臺北市第 43 屆中小學科學展覽會

作品說明書

科　　別：生活與應用科學類

組　　別：國小組

作品名稱：節能減碳環保小魚缸泵

關 鍵 詞：節能泵、連通管泵

編　　號：

作品名稱：節能減碳環保小魚缸泵

摘要

我們利用連通管原理將水流入水桶，把水桶內的空氣擠壓到水桶上方的空氣管內，讓空氣注入小魚缸的方法，做成一種節能小魚缸泵，希望可以讓我們取代家裡原有 24 小時一直運轉的泵。如此便能繼續保持大家對養魚的樂趣，又可以節能減碳，對地球環保盡一份心力!

壹、研究動機

一般家裡魚缸所用的空氣泵，需要 24 小時不停的運作，這樣的運轉方式十分耗費電力。如果我們可以找到一個方法來解決這個問題，那該有多好，於是我們想起了四年級自然課本，認識連通管的單元中，曾經學過的連通管原理：在各式形狀、大小不同的管子中注入水，並透過一通道使之連通在一起，一開始每個管子內的水位會不同，但是水會從高處往低處流動，直到各個管內的水面都一般高爲止。造成這樣的結果主要是受到水的壓力和水的深度之影響，當水位愈高，水所承受的壓力就愈大，自然而然就會往水壓低的地方流動，唯有保持各水管水的深度一樣，水壓才能夠達到平衡，水面也才能靜止不動。生活中有許多應用連通管原理的例子，例如：透過連通管原理我們可以測量水平；改變水管兩端的高度，最後兩端的水面仍然一樣高；如果有一端特別低，水則會從低的那端流出來。

如果我們利用連通管原理，讓處於高位水桶內的水自然而然從高處往低處流動至處於低位的水桶，因爲置於低處的水桶是封閉的，唯一的開口就是通往魚缸的管子，是以低位水桶的水量逐漸上升，擠壓到低位水桶中原本的空氣，這些無處可躲的空氣只好順著管子進到了魚缸，成爲幫助魚兒呼吸最好的氣體。

貳、研究目的

透過連通管原理及能量不滅定律，製造一個節能減碳的空氣泵。

參、研究設備及器材

在設計節能減碳的小魚缸泵的過程中，我們遇到了一個問題，當處於高位水桶中的水都流進低位水桶中後，是否又得靠人力，重新轉換高低水位水桶的位置，讓水流動方向相反，形成另一個循環？那麼人不就得二十四小時隨時待命，在那將水桶的位置移上移下，這樣的過程似乎比我們原先的構想麻煩許多，當我們拋出這個困難時，同組的成員立刻提到子軒爸爸剛好是電子這方面的專家，我們可以請他幫我們想想辦法，或許利用電路的設計，可以製作出不用人力操控方式的控制器，在子軒爸爸大力贊助下，「微電腦計時控制器」因此誕生了，這臺微電腦控制器的誕生，大大的協助我們有效且精準的控制水桶交換的時間。

我們將我們的研究設備及器材，分成三大類，第一類是微電腦控制器，第二類是水桶升降木架組，第三類則是配件組，因微電腦控制器是屬於較爲複雜專業的部分，是以在此我們著墨於另兩類設備及器材的介紹，各項器材分述如下：

一、微電腦計時控制器一組：

如圖(一)、圖(二)所示。

圖 (一)

圖 (二)

二、水桶升降木架組

(一)木板數片：

用以架設框架、固定水桶，以利動能傳送。

(二)釘子數根：

用以固定木板。

(三)定滑輪組二組：

主要是帶動兩個 1900c.c 水桶上下移動用。

(四) 1900c.c.水桶二個：

在於締造出兩個不同高低水位的容器，如圖(三)所示。

圖 (三)

三、配件組

(一)70cm 空氣管二支：

輸送空氣至魚缸中的管子，需要二支是因爲當高低水桶的水桶交換位置時，輸送空氣至魚缸中的管子就會不同，皆是由低水位的管子傳送空氣至魚缸中。

(二)40cm 水管一支：

主要在於輸送高水位水桶內的水至低水位水桶內。

(三)20cm＊14cm＊14cm 小魚缸一只

實驗用的魚兒是養在此魚缸中，透過由空氣管中傳送的空氣維持水中的含氧量。

(四)馬達一個

提供外接能源，促使高低水位的水桶轉換上下位置，如(圖四)所示。

(五)電源供應器二個：

一個是 DC12V 電源供應器，該電源供應器供應電源於電動馬達上；另一個是 DC5V 電源供應器，用於供應微電腦計時控制器電源，如(圖五)所示。

圖 (四)

圖 (五)

肆、研究過程或方法

一、 實驗原理

(一)連通管原理：

我們利用連通管原理，讓水從一個高處的水桶流到另一個低處的水桶內，流入的水將低處水桶內的空氣擠壓到空氣管內，讓空氣注入到魚缸，製造出空氣泡供魚兒使用。用此方式就可以達到我們想要的結果 - 「不用電動泵，就能產生空氣泡」

(二)能量不滅定律：

水由高處往低處流，釋放出能量擠出空氣泡。失去的能量，必須由外界給予，才能讓水回到高處令它能夠循環的工作。因此我們需要依賴電動馬達，將低處裝滿水的水桶提昇到高處，DC12V 電源供應器，在此扮演提供電源帶動高低水位水桶位置轉換的重要角色。

