介面實驗

實驗六

PWM 輸出及橋式驅動電路

班級：機械 3A

學號：108303013

姓名：黃鉦淳

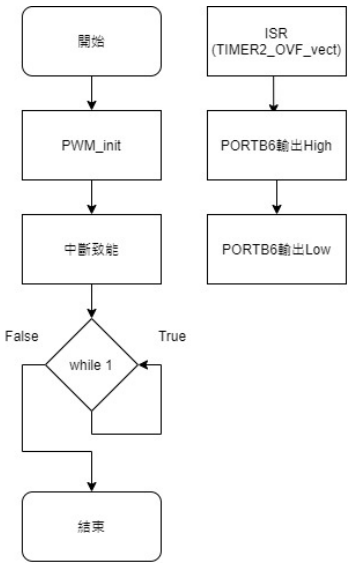
日期：110/12/16

介面實驗工作日誌

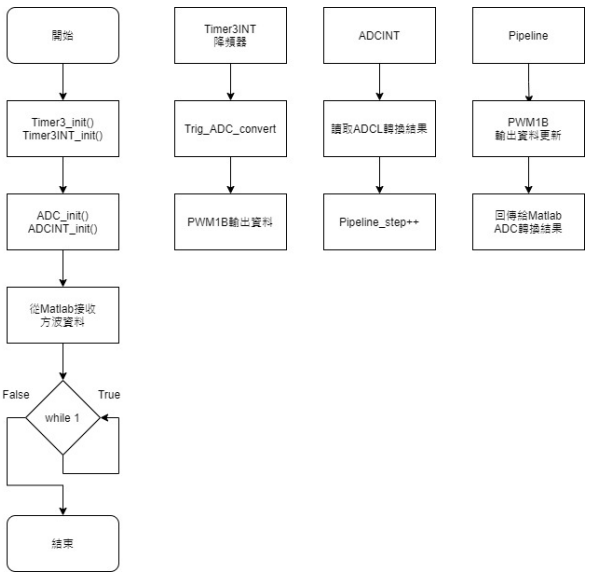
實驗六　　110 年 12 月 16 日

組別		姓名	黃鉦淳	學號	108303013
實驗起始時間		110/8/5		費時	4 個月
實驗結束時間		110/12/16			
所遭遇問題	載到舊版的 Matlab 函式庫☹				
解決方法	狂問學長				
完成項目及心得	及格了!				
調查	□是否有看課程講解影片 是否實用？有何建議？		□是否有看實驗教學影片 是否實用？有何建議？		

