工科專論期末報告格式

組員: 涂皓瑋 B06505001 朱紹勳 B06501018 陳威宇 B06204013 黃智遠 B06505049

1. 分工表:

組員 涂皓瑋: 船模設計

組員 朱紹勳: 程式、電路設計

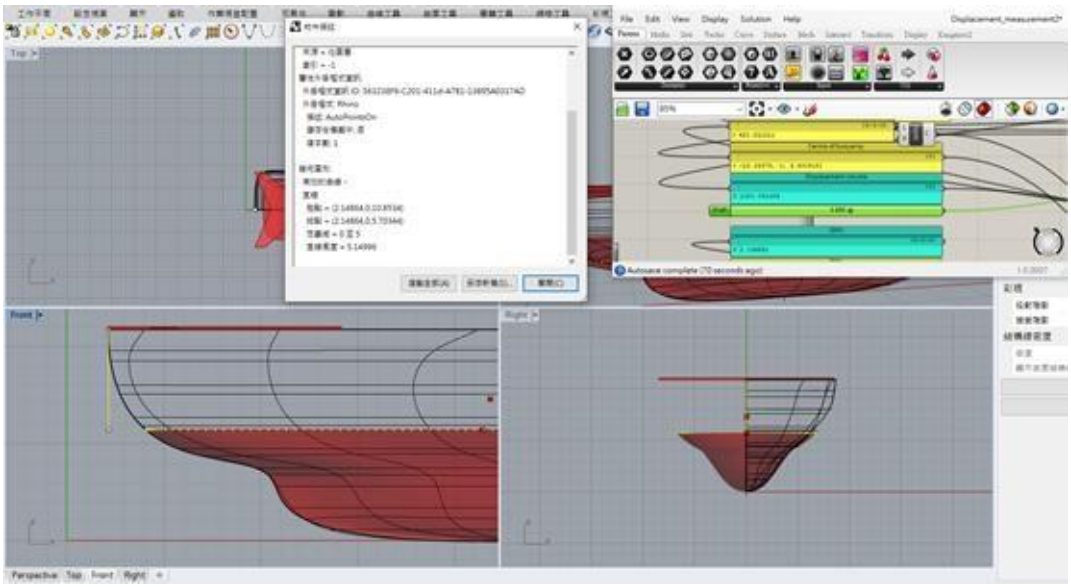
組員 陳威宇: 程式、測試

組員 黃智遠: 打雜

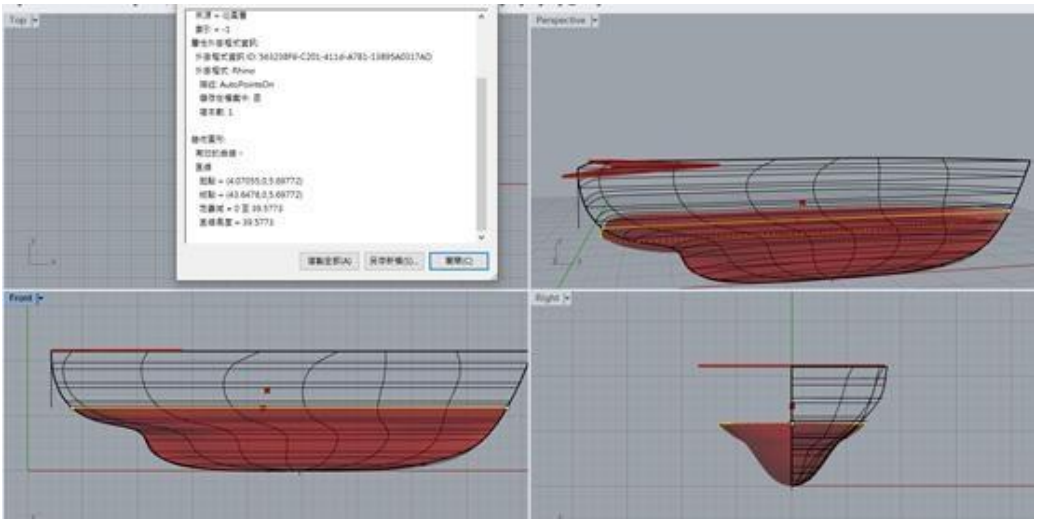
2. 船體設計:

乾舷約為 5.15cm

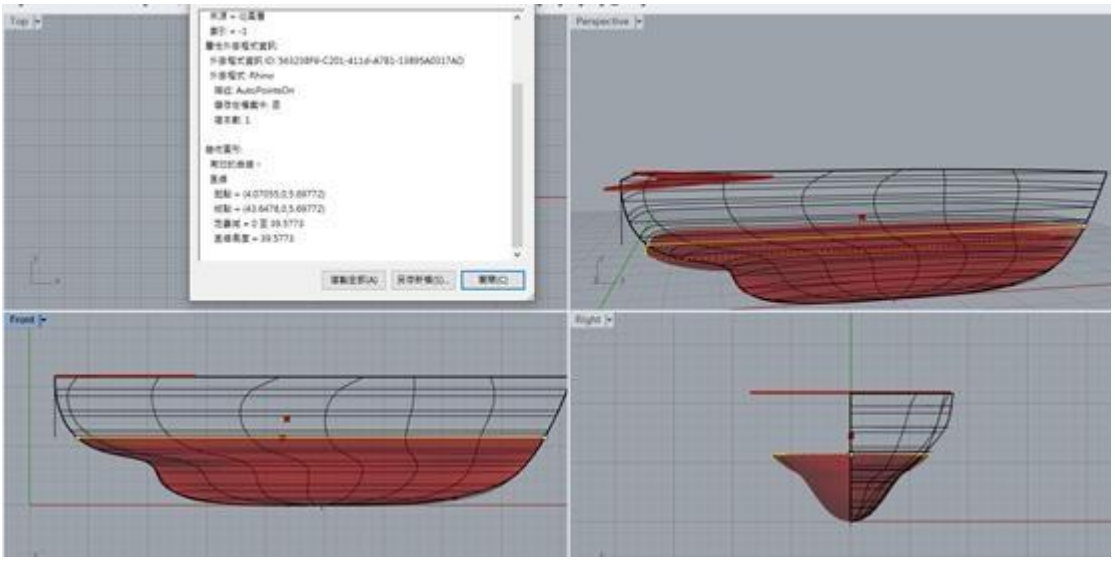
吃水 1020g 時 GM 約為 2.13cm



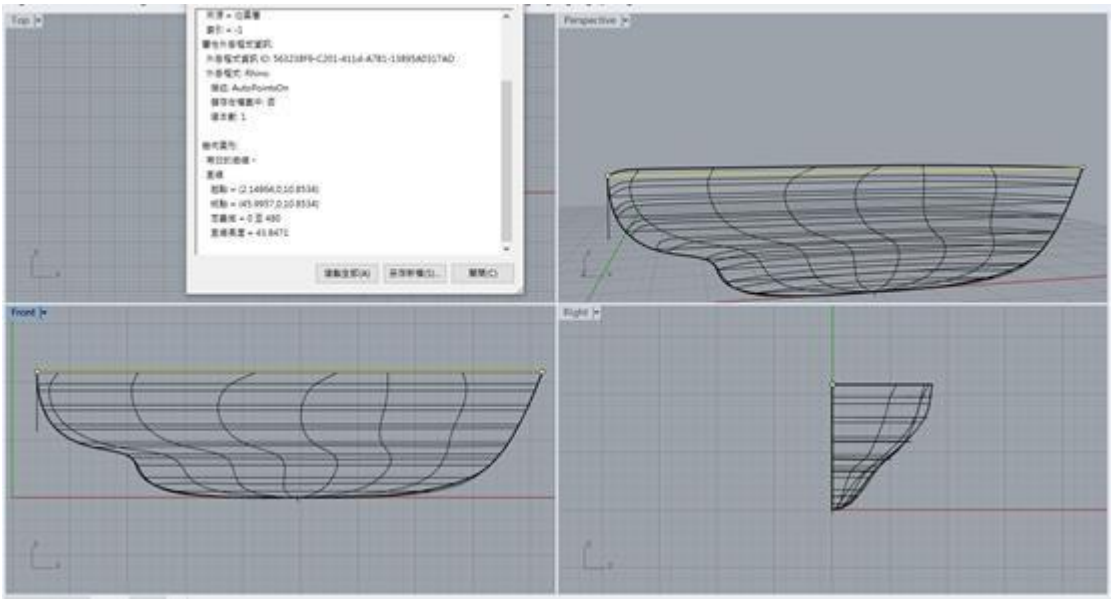
吃水 1020g 時，水線長度為 39.58cm



吃水 1020g 時，水線寬度為 12.33cm



船身最長處為 43.85cm



船體設計為下窄上寬，原因是為了保留螺槳空間以及增加 right arm 大小。

電路設計:
我們使用的 STM32F303RE 開發版，主要負責以下兩種功能: 超聲波感應與馬達控制。

我們總共使用了五個超聲波感應器，為了避免超聲波錯誤的接收到其他發出的 trigger 信號而干擾，船頭設置一個(與船隻方向相同)，船身右前、左前、右後、左後各設置一個(與船隻方向成 90 度)，彼此位置盡量拉開。

設計中，我們總共使用了七個 Timer，其中的五個 Timer: Timer2、Timer3、Timer15、Timer16、Timer4，分別負責一個超聲波感測器的 echo 信號，Timer1 負責 interrupt 與 trigger 信號的發送；Timer8 下的兩個 channel 負責兩個螺槳的旋轉。
以下表格顯示五個超聲波與對應 Timer，和 5V, Ground, trigger, echo 與 STM32 開發版的連接 pin 腳。

	綁定 Timer	5V/Gnd	Trigger	Echo
前方超聲波	Timer 2	5V/GND 集中供電	PA1	PA0
左方超聲波	Timer 3			PA6
右方超聲波	Timer 15			PB14
左後超聲波	Timer 16			PA12
右後超聲波	Timer 4			PA11

以上五個Timer 的的channel1 的channel1 設成 input capture direct mode，prescaler 皆設成 71，counter period 依據該 Timer 對應的 register 為 16bit 或 32bit 設成 0xffff、0xffffffff，另外需要在 NVIC 下開啟 global interrupt。