二、 裝置操作方式介紹

(一) 微電腦計時控制器

「微電腦計時器」實體按鍵部位，如(圖六)所示。

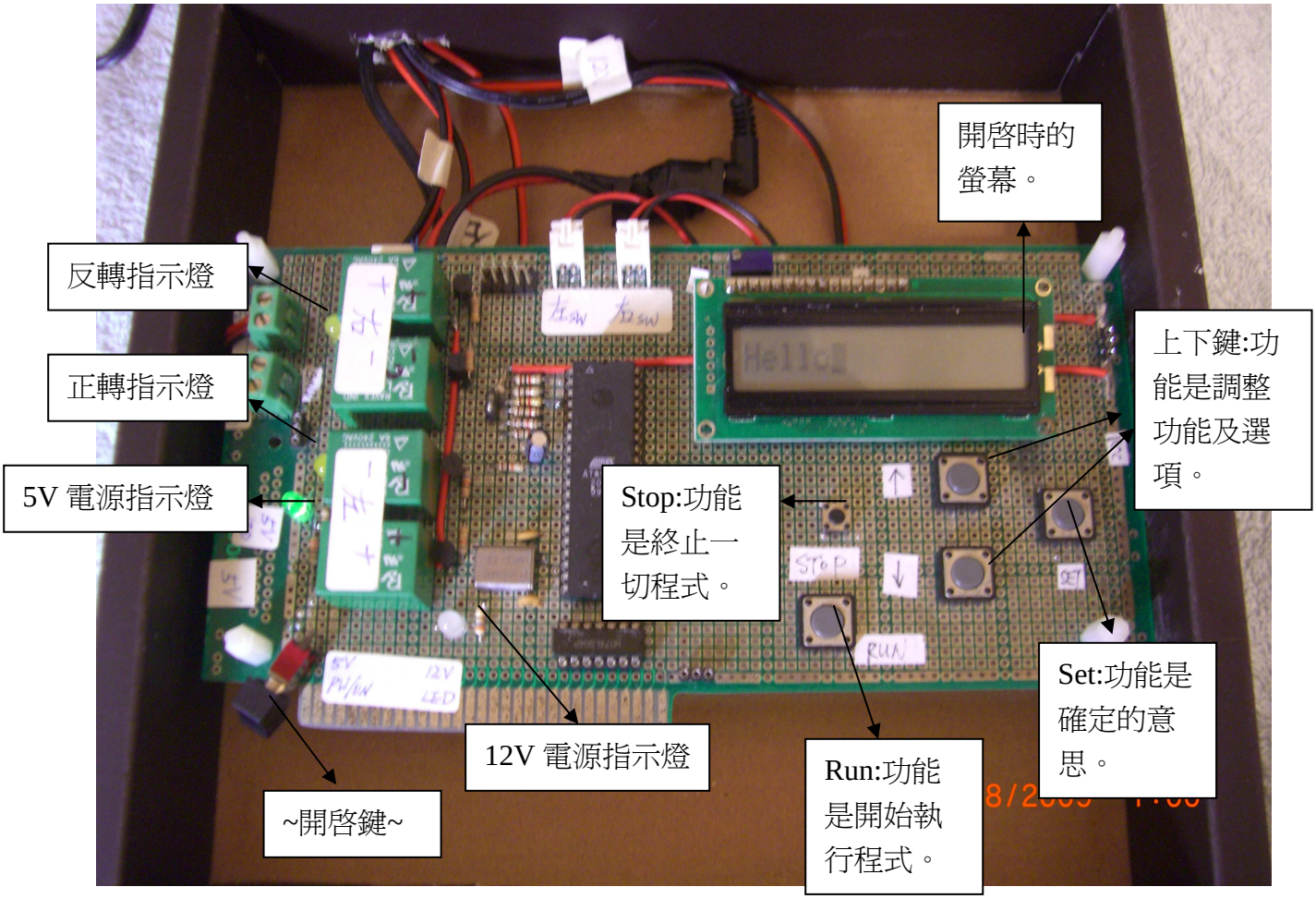


圖 (六)

1.按鍵功能介紹：

(1)Set：功能是確定的意思。

(2)Stop：功能是終止一切程式。

(3)Run：功能是開始執行程式。

(4)↑和↓： 功能是調整時間、使用功能選項。

2.使用功能介紹：

(1)Hello : 歡迎光臨(開機的主畫面)，在此時可執行 Run 的功能。

(2)SetInit : 在停止運轉的狀態，可用此功能利用↑和↓鍵來手動調整水桶的位置。

(3)Timer : 手動設定水桶交換的時間，時間設定範圍從 1 分鐘到 250 分鐘。

(4)Counter: 計時一桶水流完所需要的時間，用此來設定水桶交換的時間，不足一分鐘的秒數以一分鐘計算。

3.按鍵操作使用方式介紹：

(1)利用↑和↓鍵來移動設定的功能選項。

(2)按下 Set 鍵來確認選取進入功能項工作。

(3)設定工作完成後，按下 Set 鍵結束自動回到 Hello 歡迎光臨的主畫面。

4.計時控制器機構方塊圖：

如(圖七)所示，微電腦計時控制器內部程式碼，詳見(附件一)。

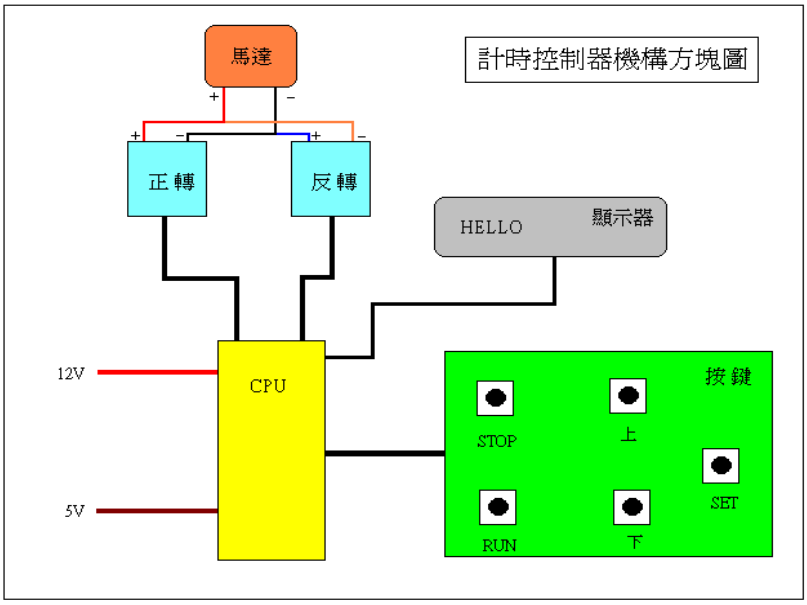
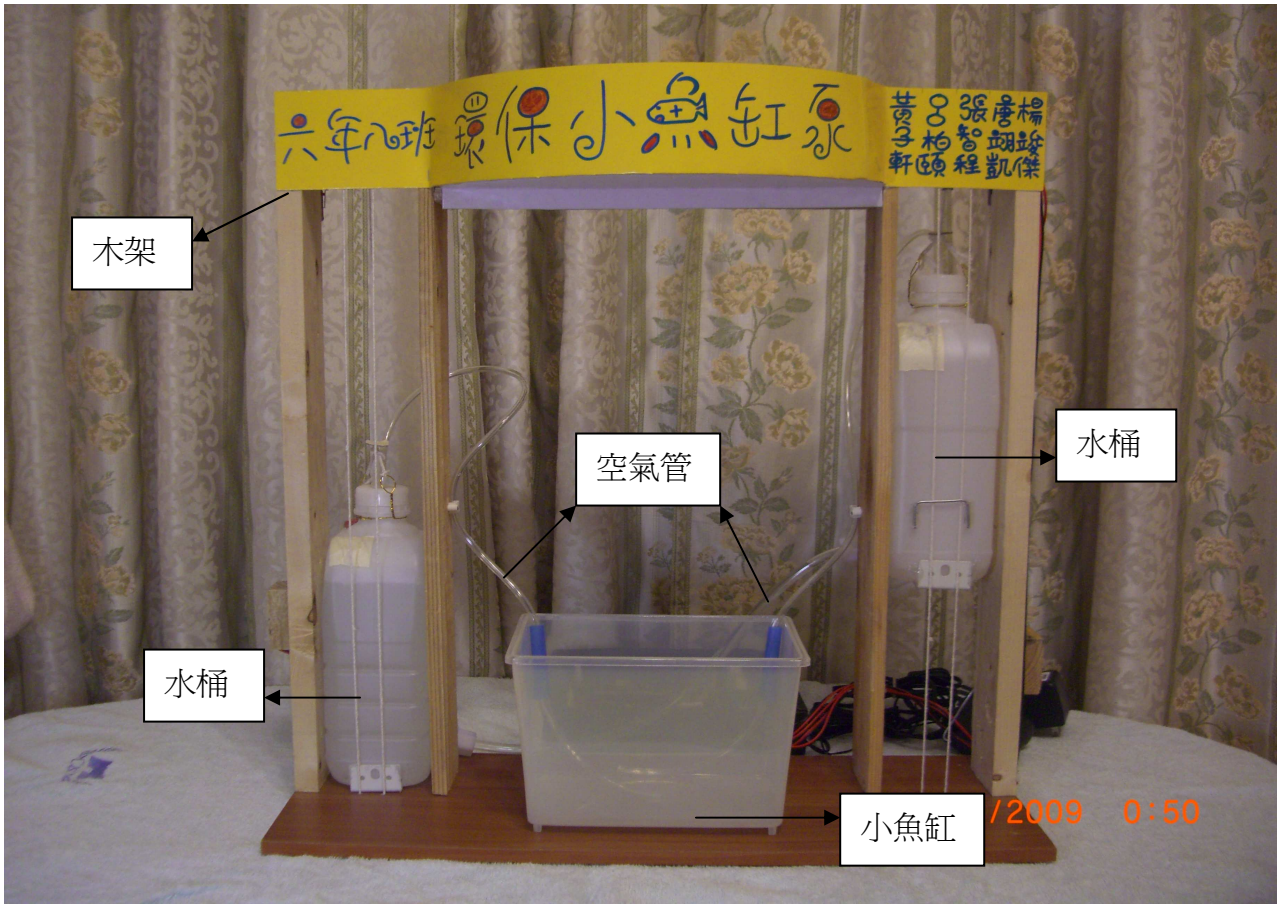