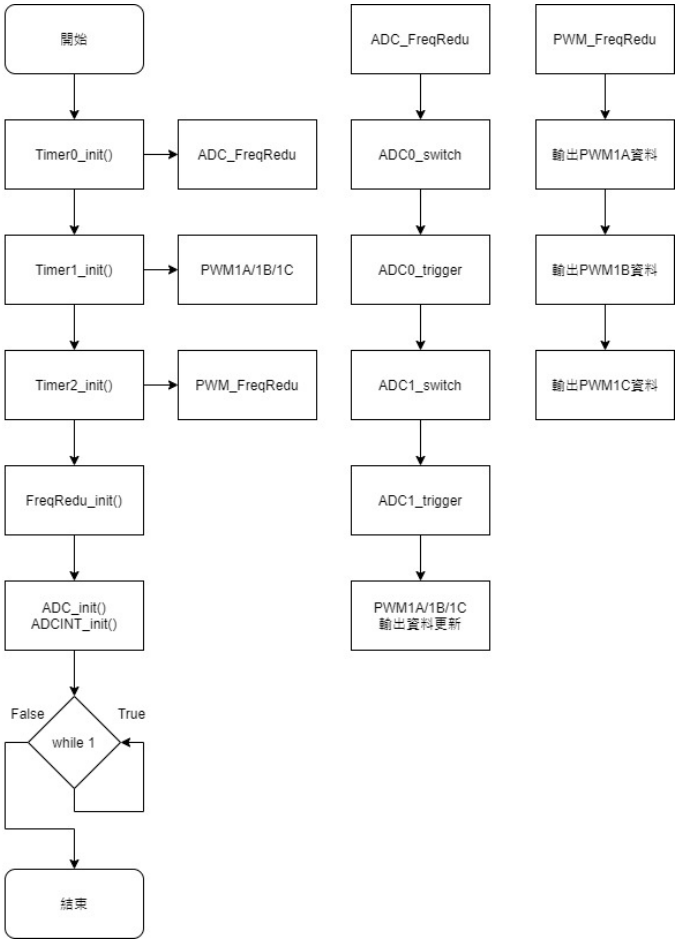
一、流程圖



實驗一



實驗二



實驗三

二、程式碼

實驗一

Language : C

```
#include "c4mlib.h"

void PWM_init();

ISR(TIMER2_OVF_vect)
{
	REGFPT(&PORTB, 0x40, 6, 1);
	printf("OVERFLOW\n");
	REGFPT(&PORTB, 0x40, 6, 0);
}

int main()
{
	C4M_DEVICE_set();
	PWM_init();
	sei();
	while (1)
	{
		;
	}
	return 0;
}

void PWM_init()
{
	REGFPT(&TCCR2, 0x48, 3, 9); //Fast PWM
	REGFPT(&TCCR2, 0x07, 0, 4); //除頻值 1024
	REGFPT(&TCCR2, 0x30, 4, 3); //正脈波
	REGFPT(&TIMSK, 0x40, 6, 1); //Overflow Flag 中斷致能
	OCR2 = 107;
	DDRB = (DDRB & (~(0xc0))) | 0xc0; //PB6 觸發腳位 || PB7 OC2
}
```

