅動軟體進入

WINICE 51/52的模擬世界。

燒錄命令

在Config命令群中有以下二組命令可供燒錄器使用

燒錄89C51/52：Config → 89C51/52 PROGRAM

驗証89C51/52：Config → 89C51/52 VERIFY

燒錄89C51/52 來驗証實驗結果，是本燒錄器設計的一個目的，為了簡化使用者操作程序；因此有很多在專業燒錄器上看到的功能諸如Security Bit、Edit、Pin-Continuity等，本燒錄器並不提供。同時在命令的設計上亦力求一個命令從頭做到尾，不要有太多的設定。

命令：***Config ->89C51/52 PROGRAM***

功能：執行燒錄89C51/52 之命令.

操作：在CONFIG命令群中選擇到本命令後，會出現如圖4.45之燒錄畫面。

首先必須輸入一個待燒錄資料的檔案名稱. 再選擇要不要作Verify的動作，最後在GO處按下Enter鍵即開使進行燒錄之程序。在燒錄之過程中，燒錄之Status及燒錄計數都會同時顯示在畫面中。在畫面中可以看到燒錄器的硬體結構及燒錄程序的提醒字句，為的是要再一次地提醒使用者相關之燒錄注意事項，以免因一時疏忽而損壞了設備。正確之

燒錄程序，請依下列步驟實施之

請確認在待燒錄插座上無待燒錄8051之IC

利用Comnfig 89C51/52 PROGRAM或Comnfig 89C51/52 VERIFY

命令完成燒錄或驗証前之設定。

將待燒錄89C51/52之IC置入插座內(注意方向)

按下”GO”鍵，進行燒錄或驗証工作。

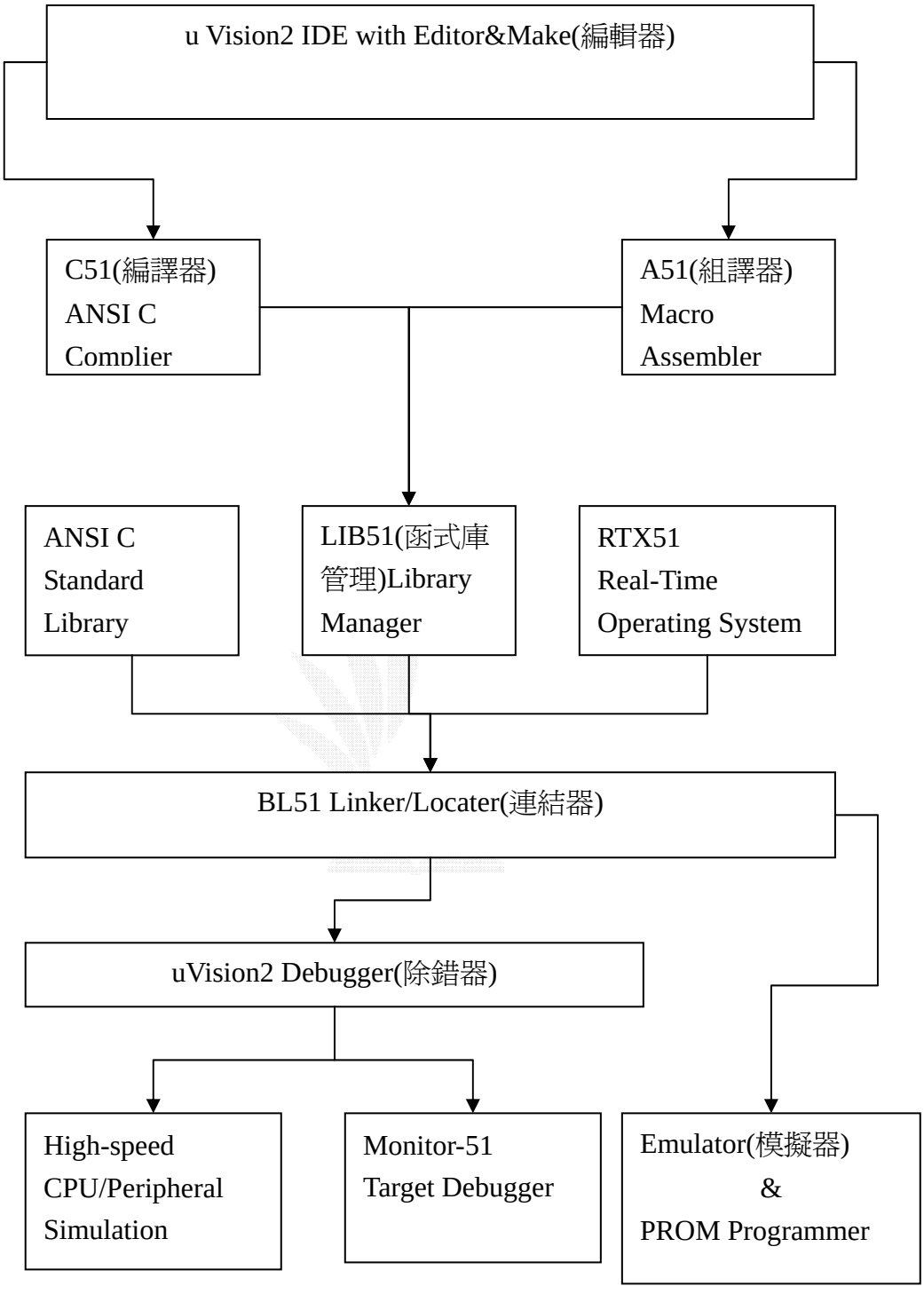
當完成上述燒錄或驗証工作時，取下待燒錄或驗証之IC

Windows版

圖4.41
89C51/52 Program
視窗

2.4 程式工具 Keil C 介紹

u Vision2 視窗版是一個獨立而且功能強大的整合性開發環境，結合了計畫(project)經營管理、原始程式編輯器(editor)、組譯器(assembler)、編譯器(complier)、連結器(linker/locater)、程式除錯器(debugger)等功能，其方塊流程圖如下圖所示。



組合語言與 C 語言的優缺點：

1) 所有有牽涉到精確計時的副程式中，用組合語言會有更快的反應

度。

2) 所有 8051 C 語言編譯程式都必小心撰寫，以免造成當機

3) C 語言編譯時可指定對執行速度或是程式碼 CODE SIZE 進行最佳化。

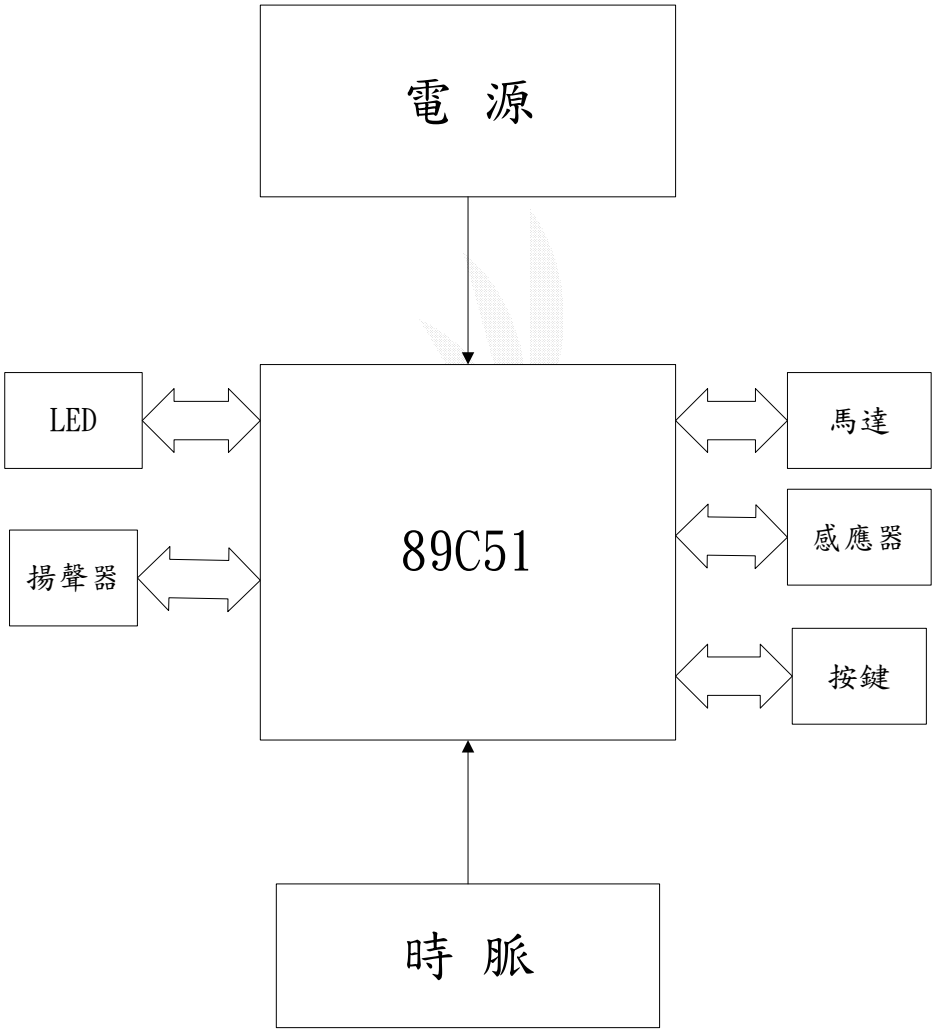
4) 大部份 C 語言的最佳化僅僅是針對該語言而已，並未對 8051 的程式碼進行最佳化。

5) 當 C 語言處理中斷時的速度並不慢,只是在使用時須瞭解其進入及返回原程式的步驟，以必免造成當機。

6) C 語言所寫的控制系統比組合語言而言，有更多不確定性存在，這些不確定性包含了我們程式寫法的執行錯誤(RUNTIME ERROR)，以及編譯程式的隱藏性 BUG，所以必需嚴謹的驗證程式的步驟。

第三章 電腦鼠介紹

電腦鼠就是一個獨立目主的運動型機器人。一個擁有獨立的電源，能夠自我思考、做判斷，並且能夠運動的機器人。它的目的在於找尋一個未知迷宮的終點，並且尋求找到最快路徑到達終點。



一隻電腦鼠的架構應分四大部分：

一、 電腦鼠的頭腦—微處理機

為了要達到走冗迷宮的目的，電腦鼠的頭腦必須能夠有以下功能

1) 記憶設計者軟體中預先所採取的策略。

2) 能夠接受並解讀感測器所傳進來的訊號。

3) 能夠根據預先設定好的情況與現在的狀況做比較，然後下指令控制馬達。

4) 最後能夠記憶迷宮的地圖。

二、 電腦鼠的眼睛—感測器

在電腦鼠常用的感測器有三種：

1)機械式感測器。

機械式感測器是一種稱為微動開關(Microswitch)的機械裝置。它的原理是利用只要很微小的接觸力，推動開關上的扳手，電路就會打開成為 ON 的狀態，電路就會將這個訊號送給微處理機，電腦鼠的大腦於是就會曉得現在的位置。