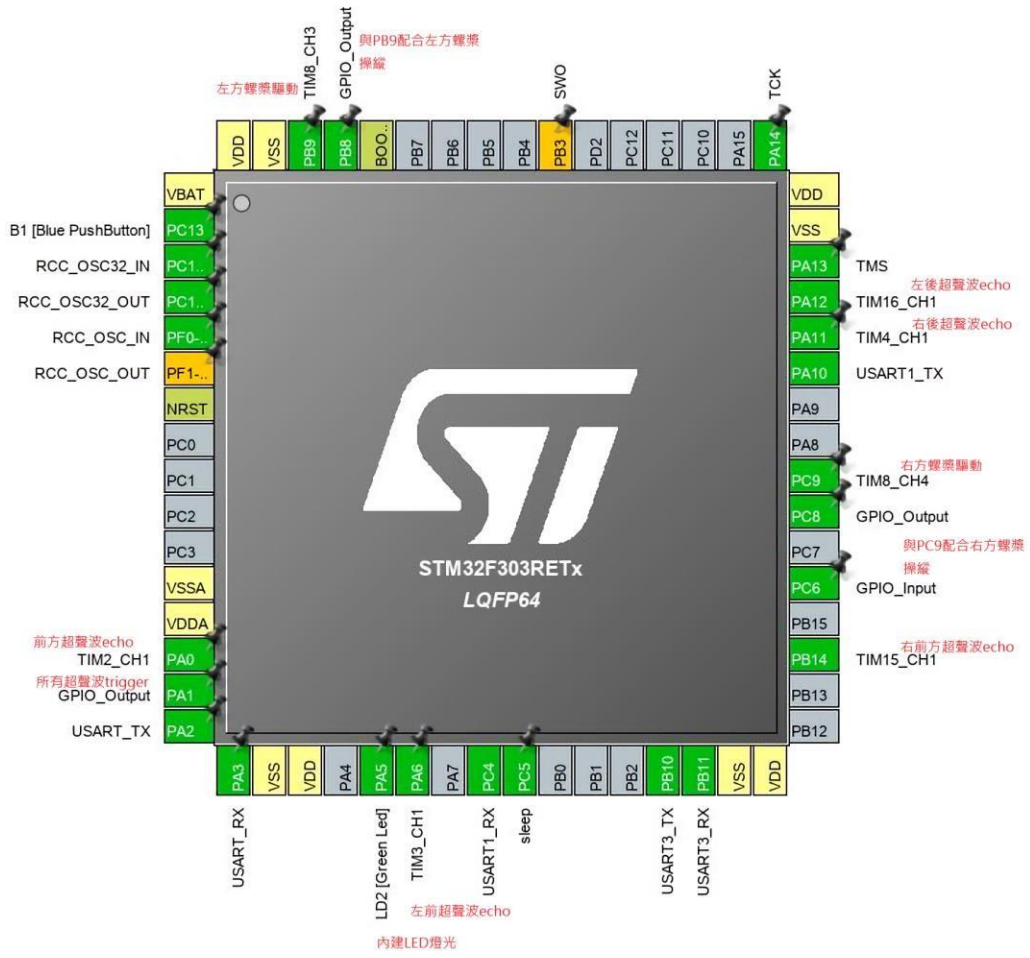


Fig. 1 Schematic mapping

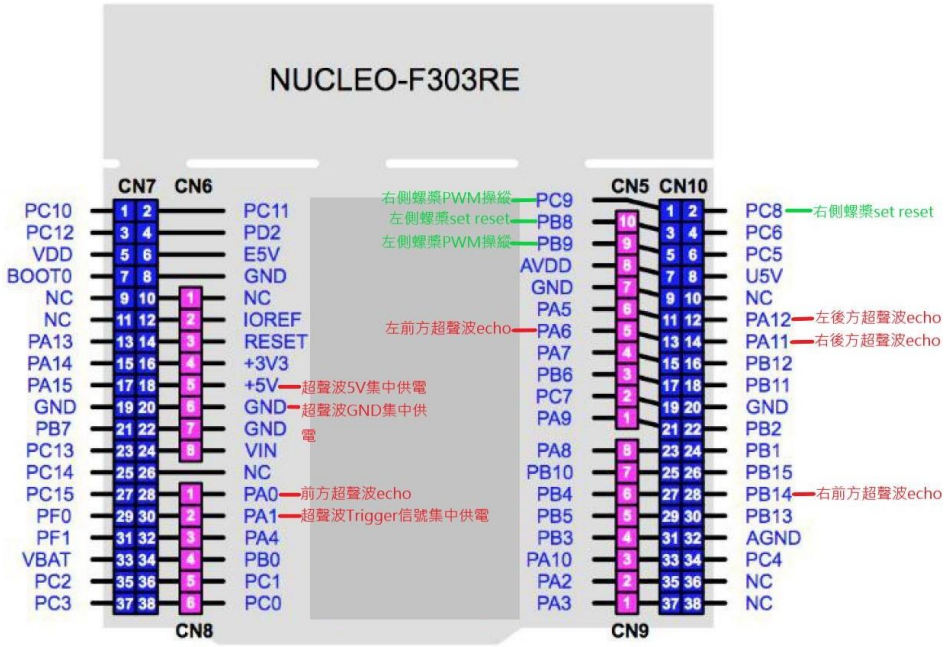


Fig. 2 STM32F303RE pinout map

針對學期初因應不同環境狀態寫成不同的 case，就我認為過於複雜，在測試除錯上也會帶給我極大的困擾。這次的比賽賽道經過多次的調整，最終版的的容錯餘裕度比較大，再加上船體本身設計穩定，因此我判斷簡單的演算法再加上多次的事先測試就可以完成比賽需求。程式的演算法以前方距離為主要判斷基礎，在不同距離下進行大小程度的左右轉彎、倒退等等。

```
#include "main.h"
#define V1 70
#define V2 75
#define V3 80
#define V4 85
#define V5 90
```