實驗二

Language : C

```
#include "c4mlib.h"

void ADC_init();
void ADCPostPro_step(void *VoidStr_p);
void PWMPrePro_step(void *VoidStr_p);

void timer3_init();

typedef struct
{
	uint16_t *InData_p;		/*Pointer points to the Input Data Source */
	uint16_t *DataList_p;		/* Pointer points to the buffer array */
	uint8_t DataLength;		/* Length of Datalist */
	uint8_t DataCount;		/*Data Count of the data in list*/
	uint8_t TaskId;			//The TaskId got after registered
	uint8_t NextTaskNum;		//Number of Next Task
	uint8_t *NextTask_p;		//pointer to the List of TaskId for NextTasks
	volatile uint8_t TrigCount; //Triggered Counter
} ADCPostProStr_t;

typedef struct
{
	uint16_t OutData;
	uint16_t *OutData_p;		/*Pointer points to the Output Data Source
*/
	uint16_t *DataList_p;		/* Pointer points to the Out buffer array */
	uint8_t DataLength;		/* Length of Datalist */
	uint8_t DataCount;		/*Data Count of the data in list*/
	uint8_t TaskId;			//The TaskId got after registered
	uint8_t NextTaskNum;		//Number of Next Task
	uint8_t *NextTask_p;		//pointer to the List of TaskId for NextTasks
	volatile uint8_t TrigCount; //Triggered Counter
} PwmPreProStr_t;
```

```
#define ADCPOSTPRO_LAY(ADCPostPro_str, ListNum, _NextTaskNum,
InDataAdd) \
    uint16_t ADCPostPro_str##_DataList[ListNum];
\
    uint8_t ADCPostPro_str##_NextTaskList[_NextTaskNum];
\
    ADCPostProStr_t ADCPostPro_str =
\

{                                                                    \
        .InData_p = InDataAdd,
\
        .DataList_p = ADCPostPro_str##_DataList,
\
        .DataLength = ListNum,
\
        .DataCount = 0,
\
        .TaskId = 0,
\
        .NextTaskNum = _NextTaskNum,
\
        .NextTask_p = ADCPostPro_str##_NextTaskList,
\
        .TrigCount = 0}

#define PwmPrePro_LAY(PwmPreProStr, ListNum, _NextTaskNum) \
    uint16_t PwmPreProStr##_DataList[ListNum];              \
    uint8_t PwmPreProStr##_NextTaskList[_NextTaskNum];      \
    PwmPreProStr_t PwmPreProStr = {                         \
        .DataList_p = PwmPreProStr##_DataList,              \
        .DataLength = ListNum,                              \
        .DataCount = 0,                                     \
        .TaskId = 0,                                        \
        .NextTaskNum = _NextTaskNum,                        \
        .NextTask_p = PwmPreProStr##_NextTaskList}

int main()
```

```
{
    C4M_DEVICE_set();

    DDRB = 0xff;

    //設定 Timer
    TIM3_HW_LAY();
    hardware_set(&TIM1_3HWSet_str);

    //設定 Timer3 中斷
    TIMHWINT_LAY(TIMINT_Str, 3, 1);
    timer3_init();

    //設定 PWM
    PWM2_HW_LAY();
    hardware_set(&PWM2HWSet_str);

    //設定 PWM 中斷
    PWMHWINT_LAY(PwmInt_Str, 0, 1);
    hardware_set(&PwmInt_Str);

    //設定 ADC
    ADC_HW_LAY();
    hardware_set(&ADCHWSet_str);
    ADC_init();

    //設定 ADC 中斷
    ADCHWINT_LAY(ADCHWINT_Str, 0, 2);
    hardware_set(&ADCHWINT_Str);

    uint16_t ADC_readData;
    ADCPOSTPRO_LAY(ADCPostPro_Str, 200, 1, &ADC_readData);
    PwmPrePro_LAY(PwmPrePro_Str, 200, 1);

    int period[2];
    for (int i = 0; i < sizeof(period) / sizeof(int); i++)
        period[i] = 100;
```

```
    FREQREDU_LAY(PWMFreqRedu_Str, 2, 2, &OCR3A, 2, &period);
    int8_t freq_TaskID[2];
    uint8_t one = 1;
    RT_FLAG_IO_LAY(ADCTrig_Str, 0, &ADCSRA, 0x40, 6, &one);
    freq_TaskID[0] = FreqRedu_reg(&PWMFreqRedu_Str,
&RealTimeFlagPut_step, &ADCTrig_Str, 1, 1);

    RT_REG_IO_LAY(PWMSend_Str, 0, &PORTB, 1,
&PwmPrePro_Str.OutData);
    freq_TaskID[1] = FreqRedu_reg(&PWMFreqRedu_Str,
&RealTimeRegPut_step, &PWMSend_Str, 1, 1);

    FreqRedu_en(&PWMFreqRedu_Str, freq_TaskID[0], ENABLE);
    FreqRedu_en(&PWMFreqRedu_Str, freq_TaskID[1], ENABLE);

    //設定 Pipeline
    PIPELINE_LAY(2, 4, 10);
    //PWM 訊號準備
    Pipeline_reg(&SysPipeline_str, &PWMPrePro_step, &PwmPrePro_Str,
NULL);
    Pipeline_reg(&SysPipeline_str, &ADCPostPro_step, &ADCPostPro_Str,
NULL);

    uint8_t TaskID[3];

    //將降頻器登入進 Timer 中斷
    TaskID[0] = HWInt_reg(&TIMINT_Str, &FreqRedu_step,
&PWMFreqRedu_Str);
    HWInt_en(&TIMINT_Str, TaskID[0], ENABLE);

    //將 ADC 結果登入進 ADC 中斷
    RT_REG_IO_LAY(ADCRead_Str, 1, &ADCL, 2, (uint8_t
*)&ADC_readData);
    TaskID[1] = HWInt_reg(&ADCHWINT_Str, &RealTimeRegGet_step,
&ADCRead_Str);
    HWInt_en(&ADCHWINT_Str, TaskID[1], ENABLE);

    //將 pipeline 登入進 ADC 中斷
```

```
	TaskID[2] = HWInt_reg(&ADCHWINT_Str, &Pipeline_step,
&SysPipeline_str);
	HWInt_en(&ADCHWINT_Str, TaskID[2], ENABLE);

	sei();
	TRIG_NEXT_TASK(0);

	HMI_snget_matrix(HMI_TYPE_UI16, 1, PwmPrePro_Str.DataLength,
PwmPrePro_Str.DataList_p);

	while (1)
	{
		;
	}
	return 0;
}

void ADC_init()
{
	//設定內部 2.56V
	REGFPT(&ADMUX, 0xC0, 6, 3);

	//設定 10 位元轉換靠右
	REGFPT(&ADMUX, 0x20, 5, 0);

	//設定非連續或觸發轉換
	REGFPT(&ADCSRA, 0x20, 5, DISABLE);

	//設定 ADC1 F1 輸入
	REGFPT(&DDRF, 0x02, 0, 0);

	//設定致能 ADC
	REGFPT(&ADCSRA, 0x80, 7, ENABLE);

	//設定致能 ADC Interrupt
	REGFPT(&ADCSRA, 0x08, 3, ENABLE);

	//設定 ADC 時脈 clk/128
```

```
	REGFPT(&ADCSRA, 0x07, 0, 7);

	//設定 ADC1 and GND
	REGFPT(&ADMUX, 0x1f, 0, 1);
}

void timer3_init()
{
	//normal mode
	REGFPT(&TCCR3A, 0x03, 0, 0);
	//normal mode
	REGFPT(&TCCR3B, 0x18, 3, 1);
	//設定 timer 時脈 clk/1024
	REGFPT(&TCCR3B, 0x07, 0, 5);

	OCR3A = 269;
	//設定 timer3A 致能
	REGFPT(&ETIMSK, 0x10, 4, 1);

	// freq = 11059200 / (2 * 1024 * (1 + 100)) = 53.46
}

void PWMPrePro_step(void *VoidStr_p)
{
	volatile PwmPreProStr_t *Str_p = (PwmPreProStr_t *)VoidStr_p;

	if (Str_p->DataCount == Str_p->DataLength)
	{
		Str_p->DataCount = 0;
	}

	//PB6 輸出
	Str_p->OutData = Str_p->DataList_p[Str_p->DataCount] << 6;
	Str_p->DataCount++;
	TRIG_NEXT_TASK(1); /* pipeline task trigger*/
}

void ADCPostPro_step(void *VoidStr_p)
```

```
{
	volatile ADCPostProStr_t *Str_p = (ADCPostProStr_t *)VoidStr_p;

	Str_p->DataList_p[Str_p->DataCount] = *(Str_p->InData_p);

	if ((Str_p->DataCount + 1) == (Str_p->DataLength))
	{
		cli();
		HMI_snput_matrix(HMI_TYPE_UI16, 1, Str_p->DataLength, Str_p-
>DataList_p);
		Str_p->DataCount = 0;
		sei();
	}
	else
		Str_p->DataCount++;
	TRIG_NEXT_TASK(0);
}
```