2)光感測器。

光感測器的原理跟超音波的原理相近，且構造簡單又有很快的反應能力，故光感測器是電腦鼠上最常用的感測器。光感測器的原理可以分成反射型和遮斷型兩種。

一、 反射型：所謂的反射型就是光的發射器和接收器位於偵測物體的同側，利用光線的反射原理是否有接收來判斷壁面的有無。

二、 遮斷型：而遮斷型則是光的發射器和接收器位於物體的兩側。當物體通過時，光線就被遮斷，以這樣的方式來決定壁面是否存在。

以下是感光元件的種類介紹：

	光感測的原理	靈敏度
光二極體	由於光照使得在 P-N 按合處產生光電流	較差
光電晶體	由於光照使得在基極和集極間產生 電流，，形成電晶體的基極電流	其次

光敏電阻	由於光照使得電阻發生改變	最佳

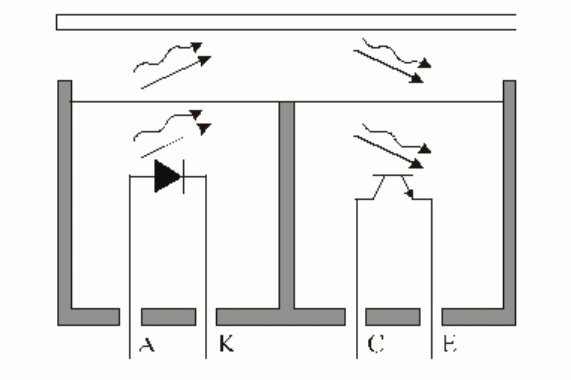
表　常用的光電轉換元件的特微及比較

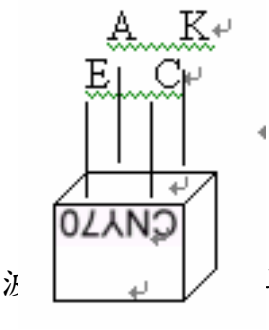
設計有三個紅外線光反射器型的光感測器，其編號分別為 R、C、L。電腦鼠希望做到將編號為 R 的光感測器用手遮蓋住時，七段顯示器 SEG1 的顯示值減 1，放開再蓋住時又再減 1；而將編號為 C 的光感測器用手遮蓋住時，七段顯示器的顯示值清除為 0；另外將編號為 L 的光感測器用手遮蓋住時，七段顯示器上的顯示值加 1，放開再蓋住時又再加 1。七段顯示器使用已編碼的控制埠 P2，而光感測器的資料讀入埠為 P0.0～P0.2，各別對應為 R、C、L。CNY70 的內部結構，其中包含紅外線發光二極體、光電晶體，以及光濾波器，其功能分別是：

1. 紅外線發光二極體：類似發光二極體(LED)的功能，當 PN 二端加上順向偏壓時可發出波長為 800 nm 的紅外線不可見光。

2. 光電晶體：為一個對紅外線波長具敏感反應的光偵測元件，當光電晶體受紅外線光照射時為低阻抗，而未受光時呈現高阻抗。

3. 光濾波器：為一僅讓波長為紅外線附近光譜通過的濾光透鏡，可用來加強光電晶體的抗雜訊能力（紅外線以外不可見與可見光的干擾）。





超音波耳

超音波麥可風接收由壁面反射回來的訊號，以決定壁面的有無。

三、 電腦鼠的腳—馬達

電腦鼠的馬達又可分為兩種，

1) 步進馬達。

2) 直流馬達。

下圖是步進馬達和直流馬達之比較：

	步進馬達	直流馬達
效率	效率低、笨重、體積大不易運轉	效率高，起動瞬間會吃很大電流，高速運轉時省電
轉矩	有靜止轉矩(靜止時有磁力固定住)	無靜止轉矩 起動轉矩大

	起動轉矩較小，加大負荷即出現同步失調(pulse loss) 高速運轉時力矩下降	高速運轉時力矩不降
控制性	開路控制即可(open loop)	需回饋控制(close loop) 要有好的控制程式或專用控制 IC
其他	在某些特定頻率會產生共振 價格較貴	有電刷雜音，使用久電刷會損耗

表　步進馬達和直流馬達的比較

四、　電腦鼠的智慧

而根據以前歷屆比賽可以得知一隻電腦鼠最起碼要具備以下三種智慧：

一、　迷宮搜尋的智慧—包括在行進中搜集壁面的資料,控制行進的動作和位置的修正，以及利用什麼的策略才能找到迷宮的終點，和如何去記錄迷宮的地圖。

二、　計算最快路徑的智慧—也就是如何由記錄好的地圖,計算出最

適於這隻電腦的最快路徑。

三、 找最快路徑上的衝刺的智慧—不同於迷宮搜尋時的行進，要如何能讓電腦鼠在最短時間內走完全程，而不失去平衡或迷失在迷宮之中。

而我們所使用的電腦鼠跟上述的一般電腦鼠的功能有以下相異之處：

一、 在電腦鼠的微處理器上，我們採取的 IC 是 AT89C51 這顆單晶片，且沒有其他的擴充 IC 來輔助，不像以前其他先進所做的電腦鼠那麼功能強大。

二、 在電腦鼠的感測器中，我們所採取的方式是用光感測器。因為光感測器在實作方面是比機械式感測器和超音波感測器來的簡易，且比較不受到外界干擾。

三、 在電腦鼠的馬達方面，我們所用的是直流馬達，不似以前人們所用的步進馬達，因為直流馬達的轉速是比較難控制的，不像步進馬達這麼容易控制使用。

四、 在電腦鼠的智慧方面，我們可以做到讓電腦鼠能夠有迷宮搜尋的智慧和能夠計算最快路徑這兩點。因為最快路徑上的衝刺需要能夠精準的控制馬達，而我們所用的直流馬達很難做到這一點。我們這個專題是以原本的只能根據貼好的路徑走的伺服器自走車

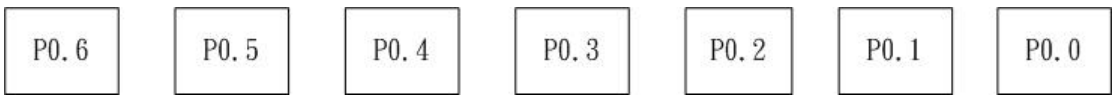
加以演變成現在的能夠搜尋迷宮的電腦鼠。

第四章 迷宮介紹與實驗說明

4.1 迷宮介紹

為了規劃我們的迷宮，我們要兼顧迷宮地形跟感應器的配合，迷宮太簡單沒有意義，而太複雜的地形感應器又偵測不到。除此之外，因為我們只有一排的 7 顆感應器，所以沒有辦法像普通電腦鼠一樣有一個延伸出去的前端的感應器，可以偵測牆壁，所以更增加我們規劃路徑的困難。

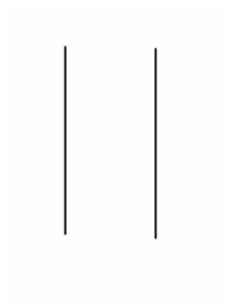
我們採用不會反光的黑色膠帶，貼在 8K 的白色厚紙板上，一個厚紙版為一個地形單位，然後我們把我們的迷宮路徑分為七大類，以這七大類為基礎就可以拼湊出很多迷宮地形來測試。以下是各別的介紹。



一、 直線：

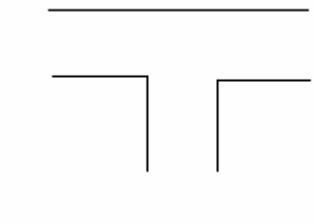
為了確保至少有一個感應器感測的到，我們的路徑厚度至少要大於 2.5cm，這樣才會至少讓一個感應器偵測的到，而兩路徑寬度則是根據感應器 P0.1 跟 P0.5 的距離。當感應器 P0.5 跟

