遙控三輪車

張恩、蔣幼齡
中國文化大學光電物理系

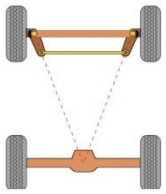
摘要

很多機車廠陸續推出了類似二輪機車單三角台設計的三輪機車。我想要試著將汽車的阿克曼轉向機構加入三輪機車內並比較兩者的優劣。此外，加入遙控系統，使我們可以觀察三輪車的運動狀態。

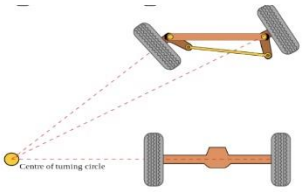
阿克曼轉向幾何

圖一、圖二為阿克曼轉向幾何(Ackermann steering geometry)是一種為了解決交通工具轉彎時，內外轉向輪路徑指向的圓心不同的幾何學，這個想法是由德國車輛工程師蘭肯斯伯格(Lankensperger)於1817年提出。

依據阿克曼轉向幾何設計的車輛，沿著彎道轉彎時，利用四連桿的相等曲柄使內側輪的轉向角比外側輪大大約2~4度，使四個輪子路徑的圓心大致上交會於後軸的延長線上瞬時轉向中心，讓車輛可以順暢的轉彎。



圖一、阿克曼轉向幾何-直行

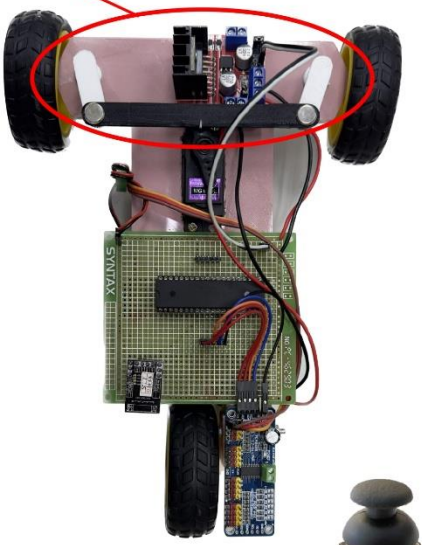


圖二、阿克曼轉向幾何-轉向

伺服馬達

伺服馬達(圖五)可透過PWM訊號精準控制旋轉角度，角度是0～180度。使用此特性控制轉向機構，就可以模擬出與汽車相似的轉向運動。

圖五、伺服馬達



圖三、成品

車體機構

本作品使用3D列印製作三輪車的車體與遙控器的外殼，在轉向部分使用了阿克曼轉向機構。

遙控的操作選擇使用搖桿，可以很直覺的控制車子的方向。控制部分則是使用RF無線電波傳輸搖桿訊號，傳輸距離最遠可達100公尺。

圖四、遙控器

HW-504搖桿模組

十字搖桿(圖六)為一個雙向的10kΩ電阻器，隨著搖桿方向不同，抽頭的阻值隨著變化。本模塊使用5V供電，在搖桿置中的情況下X、Y讀出的電壓皆為2.5V，當搖桿朝正X(Y)方向按下，讀出電壓會隨著增加，最大到5V；朝負方向按下，讀出電壓值會隨著減少，最小為0V。

圖六、HW504搖桿

PCA9685伺服馬達驅動板

PCA9685伺服馬達驅動板(圖七)是一款16路伺服馬達控制擴展板，最多可串聯62個驅動板，驅動992個伺服馬達。以I2C匯流排協定的方式做通信，就能夠讓主控晶片和PCA9685通信，實現同時控制多個伺服馬達。

圖七、 PCA9685伺服馬達驅動板

MUART0-B無線序列埠傳輸模組

MUART0-B為凌陽創新推出的無線傳輸模組，可以進行一對多傳輸。使用UART與微處理機做資料傳輸。傳輸距離最遠可達 80 ~ 100m。RF頻率2.4 ~ 2.48GHz。

圖八、MUART-B無限序列埠傳輸模組

參考資料

1. https://zh.wikipedia.org/wiki/阿克曼轉向幾何-阿克曼轉向幾何
2. https://components101.com/modules/joystick-module -Joystick Module
3. https://www.sunplusit.com/TW/Shop/Iot/MUART0B -MUART0-B 無線序列埠傳輸模組