Language : Matlab

```
clear;clc;close;

t = linspace(0, pi, 200);
x = square(t, 25);

for i = 1:200

    if (x(i) < 0)
        x(i) = 0;
    end

end

[port] = remo_open(8);
remo_snput_matrix(port, uint16(x));

while 1
    [data] = remo_snget_matrix(port);

    subplot(1, 2, 1);
    plot(t, 2 * x);
    title('Input PWM');
    xlabel('Time[t]');
    ylabel('Voltage[V]');
    ylim([0, 3]);

    subplot(1, 2, 2);
    data = data * 2.56/1024;
    plot(t, data);
    title('ADC convert Output PWM');
    xlabel('Time[t]');
    ylabel('Voltage[V]');
    ylim([0, 3]);
end
remo_close(port);
```

實驗三

Language : C

```
#include "c4mlib.h"

void ADC_init();
void ADC0PostPro_step(void *VoidStr_p);
void ADC1PostPro_step(void *VoidStr_p);

void PWMPrePro_step(void *VoidStr_p);

void timer0_init();
void timer1_init();
void timer2_init();

typedef struct
{
	uint16_t *InData_p;		/*Pointer points to the Input Data Source */
	uint16_t *DataList_p;		/* Pointer points to the buffer array */
	uint8_t DataLength;		/* Length of Datalist */
	uint8_t DataCount;		/*Data Count of the data in list*/
	uint8_t TaskId;			// The TaskId got after registered
	uint8_t NextTaskNum;		// Number of Next Task
	uint8_t *NextTask_p;		// pointer to the List of TaskId for
NextTasks
	volatile uint8_t TrigCount; // Triggered Counter
} ADCPostProStr_t;

typedef struct
{
	uint16_t OutData;
	uint16_t *OutData_p;	/*Pointer points to the Output Data Source */
	uint16_t *DataList_p; /* Pointer points to the Out buffer array */
	uint8_t DataLength;	/* Length of Datalist */
	uint8_t DataCount;	/*Data Count of the data in list*/
	uint8_t channel_num;
	uint16_t *OutArrList_p;
} PwmPreProStr_t;
```

```
#define ADCPOSTPRO_LAY(ADCPostPro_str, ListNum, _NextTaskNum,
InDataAdd) \
    uint16_t ADCPostPro_str##_DataList[ListNum];
\
    uint8_t ADCPostPro_str##_NextTaskList[_NextTaskNum];
\
    ADCPostProStr_t ADCPostPro_str =
\

{                                                                    \
            .InData_p = InDataAdd,
\
            .DataList_p = ADCPostPro_str##_DataList,
\
            .DataLength = ListNum,
\
            .DataCount = 0,
\
            .TaskId = 0,
\
            .NextTaskNum = _NextTaskNum,
\
            .NextTask_p = ADCPostPro_str##_NextTaskList,
\
            .TrigCount = 0}

#define PwmPrePro_LAY(PwmPreProStr, ListNum, _channel_num) \
    uint16_t PwmPreProStr##_ArrayList[_channel_num];          \
    uint16_t PwmPreProStr##_DataList[ListNum];                \
    PwmPreProStr_t PwmPreProStr = {                             \
        .DataList_p = PwmPreProStr##_DataList,                \
        .DataLength = ListNum,                                \
        .DataCount = 0,                                       \
        .channel_num = _channel_num,                            \
        .OutArrList_p = PwmPreProStr##_ArrayList}
```

```
#define ADCPOSTPRO_LAY(ADCPostPro_str, ListNum, _NextTaskNum,
InDataAdd) \
    uint16_t ADCPostPro_str##_DataList[ListNum];
\
    uint8_t ADCPostPro_str##_NextTaskList[_NextTaskNum];
\
    ADCPostProStr_t ADCPostPro_str =
\

{                                                                    \
            .InData_p = InDataAdd,
\
            .DataList_p = ADCPostPro_str##_DataList,
\
            .DataLength = ListNum,
\
            .DataCount = 0,
\
            .TaskId = 0,
\
            .NextTaskNum = _NextTaskNum,
\
            .NextTask_p = ADCPostPro_str##_NextTaskList,
\
            .TrigCount = 0}

int main()
{
    C4M_DEVICE_set();

    /*=================TIMER_INIT================*/
    //設定 Timer
    // Timer 0          => ADC_Frequency_redu
    // Timer 1 A/B/C => PWM
    // Timer 2          => PWM_Frequency_redu

    TIMHWINT_LAY(TIM0INT_Str, 0, 2);
    timer0_init();
```

```
    timer1_init();

    TIMHWINT_LAY(PWM2INT_Str, 2, 1);