P0.1 為黑色膠帶時，代表直線。



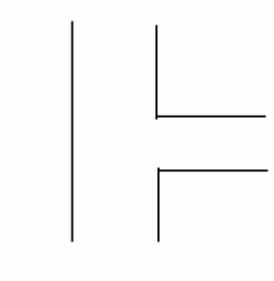
二、 T 字型路線

T 字型路徑的膠帶厚度跟寬度都跟直線一樣，我們是以當感應器什麼都沒感應到為感測到 T 字型路徑



三、 右邊有叉路

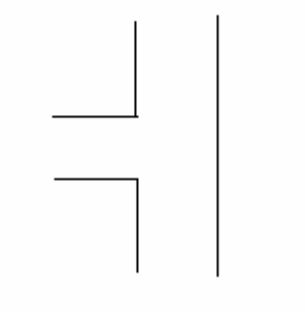
當感應器 P0.5 感應到黑色膠帶時，我們當成感測到右邊有叉路的地形。



四、 左邊有叉路

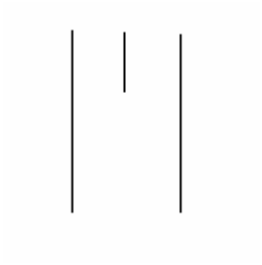
當感應器 P0.1 感應到黑色膠帶時，我們當成感測到左邊有

叉路的地形。



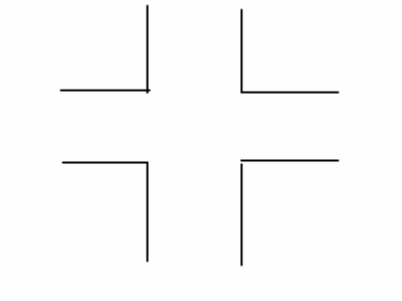
五、 斷路

當感應器 P0.5,P0.3,P0.1 感應到黑色膠帶時，我們當成感測到斷路的地形。



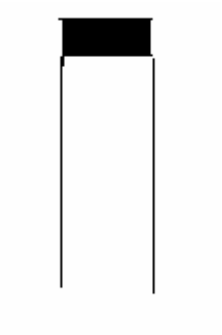
六、 十字型路徑

十字型路徑，我們是以當感應器什麼都沒感應到為感測到十字型路徑，跟 T 字型路徑一樣。



七、 終點

終點有別於斷路，斷路是感應器 P0.5,P0.3,P0.1 同時感應到，而終點則是感應器 P0.5,P0.4,P0.3,P0.2,P0.1 同時感應到。



4.2 實驗說明

這一個電腦鼠走迷宮的實驗是分成兩個步驟來進行。

一、 先搜尋整個迷宮的路徑。在搜尋迷宮路徑的方法是用資料結構裡的 Depth-First Search 來搜尋路徑並以右手法則來偵測路徑並記錄所經過的路徑。

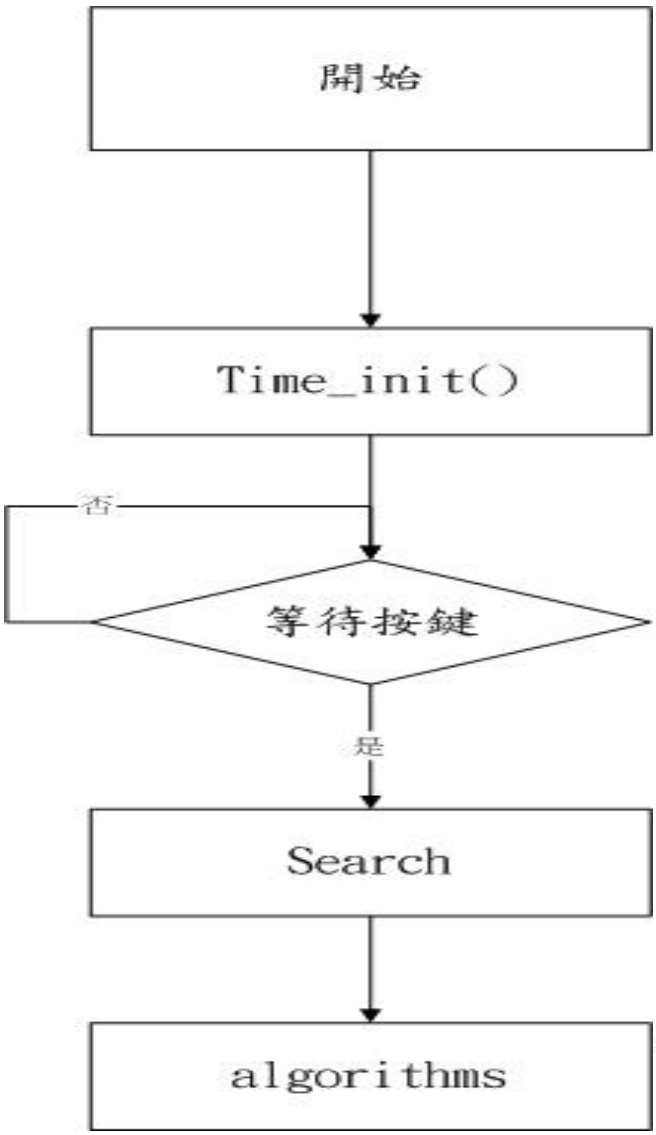
二、 再根據第一步驟所得到的路徑分析出最快的路徑。記錄電腦鼠到終點所經過的所有路徑的值丟到自己所設計的演算法處理，得到這個迷宮從起點到終點的最快路徑。

第五章　程式架構

5.1 主要程式

main()：

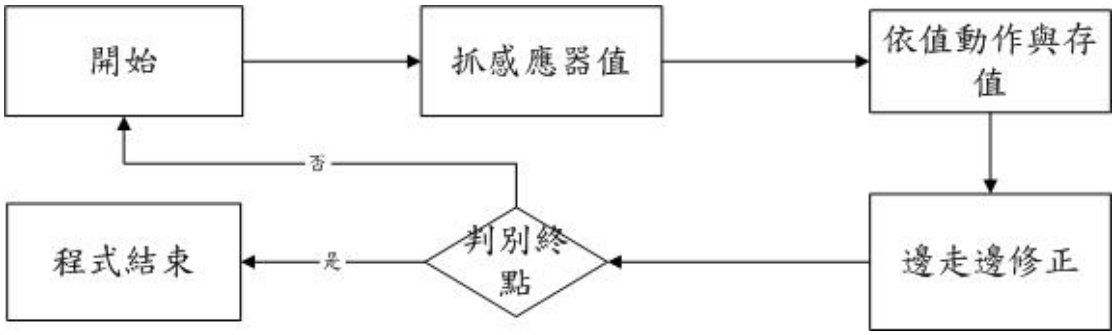
說明：當 89C51 reset 後，一開始會執行計時器初始化的動作，之後就一直等待使用者輸入按鍵，依照按鍵跳到對應的功能，search 完以後，他會直接跳到找最快路徑的步驟去，一直到到達終點才會結束。



search()：

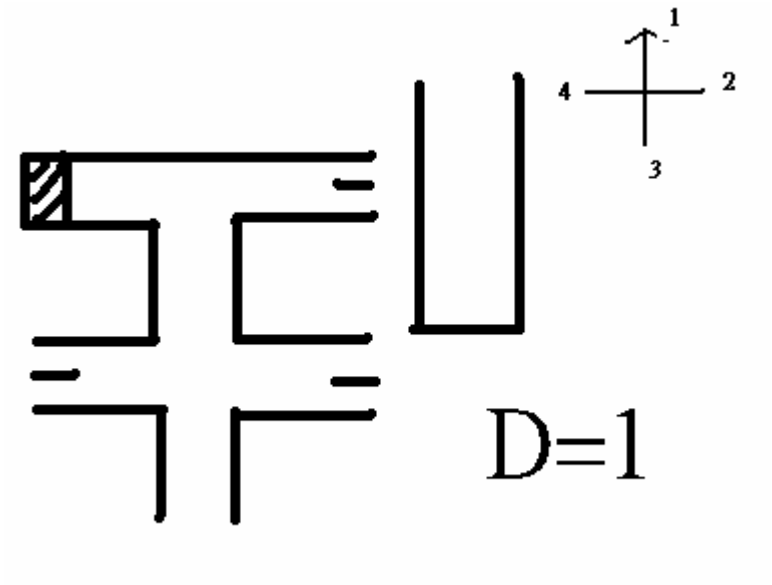
說明：search 步驟需要很多副程式來共同完成，譬如直走轉向、判斷路徑、儲存絕對方向等，這些會留到後面的副程式來各別介紹。Search 的主要流程如下，一開始他會他會抓感應器的值來判斷目前的地形，當直線他會直走，當有叉路時，他會依照 DFS 的右手法則的右轉->直走->左轉的優先權來轉向，轉向完他會將絕對方向存入陣列，之後就

會一直重複，如果到終點才會跳到找最快路徑步驟去執行。



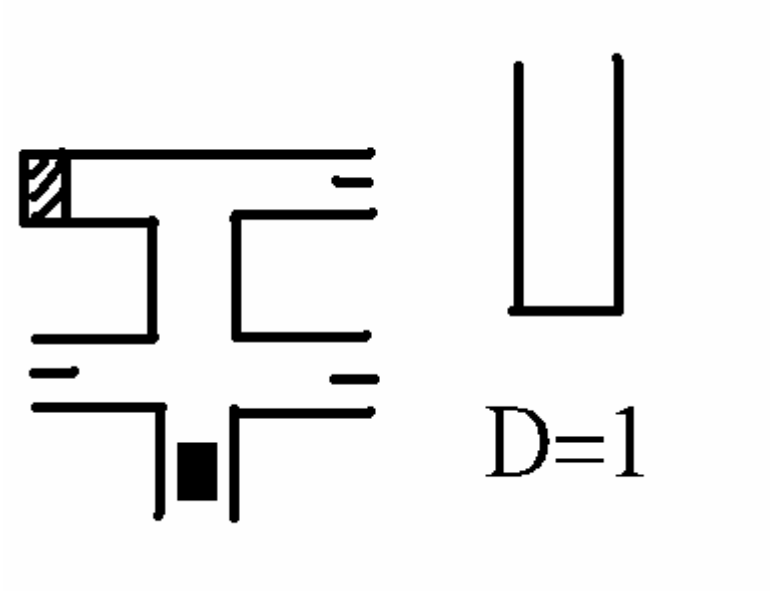
SEARCH 圖解

1.



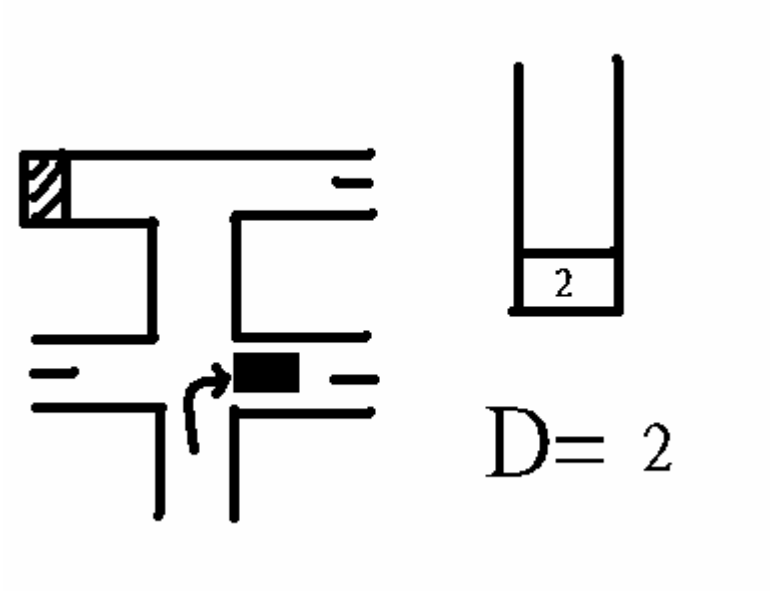
說明:D 表示目前的絕對方向

2.



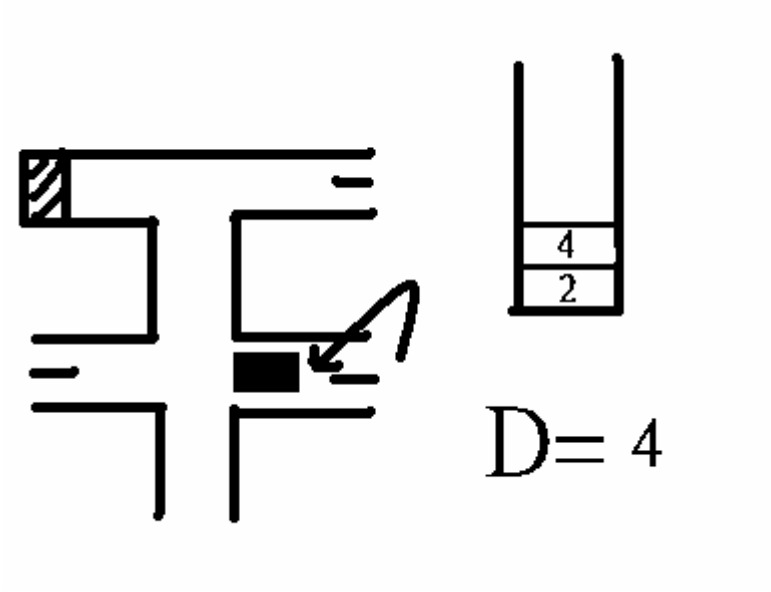
說明:我們定一開始的起點為方向 1，電腦鼠起點定在迷宮的最下方

3.



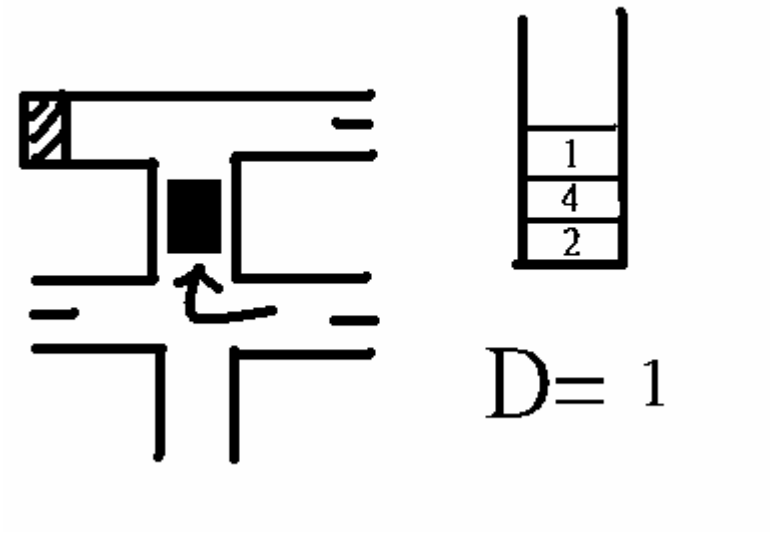
說明:當電腦鼠走到十字路口時，電腦鼠依 DFS 的右手法則向右轉，因為電腦鼠向右轉，所以我們將絕對方向轉換為 2，然後將 2 存入堆疊。

4.