110 學年度第 2 學期

專題討論(三)

遙控三輪車

系級：光電物理二

學號：A9214391

姓名：張恩

指導老師：蔣幼齡 教授

中華民國 111 年六月

光電物理學系 二 年級學生 張 恩 學號 A9214391

修習 專題討論(三) 所完成之論文

遙控三輪車

內容與格式經論文指導老師審查，認為符合繳交標準。

指導老師：

蔣 幼 齡

中華民國 111 年六月

目錄

一、 中文摘要

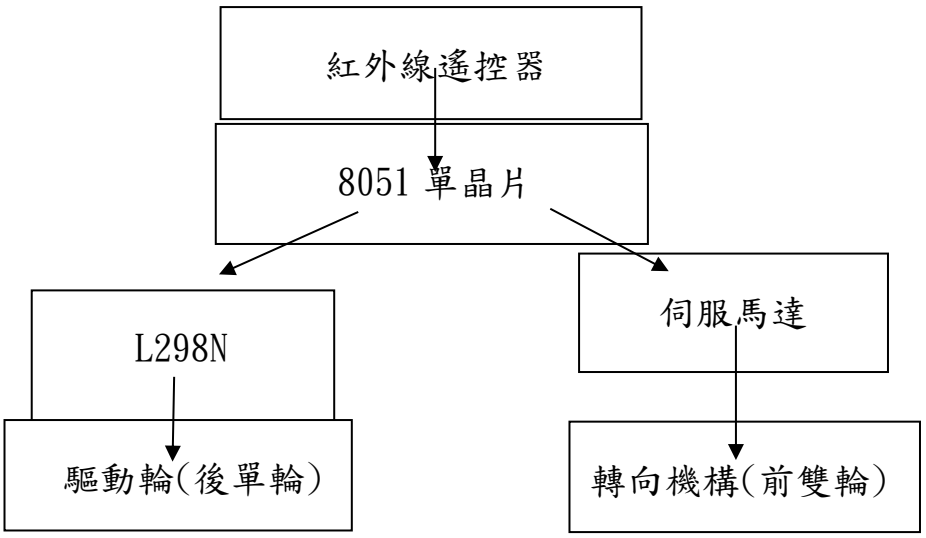
現在很多機車車廠都陸續推出三輪機車，但大多是類似二輪機車單三角台設計。所以我想要試著將汽車的阿克曼轉向機構加入三輪機車內，並比較兩者的優劣。

二、 英文摘要

Now many motorcycle manufacturers have successively launched three-wheeled motorcycles, but most of them are similar to the single-triangle design of two-wheeled motorcycles. So I wanted to try adding the car's Ackerman steering mechanism to the trike and compare the pros and cons of the two.

三、元件架構

遙控三輪車元件架構如圖一，以紅外線遙控模組負責接收訊號，傳送至 8051 單晶片後，在遊 8051 單晶片控制直流馬達、伺服馬達，分別完成直行、後退及轉向的動作。



圖一、架構圖

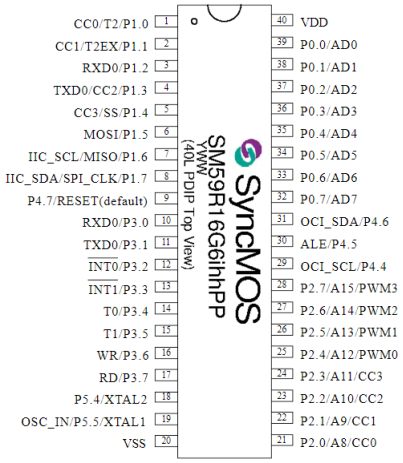
四、 8051 單晶片介紹

我使用的 8051(型號：新茂 SM59R16G6)如圖二、三，為 8 位元的單晶片微控制器，有 64KB 內部城市記憶體(ROM)，具有邏輯代數

運算功能，(位元邏輯)，工作電壓 5V、輸出/入電流 20mA 以下，常使用組合語言及 C 語言編寫程式。

- 40 隻腳位(圖三)
- 類比數位轉換(ADC)，P1
- 4 組 8 位元的 I/O 埠，P0~P3
- 2 組 16 位元計時/計數器
- 4 個 PWM 輸出腳位，P2_4~P2_7
- 10 個中斷源，1 組雙全工串列埠(UART)，1 組積體匯流排電路(I2C)，P1_6-P1_7