    timer2_init();

    /*================PWM================*/

    PwmPrePro_LAY(PwmPrePro_Str, 180, 3);
    //設定 PWM 降頻器
    uint8_t PWM_period[3];
    for (int i = 0; i < sizeof(PWM_period) / sizeof(uint8_t); i++)
        PWM_period[i] = 10;

    FREQREDU_LAY(PWMFreqRedu_Str, 3, 3, &OCR2, 1, &PWM_period);

    RT_REG_IO_LAY(PWM1ASet_Str, 0, &OCR1AL, 2,
(PwmPrePro_Str.OutArrList_p + 0));
    RT_REG_IO_LAY(PWM1BSet_Str, 1, &OCR1BL, 2,
(PwmPrePro_Str.OutArrList_p + 1));
    RT_REG_IO_LAY(PWM1CSet_Str, 2, &OCR1CL, 2,
(PwmPrePro_Str.OutArrList_p + 2));

    uint8_t PWM_freq_TaskID[3];
    PWM_freq_TaskID[0] = FreqRedu_reg(&PWMFreqRedu_Str,
&RealTimeRegPut_step, &PWM1ASet_Str, 1, 0);
    PWM_freq_TaskID[1] = FreqRedu_reg(&PWMFreqRedu_Str,
&RealTimeRegPut_step, &PWM1BSet_Str, 1, 1);
    PWM_freq_TaskID[2] = FreqRedu_reg(&PWMFreqRedu_Str,
&RealTimeRegPut_step, &PWM1CSet_Str, 1, 2);

    FreqRedu_en(&PWMFreqRedu_Str, PWM_freq_TaskID[0], ENABLE);
    FreqRedu_en(&PWMFreqRedu_Str, PWM_freq_TaskID[1], ENABLE);
    FreqRedu_en(&PWMFreqRedu_Str, PWM_freq_TaskID[2], ENABLE);

    /*================ADC================*/
    //設定 ADC
    // ADC_HW_LAY();
```

```
    // hardware_set(&ADCHWSet_str);
    ADC_init();

    //設定 ADC 降頻器
    uint8_t ADC_period[9];
    for (int i = 0; i < sizeof(ADC_period) / sizeof(uint8_t); i++)
        ADC_period[i] = 10;

    FREQREDU_LAY(ADCFreqRedu_Str, 9, 9, &OCR0, 1, &ADC_period);

    unsigned int ADC0_read_Data;
    unsigned int ADC1_read_Data;

    ADCPOSTPRO_LAY(ADC0_PostPro_Str, 180, 0, &ADC0_read_Data);
    ADCPOSTPRO_LAY(ADC1_PostPro_Str, 180, 1, &ADC1_read_Data);

    RT_REG_IO_LAY(ADC0_result_Str, 3, &ADCL, 2, (uint8_t
*)&ADC0_read_Data);
    RT_REG_IO_LAY(ADC1_result_Str, 4, &ADCL, 2, (uint8_t
*)&ADC1_read_Data);

    unsigned char zero = 0;
    unsigned char one = 1;

    //切換到單通道 ADC0 and GND
    RT_FLAG_IO_LAY(ADC0_switch_channel_Str, 0, &ADMUX, 0x1f, 0,
&zero);
    //觸發單通道
    RT_FLAG_IO_LAY(ADC0_trigger_convert_Str, 1, &ADCSRA, 0x40, 6,
&one);

    //切換到單通道 ADC1 and GND
    RT_FLAG_IO_LAY(ADC1_switch_channel_Str, 2, &ADMUX, 0x1f, 0,
&one);
    //觸發單通道
    RT_FLAG_IO_LAY(ADC1_trigger_convert_Str, 3, &ADCSRA, 0x40, 6,
&one);
```

```
    uint8_t ADC_freq_TaskID[9];
    ADC_freq_TaskID[0] = FreqRedu_reg(&ADCFreqRedu_Str,
&RealTimeFlagPut_step, &ADC0_switch_channel_Str, 1, 0);
    ADC_freq_TaskID[1] = FreqRedu_reg(&ADCFreqRedu_Str,
&RealTimeFlagPut_step, &ADC0_trigger_convert_Str, 1, 1);
    ADC_freq_TaskID[2] = FreqRedu_reg(&ADCFreqRedu_Str,
&RealTimeRegGet_step, &ADC0_result_Str, 1, 2);
    ADC_freq_TaskID[3] = FreqRedu_reg(&ADCFreqRedu_Str,
&ADC0PostPro_step, &ADC0_PostPro_Str, 1, 3);
    // ADC_freq_TaskID[3] = FreqRedu_reg(&ADCFreqRedu_Str,
&Pipeline_step, &SysPipeline_str, 1, 3);

    ADC_freq_TaskID[4] = FreqRedu_reg(&ADCFreqRedu_Str,
&RealTimeFlagPut_step, &ADC1_switch_channel_Str, 1, 4);
    ADC_freq_TaskID[5] = FreqRedu_reg(&ADCFreqRedu_Str,
&RealTimeFlagPut_step, &ADC1_trigger_convert_Str, 1, 5);
    ADC_freq_TaskID[6] = FreqRedu_reg(&ADCFreqRedu_Str,
&RealTimeRegGet_step, &ADC1_result_Str, 1, 6);
    ADC_freq_TaskID[7] = FreqRedu_reg(&ADCFreqRedu_Str,
&ADC1PostPro_step, &ADC1_PostPro_Str, 1, 7);

    ADC_freq_TaskID[8] = FreqRedu_reg(&ADCFreqRedu_Str,
&PWMPrePro_step, &PwmPrePro_Str, 1, 8);

    // ADC_freq_TaskID[7] = FreqRedu_reg(&ADCFreqRedu_Str,
&Pipeline_step, &SysPipeline_str, 1, 7);
    // ADC_freq_TaskID[8] = FreqRedu_reg(&ADCFreqRedu_Str,
&Pipeline_step, &SysPipeline_str, 1, 8);

    FreqRedu_en(&ADCFreqRedu_Str, ADC_freq_TaskID[0], ENABLE);
    FreqRedu_en(&ADCFreqRedu_Str, ADC_freq_TaskID[1], ENABLE);
    FreqRedu_en(&ADCFreqRedu_Str, ADC_freq_TaskID[2], ENABLE);
    FreqRedu_en(&ADCFreqRedu_Str, ADC_freq_TaskID[3], ENABLE);
    FreqRedu_en(&ADCFreqRedu_Str, ADC_freq_TaskID[4], ENABLE);