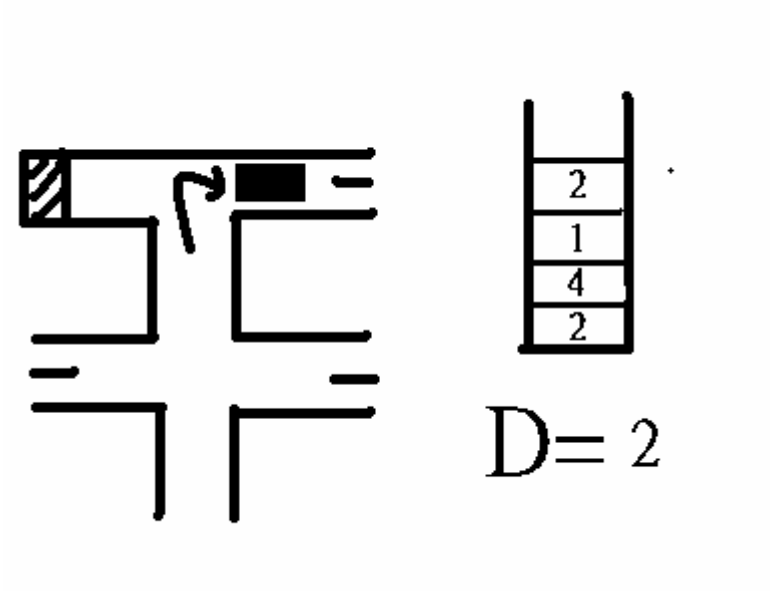
說明:當電腦鼠遇到斷路的時候，他會迴轉，同時將絕對方向設為 2 的相反->4，然後將他存入堆疊。

5.



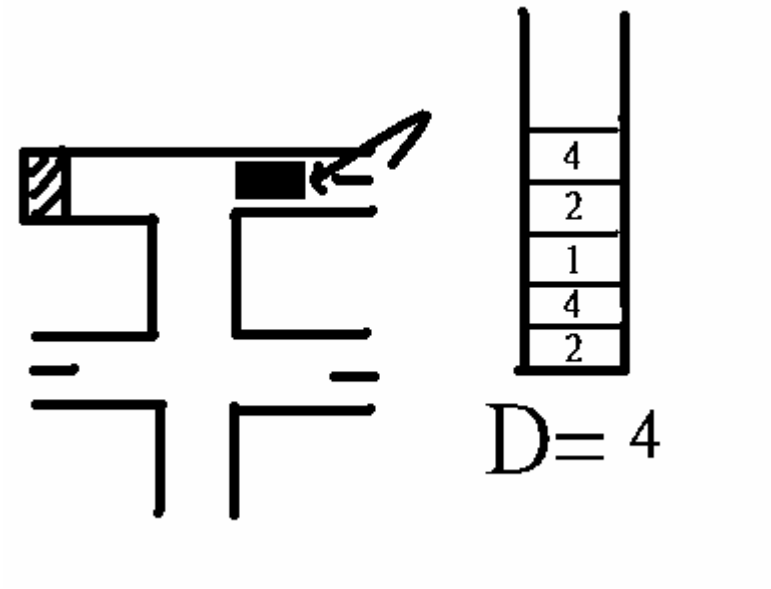
說明:當電腦鼠遇到十字路口時，他一樣依右手法則右轉，然後將方向轉換為 1，並將方向存入堆疊。

6.



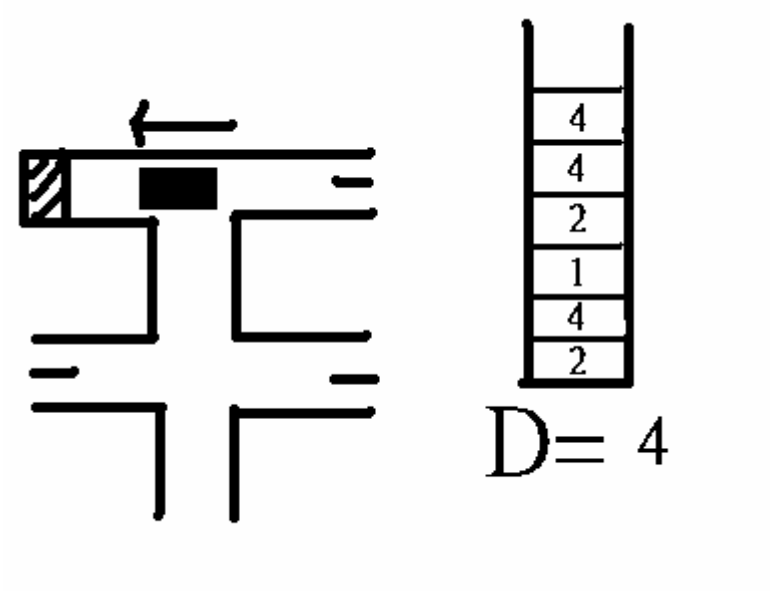
說明:當電腦鼠遇到 T 字行路口時，他依照右手法則再一次右轉，然後將轉換後的方向 2，存入堆疊。

7.



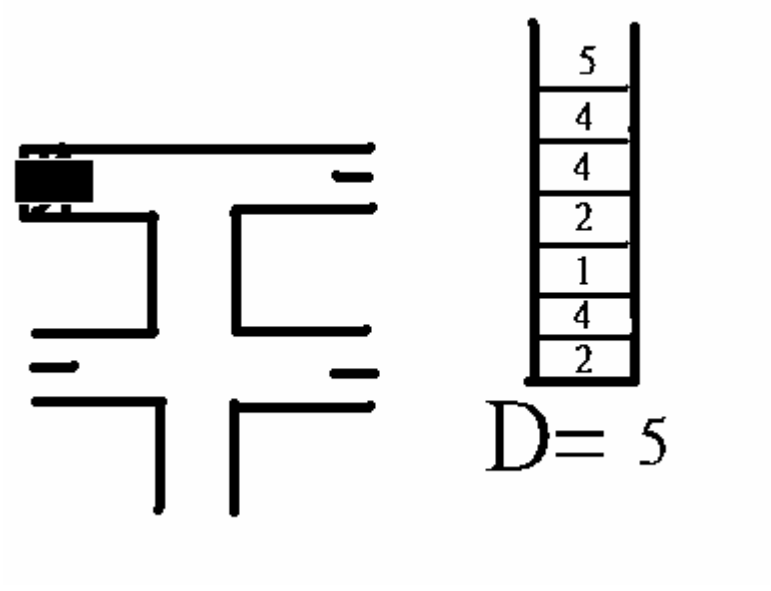
說明:當電腦鼠再次遇到斷路時，他一樣會迴轉，然後取相反的方向，存入堆疊。

8.



說明:當電腦鼠遇到以他的方向來說是左 T 字型的路徑時，他會依照 DFS 的右手法則定理，轉向的優先權為右轉>直走>左轉，所以這時候他會選擇直走，然後直接將目前的方向存入堆疊，不用轉換。

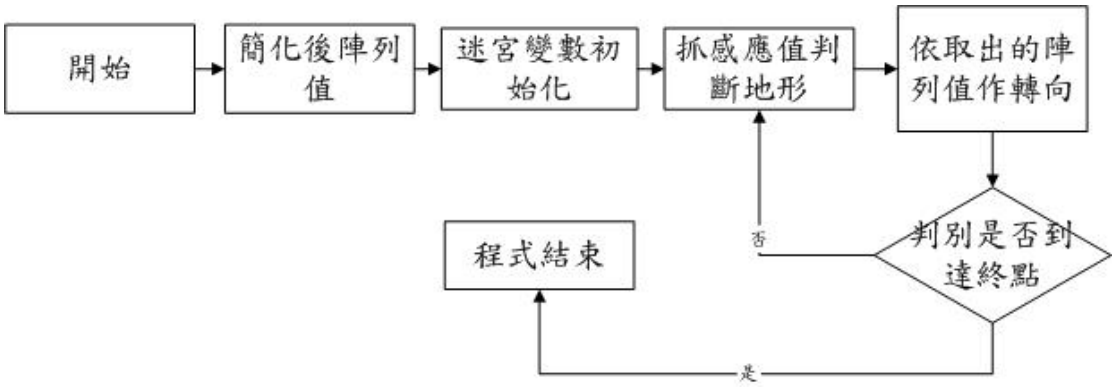
9.



說明:當電腦鼠到達終點時，他就會停止動作，然後將終點的代表符號(5)存入堆疊，有利於我們判斷，他是否抓到終點的地形。

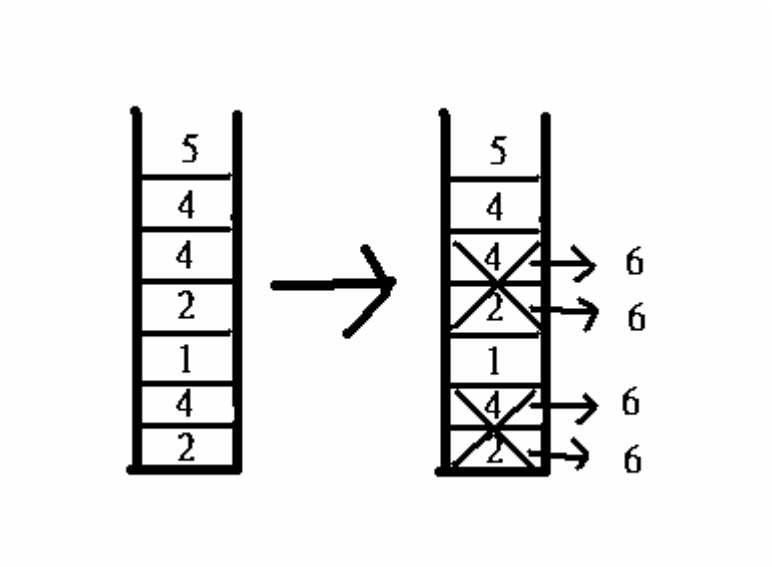
ALGORITHM()：

說明：找最快路徑步驟一開始會馬上處理 search 產出的陣列，產生最快的路徑。迷宮初始化，是為了讓電腦鼠再次重新從終點開始走。最快路徑的步驟跟 search 步驟大致相同，最大的不同是，當有叉路時，電腦鼠並不會依照 DFS 的右走法則來轉向，電腦鼠會把絕對路徑從陣列取出，並轉成相對路徑，同時依照相對路徑來轉向。



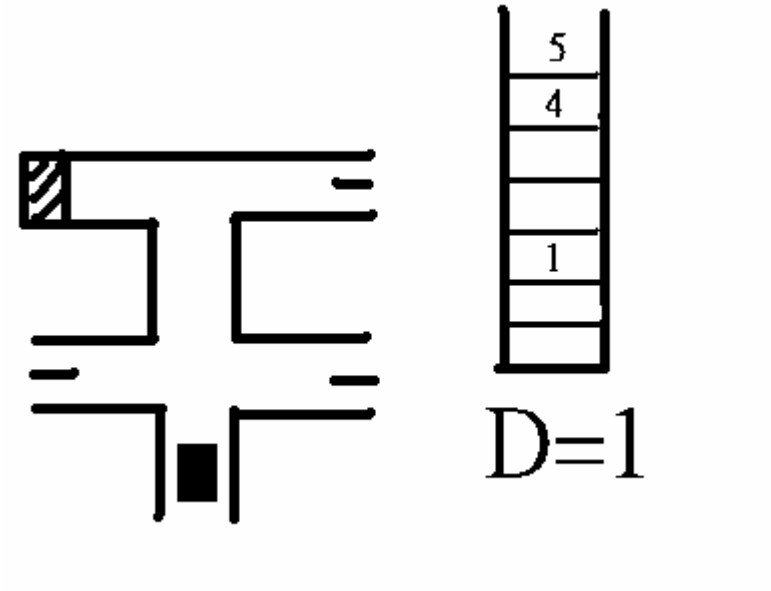
找最快路徑圖解

1.