圖二、8051 單晶片



圖三、8051 腳位圖

五、元件介紹

(1)紅外線遙控模組

紅外線模組(如圖四)是由紅外線發射器及接收器組成，以紅外線發光 LED 發射波長 940nm 的紅外線、38KHz 的載波平率來傳送信號，在每個按鍵備案下發送編碼十，會送出一組 8 位元的代碼及一組 8 位元的反代碼，當兩組代碼相互以 16 進位互補時，才為有效代碼。

圖四、紅外線遙控模組

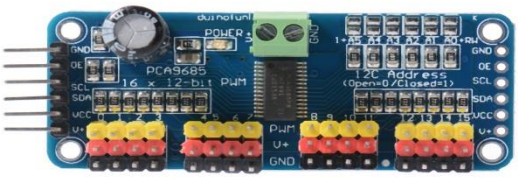
1	0XBA45	0	0XE61A
2	0XB946	*	0XE916
3	0XB847	#	0XF20D
4	0XBB44	↑	0XE718
5	0XBF40	↓	0X9D62
6	0XBC43	→	0X946B
7	0XF807	←	0XF708
8	OXEA15	OK	0XE31C
9	0XF60A		

圖五、各按鈕編碼對照表

(2)PCA9685 伺服馬達驅動板

在本專題使用的伺服馬達需用 PWM 訊號進行控制，但 8051 單晶片內建的 PWM 輸出訊號腳的頻率太高，無法與伺服馬達所需的頻率匹配。使用中斷副程式模擬的 PWM 訊號也因解析度太差，導致無法精準地控制轉向的角度。所以在本專題中我使用 PCA9685 驅動板控制。

PCA9685 伺服馬達驅動板是一款 16 路伺服馬達控制擴展板，此驅動板最多可串聯 62 個驅動板，總共可驅動 992 個伺服馬達。以 I2C 匯流排協定的方式做通信，就能夠讓主控晶片和 PCA9685 通信，實現同時控制多個伺服馬達。



圖六、PCA9685 伺服馬達驅動板

(3)伺服馬達

伺服馬達(servo motor)(圖七)，因常用於搖控飛機模型，所以又常稱為 RC 伺服機(RC Servo，Radiio Control Servo，Remote Control Servo)、舵機。

伺服馬達裡含有直流馬達、齒輪箱、軸柄、以及控制電路，我們可透過 PWM 訊號精準控制伺服馬達的旋轉角度，角度是 0 到 180 度。

圖七、伺服馬達

伺服馬達控制馬達角度的脈衝寬度是 0.5ms~2.5ms，本專題使

用頻率 50Hz 的 PWM 訊號，脈衝持續時間與轉動角度關係如圖八。不同顆馬達可能會因內部電位計有些微差異而產生誤差，引此使用前須做檢測並校正。

脈衝持續時間(ms)	角度(˚)
0.5	0
1.0	45
1.5	90
2.0	135
2.5	180

圖八、脈衝時間與轉動角度關係

(4)直流馬達

直流馬達(圖九)的基本構造包括「電樞」、「場磁鐵」、「集電環」、「電刷」。

- 電樞：可以繞軸心轉動的軟鐵芯纏繞多圈線圈。
- 場磁鐵：產生磁場的強力永久磁鐵或電磁鐵。
- 集電環：線圈約兩端皆至兩片半圓形的集電環，隨線圈轉動，可供改變方向的變相器。每轉動半圈(180 度)，線圈上的電流方向就改變一次。
- 電刷：通常使用炭製成，集電環接觸固定位置的電刷，用以接至電源。