    FreqRedu_en(&ADCFreqRedu_Str, ADC_freq_TaskID[5], ENABLE);
    FreqRedu_en(&ADCFreqRedu_Str, ADC_freq_TaskID[6], ENABLE);
    FreqRedu_en(&ADCFreqRedu_Str, ADC_freq_TaskID[7], ENABLE);
    FreqRedu_en(&ADCFreqRedu_Str, ADC_freq_TaskID[8], ENABLE);
```

```
	//設定 Pipeline
	// PIPELINE_LAY(3, 5, 10);
	// PWM 訊號準備
	// uint8_t pipeline_ID[3];
	// pipeline_ID[0] = Pipeline_reg(&SysPipeline_str, &ADC0PostPro_step,
&ADC0_PostPro_Str, NULL);
	// pipeline_ID[1] = Pipeline_reg(&SysPipeline_str, &ADC1PostPro_step,
&ADC1_PostPro_Str, NULL);
	// pipeline_ID[2] = Pipeline_reg(&SysPipeline_str, &PWMPrePro_step,
&PwmPrePro_Str, NULL);

	/*================TOTAL================*/

	uint8_t TaskID[2];

	//將 PWM 降頻器登入進 Timer2 中斷
	TaskID[0] = HWInt_reg(&PWM2INT_Str, &FreqRedu_step,
&PWMFreqRedu_Str);
	HWInt_en(&PWM2INT_Str, TaskID[0], ENABLE);

	//將 ADC 降頻器登入進 Timer0 中斷
	TaskID[1] = HWInt_reg(&TIM0INT_Str, &FreqRedu_step,
&ADCFreqRedu_Str);
	HWInt_en(&TIM0INT_Str, TaskID[1], ENABLE);

	HMI_snget_matrix(HMI_TYPE_UI16, 1, PwmPrePro_Str.DataLength,
PwmPrePro_Str.DataList_p);

	TRIG_NEXT_TASK(0);
	sei();

	while (1)
	{
		;
	}

	return 0;
```

```
}

void ADC_init()
{
    //設定參考電壓:外部 AVCC
    REGFPT(&ADMUX, 0xc0, REFS0, 1);

    //設定 10 位元轉換靠右
    REGFPT(&ADMUX, 0x20, ADLAR, 0);

    //設定非連續或觸發轉換
    REGFPT(&ADCSRA, 0x20, ADFR, DISABLE);

    //設定 ADC0:2 F0:2 輸入
    REGFPT(&DDRF, 0x07, 0, 0);

    //設定致能 ADC
    REGFPT(&ADCSRA, 0x80, ADEN, ENABLE);

    //設定 ADC 時脈  clk/128
    REGFPT(&ADCSRA, 0x07, ADPS0, 7);
}

void timer0_init()
{
    REGFPT(&TCCR0, 0x48, WGM01, 1); /*CTC*/
    REGFPT(&TCCR0, 0x07, CS00, 5);    /*clk/128*/
    REGFPT(&TIMSK, 0x02, OCIE0, 1); /*致能中斷*/
    OCR0 = 107;

    //時間: 0.02 [s]
}

void timer1_init()
{
    //  設定耦合 OCR1A/1B/1C
    REGFPT(&TCCR1A, 0xc0, COM1A0, 2);
    REGFPT(&TCCR1A, 0x30, COM1B0, 2);
```

```
    REGFPT(&TCCR1A, 0x0c, COM1C0, 2);

    // PWM, Phase and Frequency Correct | Top : ICRn
    REGFPT(&TCCR1A, 0x03, WGM10, 0);
    REGFPT(&TCCR1B, 0x18, WGM12, 2);

    ICR1 = 255;

    REGFPT(&TCCR1B, 0x07, CS10, 1); /*clk/1*/

    REGFPT(&DDRB, 0xe0, 5, 0x07);
}

void timer2_init()
{
    REGFPT(&TCCR2, 0x48, WGM01, 1); /*CTC*/
    REGFPT(&TCCR2, 0x07, CS00, 5);    /*clk/128*/
    REGFPT(&TIMSK, 0x80, OCIE2, 1); /*致能中斷*/
    OCR2 = 107;

    //時間: 0.02 [s]
}

void ADC0PostPro_step(void *VoidStr_p)
{
    volatile ADCPostProStr_t *Str_p = (ADCPostProStr_t *)VoidStr_p;
    Str_p->DataList_p[Str_p->DataCount] = *(Str_p->InData_p);

    if ((Str_p->DataCount + 1) == (Str_p->DataLength))
    {
        cli();
        HMI_snput_matrix(HMI_TYPE_UI16, 1, Str_p->DataLength, Str_p-
>DataList_p);
        Str_p->DataCount = 0;
        sei();
    }
    else
        Str_p->DataCount++;
```

```
    TRIG_NEXT_TASK(1);
}

void ADC1PostPro_step(void *VoidStr_p)
{
    volatile ADCPostProStr_t *Str_p = (ADCPostProStr_t *)VoidStr_p;
    Str_p->DataList_p[Str_p->DataCount] = *(Str_p->InData_p);

    if ((Str_p->DataCount + 1) == (Str_p->DataLength))
    {
        cli();
        HMI_snput_matrix(HMI_TYPE_UI16, 1, Str_p->DataLength, Str_p-
>DataList_p);
        Str_p->DataCount = 0;
        sei();
    }
    else
        Str_p->DataCount++;

    TRIG_NEXT_TASK(2);
}

void PWMPrePro_step(void *VoidStr_p)
{
    // printf("PWMPrePro_step\n");
    volatile PwmPreProStr_t *Str_p = (PwmPreProStr_t *)VoidStr_p;

    int DataIdx[3];

    if (Str_p->DataCount == Str_p->DataLength)
    {
        Str_p->DataCount = 0;
    }

    for (int i = 0; i < Str_p->channel_num; i++)
    {
        DataIdx[i] = Str_p->DataCount + i * Str_p->DataLength / 3;
```

```
            if (DataIdx[i] > Str_p->DataLength)
                DataIdx[i] -= Str_p->DataLength;

            Str_p->OutArrList_p[i] = Str_p->DataList_p[DataIdx[i]];
        }
        Str_p->DataCount++;

        TRIG_NEXT_TASK(0);
}
```