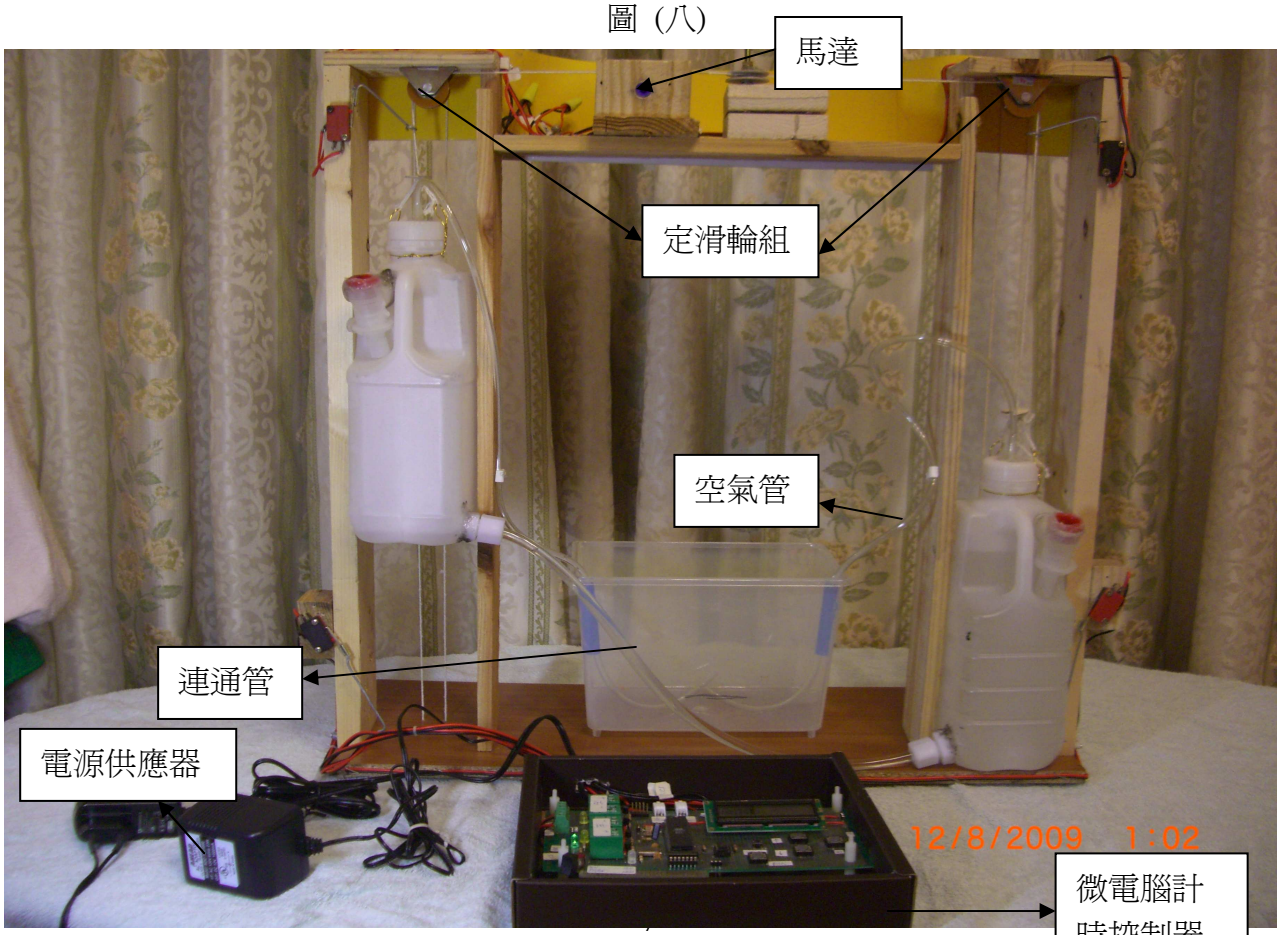
圖 (七)

(二)節能泵

「節能泵」實體正、反面如(圖八)、(圖九)所示。



圖（八）



圖（九）

1.節能泵操作方式：

(1)利用↑和↓鍵來選取需要設定的功能選項，按下 Set 鍵來確認選取進入功能項工作。

(2)進入 SetInit 功能項，將水桶調整到適當的位置。

(3)進入 Timer 或 Counter 功能項，定出水桶交換的時間。時間設定只需要做一次，會永遠記錄在計時控制器內不會消失，亦可重覆相同的動作進行修改內存的時間。