圖九、直流馬達

六、成品

目前車輛已可以在接收紅外線訊號後，完成前進、後退及轉向。

將來會改進傳輸方式，改用 RF 傳輸，以改善傳輸的準確率及傳輸距離。

六、參考資料

1. https://autos.yahoo.com.tw/main/index.php/news/ 【二輪解密】避震之前叉看起來不太一樣！非典型設計四重奏！

2.https://zh.wikipedia.org/wiki/阿克曼轉向幾何-阿克曼轉向幾何

3.https://kknews.cc/zh-tw/car/28oxqjy.html-汽車阿克曼轉向機構原理

4. http://www.syncmos.com.tw/products_file/ISSFD-M059_I_SM59R16G6_20150427(EN).pdf. -SM59R16G6W 產品資訊

七、工作日誌

時 間	地 點	內　　容
111/01/09	義 221	3D 圖製作
111/01/15	義 221	PWM 程式撰寫
111/01/23	義 221	3D 列印部件
111/02/20	義 221	伺服馬達測試
111/03/05	義 221	直流馬達程式測試
111/03/12	義 221	IR 程式測試
111/03/26	義 221	PCA9685 控制板測試
111/04/09	義 221	程式完成
111/04/23	義 221	電路測試完成
111/06/25	義 221	機構重新調試

附錄 ： 程式碼

主程式

```
#include "SM59R16G6.h"
#include "motor.h"
#include "IR_remote.h"
#include "control_middle.h"
#include "pca9685_for_i2c.h"
main()
{
	unsigned char angle = 90;
	reset();
	ini_remote();
	begin();
	PWMfreaq_set();

	while(1)
	{
		switch( key_edge)
		{
			case 0XB9 ://2

				forward();
				if(angle > 90)
				{
					angle = middle_L(angle);
				}
				else
				{
					angle = middle_R(angle);
				}
				key_edge=0;
				while(key_level!=0);
				break;

			case 0XEA ://8
```

```
                backward();
                if(angle > 90)
                {
                        angle = middle_L(angle);
                }
                else
                {
                        angle = middle_R(angle);
                }

key_edge=0;
while(key_level!=0);
                break;

  case 0XBF ://5

                astop();
                if(angle > 90)
                {
                        angle = middle_L(angle);
                }
                else
                {
                        angle = middle_R(angle);
                }

//reset();
                key_edge=0;
while(key_level!=0);
                break;

  case 0xBB ://4

                angle = right(angle);
                key_edge=0;
                while(key_level!=0);
                break;
```

```
            case 0xBC ://6

                        angle = left(angle);
                        key_edge=0;
                        while(key_level!=0);
                        break;

            }
    }
}
```

副程式 control_middle.C

```
#include "SM59R16G6.h"
#include "pca9685_for_i2c.h"

#define board1    0x40
#define pin0              0x06

void delay (int count)
{
                  int i,j;
                  for(i = 0;i<count;i++)
                  {
                  for(j = 0;j<count;j++);
                  }
}



unsigned int adj_angle_L (unsigned int i)
{
      unsigned int angle_L;
      angle_L = (102 + i*410/180) % 256;
      return angle_L;
}
unsigned int adj_angle_H (unsigned int i)
{
      unsigned int angle_H;
      angle_H = (102 + i*410/180) / 256;
      return angle_H;
}

void reset(void)
{
      int i = 90;
      setPWM(board1,pin0,0x00,0x00,adj_angle_L(i),adj_angle_H(i));//90
      delay(100);
}
```

```
unsigned char middle_L(int xx)//over 90
{
        int i;
        for (i = xx;i > 89;i--)
        {
                setPWM(board1,pin0,0x00,0x00,adj_angle_L(i),adj_angle_H(i));//90
           delay(100);
        }
        return i+1;
}

unsigned char middle_R(int xx)//below 90
{
        int i;
        for (i = xx;i < 91;i++)
        {
                setPWM(board1,pin0,0x00,0x00,adj_angle_L(i),adj_angle_H(i));//90
           delay(100);
        }

        return i-1;
}

unsigned char left (int xx)
{
        int i;
        for (i = xx;i < 136;i++)
          {
                setPWM(board1,pin0,0x00,0x00,adj_angle_L(i),adj_angle_H(i));//135
           delay(100);
          }
         return i-1;
}

unsigned char right (int xx)
{
        int i;
        for (i = xx;i > 44;i--)
```

```
        {
            setPWM(board1,pin0,0x00,0x00,adj_angle_L(i),adj_angle_H(i));//45
          delay(100);
        }
        return i+1;
}
```