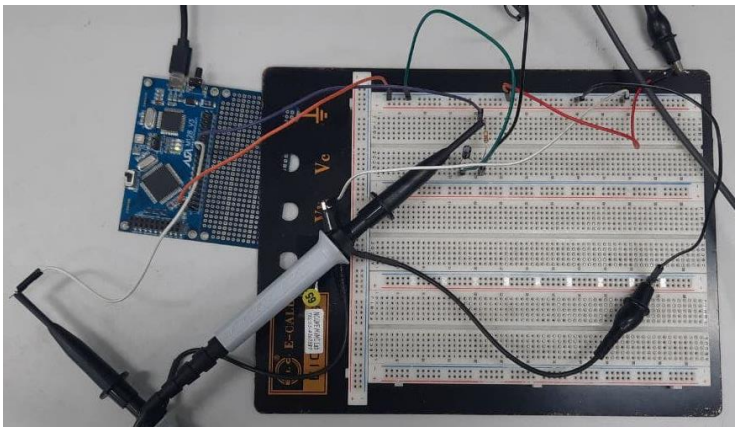
Language : Matlab

```
clear;clc;close;

port = remo_open(6);

t = linspace(0, 2 * pi, 180);
x = uint16((sin(t)+1) * 255/2);

volt_ref = 4.65;

remo_snput_matrix(port, x);

while 1
    [data1] = remo_snget_matrix(port);
    [data2] = remo_snget_matrix(port);
    data1 = double(data1)/1024*volt_ref;
    data2 = double(data2)/1024*volt_ref;

    plot(t,data1,t,data2);
    xlim([0 2*pi]);
    ylim([0 volt_ref]);
    xlabel('Time[t]');
    ylabel('Volt[V]');
    title('PWM Three-phase Sine Wave');
    legend('sin(t)','sin(t+\pi/3)');
end

remo_close(port);
```