(4)回到 Hello 歡迎光臨的主畫面後，即可按下 Run 鍵，令這節能泵開始運轉。

2. 節能泵的機構圖：

如(圖十)所示。

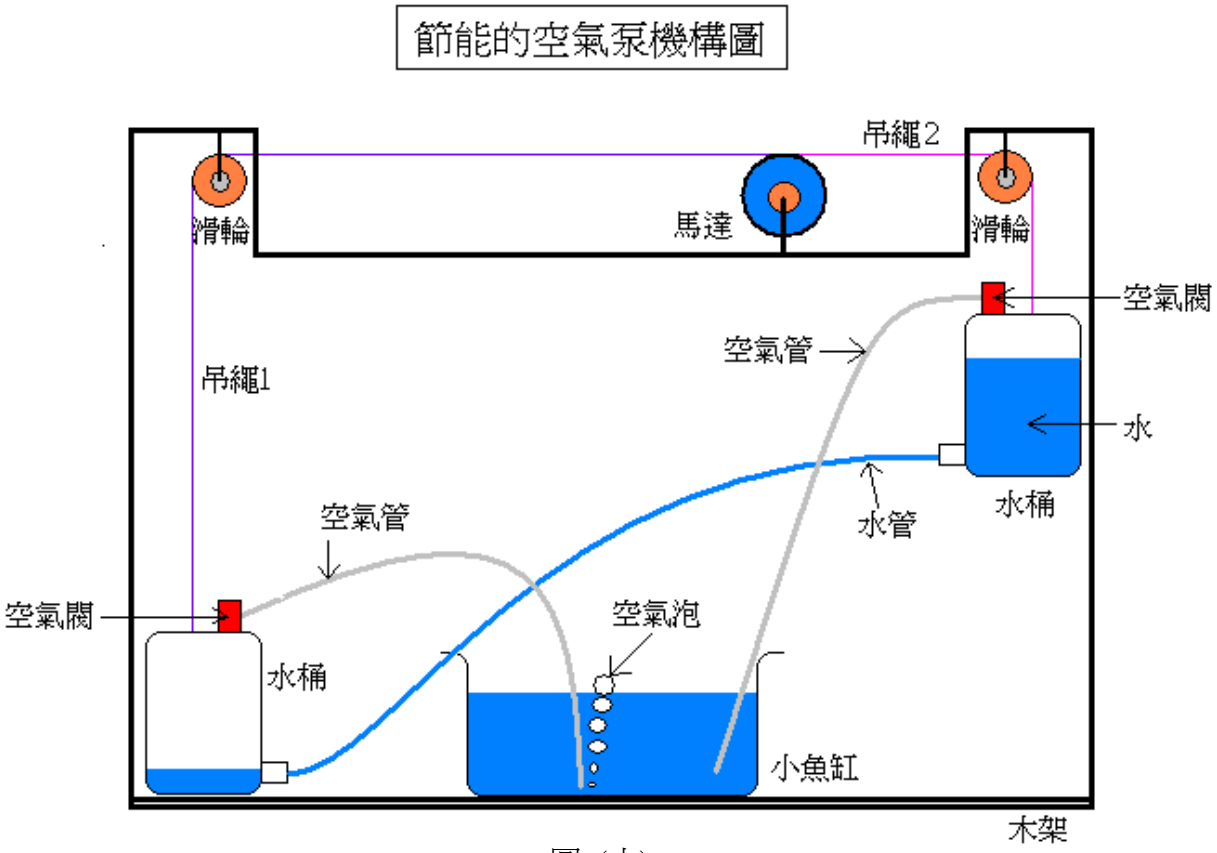


圖 (十)

伍、研究結果

依據我們所設計出的節能減碳小魚泵的操作實驗，我們有以下幾點發現，依序整理如下：

一、水桶提升高度越高，所產生的氣泡越強勁：

本次實驗使用 1900cc 的水桶，內置約 1500cc 的水。使用 DC12V 的馬達將水桶提昇約 25cm 高，就可以成功的產生空氣泡。若將水桶提昇到 50cm 高時，可以看見在小魚缸中產生出較強勁的空氣泡。

二、水桶的蓄水量增加一倍，所產生的氣泡也會加倍。

若是將 1500cc 的水桶換成 3000cc 的水桶，預計可以產生 2 分鐘的空氣泡。就節省能源的目的，功效就可增加一倍。

三、水管管徑，影響空氣管中空氣流出的速度：

原本使用相同內徑 5mm 的水管和空氣管，提昇水桶約 25cm 高，結果空氣泡在水桶內的水約剩下 800cc 時就無法順利產生。因此我們將水管的內徑改爲 6mm 加速水流，而空氣管不變的條件下，就能夠全程流出。

四、水桶由高水位流至低水位的過程中，因爲不需馬達運轉，因此可節省能源：

我們利用功能 Counter 測出一桶 1500cc 的水流完所需的時間，大概是 1 分鐘，結果能夠在小魚缸中產生出適量的空氣泡，和我們預期的結果吻合。同時在水流動的過程中，不需要馬達的運轉，是可以達到節省能源的目的。

五、以公式計算節能減碳環保小魚缸泵與一般家用魚缸泵耗電量之比較，我們設計的小魚缸泵是較爲省電。

(一)DC12V 的小馬達提昇裝滿 3 公升水的水桶，約需要 3 秒鐘。因此每小時的能源消耗計算如下：

60mins / 2mins = 30 times

3secs * 30 times = 90 secs

能源消耗計算公式：

1 焦 = 1 W * 秒

1 度 = 1 千瓦時 = 1 千瓦 * 小時 = 1000 瓦* 3600 秒 = 3600000 焦

1.理論值電力消耗估算：

(1)平常在家裡魚缸用的空氣泵，110V 電力消耗約 - 3W，使用 1 小時，

因此消耗　　3 * 3600 = 10800 焦

(2)環保小魚缸泵， 12V 電力消耗約 － 1.8W，使用 1 小時，

因此消耗 1.8 * 90 = 62 焦

如此可知，每小時只用掉 90 秒的電力，和平常空氣泵的 3600 秒比起來，其他時間的電力就全省下來了。

六、以三用電錶實際測量節能減碳環保小魚缸泵與一般家用魚缸泵耗電量之比較，我們設計的小魚缸泵仍較爲省電，約省下 87.5 %的耗電量。

(一)實際值電力消耗估算：

1.平常在家裡魚缸用的空氣泵，實際電流量測圖如(圖十一)、(圖十二)所示。

110V 電力消耗約 － 32.3mA，

計 110V * 32.3mA / 1000 = 3.553 W

使用 1 小時，因此消耗 3.553 * 3600 = 12790.8 焦

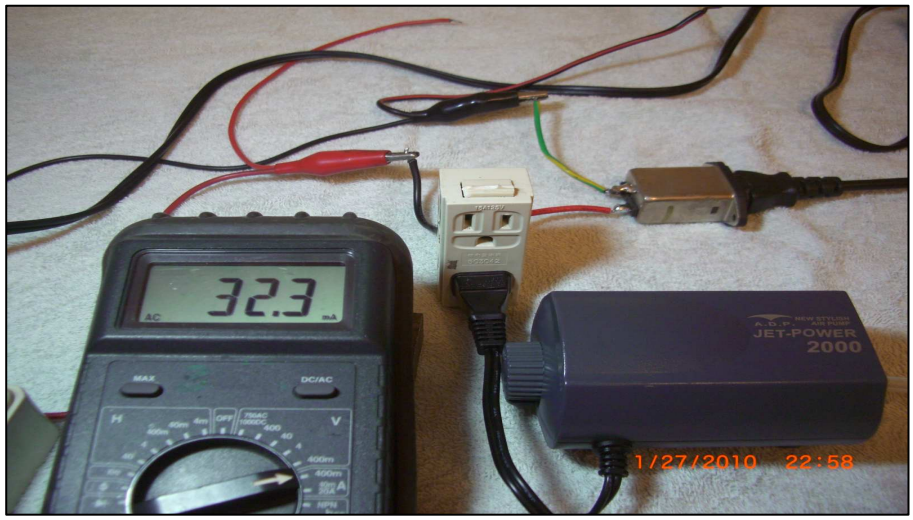


圖 (十一)