Fig.3.1

圖 3.1 顯示C 語言的巨集，定義會使用到五個段位的馬達 PWM。經過測試，馬達會在 PWM=70 實開始運作，然而 PWM 90 以上的加減速過大，所以不採用。

```
TIM_HandleTypeDef htim1;
TIM_HandleTypeDef htim2;
TIM_HandleTypeDef htim3;
TIM_HandleTypeDef htim4;
TIM_HandleTypeDef htim8;
TIM_HandleTypeDef htim15;
TIM_HandleTypeDef htim16;

UART_HandleTypeDef huart1;
UART_HandleTypeDef huart2;
UART_HandleTypeDef huart3;

/* USER CODE BEGIN PV */
int Front;
int Left;
int Right;
int FrontRight;
int FrontLeft;
int RearRight;
int RearLeft;
int RightSlope;
int LeftSlope;
//int Rear;

char tosend[20] = { 0 };
char test[30] = { 0 };
int turnleft;
int turnright;
int tmpcenter2;
int tmpcenter;
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_TIM1_Init(void);
static void MX_TIM2_Init(void);
static void MX_USART2_UART_Init(void);
static void MX_TIM8_Init(void);
static void MX_TIM3_Init(void);
static void MX_TIM4_Init(void);
static void MX_TIM15_Init(void);
static void MX_USART1_UART_Init(void);
static void MX_USART3_UART_Init(void);
static void MX_TIM16_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */
```

Fig.3.1

```c
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
void PrintDistance(double raw_distance) {
    int distance_interger = (int) raw_distance;
    int distance_float = (int) ((raw_distance - distance_interger) * 100);
    sprintf(tosend, "%d.%02d\r\n", distance_interger, distance_float);
    HAL_UART_Transmit(&huart2, tosend, sizeof(tosend), 0xffff);
}

void DebugLED(int condition) {
    if (condition == 0) {
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
    } else if (condition == 1) {
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
    } else {
        return;
    }
}

void MotorSet(int LeftThrust, int RightThrust) {
    if (abs(LeftThrust) > 100 || abs(RightThrust) > 100) {
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_8, GPIO_PIN_RESET);
        TIM8->CCR3 = 0;
        TIM8->CCR4 = 0;
        return;
    }
    if (LeftThrust > 0) {
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8, GPIO_PIN_RESET);
        TIM8->CCR3 = LeftThrust;
    }
    if (LeftThrust <= 0){
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8, GPIO_PIN_SET);
        TIM8->CCR3 = 100 - (abs(LeftThrust));
    }
    if (RightThrust > 0) {
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_8, GPIO_PIN_RESET);
        TIM8->CCR4 = RightThrust;
    }
    if (RightThrust <= 0){
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_8, GPIO_PIN_SET);
        TIM8->CCR4 = 100 - (abs(RightThrust));
    }
}
void MaxPowerTest(){
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_8, GPIO_PIN_RESET);
    TIM8->CCR3 = 100;
    TIM8->CCR4 = 100;
}
```

Fig.3.2 API

去年小組的自走車也是我負責，在工海專論中為了讓同學也能協助測試與調參，特別將一些複雜的程式碼封裝起來，以減少同學在調整時的失誤。另外有善用開發板上面的 LED 燈，將燈光亮案寫在某個程式區塊，以方便在測試時檢驗是否有執行我們所預期的指令。另外在在連接電腦時，可以利用 serial port terminal 印出指定超聲波的距離。螺槳操縱上，有提供反轉的功能，實踐上就是將與 timer6 channel 3,4 配合的 GPIO 設成高電位或低電位，並換算輸入的 PWM。