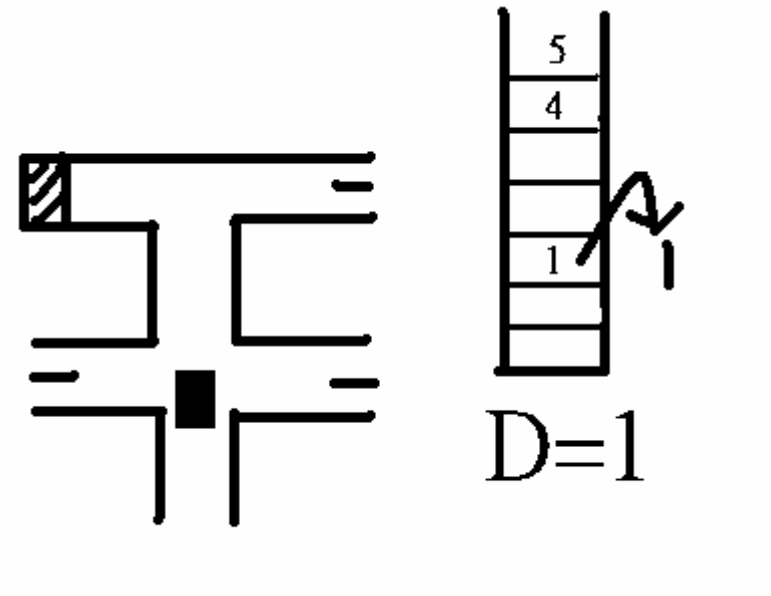
說明: 這是找最快路徑步驟最重要的一部分，他會處理 search 傳來的陣列，然後將相反方向的路徑消掉，其中如果是 24 跟 13 兩各值相連的話，就可以消掉，我們寫入 6 代表這各陣列的值已經被處理過了。

2.



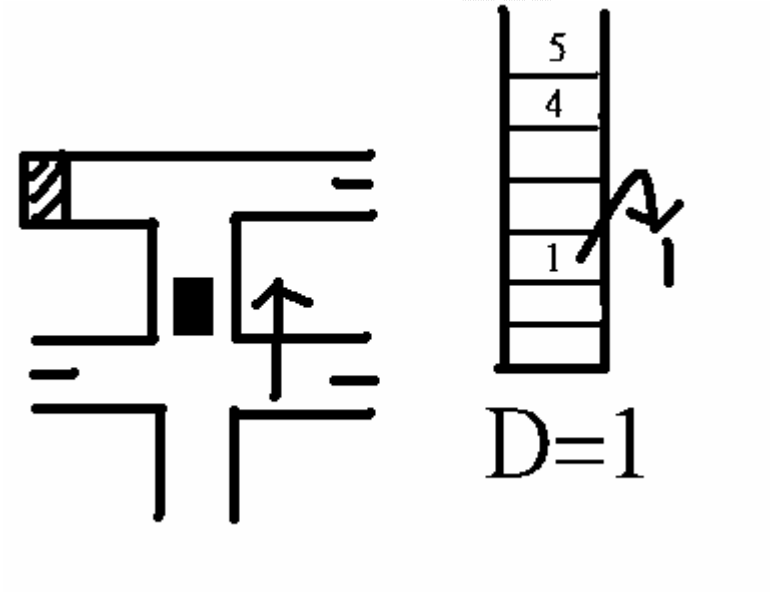
說明: 當 search 步驟做完時，我們將電腦鼠拉回原點，然後從原點開始作找最快路徑的步驟，其中 D 代表電腦鼠目前的方向，我們是預設 1，然後我們把 6 擦去不看，是為了陣列的簡潔易看。

3.



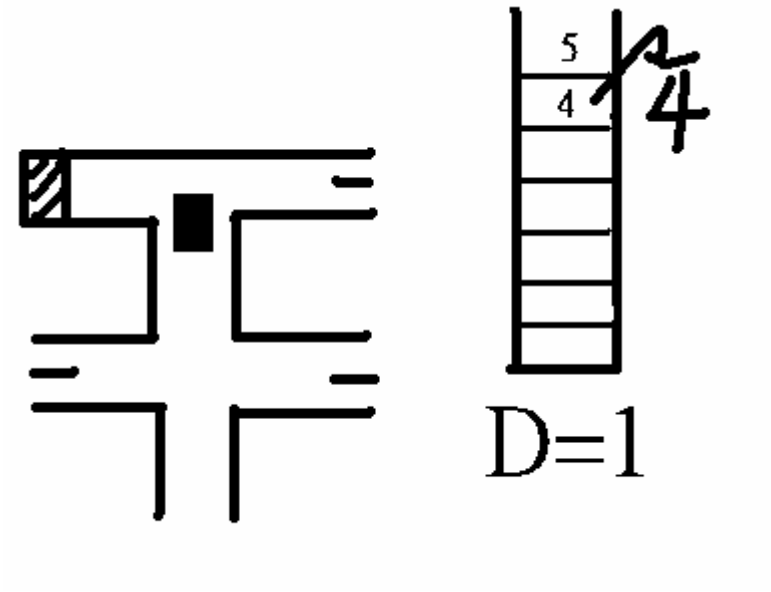
說明: 當電腦鼠走到十字路徑時，我們從陣列的底部開始往上找，將陣列裡面非 6 或 5 的值取出，所以我們取 1。

4.



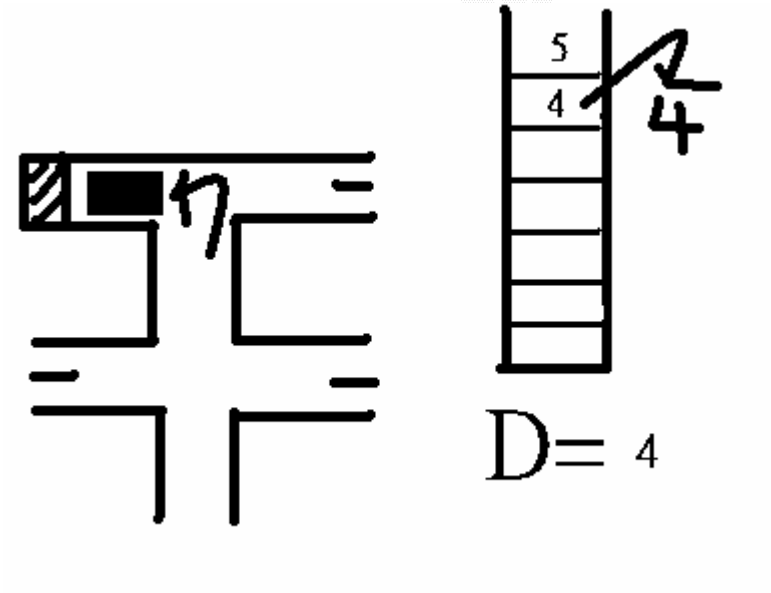
說明:現在這各步驟，我們將取出的值 1 跟目前的方向作比對，如果相同，他就會選擇直走。

5.



說明: 當電腦鼠再次遇到叉路時，他一樣會由上一次找到的陣列的指標開始往上找非 5 或 6 的值，所以這一次我們找到 4。

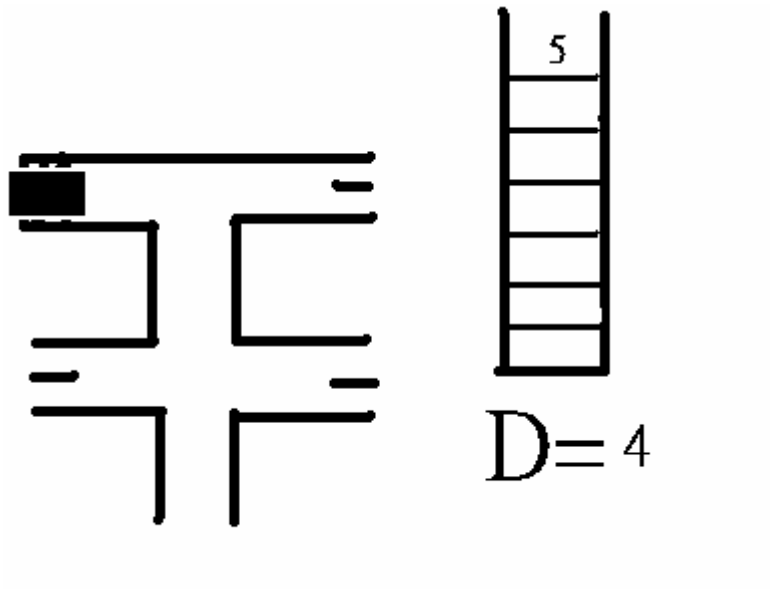
6.



說明:取出的 4 會與目前的方向作比對，目前我們的方向是 1，依照我們定的座標值，取出來的 4，代表電腦鼠要左轉，左轉完後，我們如同前面 search 步驟一樣，將目前的方向轉換為 4，只是轉換後，我們

並沒有存入陣列。

7.