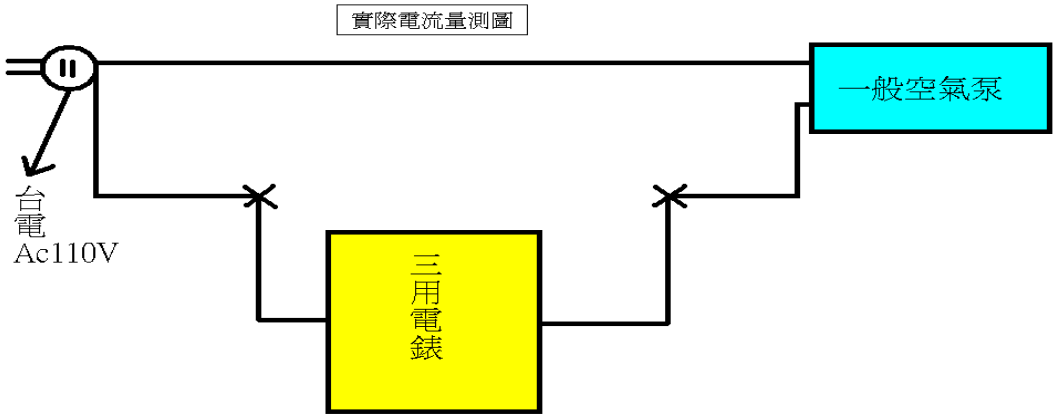


圖 (十二)

2.環保小魚缸泵，實際電流量測圖，如(圖十三)、(圖十四)所示。

(1)12VDC 直流馬達和計時控制器，電力消耗約 － 55.5mA，

計 110V * 55.5mA / 1000 = 6.105W

使用 1 小時，因此消耗 6.105 * 90 = 549.45 焦

(2)計時控制器電力消耗約 － 2.7mA，

計 110V * 2.7mA / 1000 = 0.297W

使用 1 小時，因此消耗 0.297 * (3600 － 90) = 1042.47 焦

合計消耗 549.45 + 1042.47 = 1591.92 焦

圖 (十三)

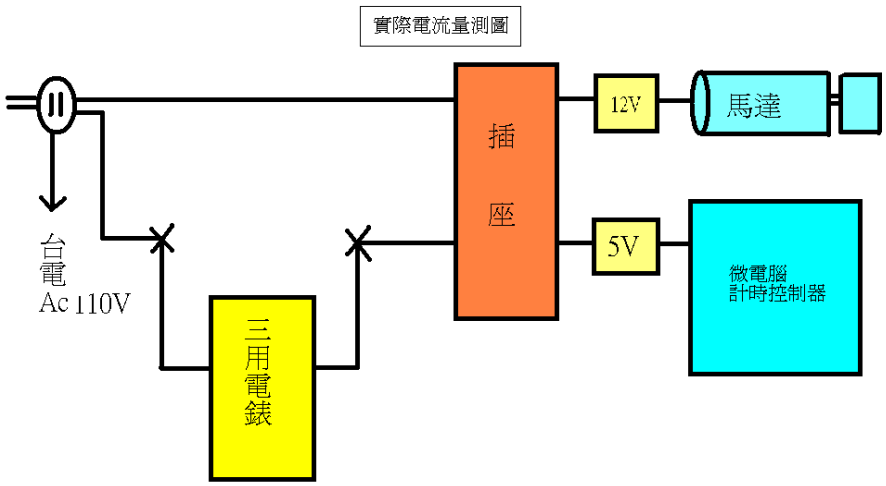


圖 (十四)

目前所使用的 12V 及 5V 大安培數(2.8A)電源供應器，是在家中隨手取得的，電力消耗約 – 39.7mA，並未考量是否合乎環保。而直流馬達及計時控制器電力合計消耗約 55.5 mA，因此我們大約只要 200 mA 的電源供應器即可。所以目前只將直流馬達及計時控制器的電力消耗列入計算。

因此依實際電力消耗值估算比較：

1591.92 / 12790.8 * 100 % = 12.5 %

用此方法，我們約可節省 87.5 % 的電量，如此就能達到節能減碳的效果。

陸、討論

依據實驗結果，我們做了一些討論，茲整理如下：

一、 以目前的設備，1500c.c.可產生 1 分鐘的氣泡。如果我們希望產生 60 分鐘的氣泡，必須要用 90 公升的水才能達成，這需要非常大的水桶，基本上是不可行的，而且那 90 公升的水需要更大的馬達才能將它提到高處，不見得能省到能源。

二、 以目前的設備，用 DC12V 的小馬達，至少可以提升 3 公升的水，因此 3 公升的水約可以提供連續 2 分鐘的氣泡時間。

三、水桶的高度、水桶大小、水管粗細及馬達的位置的影響。

(一)水桶的高度: 水桶提升的位置越高，能產生較大壓力的空氣，產生出的空氣泡效果越良好。

(二)水桶大小: 水桶越大，水量越多，可產生較持久的空氣泡，而水桶的量不能太大，若令小馬達提不起來時，反而會耗電。

(三)水管粗細: 原本使用水管和空氣管口徑是 1:1 的比例，造成水桶裡的水流到一半時就產生水壓不夠的現象，就擠壓不出空氣泡，因此將口徑改爲 2:1 的比例時，產生空氣泡的效果就較良好。

(四)馬達的位置: 馬達位置是以最省力的地方擺置，因此選與水桶最高點平行的位置放置，使用最短的提昇距離，可以消耗最少的電力。

四、一般的小魚缸及泵體積小，不佔太大的空間；反觀現在設計的節能泵，就必須要有一定的擺放位置才行，這是日後有興趣再做此研究的同好，可以再修正的地方。

柒、結論

結果我們成功的做出這個節能減碳環保小魚缸泵，可以不打擾我們的安寧，並且省電，成功的產生出空氣泡，也可以持續我們養魚的樂趣，雖然只能做到兩分鐘的氣泡時間，但足以做到節能減碳的理論。