副程式 IR_remote.C

```
//#############################################################################
//#        IR_remote decoder                                                  #
//#        using INT0 and TIMER0 to capture the high time for decoding        #
//#        sensor output connect to INT0                                      #
//#############################################################################

#include "SM59R16G6.h"
//#include "Debug_Pin.h"
//define state mode

#define        IDLE             0
#define        BYTE0             1
#define        BYTE1             2
#define        BYTE2             3
#define        BYTE3             4
#define        KEY_HOLD        5

#define        NEW_FRAME_LOW            (4500*0.9)/4.34
#define        NEW_FRAME_HIGH           (4500*1.1)/4.34
#define        SAME_KEY_LOW             (2200*0.9)/4.34
#define        SAME_KEY_HIGH            (2200*1.1)/4.34
#define        ONE_LOW                  (1690*0.9)/4.34
#define        ONE_HIGH                 (1690*1.1)/4.34
#define        ZERO_LOW                 (560*0.82)/4.34
#define        ZERO_HIGH                (560*1.1)/4.34
#define        HEADER_LOW               (4500*0.9)/4.34
#define        HEADER_HIGH              (4500*1.1)/4.34
#define        KEYHOLD_PERIOD           (98190*1.2)/4.34
#define        KEYHOLD_FRAME_LOW      (2250*0.9)/4.34
#define        KEYHOLD_FRAME_HIGH    (2250*1.1)/4.34

#define        CLEAR_TIMER0             TL0=0;TH0=0
#define        SET_KEYHOLD_TIMER      TH0=(65535-KEYHOLD_PERIOD)/256;\
                                      TL0=256-(KEYHOLD_PERIOD-
((KEYHOLD_PERIOD/256)*265));
```

```
unsigned char           data0;
unsigned char           data1;
unsigned char           data2;
unsigned char           data3;
unsigned char           key_edge=0;
unsigned char           key_level=0;
char        state=0;
char        cnt=0;

    void ini_remote(void)
    {   TMOD&=0xF0;         //set timer0 16 bits timer
        TMOD|=0x09;
        PFCON&=0xFC;        //set timer0 clk using OSC/96
        PFCON|=0x02;        //OSC 22.1184M (4.34us/clk;284ms period)
        CLEAR_TIMER0;
        ET0=1;              //enable timer0 inturrupt

        IT0=1;              //set EXT0 falling edge trigger
        EX0=1;              //enable EXT0 interrupt
        EA=1;               //enable all interrupt
    }


    void ir_decode(void) interrupt d_INT0_Vector using 3
    {   int timer;
        timer=(int)TH0*256+TL0;         //collect timer data for process
        CLEAR_TIMER0;                   //clear timer for next capture
        TR0=1;                          //arm timer0 capture action

        switch(state){                  //wait for legit header
            case IDLE:
                if(timer>NEW_FRAME_LOW && timer<NEW_FRAME_HIGH)
                {   data0=0;            //prepare to receive bit string data
                    cnt=8;
                    state=BYTE0;
                }
                break;
```

```
case BYTE0:
    data0>>=1;
    if(timer>ONE_LOW && timer<ONE_HIGH)data0|=0x80;
    else if(timer>ZERO_LOW && timer<ZERO_HIGH);
    else
    {   state=IDLE;
        //TOOGLE_TEST_PIN1;TOOGLE_TEST_PIN1;//##########
        break;
    }
    if(--cnt==0)
    {   data1=0;
        cnt=8;
        state=BYTE1;
    }
    break;

case BYTE1:
    data1>>=1;
    if(timer>ONE_LOW && timer<ONE_HIGH)data1|=0x80;
    else if(timer>ZERO_LOW && timer<ZERO_HIGH);
    else
    {   state=IDLE;
        //TOOGLE_TEST_PIN1;TOOGLE_TEST_PIN1;//##########
        break;
    }
    if(--cnt==0)
    {   data2=0;
        cnt=8;
        state=BYTE2;
    }
    break;

case BYTE2:
    data2>>=1;
    if(timer>ONE_LOW && timer<ONE_HIGH)data2|=0x80;
    else if(timer>ZERO_LOW && timer<ZERO_HIGH);
    else
    {   state=IDLE;
```

```
                //TOOGLE_TEST_PIN1;TOOGLE_TEST_PIN1;//##########
                break;
            }
            if(--cnt==0)
            {   data3=0;
                cnt=8;
                state=BYTE3;
            }
            break;

        case BYTE3:
            data3>>=1;
            if(timer>ONE_LOW && timer<ONE_HIGH)data3|=0x80;
            else if(timer>ZERO_LOW && timer<ZERO_HIGH);
            else
            {   state=IDLE;
                //TOOGLE_TEST_PIN1;TOOGLE_TEST_PIN1;//##########
                break;
            }
            if(--cnt==0)
            {   if((data0==0) && ((data0+data1)==0xFF) && ((data2+data3)==0xFF))
                {   key_edge=data3;
                    key_level=data3;
                    SET_KEYHOLD_TIMER;
                    state=KEY_HOLD;
                }
                else state=IDLE;
            }
            break;

        case KEY_HOLD:
            if(timer>KEYHOLD_FRAME_LOW && timer<KEYHOLD_FRAME_HIGH)
            {   SET_KEYHOLD_TIMER;
            }
            else state=IDLE;
            break;
    }
}
```

```
void timer0_overflow(void) interrupt d_T0_Vector using 2
{     //TOOGLE_TEST_PIN2;
      TR0=0;
      CLEAR_TIMER0;
      key_level=0;
}
```