我在設計程式與實際測試中發現到，我必須確實的了解硬體狀態，例如接線、電池、馬達、超聲波，並確保他們在每一次的正常運作；如此在船隻實際測試發生不盡人意的表現時，才有可以信賴的判斷依據讓我找到出錯的關鍵點。

```
void PrintDistance(double raw_distance)
```

用來協助印出距離

適用情況:

(1) 作為超聲波與系統的檢測

(2) 協助調參時判斷

已經寫在 **HAL_TIM_IC_CaptureCallback** 中，每一個 htim→Instance 下皆有一組已經註解起來的 PrintDistance(distance);

將註解消除，經過 build → debug as → 按下上方的開始鍵

(確保 serial port terminal 有連接，每次重接上 usb 後皆要重新連線)

就可以在 serial port terminal 看到印出距離。

- 注意: 一次測試只能印出一個方向的距離(一次只能反註解一個)
  因為共用所有的數字皆共用一組 char array 和 usart，會互相干擾。

```
void DebugLED(int condition)
```

控制 LED 燈，condition 為 1 是亮，0 為暗，其他數字不會有反應。

適用情況:

(1) 作為狀態標示，在試跑時可知道正在執行程式碼哪一段

(2) 協助除錯

- 注意: 記得要關燈 DebugLED(0);

```
void MotorSet(int LeftThrust, int RightThrust)
```

簡化程式碼，在控制馬達時只需輸入左右馬力百分比。

兩個參數的輸入範圍皆為 100~－100，輸入範圍外的值默認馬力為 0

適用情況:

(1) 馬達控制

- 範例:

```
MotorSet(100,100)     // 左右輪全速前進
MotorSet(0,0)         // 左右輪停止
MotorSet(-100,-100)   // 左右輪全速後退
MotorSet(-50,50)      // 左輪半速倒退，右輪半速前進(原地左轉)
```

硬體檢測:

上電前

(一) 外部接線是否有脫落，對照上方腳位圖

驅動版是否接對(接在背面，四個 motor 接點朝下(stm32 LD1 為上))

(二) 電池需 1.4V 以上，才能保證效能

上電後

(三) 不定時檢查是否有發(電線、感測器)、燒焦味

(四) 執行超聲波感測器檢測，依次註解消除 PrintDistance(distance)
是否走正確印出距離，每個超聲波感測器執行一次

編譯與燒錄

(五) 在 USB 供電、電池供電切換前
下電 → E5V 、 U5V 切換 → 上電

(六) 按下 debug as 後，是否進入 debug 頁面，LD1 是否有綠紅綠紅切換

以上是測試前會進行的確認動作。

```
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef* htim) {
    if (htim->Instance == TIM2) {
        if (HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0) == 1) {
            __HAL_TIM_SET_COUNTER(&htim2, 0);
        } else {
            int Frontcnt = __HAL_TIM_GET_COUNTER(&htim2);
            double Frontdistance = Frontcnt / (double) 58;
            Front = (int) Frontdistance;
//          PrintDistance(Frontdistance);

        }
    }
    if (htim->Instance == TIM3) {
        if (HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_6) == 1) {
            __HAL_TIM_SET_COUNTER(&htim3, 0);
        } else {
            int Leftcnt = __HAL_TIM_GET_COUNTER(&htim3);
            double Leftdistance = Leftcnt / (double) 58;
            FrontLeft = (int) Leftdistance;
//          PrintDistance(FrontLeft);
            double test = (FrontRight-FrontLeft);
//          PrintDistance(test);
        }
    }
    if (htim->Instance == TIM15) {
        if (HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_14) == 1) {
            __HAL_TIM_SET_COUNTER(&htim15, 0);
        } else {
            int Rightcnt = __HAL_TIM_GET_COUNTER(&htim15);
            double Rightdistance = Rightcnt / (double) 58;
            FrontRight = (int) Rightdistance;
            PrintDistance(Rightdistance);
        }
    }
    if (htim->Instance == TIM4) {
        if (HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_11) == 1) {
            __HAL_TIM_SET_COUNTER(&htim4, 0);
        } else {
            int RearRightcnt = __HAL_TIM_GET_COUNTER(&htim4);
            double RearRightdistance = RearRightcnt / (double) 58;
            RearRight = (int) RearRightdistance;
            RightSlope = (RearRight-FrontRight);
//          PrintDistance(RearRightdistance);
//          PrintDistance(RightSlope);

        }
    }
    if (htim->Instance == TIM16) {
        if (HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_12) == 1) {
            __HAL_TIM_SET_COUNTER(&htim16, 0);
        } else {
            int RearLeftcnt = __HAL_TIM_GET_COUNTER(&htim16);
            double RearLeftdistance = RearLeftcnt / (double) 58;
            RearLeft = (int) RearLeftdistance;
            LeftSlope = (RearLeft-FrontLeft);
//          PrintDistance(RearLeftdistance);
        }
    }
```