捌、參考資料及其他

~無~

附件一

計時控制器內部程式碼

```
/* 2009-11-16-Niko, Controller of motor timer system */
#include "io51.h"
#include "string.h"
/* define pinname of 89C51 chip */
#define Key_Up      P1_0            /* Key press Up      status    */
#define Key_Dn      P1_1            /* Key press Down status    */
#define Key_Set     P1_2            /* Key press Set    status    */
#define Left_sw     P1_3            /* Bottom left sw */
#define Right_sw    P1_4            /* Bottom right sw */
#define SW_Stop     P1_5            /* Flag of stop immediately */
#define SDA         P2_0            /* For EEPROM used    */
#define SCL         P2_1            /* For EEPROM used    */
#define TurnMotor   P2_2            /* Motor turning in normal */
#define AntiTurnMotor    P2_3    /* Motor turning in anti way */
#define LCD_EN      P2_4            /* lcd display single - E    */
#define LCD_RS      P2_6            /* lcd display single - RS */
#define SW_RUN      P2_7            /* Flag of start up the program */
/* LCD const define */
#define LCD_Cmd             0                   /* LCD, set RS=0 for Command Input */
#define LCD_Data            1                   /* LCD, set RS=1 for Data Input */
/* declare variable name and value */
static const char LCDDisp[5][11]={
                                    "Hello\0",
                                    "SetInit\0",
                                    "Timer\0",
                                    "Counter\0",
                                    "\0"
                                    };
/* LCD controller command code */
static const char CLR_DISP              = 0x01;     /* 0000-0001, clear display */
static const char ENTRY_INC     = 0x06;     /* 0000-0110, set cursor position counter
increase and display not move */
static const char DISP_ON               = 0x0f;     /* 0000-1111, display on, cursor on,
cursor blink */
static const char DISP_OFF              = 0x08;     /* 0000-1000, display, cursor off */
static const char DD_RAM_ADDR  = 0x80;      /* 1100-0000, set DD RAM    address */
static const char DISP_1Line_8Bit   = 0x30;     /* 0011-0100, set 1 line 8 bits display */

/* LCD variable define */
sfr             LCD_DATA    = 0x80;             /* P0 Data I/O    */
sfr             DATABUS     = 0xb0;             /* P3 Data I/O    */
static const int    COUNT_10ms = 1900;          /* The LCD 20ms delay counter */
static const int    COUNT_100us = 20;                   /* The LCD 100us delay
counter */
/* variable define */
static const char DELAY1        = 100;
static const char ENABLE        = 1;
static const char DISABLE       = 0;
static const char POWER_ON      = 0;
static const char FREE          = 1;
static const char BUSY      = 0;
static const char KEYPRESS      = 0;
/* Public variabel define */
char        Running;
char        beginTimer;
/* System running status */
enum        Running_Status
{
            AllStop,
            SetInit,
            Timer,
            Counter,
            Go
```

```
	};
	/* KeyPress sequence number */
	enum	KeyPress_sequence
	{
		UP_Key,
		Down_Key,
   Set_Key,
		Stop_Key,
		Run_Key,
		None
	};
	/* module funtion */
	/* ----------------------------------	*/
	/* FUNCTION: DELAY			*/
	/*					*/
	/* ----------------------------------	*/
	void DELAY(unsigned int value)
	{
		while(value != 0)
		value--;
		   return ;
	}
	/* ----------------------------------		*/
	/* FUNCTION: LCD_L_Delay			*/
	/*		Delay for atleast 10 ms		*/
	/*		counter : 1900 for 10ms		*/
	/*			   20 for 100us		*/
	/* ----------------------------------		*/
	void LCD_L_Delay(int counter)
	{
		unsigned int i;
		for(i=0; i<counter; i++){}
		return ;
	}
	/* ----------------------------------------		*/
	/* FUNCTION: LCDCOMMAND				*/
	/*		Set LCD driver command			*/
	/*     _  ________________________________	*/
	/* RS  _>--<________________________________	*/
	/*						*/
	/*     ______					*/
	/* RW        \________________________________	*/
	/*						*/
	/* E   _____________/   ___________________\___	*/
	/*     ______________            _______	*/
	/* DB  ______________>-------------<_______	*/
	/*						*/
	/* ----------------------------------------	*/
	void LCDWRITE(char Cmd_or_Data, char datacomm)
	{
		LCD_DATA = datacomm;		/* send to LCD data bus */
		if (Cmd_or_Data == 1)		/* Set for Data input */
			LCD_RS = 1;
		else
			LCD_RS  =  0;				/*  Set  for  command
input */
		DELAY(2);
		LCD_EN = 1;
		DELAY(DELAY1);
		LCD_EN = 0;
		DELAY(DELAY1);
		LCD_L_Delay(COUNT_100us);		/* Delay 100uS for execution */
		LCD_L_Delay(COUNT_10ms);		/* Delay 10mS for execution */
		return ;
	}
	/* ----------------------------------	*/
	/* FUNCTION: LCD_Set_Cursor	*/
	/*	CurX = Position ( 0 to 15) */
	/* Current LCD spec: 1 * 16	*/
	/* ----------------------------------	*/

	void LCD_Set_Cursor(unsigned char CurX)
```

```
{
        LCDWRITE( LCD_Cmd, DD_RAM_ADDR+CurX);
        return ;
}
/* ---------------------------------- */
/* FUNCTION: OpenLCD                  */
/* ---------------------------------- */
void OpenLCD(void)
{
        LCD_EN = 0;                     /* set to low */
        LCD_RS = 0;
        LCDWRITE(  LCD_Cmd,  DISP_1Line_8Bit);          /* make sure is been setting */
        LCDWRITE( LCD_Cmd, DISP_1Line_8Bit);
        LCDWRITE( LCD_Cmd, DISP_1Line_8Bit);
        LCDWRITE( LCD_Cmd, DISP_ON) ;
        LCDWRITE( LCD_Cmd, ENTRY_INC) ;
        LCDWRITE( LCD_Cmd, CLR_DISP) ;
        LCD_Set_Cursor(0);                              /* move cursor to left begining position */
        return ;
}
/* ----------------------------------  */
/* FUNCTION: PutStringLCD              */
/*         Display the string from left */
/*         to right, position auto add 1 */
/* ----------------------------------  */
void PutStringLCD( unsigned char *String, char String_len)
{
        char            count_i;
        unsigned char   Str_Temp ;
      LCDWRITE( LCD_Cmd, CLR_DISP);     /* clear LED */
        if (String_len > 16)            /* Max length 16 char */
                String_len = 16;
        for (count_i = 0; count_i < String_len; count_i++)
        {
                Str_Temp = *String;
                if (Str_Temp == 0 || Str_Temp == '\0')
                      break;
                LCDWRITE( LCD_Data, Str_Temp) ;
                String++;
        }
        return ;
}
/* ---------------------------------- */
/* FUNCTION: LCD_display   */
/* ---------------------------------- */
void LCD_display(char sel_data)
{
      PutStringLCD(LCDDisp[sel_data], 11);
      return ;
}
/* ----------------------------------  */
/* FUNCTION: DELAY0                    */
/*                                     */
/*          50ms each time,      */
/* value=20 then have 1 second         */
/* ----------------------------------  */
void DELAY0(unsigned int value)
{
      TR0=1;
      while(value != 0)
      {     TH0=76;
            TL0=1;
            while(TF0 != 1);
            TF0=0;
            value--;
            if (SW_Stop==KEYPRESS)
            {
```

```
        Running   = AllStop;
                  break;
              }
        }
        TR0=0;
}
/* ---------------------------------- */
/* FUNCTION: EEPCLK          */
/* ---------------------------------- */
void EEPCLK(void)
{
        char i;
        SCL=1;
        for(i=0;i<3;i++);
        SCL=0;
        for(i=0;i<3;i++);
        return ;
}
/* ---------------------------------- */
/* FUNCTION: Send_data       */
/* ---------------------------------- */
void Send_data(char mdata)
{
   char         i,j;
   DATAIN = mdata;