說明: 此時當電腦鼠繼續走時，他到最後會偵測到終點的路徑，整各實驗就算完成。

5.2 副程式說明

計時器初始化

void time_init(void);

說明：將計時器 0 設定為每 0.1MS 執行一次，計時器 1 每 50MS 執行一次。

延遲 10ms

void delayX10ms(unsigned char);

說明：使用 for 迴圈做延遲，因為每個編譯器都不太一樣，所以只能用在不是很精確的延遲上。

第二個延遲 10ms

void delay2X10ms(unsigned char);

說明：因為怕計時中斷呼叫同一個 delayX10ms()，會產生程式錯亂，所以特別設一個延遲專給計時器呼叫。

揚聲器發聲

void beep(void);

說明：讓揚聲器產生單音。

Push 值

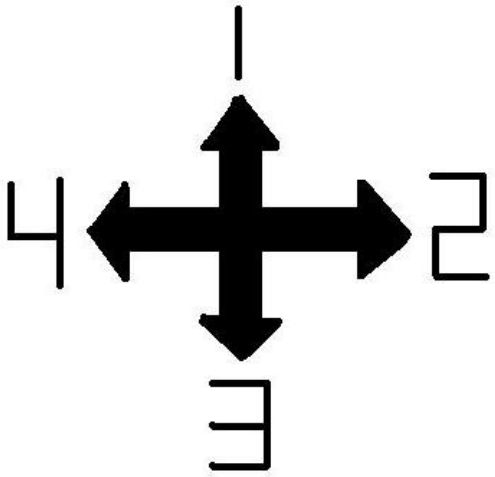
void push(unsigned char);

說明：假如有轉向，將絕對方向 push 到陣列。

修正絕對方向

void adjust_direction(unsigned char t);

說明：因為轉向後，絕對方向會改變，所以需要依照智慧鼠轉的方向，來修正絕對方向。



右轉->方向加一

左轉->方向減一

修飾陣列值

void modulate(void);

說明：除去陣列中多走的路徑。

演算法

void algorithm(void);

說明：行走的方式跟 seach 一樣，但是當他走到叉路時，他會從堆疊

取出路徑，去做轉向，而不是根據右手法則。

感應值

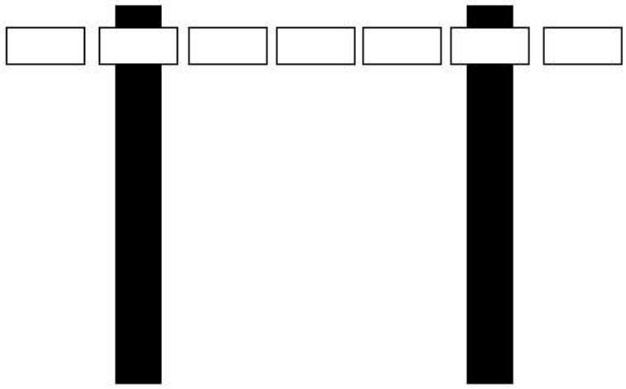
char sensor(void);

說明：抓取感應的值，回傳判斷出的路徑。

感應的值	判斷出的路徑
0x22、0x66、0x44、0x4C、0x33、0x15、0x1F	直線
0x2A、0x7E、0x540x7C、0x3F、0x15、0x1F	牆壁
0x00	T 或者是十字
0x02、0x04、0x60、0x07、0x03、0x01	左 T
0x20、0x10、0x18、0x30、0x60、0x40	右 T
0x1C、0x1E、0x0E、0x3C、0x38	終點
0x63、0x67、0x47、0x4F、0x73、0x71、0x79	最兩邊感應到

感應器與迷宮路徑的模擬圖

由右到左為感應器 P0.0 到 P0.6



演算法步驟的反應動作

void algorithm_action(unsigned char t);

說明：將陣列值取出，並將取出的絕對方向，轉換成智慧鼠的相對方向，然後轉向。

搜尋的反應動作

void action(unsigned char t);

說明：依照右手法則，右轉->直走->左轉，來判斷該轉哪個方向。

邊走邊修正

void correct_straight(void);

說明：依照感應器的來修正直走。

感應的值	判斷結果

0x4C	稍微偏右
0x44	有點偏右
0x66	嚴重偏右
0x19	稍微偏左
0x11	有點偏左
0x33	嚴重偏左

轉彎後修正

void correct_turn(void);

說明：依照感應器的來修正轉彎後的智慧鼠。

感應的值	判斷結果
0x08、0x18、0x0C	無法判斷，往前走一點點再測試一次
0x00	無法感應到，先往右轉，如果經過一段時間還未偵測到路徑，就改往右轉
0x10、0x30	有點偏右
0x20、0x60、0x40	嚴重偏右
0x04、0x06	有點偏左

0x02、0x03、0x01	嚴重偏左

走直線

void straight(void);

說明：- 用計時器中斷定時呼叫直走函式

- 控制馬達開關，限制速度

- 每 50ms 呼叫一次

右轉

void turnleft(void);

說明：使用計時器 1 控制馬達該走的時間較準確，轉向的同時有很多停頓，為的是讓馬達的慣性到最小。

左轉

void turnright(void);

說明：使用計時器 1 控制馬達該走的時間較準確，轉向的同時有很多停頓，為的是讓馬達的慣性到最小。

迴轉

void goback(void);

說明：左馬達轉 270 度，右馬達轉 90 度，就可以達到迴轉的目的。

第六章　結論

在做完這個專題實驗後，知道我們的電腦鼠已經可以作出搜尋迷宮且能夠根據之前所搜尋的迷宮找出最快路徑。但在多次測試過程之中，也發現到搜尋迷宮時，偶爾會有路徑判斷錯誤，導致不可預期的錯誤產生。經過我們討論的結果應該有兩種因素，會使感應器在抓路徑時，發生錯誤。一、因為車子有所偏離路徑，且微調並沒有馬上修正過來，導致在抓感應器值那瞬間會感應成其他路徑。二、在轉彎過程當中，因為車子的前輪卡到厚紙版上的路徑，導致無法順利過彎。我們初構的想法，軌道只有一條，但微鼠的直流馬達轉動似乎不是很同步，這導致我們的微鼠，在行走時會產生些微的歪斜，前進數步之後，進而偏離整個軌道，這使的感應值錯亂，產生到底是轉彎還是偏差的模糊界定。這便使我們決定更改整個地圖的製法，使用雙軌道，當車子在雙軌上行走時，行走偏差可以使程式做微調，遇到路口轉彎時，一邊感應值沒有感應到，程式就知道車子該轉彎了，這樣的做法，可以大幅度降低上述兩種問題的錯誤率，雖然沒有辦法百分之百決解，但是經過多次嘗試，取成功搜尋到終點的那一次。同樣的，在找最快路徑步驟時，車子也可能會偏離路徑，也是需要多次嘗試，就可以達成我們這次專題實驗的目的。

第七章 心得

劉明機個人心得：

看到日本的電子寵物，覺得滿有趣的，所以就打算專題做相關的題目，後來我一個人去找陳啟鏘老師，說我想做會走迷宮的電腦鼠以後，老師說我一個人會太少，可以做的東西有限，所以我就找了江佳霖跟程毓北一起做。

後來經過老師的建議，我們決定買別人的產品來實做電腦鼠，而不是自己做。所以我們就在網路上找到了益眾科技，知道他們是做電腦鼠方面，滿有名的一家公司。我們就決定跟他們買，不過我們買的時候，沒有問清楚他們的規格，導致後面實作上的重重困難。同時我們從老師那邊挖來一台 WINICE 8051/52-E 的 ICE，老師說這是以前買的，可是之前的學長說這台可能壞掉了，叫我們回去用用看，只是沒想到這台的燒錄功能，是我們之後專題不可或缺的工具。

暑假初，我們買回來智慧鼠以後，決定先從他原有的功能研究，直到我們摸熟機器後，再發展出我們的智慧鼠。一開始因為我們還不是很懂 8051 的相關東西，連他機器有問題我們也不知道，就傻傻的在哪邊測試，後來才知道他的變壓器有問題，浪費了我們很多時間。

當我們已經摸熟機器跟他原有的功能以後，就已經九月多了，會

摸這麼久的原因，是因為我們不知道他的變壓器壞了，導致馬達無法正常運作，還一直以為是我們程式的問題，之後我們把機器送回去給他們修，當他們把機器送回來以後，我們的專題才算真的開始在做，不過之前的努力並沒有白費，反而因為我們已經把 8051 的基礎打好了，之後開發起來也滿快的。

因為買回來的智慧鼠的硬體上有太大的限制，像他只有一排感應器，沒有辦法感應到前端的路徑，他的馬達是直流馬達，沒有辦法很精確的控制所走的距離，馬達只能直走不倒轉，馬達轉太快，轉彎會過頭等，做這個專題讓我了解到硬體對軟體上的功能限制有多麼大，我們花在調整轉彎的方向，跟測試抓到的路徑值，就花了大半的時間，可以說是浪費很多時間在無意義的東西上，幾度導致我們想放棄留到下學期做，可是陳啟鏘老師鼓勵我們，說希望我們上學期做出來，我們也不想拖到下學期，所以還是一同努力去做，到最後雖然沒有做到百分之百，但是終於還是做出來了。所以很感謝陳啟鏘老師的指導，跟益眾科技的員工，耐心地回答我們的問題，還有江佳霖、程毓北的共同參與專題製作，才能完成這個專題。

江佳霖個人心得：

一開始決定做電腦鼠這個專題時，因為什麼都不太懂，就從圖書

館裡找了許多有關於電腦鼠的書，也在網路上找了之前電腦鼠比賽的資料。了解到一隻電腦鼠應該有那些功能和那些零件，知道這一點後也面臨了該決則買零件來做電腦鼠或者該直接買成品來做這個專題？這個時候我們問陳啟鏘老師，該如何決擇。因為資工系對於硬體上的學習僅有一些基本理論上的教學，所以要作出一隻電腦鼠可能要花費很多時間。因此老師是建議我們直接買成品來做。我們也從網路上找有賣電腦鼠的資料，也找到益眾科技這間公司。看了他們的電腦鼠的價目表，才知道原來電腦鼠的硬體真的是蠻貴的，從 3000 多到 10000 多的價錢都有。最後是買了 4000 多的伺服器自走車，拿到成品時，才知道伺服器自走車它只是純 8051 單晶片的電腦鼠，沒有任何的擴充插巢，這麼一來這台伺服器自走車勢必無法做到走迷宮的功能，當時知道這件是後就一直不知道該如何是好。這個時候我們曾想過換題目，但錢都花了，所以就硬著頭皮繼續作下去。一開始因為對伺服器自走車的一些功能和抓感應器值這方面的問題花了蠻多時間。後面則是為了讓伺服器自走車能夠精準的走，必需不斷的測試，改程式、燒錄、再測試。最後把原本的伺服器自走車改成有初步功能的電腦鼠。做完這個專題之後才知道軟體跟硬體的配合是相當重要，當發現錯誤時必需去查看是軟體或硬體導致錯誤發生。如果不懂一些硬體的基本觀念

的話，想要進一步去用軟體來操作硬體是不可能的事情。我也從這個專題裡了解到彼此分工的重要性，因一開始沒有分工，大家都隨便做，導致進度落後。幸好最後了解到這一點，才能完成這個專題。最後要感謝老師的指導跟鼓勵，我們才能做完這個專題。

程毓北個人心得：

8051 所發展的系統，廣泛應用在各界，其中以及時系統中的控制為多常常利用在 3C 家電控制、工業機械的控制器、車輛的零件等等.. 這些都是需要對外在環境立即作反應的設計，體積小、執行效率快，正好符合業界的『快、好、省』需求。然而，新一代的需要，單晶片已經不敷現代使用的需求，INTEL 早在八零年代末期就收手不作 8051，他們意識到 8051 的發展到了極限，站在研發的觀點，這已經到了瓶頸了，技術必須再突破，就必須捨棄傳統的做法。儘管留下來的 8051 系列背後，還是有很多賺頭‥ 。隨著科技進步，現在許多 IC 大廠都忙著研發新的製程，新一代的 IC 內嵌 System 核心(SOC)已成為 IC 設計主流， SOC 加強了系統整合能力其集成式的功能大大降低開發設計的週期，面對未來更多變更智慧化的潮流，SOC 展現更寬闊的發展前景。這次的專題中，我學習到很多關於單晶片方面的知識，學習了之後有了更深刻的感受，不是感到學習了很多，反而看到了許多應用

技術，才驚覺學習的路還很長。台灣目前面臨企業轉型， 朝向精密工業發展，未來幾年，台灣將成為世界 IC 研發中心，這類的資訊豐富，研究也很多 CHIP 的未來展望，倒是給自己一條明路走。