Fig.3.3 超聲波感測公式

五個超聲波共用 Timer1 的 interrupt 和 trigger 信號，並使用各自的 Timer 處理 echo 信號。在收到回彈信號後，將 echo 拉到高戀位，觸發之後會提到的 HAL_TIM_IC_CatputeCallback ，並使用_HAL_TIM_SET_COUNTER(&htim,x)

將 counter register 設成 0；由於我們對於每一個超聲波的 input capture 訂為 both edge，隨後 echo 回到低電位時用_HAL_TIM_SET_COUNTER(&htimx)取出 counter register 的值(等同於時間)，再用上換算公式得到距離。

```
if(Front>80){
    if(FrontRight<7) MotorSet(-V1,V1);
    else if(FrontLeft<7) MotorSet(V1,-V1);
    else if(RightSlope>9){
        DebugLED(0);
        MotorSet(V3,V4);
    }else if (LeftSlope >9){
        DebugLED(1);
        MotorSet(V4,V3);
    }else{
        DebugLED(0);
        MotorSet(V4,V4+2);
    }
}
else if(Front>75&&Front<80){
    if(FrontRight<7) MotorSet(-V1,V1);
    else if(FrontLeft<7) MotorSet(V1,-V1);
    DebugLED(0);
    MotorSet(0,0);
}
else if(Front>65&&Front<75){
    if(FrontRight<7) MotorSet(-V1,V1);
    else if(FrontLeft<7) MotorSet(V1,-V1);
    else if(RightSlope>7){
        DebugLED(0);
        MotorSet(V3,V4);
    }else if (LeftSlope >7){
        DebugLED(1);
        MotorSet(V4,V3);
    }else{
        DebugLED(0);
        MotorSet(V4,V4+1);
    }
}
else if(Front>60&&Front<65){
    DebugLED(0);
    if(FrontRight<7) MotorSet(-V1,V1);
    else if(FrontLeft<7) MotorSet(V1,-V1);
    else MotorSet(0,0);
}
else if(Front>50&&Front<60){
//  DebugLED(1);
    if(FrontRight<7) MotorSet(-V1,V1);
    else if(FrontLeft<7) MotorSet(V1,-V1);
    else if((FrontRight-FrontLeft)>15){
        DebugLED(0);
        MotorSet(V3,V2);
    }else if((FrontLeft-FrontRight)>15){
        DebugLED(0);
        MotorSet(V2,V3);
    }else MotorSet(V3,V3);
}
```