   SDA=DATAIN_7;
   EEPCLK();
   SDA=DATAIN_6;
   EEPCLK();
   SDA=DATAIN_5;
   EEPCLK();
   SDA=DATAIN_4;
   EEPCLK();
   SDA=DATAIN_3;
   EEPCLK();
   SDA=DATAIN_2;
   EEPCLK();
   SDA=DATAIN_1;
   EEPCLK();
   SDA=DATAIN_0;
   EEPCLK();
   SCL = 1;                                  /* Ack      */
   DELAY(DELAY1);
   while(SDA)
   {     SCL = 0;
         DELAY(DELAY1);
         SCL = 1;
         DELAY(DELAY1);
   }
   SCL = 0;
   return ;
}
/* ---------------------------------- */
/* FUNCTION: EEP_read         */
/* ---------------------------------- */
char EEP_read(char mdata)
{
        char      ret_data,i,j,k;
        SCL = 1;                                  /* start condition              */
        SDA = 1;
        SDA = 0;
        SCL = 0;
        Send_data(0xa0);                          /* send device address       */
        Send_data(mdata);                         /* send word address         */
        SCL = 1;                                  /* start condition              */
        SDA = 1;
        SDA = 0;
        SCL = 0;
        Send_data(0xa1);                          /* send device address       */
        DATAIN = 0;
```

```
    SCL = 1;                                    /* read data7                  */
        DATAIN_7=SDA;
        SCL = 0;
        DELAY(DELAY1);
        SCL = 1;                                /* read data6                  */
        DATAIN_6=SDA;
        SCL = 0;
        DELAY(DELAY1);
        SCL = 1;                                /* read data5                  */
        DATAIN_5=SDA;
        SCL = 0;
        DELAY(DELAY1);
        SCL = 1;                                /* read data4                  */
        DATAIN_4=SDA;
        SCL = 0;
        DELAY(DELAY1);
        SCL = 1;                                /* read data3                  */
        DATAIN_3=SDA;
        SCL = 0;
        DELAY(DELAY1);
        SCL = 1;                                /* read data2                  */
        DATAIN_2=SDA;
        SCL = 0;
        DELAY(DELAY1);
        SCL = 1;                                /* read data1                  */
        DATAIN_1=SDA;
        SCL = 0;
        DELAY(DELAY1);
        SCL = 1;                                /* read data0                  */
        DATAIN_0=SDA;
        SCL = 0;
        DELAY(DELAY1);
        SCL = 1;
        DELAY(DELAY1);
        SCL = 0;                                /* stop    condition           */
        SDA = 0;
        SCL = 1;
        SDA = 1;
        ret_data = DATAIN;
        return (ret_data);
}
/* ---------------------------------- */
/* FUNCTION: EEP_write                */
/* ---------------------------------- */
void EEP_write(char addin, char mdata)
{
        char    i,j;
        SCL = 1;                                /* start condition             */
        SDA = 1;
        SDA = 0;
        SCL = 0;
        Send_data(0xa0);                        /* send device address         */
        Send_data(addin);                       /* send word address           */
        Send_data(mdata);                       /* send data                   */
        SCL = 0;                                /* stop    condition           */
        SDA = 0;
        SCL = 1;
        SDA = 1;
        SCL = 1;                                /* start condition             */
        SDA = 1;
        SDA = 0;
        SCL = 0;
        Send_data(0xa0);                        /* send device address         */
        SCL = 1;                                /* Ack                         */
        DELAY(DELAY1);
        while(SDA)
        {       SCL = 0;
                DELAY(DELAY1);
                SCL = 1;
```

```
while (Key_Dn==DISABLE) {}
                        break;
                }
                if (Key_Set==DISABLE)
                {       key_press=Set_Key;
                        while (Key_Set==DISABLE) {}
                                break;
                }
                if (SW_Stop==DISABLE)
                {       key_press=Stop_Key;
                                break;
                }
                if (SW_RUN==DISABLE)
                {       key_press=Run_Key;
                        while (SW_RUN==DISABLE) {}
                                break;
                }
          }
        return (key_press);
}
/* ---------------------------------- */
/* FUNCTION: KeyCheck         */
/* ---------------------------------- */
char KeyCheck(void)
{
        char key_press;
        while(1)
        {
                if (Key_Up==DISABLE)
                {       key_press=UP_Key;
                        while (Key_Up==DISABLE) {}
                                break;
                }
                if (Key_Dn==DISABLE)
                {       key_press=Down_Key;
                        while (Key_Dn==DISABLE) {}
                                break;
                }
                if (Key_Set==DISABLE)
                {       key_press=Set_Key;
                        while (Key_Set==DISABLE) {}
                                break;
                }
                if (SW_Stop==DISABLE)
                {       key_press=Stop_Key;
                                break;
                }
                if (SW_RUN==DISABLE)
                {       key_press=Run_Key;
                        while (SW_RUN==DISABLE) {}
                                break;
                }
                key_press=None;
                break;
        }
        return (key_press);
}
/* ---------------------------------- */
/* FUNCTION: Time_display    */
/* ---------------------------------- */
void Time_display(char mins, char secs)
{
         char NumData[9];
         memset(NumData, 0, sizeof(NumData));
sprintf(NumData, "%d:%d", mins, secs);
         PutStringLCD(NumData, 9);
         return ;
}
/* ---------------------------------              */
/* FUNCTION: Do_running                           */
```

```
/*    if doing = Timer,                                    */
/*                    then count down until 0,             */
/*                and set timer again and run              */
/*    then if doing = Counter,                             */
/*                    then count + 1,                      */
/*                until STOP or SET key press              */
/* ----------------------------------          */
char Do_running( char nowtime, char doing )
{
      char keypress, currtime, failcounter, normal, currsec, delayCounter, MotorCounter;
      char direction;
      TurnMotor = FREE;
      AntiTurnMotor = FREE;
      normal=1;               /* normal */
      currtime = nowtime;
      MotorCounter=5;
      if ((Right_sw == KEYPRESS) && (Left_sw == KEYPRESS))
      {
            PutStringLCD("SW Error\0", 9);
            normal=0;
      }
      else
      {
            if (Left_sw == KEYPRESS)
            {
                  PutStringLCD("Turning\0", 9);
                  Running   = Go;
                  TurnMotor = BUSY;
                  direction = 0;
            }
            else
            {           /* include Right_sw == KEYPRESS && none sw been KEYPRESS */
                  PutStringLCD("AntiTurn \0", 8);
                  Running   = Go;
                  AntiTurnMotor = BUSY;
                  direction = 1;
            }
      }
      currsec=0;
      while (normal)
      {
            Time_display(currtime, currsec);
            if (doing == Timer)
            {
                  if (currsec == 0)
                  {
                        currsec = 59;