最後感謝陳啟鏘 指導老師　的技術指導以及使用器材。

老師的督導讓我們能如期順利的完成。

以及同組的同學的賣力演出讓這部作品如期推出...。

附錄一：參考文獻

[1] 王其宏、李啟誌、林宏宇，動手做電腦鼠—設計原理與實作，全欣資訊圖書，81 年 3 月 2 日 初版。

[2] 吳一農、林家德，電腦鼠製作—神通鼠秘笈，松崗電腦圖書資料股份有限公司，81 年 6 月 初版。

[3] 蔡朝洋，單晶片微電腦 8051/8951 原理與應用，全華科技圖書股份有限公司，91 年 7 月 初版。

[4] 楊明豐，8051 單晶片設計實務—組合語言版，碁峯資訊股份有限公司，2003 年 04 月 二版。

[5] 楊明豐，8051 單晶片 C 語言設計實務—使用 Keil C，碁峯資訊股份有限公司，2003 年 03 月 初版。

[6] 陳明熒，單晶片 8051 實作入門—最新版，文魁資訊股份有限公司，2002 年 8 月 初版。

[7] Han-Way Huang，Using the MCS-51 Microcontroller，OXFORD UNIVERSITY PRESS，2000 年。

[8] 林伸茂，8051 單晶片—徹底研究經驗篇，旗標出版股份有限公司，91 年 7 月 初版

附錄二：程式碼

```
//9212011 感應器的值要反轉..終點的表示法要變動..
//9212031 將 mouseState 改為 mouseState..加入 ab_direction..開始做 search..
//9212051 將陣列值反映到 led 上..微調 for 電池..top[0]當陣列第一個
//          當最兩邊感應到時..往前走..直到空白或終點為止
//          轉彎修正 18 0c 會重複到..也要當往前走
#include <AT89X51.H>

//----------全域變數----------

unsigned char time0_temp=0;
unsigned char time1_temp=0;
unsigned char mainState=0;        //功能鍵
unsigned char mouseState=0;       //電腦鼠的狀態
unsigned char ab_direction=1;     //絕對方向

bit correct=0;                    //禁止中斷影響
//bit norepeat=0;                     //用在左 T..讓他不要重複存值
bit GOTOEND=0;                      //走到終點

unsigned char r=20;                 //轉 90 度所需要的時間 25->22->20->15

unsigned char step=0;             //走幾格
unsigned char littleStep=1;       //走一小步
unsigned char countStep=0;        //4in1 記步器

unsigned char maxStack=30;
unsigned char stack[30]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
    //stack[0]不用
unsigned char top=0;              //top=0 當作 pop 底
unsigned char top2=0;

//----------函數宣告----------
void time_init(void);
void delayX1ms(unsigned char);
void delayX5ms(unsigned char);
```

```
void delayX10ms(unsigned char);
void delay2X10ms(unsigned char);
void beep(void);

void push(unsigned char);
//char pop(void);

void adjust_direction(unsigned char t);

void modulate(void);
void algorithm(void);

char sensor(void);

void algorithm_action(unsigned char t);
void action(unsigned char t);
void correct_straight(void);
void correct_turn(void);
void straight(void);
void turnleft(void);
void turnright(void);
void goback(void);
//-----------------------------

void main(){
//-----------宣告區-------------
      unsigned char temp=0;
      unsigned char temp2=0;
      unsigned char i=0;
      bit   temp_bit=0;
//-----------初始化區-----------
      time_init();
//------------------------------
      while(mainState==0){
            P3=0xff;
            temp=P3;
            temp=~temp;
```

```
            if (temp==0x01){            //test1 感應器輸出到 led
                mainState=1;
            }
            else if (temp==0x02){       //test2 動作測試
                mainState=2;
            }
            else if(temp==0x04){        //test3 地形偵測
                mainState=3;
            }
            else if(temp==0x08){        //test4 地形判斷
                mainState=4;
            }
            else if(temp==0x10){        //test5 連續地形判斷
                mainState=5;
            }
            else if(temp==0x20){
                mainState=6;            //test6 邊走邊修正+連續地形判斷
            }
            else if(temp==0x40){
                mainState=7;            //test7 搜尋
            }
            else if(temp==0x80){
                mainState=8;            //test8
            }
            else{}
        }
    //------------test1-----------------
        while(mainState==1){
            P2=P0;
        }
    //------------test2-----------------
        while(mainState==2){
            P3=0xff;
            temp=P3;
            temp=~temp;
            if (mouseState!=0){//有動作正在進行中
                continue;
```

```
            }
            switch(temp){
                case 0x01:          //直走一格
                    mouseState=1;
                    straight();
                break;
                case 0x02:          //左轉
                    mouseState=2;
                    turnleft();
                break;
                case 0x04:          //右轉
                    mouseState=3;
                    turnright();
                break;
                case 0x08:          //迴轉
                    mouseState=4;
                    goback();
                break;
                default:;
            }
        }
//------------test3-----------------
        while(mainState==3){
        }
//------------test4-----------------
        while(mainState==4){
        }
//------------test5-----------------
        while(mainState==5){
        }
//------------test6-----------------
        while(mainState==6){
        }
//-------------test7----------------
        while(mainState==7){
            P3=0xff;
            temp=P3;
```

```
temp=~temp;
if (temp==0x01){
    while (GOTOEND==0){
        if(mouseState==1){
            temp2=sensor();
            if (temp2!=0){
                action(temp2);
            }else {}

            correct_straight();
        }else if(mouseState==0){
            step=10;
            mouseState=1;
            straight();
        }else {}
        delayX10ms(1);
        P2=~((top&0x0f)|(stack[top-1]<<4));
    }
    i=0;
    do{
        P2=~0x00;
        delayX10ms(10);
        P2=~((i&0x0f)|(stack[i]<<4));
        if (i>=top){
            i=0;
        }else {
            i++;
        }
        delayX10ms(250);
        P3=0xff;
        temp=P3;
        temp=~temp;
    }while(temp!=0x02);
    beep();
    beep();
    beep();
```

```
                algorithm();
            }
            else {}
        }

//-------------test8-------------------
        while(mainState==8){
        }
//---------------------------------------
}
void algorithm_action(unsigned char t){
//      P2=~0x00;
//      delayX10ms(10);
        while(stack[top2]==6){
            top2++;
        }
        P2=~stack[top2];
        if(t==3||t==4||t==5){
            switch(stack[top2]){
                case 1:
                    if (ab_direction==4){//right
                        mouseState=3;
                        turnright();
                        correct_turn();
                        ab_direction++;
                    }
                    else if(ab_direction==2){//left
                        mouseState=2;
                        turnleft();
                        correct_turn();
                        ab_direction--;
                    }
                    else {
                    }
                    break;
                case 2:
                    if (ab_direction==1){//right
```

```
                mouseState=3;
                turnright();
                correct_turn();
                ab_direction++;
        }
        else if(ab_direction==3){//left
                mouseState=2;
                turnleft();
                correct_turn();
                ab_direction--;
        }
        else {
        }
        break;
case 3:
        if (ab_direction==2){//right
                mouseState=3;
                turnright();
                correct_turn();
                ab_direction++;
        }
        else if(ab_direction==4){//left
                mouseState=2;
                turnleft();
                correct_turn();
                ab_direction--;
        }
        else {
        }
        break;
case 4:
        if (ab_direction==1){//left
                mouseState=2;
                turnleft();
                correct_turn();
                ab_direction--;
        }
```

```
                else if(ab_direction==3){//right
                    mouseState=3;
                    turnright();
                    correct_turn();
                    ab_direction++;
                }
                else {
                }
                break;
            default:
                ;
        }
        top2++;
        if (ab_direction==0){
            ab_direction=4;
        }
        else if(ab_direction==5){
            ab_direction=1;
        }
        else{
        }
    }
    else if(t==6){
        GOTOEND=1;
        mouseState=0;
        P1_0=1;
        P1_1=1;
    }
    else{
    }
}
void action(unsigned char t){
    unsigned char temp;
    switch (t){
    case 1://直線
        break;
    case 2://牆
```

```
        mouseState=4;
        goback();
        correct_turn();
        adjust_direction(2);
        push(ab_direction);
        break;
case 3://T
        mouseState=3;
        turnright();
        correct_turn();
        adjust_direction(3);
        push(ab_direction);
        break;
case 4://左 T
        adjust_direction(4);
        push(ab_direction);
        while ((temp = sensor())!=1){
            if(mouseState==0){
                step=10;
                mouseState=1;
                straight();
            }
        }
        break;
case 5://右 T
        mouseState=3;
        turnright();
        correct_turn();
        adjust_direction(5);
        push(ab_direction);
        break;
case 6://終點
        break;
case 7://最兩邊感應到
        correct=1;
        P1_0=0;
        P1_1=0;
```

```
            delayX10ms(2);
            P1_0=1;
            P1_1=1;
            correct=0;
            if ((temp = sensor())==4){
                goto action_7;
            }

            temp = sensor();
            while(!(temp==3||temp==6)){//直到 T 或終點..才會繼續感應下一個值
                if(mouseState==0){
                    step=10;
                    mouseState=1;
                    straight();
                }
                temp = sensor();
            }
            if (temp==6){
                GOTOEND=1;
                mouseState=0;
                P1_0=1;
                P1_1=1;
                adjust_direction(6);
                push(ab_direction);
            }
action_7:
            break;
        default:
            beep();
        }
}
void adjust_direction(unsigned char t){
        switch (t){
            case 1:
                break;
            case 2://牆
                if (ab_direction==1){
```

```
            ab_direction=3;
        }
        else if (ab_direction==2){
            ab_direction=4;
        }
        else if (ab_direction==3){
            ab_direction=1;
        }
        else if (ab_direction==4){
            ab_direction=2;
        }
        else {
            beep();
        }
        break;
    case 3://T
        ab_direction++;
        if (ab_direction==0){
            ab_direction=4;
        }
        else if(ab_direction==5){
            ab_direction=1;
        }
        else {}
        break;
    case 4://左 T
        break;
    case 5://右 T
        ab_direction++;
        if (ab_direction==0){
            ab_direction=4;
        }
        else if(ab_direction==5){
            ab_direction=1;
        }
        else {}
        break;
```

```
            case 6://終點
                ab_direction=5;
                break;
            default:
                ;
        }
}
void modulate(){
        bit modulation=0;
        unsigned char i=0,j=0;

modulateStart:

        while (stack[i]!=5&&i<top){
            while (stack[i]==6&&i<top){
                i++;
            }
            j=i+1;
            while (stack[j]==6&&j<top){
                j++;
            }

            if (stack[i]==1&&stack[j]==3){
                stack[i]=6;
                stack[j]=6;
                modulation=1;
                i++;
            }
            else if (stack[i]==3&&stack[j]==1){
                stack[i]=6;
                stack[j]=6;
                modulation=1;
                i++;
            }
            else if (stack[i]==2&&stack[j]==4){
                stack[i]=6;
                stack[j]=6;
```

```
                modulation=1;
                i++;
            }
            else if (stack[i]==4&&stack[j]==2){
                stack[i]=6;
                stack[j]=6;
                modulation=1;