副程式 motor.C

```
#include "SM59R16G6.h"

/*void adelay(int count)
{
	int i,j;
	for(i=0;i<count;i++)
	{
		for(j=0;j<count;j++);
	}
}*/

void forward(void)
{
	P1_0=1;
	P1_1=0;
	//adelay(500);
}

void backward(void)
{
	P1_0=0;
	P1_1=1;
	//adelay(500);
}

void astop(void)
{
	P1_0=0;
	P1_1=0;
	//adelay(500);
}
```

副程式 pca9685_for_i2c.C

```
#include "SM59R16G6.h"

sbit sda = P1^7;
sbit scl = P1^6;


    void usdelay(unsigned int delay_cnt) //for 22.1MHz OPT:4 ~(2.33+delay_cnt*1.5)uS
    {
            while (delay_cnt>0)delay_cnt--;
    }



    void start()//start
    {
                    scl = 1;
        usdelay(5);     //bus free time between stop and start >4.7us
        sda = 1;
        usdelay(5);     //setup times (r)START condition >4.7us
        sda = 0;
        usdelay(4);     //hold time (r)START condition >4us
    }

    void send_8bits(unsigned char byte_data)//write
    {
            unsigned char bit_ptr=0x80;

        while(bit_ptr)
        {
                                scl = 0;
          sda =(byte_data & bit_ptr);
          bit_ptr>>=1;
         usdelay(5);
          scl = 1;
          while(scl==0);    //in case of slave time strench
          usdelay(3);
```

```
    }
    scl = 0;       //finish bit data transfer
    sda = 1;       //release sda line for ACK
}

 char read_8bits(void)
      {
            unsigned char bit_ptr=0x80;
            unsigned char dt=0;
            P1_5 =0;
            P1_5 =1;
            sda = 1;

            while(bit_ptr)
            {
                  scl = 1;
                  if(sda)
                  {
                        dt|=bit_ptr;
                  }
                  scl = 0;
                  bit_ptr>>=1;
                  scl =0;

            }
            P1_5 =0;
            return dt;

      }


char pulling_ack(void)//ACK
{
            sda = 1;                  //assure sda line is released
    usdelay(4);                 //3.4us for valid data
    if(sda == 0)
    {
                              scl = 1;
```

```
        while(scl==0);    //in case clock stretching
        usdelay(4);
        scl = 0;
        return 0;
    }
    else
    {
                                scl = 1;
        while(scl==0);    //in case clock stretching
        usdelay(4);
        scl = 0;
        return 1;
    }
        }
void return_nack(void)
        {
                sda = 1;
                scl = 1;
                scl = 0;
        }
void stop(void)//stop
{
        sda = 0;
    scl = 1;
    usdelay(5);
    sda = 1;
}

void pca9685_write_command(char add, char rag ,char dat)
{
                add=add*2;
                start();
    send_8bits(add);
    pulling_ack();
    send_8bits(rag);
    pulling_ack();
                send_8bits(dat);
    pulling_ack();
```

```
        stop();
    }
        void setPWM(char add, char rag ,char LEDn_ON_L,char LEDn_ON_H,char
LEDn_OFF_L,char LEDn_OFF_H)
        {
                add=add*2;