                        if (currtime == 0)
                        {
                             currtime = nowtime-1;
                             MotorCounter=5;
                             Running     = Go;
                             if (Left_sw == KEYPRESS)
                             {
                               direction = 0;
                                    TurnMotor = BUSY;
                               }
                               else
                               {
                                    direction = 1;
                                    AntiTurnMotor = BUSY;
                               }
                        }
                        else
                              currtime--;
                  }
                  else
                        currsec--;
            }
            else
```

```
{           /* counter */
                currsec++;
                if (currsec == 60)
                {
                        currtime++;
                        currsec=0;
                }

            }
            delayCounter=15;
            keypress=None;
            while ( delayCounter )
           {
                if ((Key_Set==DISABLE) || (SW_Stop==DISABLE))
                {
                        normal=0;
                        keypress=Stop_Key;
                        TurnMotor = FREE;
                        AntiTurnMotor = FREE;
                        if (doing == Counter)
                        {
                                if (currsec > 0)
                                        currtime++;
                                EEP_write(1, currtime);
                                beginTimer = currtime;
                        }
                       break;
                }
                if (Running == Go)
                {
                        if (((Right_sw == KEYPRESS) && (direction==0)) ||
                                ((Left_sw == KEYPRESS) && (direction==1))    ||
                                (MotorCounter <= 0))
                        {
                                TurnMotor = FREE;
                                AntiTurnMotor = FREE;
                                Running   = AllStop;
                        }
                }
                delayCounter--;
                DELAY(135);
                DELAY0(1); /* 20 /second */
            }     /* end of while ( 1 ) */
            if (MotorCounter > 0)
                MotorCounter--;
       }
       Running   = AllStop;
       keypress = Waitkey1();
       return (Stop_Key);
}
/* ---------------------------------- */
/* FUNCTION: MotorInit                */
/* ---------------------------------- */
void MotorInit( void )
{
        PutStringLCD("Initial\0", 8);
     TurnMotor = FREE;
     AntiTurnMotor = FREE;
        while ( 1 )
     {
            if (Key_Up==KEYPRESS)
            {
                TurnMotor = BUSY;
                AntiTurnMotor = FREE;
                while ((Key_Up==KEYPRESS) && (Right_sw != KEYPRESS)) {}
                TurnMotor = FREE;
                AntiTurnMotor = FREE;
            }
            if (Key_Dn==KEYPRESS)
            {
                TurnMotor = FREE;
```

```
    AntiTurnMotor = BUSY;
                while ((Key_Dn==KEYPRESS) && (Left_sw != KEYPRESS)) {
    TurnMotor = FREE;
                AntiTurnMotor = FREE;
            }
            if ((Key_Set==KEYPRESS) || (SW_Stop==KEYPRESS))
                    break;
}     /* end of while ( 1 ) */
      return ;
}
/* ---------------------------------- */
/* FUNCTION: NUM_display    */
/* ---------------------------------- */
void NUM_display(char currtime)
{
       char NumData[9];
       memset(NumData, 0, sizeof(NumData));
       sprintf(NumData, "%d mins", currtime);
       PutStringLCD(NumData, 11);
       return ;
}
/* ---------------------------------- */
/* FUNCTION: SetTime            */
/* ---------------------------------- */
void SetTime( void )
{
      char currtime, keypress;
      currtime = beginTimer;
      while ( 1 )
     {
         NUM_display(currtime);
           keypress = Waitkey1();
           switch(keypress)
           {
                case    Stop_Key:
                        break;
                case    Set_Key:
                  EEP_write(1, currtime);
                  beginTimer = currtime;
                        break;
                case    Down_Key:
                        if (currtime > 1)
                            currtime--;
                        break;
                case    UP_Key:
                  if (currtime < 250)
                            currtime++;
                        break;
           }            /* end of switch */
           if ((keypress == Stop_Key) || (keypress == Set_Key))
                        break;
      }     /* end of while ( 1 ) */
}
/* ---------------------------------- */
/* FUNCTION: SetCounter        */
/* ---------------------------------- */
void SetCounter( void )
{
      char currtime, keypress;
      currtime = 0;
      PutStringLCD("Sure ?\0", 6);
      keypress = Waitkey1();
     if (keypress == Set_Key)
     {
            Do_running(currtime, Counter);
      }
      return ;
}
/* ---------------------------------- */
/* FUNCTION: SETSYSTEM    */
/* ---------------------------------- */
void SETSYSTEM( void )
```

```
{
        P0 = 0xff;
        P1 = 0xff;
        P2 = 0xff;
        P3 = 0xff;
           Running     = AllStop;

        TMOD = 0x01;                                    /* timer 0 - timer */
        OpenLCD();
        beginTimer = EEP_read(1); /* the timer in minutes, store in this address */
        LCD_display(POWER_ON);        /* LCD display hello */
        return ;
}
main()                                                  /* main loop */
{
    char                keypress;
    char                data8, count_i;

    SETSYSTEM();
       TurnMotor = FREE;
    AntiTurnMotor = FREE;
    while ( 1 )
    {
            keypress = Waitkey();
            switch(keypress)
            {
                case    Stop_Key:
                        SETSYSTEM();
                           break;
                case    UP_Key:
                        if (Running > 0)
                        {
                                Running--;
                                  LCD_display(Running);
                        }
                        break;
                case    Down_Key:
                        if (Running < 3)
                        {
                                Running++;
                                LCD_display(Running);
                        }
                                break;
                case    Set_Key:
                        switch(Running)
                        {

                                case SetInit:
                                MotorInit();
                                        break;
                                case Timer:
                                            SetTime();
                                        break;
                                case Counter:
                                        SetCounter();
                                        break;
                        }
                        Running = AllStop;
                        LCD_display(Running);
                                break;
                case    Run_Key:
                  Do_running(beginTimer, Timer);
                                break;
            }           /* end of switch */
        }  /* end of while ( 1 ) */
}
/* end of main */
```