                i++;
            }
            else {
                i++;
                while (stack[i]==6&&i<top){
                    i++;
                }
            }
        }

        if (modulation==1){
            i=0;
            j=0;
            modulation=0;
            goto modulateStart;
        }
        else {
//          goto mainState3;
        }
}
void algorithm(){
        unsigned char temp=0;
        unsigned char i=0;
//---------------------------------
        modulate();//先處理陣列的值得到最快路徑
        i=0;
        do{
            P2=~0x00;
            delayX10ms(10);
            P2=~((i&0x0f)|(stack[i]<<4));
```

```
            if (i>=top){
                i=0;
            }else {
                i++;
            }
            delayX10ms(250);
            P3=0xff;
            temp=P3;
            temp=~temp;
        }while(temp!=0x01);

//-------search initial--------------
        mouseState=0;
        GOTOEND=0;
        correct=0;
        ab_direction=1;
        top2=0;//指到目前的陣列
//-----------------------------------
        while (GOTOEND==0){
            if(mouseState==1){
                temp=sensor();
                if (temp!=0){
                    algorithm_action(temp);
                }else {}
                correct_straight();
            }
            else if(mouseState==0){
                step=10;
                mouseState=1;
                straight();
            }else {}

            delayX10ms(1);
        }

}
char sensor(){
```

```
    unsigned char temp;

    P0=0xff;
    temp=P0&0x7F;          //把 P0.7 去掉
    if (temp==0x22||                                    //直線
        temp==0x66||temp==0x44||temp==0x4C||
        temp==0x33||temp==0x15||temp==0x1F){
        return 1;
    }
    else if(temp==0x2A||
            temp==0x7E||temp==0x54||temp==0x7C||
            temp==0x3F||temp==0x15||temp==0x1F){        //牆
        return 2;
    }
    else if(temp==0x00){                                //T or 十字
        return 3;
    }
    else if(temp==0x02||
            temp==0x04||temp==0x60||temp==0x07||
            temp==0x03||temp==0x01){                    //右 T->左 T
        return 4;
    }
    else if(temp==0x20||
            temp==0x10||temp==0x18||temp==0x30||                    //去掉
temp==0x30 0x20
            temp==0x60||temp==0x40){                    //左 T->右 T
        return 5;
    }
    else if(temp==0x1C||
            temp==0x1E||temp==0x0E||//temp==0x0F||temp==07||
            temp==0x3C||temp==0x38){//temp==0x78||temp==70){    //終點
        return 6;
    }
    else if(temp==0x63||
            temp==0x67||temp==0x47||temp==0x4F||
            temp==0x73||temp==0x71||temp==0x79){          //最兩邊感應到
            return 7;
```

```
        }
        else{                                                    //nothing
            return 0;
        }
    }
    void correct_straight(void){
        unsigned char right=5;
        unsigned char left=5;
        unsigned char temp;

        P0=0xff;
        temp=P0&0x7F;                                    //把 P0.7 去掉
        if (temp==0x4C){
            correct=1;              //禁止計時中斷影響
            delayX10ms(10);
            P1_0=1;
            P1_1=0;                                      //左馬達停一下 讓右馬
    達轉回
            delayX10ms(left);
            P1_1=1;
            delayX10ms(10);
            correct=0;
        }
        else if(temp==0x44){
            correct=1;              //禁止計時中斷影響
            delayX10ms(10);
            P1_0=1;
            P1_1=0;                                      //左馬達停一下 讓右馬
    達轉回
            delayX10ms(left-2);
            P1_1=1;
            delayX10ms(10);
            correct=0;
        }
        else if(temp==0x66){
            correct=1;              //禁止計時中斷影響
            delayX10ms(10);
```

```
            P1_0=1;
            P1_1=0;                                    //左馬達停一下 讓右馬
達轉回
            delayX10ms(left-3);
            P1_1=1;
            delayX10ms(10);
            correct=0;
        }
        else if(temp==0x19){
            correct=1;
            delayX10ms(10);
            P1_1=1;
            P1_0=0;
            delayX10ms(right);
            P1_0=1;
            delayX10ms(10);
            correct=0;
        }
        else if(temp==0x11){
            correct=1;
            delayX10ms(10);
            P1_1=1;
            P1_0=0;
            delayX10ms(right-2);
            P1_0=1;
            delayX10ms(10);
            correct=0;
        }
        else if(temp==0x33){
            correct=1;
            delayX10ms(10);
            P1_1=1;
            P1_0=0;
            delayX10ms(right-3);
            P1_0=1;
            delayX10ms(10);
            correct=0;
```

```
        }
        else {
            //return 0;
        }
    }
    void correct_turn(void){
        unsigned char right=10;
        unsigned char left=10;
        unsigned char temp;
        unsigned char temp2=0;
        bit turn=0;                 //0->先右轉..1->左轉
        correct=1;

    correct_start:

        P0=0xff;
        temp=P0&0x7F;//把 P0.7 去掉
        switch (temp){
            case 0x08:                      //0001000 往前走一點點再測試一次
            case 0x18:
            case 0x0C:
                delayX10ms(100);
                P1_0=0;
                P1_1=0;
                delayX10ms(1);
                P1_0=1;
                P1_1=1;
                delayX10ms(100);
                goto correct_start;
                break;
            case 0x00:
                while(temp==0x00){
                    if (turn==0){
                        delayX10ms(100);
                        P1_1=1;
                        P1_0=0;
                        delayX10ms(left);
```

```
                    P1_0=1;
                    delayX10ms(100);
                    if (temp2++>4){
                        turn=1;
                        temp2=0;
                    }

                }
                else {
                    delayX10ms(100);
                    P1_0=1;
                    P1_1=0;
                    delayX10ms(left);
                    P1_1=1;
                    delayX10ms(100);
                    if (temp2++>4){
                        beep();
                    }
                }
                delayX10ms(1);
                P0=0xff;
                temp=P0&0x7F;
            }
//              beep();
            goto correct_start;
            break;
        case 0x10:                  //偏右 part1
        case 0x30:
            delayX10ms(100);
            P1_0=1;
            P1_1=0;                 //左馬達停一下 讓右馬達轉回
            delayX10ms(left);
            P1_1=1;
            delayX10ms(100);
            break;
        case 0x20:                  //偏右 part2
        case 0x60:
```

```
            case 0x40:
                delayX10ms(100);
                P1_0=1;
                P1_1=0;                 //左馬達停一下 讓右馬達轉回
                delayX10ms(left+5);
                P1_1=1;
                delayX10ms(100);
                break;
            case 0x04:                  //偏左 part1
            case 0x06:
                delayX10ms(100);
                P1_1=1;
                P1_0=0;
                delayX10ms(right);
                P1_0=1;
                delayX10ms(100);
                break;
            case 0x02:                  //偏右 part2
            case 0x03:
            case 0x01:
                delayX10ms(100);
                P1_1=1;
                P1_0=0;
                delayX10ms(right+5);
                P1_0=1;
                delayX10ms(100);
                break;
            default:
                ;
        }
        correct=0;
    }

    void beep(){
        P1_7=0;
        time0_temp=19;
        while(time0_temp!=0){}
```

```
        P1_7=1;
        time0_temp=19;
        while(time0_temp!=0){}
    }

    void straight(){
        P1_0=1;
        P1_1=1;
        time1_temp=littleStep;
        countStep=0;
    }

    void turnleft(){
        P1_0=1;
        P1_1=1;
        delayX10ms(100);        //靜止一下在動

        time1_temp=r;
        P1_1=0;                 //右馬達轉
        while(time1_temp!=0){}

        P1_1=1;
        P1_0=1;
        delayX10ms(100);        //靜止一下在動

        mouseState=0;           //動作完畢
    }
    void turnright(){
        P1_0=1;
        P1_1=1;
        delayX10ms(100);        //靜止一下在動

        time1_temp=r;
        P1_0=0;                 //右馬達轉
        while(time1_temp!=0){}

        P1_0=1;
```

```
        P1_1=1;
        delayX10ms(100);        //靜止一下在動

        mouseState=0;           //動作完畢
    }
    void goback(){
        unsigned char temp1=r*3+10;      //270 度
    //  unsigned char temp2=r*2;     //180 度
    //-------------------------------------------
        P1_0=1;
        P1_1=1;
        delayX10ms(100);            //靜止一下在動

        time1_temp=temp1;           //左 270 度
        P1_0=0;
        while(time1_temp!=0){}
    //-------------------------------------------
        P1_0=1;
        P1_1=1;
        delayX10ms(100);            //靜止一下在動

        time1_temp=r;           //左邊 90 度
        P1_1=0;
        while(time1_temp!=0){}
    //-------------------------------------------
        P1_1=1;
        P1_0=1;
        delayX10ms(100);        //動作完畢

        mouseState=0;
    }
    void push(unsigned char temp){
        stack[top]=temp;
        if (top>=maxStack){
            beep();
        }
        else {
```

```
            top++;
        }
    }
    void time_init(void){
        time0_temp=0;
        time1_temp=0;

        PT0=1;          //TIME0 高優先
        PT1=0;          //TIME1 低優先
        EA=1;           //所有中斷自己搞

        TMOD=0x12; //TIME0->MODE2,TIME1->MOD1

        TH0=0x9C;    //0.1MS/1US=100->256-100=156=9C
        TL0=0x9C;
        ET0=1;          //TIME0 中斷啟動
        TR0=1;          //TIME0 啟動

        TH1=0x3C;    //50MS/1US=50000->65536-50000=15536=3CB0H
        TL1=0xB0;
        ET1=1;          //TIME1 中斷啟動
        TR1=1;          //TIME1 啟動
    }

    void delayX5ms(unsigned char temp){
        int i,j;
        for (i=0;i<temp;i++){
            for(j=0;j<600;j++){
                ;
            }
        }
    }
    void delayX1ms(unsigned char temp){
        int i,j;
        for (i=0;i<temp;i++){
            for(j=0;j<120;j++){
                ;
```

```
            }
        }
    }
    void delayX10ms(unsigned char temp){
        int i,j;
        for (i=0;i<temp;i++){
            for(j=0;j<1200;j++){
                ;
            }
        }
    }
    void delay2X10ms(unsigned char temp){
        int i,j;
        for (i=0;i<temp;i++){
            for(j=0;j<1200;j++){
                ;
            }
        }
    }

    void time0_vector(void) interrupt 1{     //TIME0 0.1Ms 中斷一次
        if (time0_temp>0){
            time0_temp--;

        }
    }

    void time1_vector(void) interrupt 3{     //TIME1 50Ms 中斷一次
        if(time1_temp>0){
            time1_temp--;
        }
        else if (time1_temp==0){
            if (mouseState==1&&correct==0){
                if(countStep<step){
                    countStep++;
                    time1_temp=littleStep;
                    P1_0=0;
```

```
                    P1_1=0;
                    delay2X10ms(1);
                    P1_0=1;
                    P1_1=1;
                    delay2X10ms(10);
                }
                else if(countStep>=step){
                    mouseState=0;
                    P1_0=1;
                    P1_1=1;
                }
                else {}
            }
            else {}
        }
        else {}
        TMOD=0x12;        //TIME0->MODE2,TIME1->MOD1
        TH1=0x3C;         //50MS/1US=50000->65536-50000=15536=3CB0H
        TL0=0x9C;
}
```