                start();
        send_8bits(add);
        pulling_ack();
        send_8bits(rag);
        pulling_ack();
                send_8bits(LEDn_ON_L);
        pulling_ack();
                send_8bits(LEDn_ON_H);
        pulling_ack();
                send_8bits(LEDn_OFF_L);
        pulling_ack();
                send_8bits(LEDn_OFF_H);
        pulling_ack();
                stop();
        }
        char pca9685_read_command( char add )
        {
                char dt;
                add=add*2+1;
                start();
        send_8bits(add);
        pulling_ack();
                dt=read_8bits();
        return_nack();
                stop();
                return dt;
        }
        void blank(char add , char rag )
        {
            add=add*2;
            start();
            send_8bits(add);
```

```
        pulling_ack();
        send_8bits(rag);
        pulling_ack();
        stop();
    }

void begin (void)
    {
        pca9685_write_command(0x40,0x00,0x80);
        pca9685_write_command(0x41,0x00,0x80);
        blank(0x40,0x00);
        blank(0x41,0x00);
        pca9685_read_command(0x40);
        pca9685_read_command(0x41);
        pca9685_write_command(0x40,0x00,0x10);
        pca9685_write_command(0x41,0x00,0x10);
        pca9685_write_command(0x40,0xFE,0x05);
        pca9685_write_command(0x41,0xFE,0x05);
        pca9685_write_command(0x40,0x00,0x00);
        pca9685_write_command(0x41,0x00,0x00);
        pca9685_write_command(0X40,0X00,0XA0);
        pca9685_write_command(0X41,0X00,0XA0);
    }
  void PWMfreaq_set(void)
    {
        blank(0x40,0x00);
        blank(0x41,0x00);
        pca9685_read_command(0x40);
        pca9685_read_command(0x41);
        pca9685_write_command(0x40,0x00,0x30);
        pca9685_write_command(0x41,0x00,0x30);
        pca9685_write_command(0x40,0xFE,0x79);//4B
        pca9685_write_command(0x41,0xFE,0x4B);
        pca9685_write_command(0x40,0x00,0x20);
        pca9685_write_command(0x41,0x00,0x20);
        pca9685_write_command(0x40,0x00,0xA0);
        pca9685_write_command(0x41,0x00,0xA0);
    }
```

副程式 control_middle.h

```
extern void reset(void);
extern unsigned char adj_angle_L(unsigned int);
extern unsigned char adj_angle_R(unsigned int);
extern unsigned char middle_L(int);
extern unsigned char middle_R(int);
extern unsigned char left(int);
extern unsigned char right(int);
```

副程式 IR_remote.h

```
extern void ini_remote(void);

extern unsigned char        key_edge;       /*nonzero hex data represent key edge data; clear after
action */
extern unsigned char        key_level;      /*nonzero hex data represent key pressed and hold */
```

副程式 motor.h

```
extern void astop(void);
extern void forward(void);
extern void backward(void);
```

副程式 pca9685_for_i2c.h

```
extern void setPWM(char add, char rag ,char LEDn_ON_L,char LEDn_ON_H,char
LEDn_OFF_L,char LEDn_OFF_H);
extern void begin(void);
extern void PWMfreaq_set(void);
```