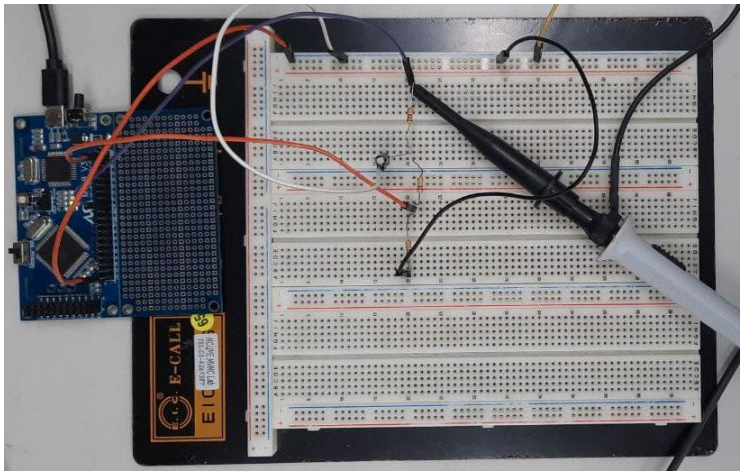
三、實驗數據

1.電路圖

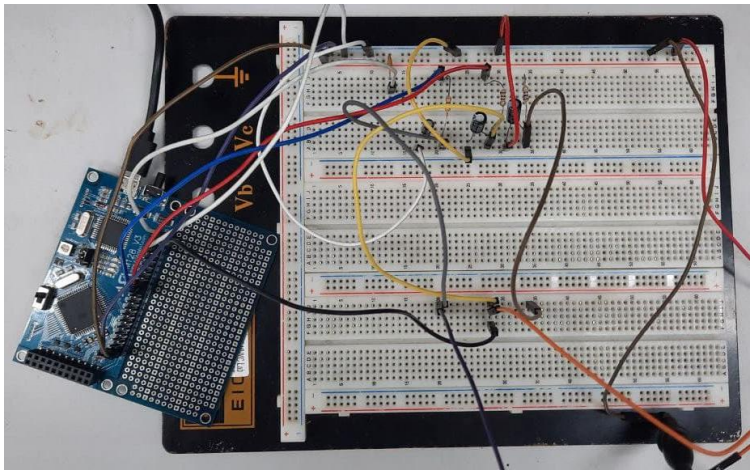
實驗一



實驗二



實驗三



2.實驗結果

實驗一

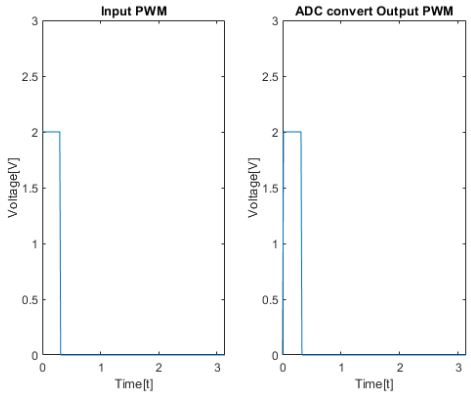
CH1 PWM產生的波型

CH2 PWM經過低通濾波器產生的波型

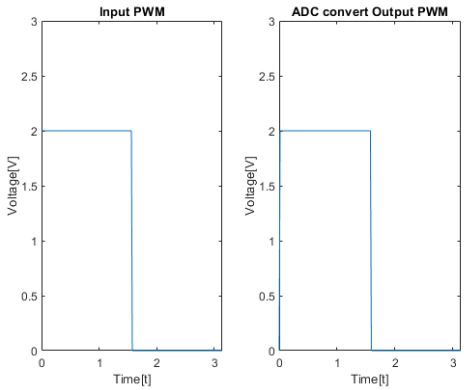
N = 256 OCR2 = 107

N = 1024 OCR2 = 107

實驗二

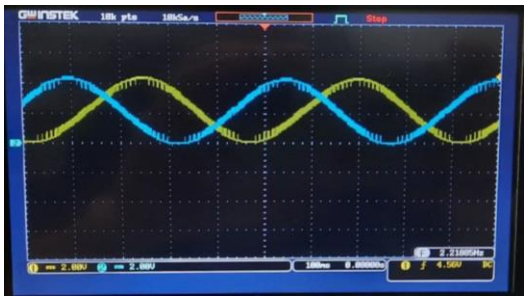


Time:0~π Duty:5%

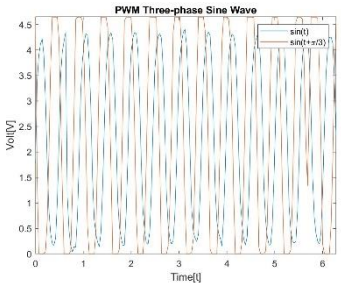


Time:0~π Duty:25%

實驗三



PWM 三相弦波



PWM 三相弦波

勘誤:橘線為 $\sin(t+2\pi/3)$

四、實驗問題

1.請討論負載為以功率電阻代替的喇叭與直接用喇叭，其電流波形有何不同，不同的原因可能是什麼?

電阻之電抗僅有實部，但喇叭之電抗包含了實部及虛部。

電阻定電壓下，輸出之電流不會隨著頻率，而有所改變(因電抗僅為實部);

喇叭定電壓下，輸出之電流會因頻率，而有所改變(因電抗包含為實部及虛部。