fig 3.4.1 距離判斷演算法

```
else if(Front>40&&Front<50){
    DebugLED(0);
    if(FrontRight<7) MotorSet(-V1,V1);
    else if(FrontLeft<7) MotorSet(V1,-V1);
    else if((FrontRight-FrontLeft)>10){
        MotorSet(V3,V2);
    }else if((FrontLeft-FrontRight)>10){
        MotorSet(V2,V3);
    }
}
else if(Front>20&&Front<40){
    if(FrontRight<7) MotorSet(-V1,V1);
    else if(FrontLeft<7) MotorSet(V1,-V1);
    else if((FrontRight-FrontLeft)>10){
        DebugLED(0);
        MotorSet(V2,-V2);
        DebugLED(0);
    }
    else if (FrontLeft-FrontRight>10){
        MotorSet(-V2,V2);
        DebugLED(0);
    }
}
else if(Front<20){
    DebugLED(0);
    MotorSet(-V4,-V4);
}

//************************Tune the above section ************************************
//*************************************************************************************

    HAL_TIM_Base_Start_IT(&htim1);
}
```

fig 3.4.2 距離判斷演算法

整個船隻的操縱主要分成幾個階段:

(1) 前方長距離

此時兩側螺槳以 PWM=85 的速度正走，並在兩種狀況下調整左右航向

(1) 左前左後之間、右前右後之間超生波距離相差太大
此演算法是避免在長距離時就發生 45 度擦撞的現象

(2) 左前 右前超聲波感測極短距離，此時極有可能擦撞牆壁
為了避免船頭接觸牆壁的摩擦力導致船首航向更偏離賽道
因此讓兩側螺槳轉向相反，盡量原地調整航向，避免繼續前進

(2) 前方中距離

我在其中的兩段五公分距離區間讓螺槳停止，以免船速在近距離時過快
因為進入直線航道，我停止使用左前左後之間、右前右後之間超生波距離的判斷法，而是單純採取左前 右前超聲波感測距離的修正法。

(3) 前方中短距離

與中距離相似，但船速有所下降，並且旋轉幅度上升，使用原地旋轉

(4) 前方短距離

此時螺槳快速反轉，以期望減速避免正面碰撞

```
int tim1Count = 7000;
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef* htim) {
    if (htim→Instance == TIM1) {
        if (tim1Count < 7000) {
            tim1Count++;
        } else {
            HAL_TIM_Base_Stop_IT(&htim1);
            if (HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_1) == 0) {
                HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, 1);
                HAL_TIM_Base_Start_IT(&htim1);
            } else {

                tim1Count = 0;
                HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, 0);

            }
        }
    }
}
```

fig 3.4.2 Timer1 的 interrupt, callback 和 trigger

使用 counter ，當 counter<7000 ， 每一個 counter period 會出發一次 HAL_TIM_PeriodElapsedCallback。當 counter=7000，將 trigger 設定成高電位促使所有超聲波發出感測距離的信號，並在下一個 counter period 時關閉 trigger 信號，同時 counter 設為 0 重新計數。

```c
int main(void) {
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_TIM1_Init();
  MX_TIM2_Init();
  MX_USART2_UART_Init();
  MX_TIM8_Init();
  MX_TIM3_Init();
  MX_TIM4_Init();
  MX_TIM15_Init();
  MX_USART1_UART_Init();
  MX_USART3_UART_Init();
  MX_TIM16_Init();
  /* USER CODE BEGIN 2 */
  HAL_TIM_PWM_Start(&htim8, TIM_CHANNEL_4);
  TIM8->CCR4 = 100; //right
  HAL_TIM_PWM_Start(&htim8, TIM_CHANNEL_3);
  TIM8->CCR3 = 100; //left
  HAL_TIM_IC_Start_IT(&htim2, TIM_CHANNEL_1);
  HAL_TIM_IC_Start_IT(&htim3, TIM_CHANNEL_1);
  HAL_TIM_IC_Start_IT(&htim4, TIM_CHANNEL_1);
  HAL_TIM_IC_Start_IT(&htim15, TIM_CHANNEL_1);
  HAL_TIM_IC_Start_IT(&htim16, TIM_CHANNEL_1);
  HAL_TIM_Base_Start_IT(&htim1);
```

啟動 uart, Timer 與相對應 interrupt, GPIO 的程式碼。