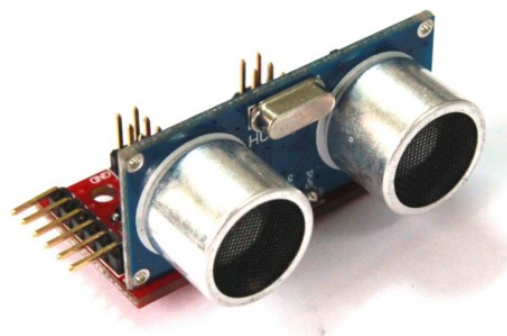


Sonar B 模組

超音波測距

版本：V2.0



產品介紹：Sonar B 模組可透過簡單的聯接，直接由利基之 Ozone 操控各項應用，藉由簡單的指令下達設定環境的條件，配合所須的偵測次數，測量距離。

應用方向：

- 可做為偵測距離之工具。
- 配合移動機構(機器人、自走車…等)，可達到避障的功能，保護機構。

產品特色：

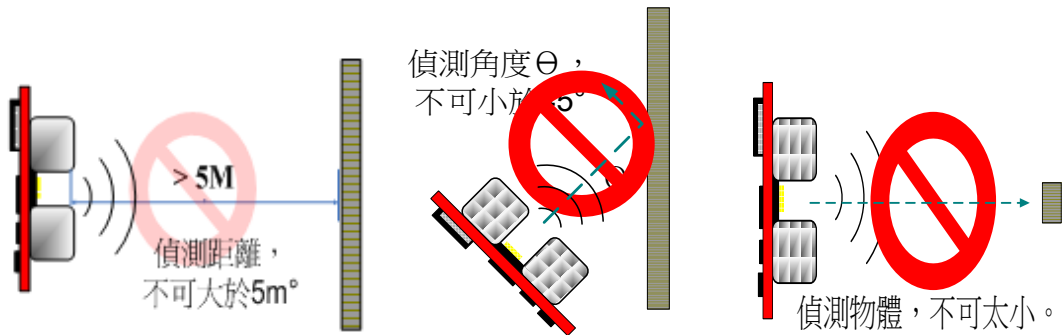
- 使用簡單，並設有單次(Ranging)、多次(RepeatRanging)等指令。
- 設有Event事件，與RepeatRanging搭配使用，可即時取得偵測結果。
- 可設定簡測結果回傳格式(us、cm、inch)。
- 額外的擴充功能，可切換為**一般模式**(四組 A/D 輸入)、**Master模式**(四組同步輸出訊號)或 **Slave 模式**(三組A/D輸入加一組同步輸入訊號)。
- 可透過I2C方式，下達指令。

產品規格：

- Supply voltage：6~12 VDC
- Size：23.4mm H × 48.3mm W × 18mm D
- Weight：11g(0.39oz)
- Detection scope：2cm ~ 5m
- 接腳定義：

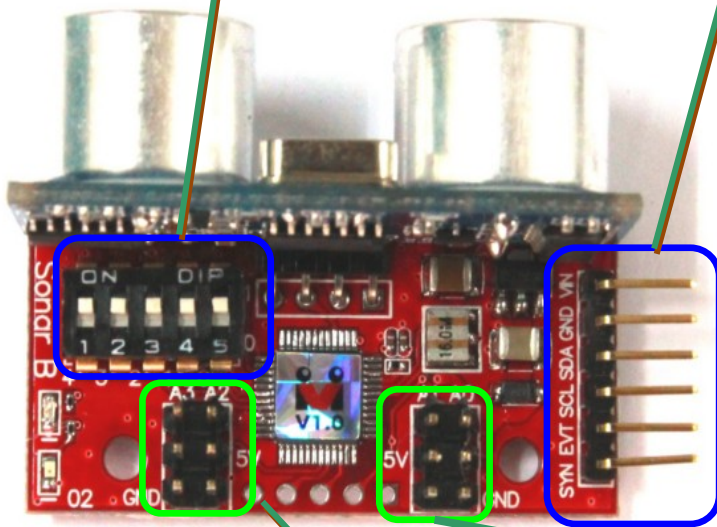
| 接腳名稱 | VIN | GND | SDA | SCL | EVT | SYN |
|------|------|-----|------|------|------|------|
| 接腳定義 | 外部電源 | 接地端 | 資料訊號 | 時脈訊號 | 事件訊號 | 同步訊號 |

- 偵測限制：



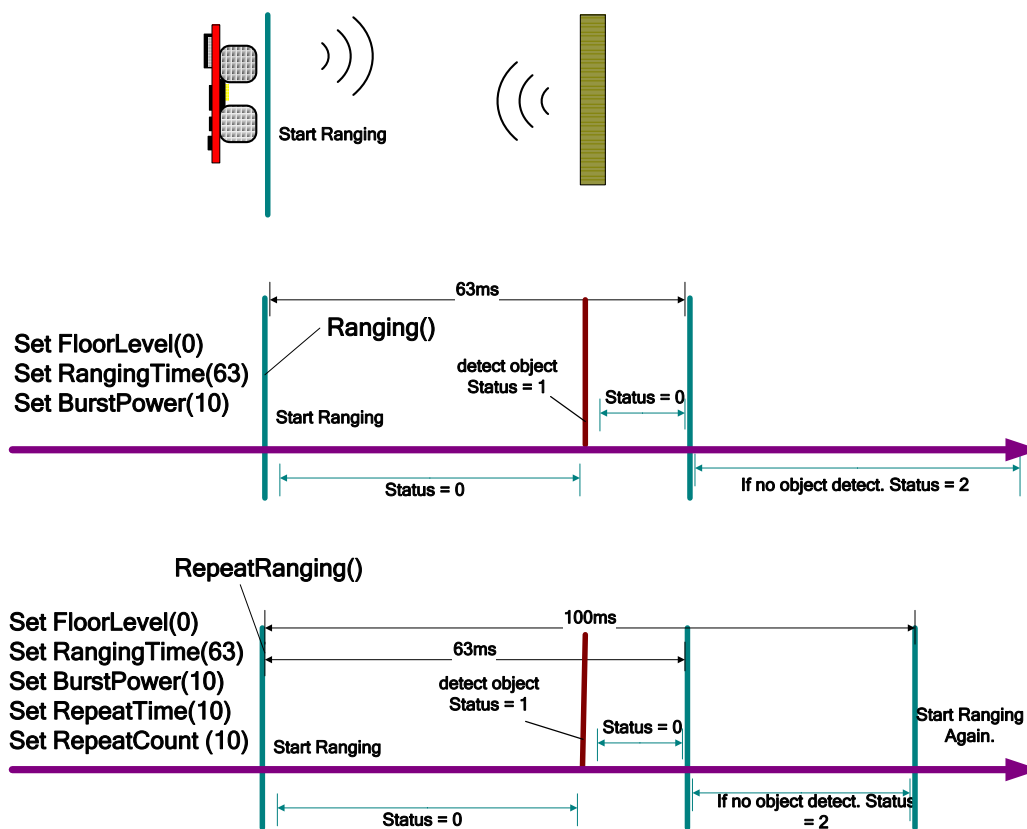
模組編號設定開關，由右至左以二進制設定Sonar B的模組編號，編號可以讓Ozone[®]操控時，判斷想要控制的模組。

cmdBUS[™]接腳，將此處腳位與Ozone[®]對應腳位相接，即可透過Ozone[®]操控Sonar B(連接時請注意腳位對應，將Vin對接Ozone[®]上的Vin腳位，若是腳位錯誤可能造成模組損毀)



擴充功能接腳，可設為

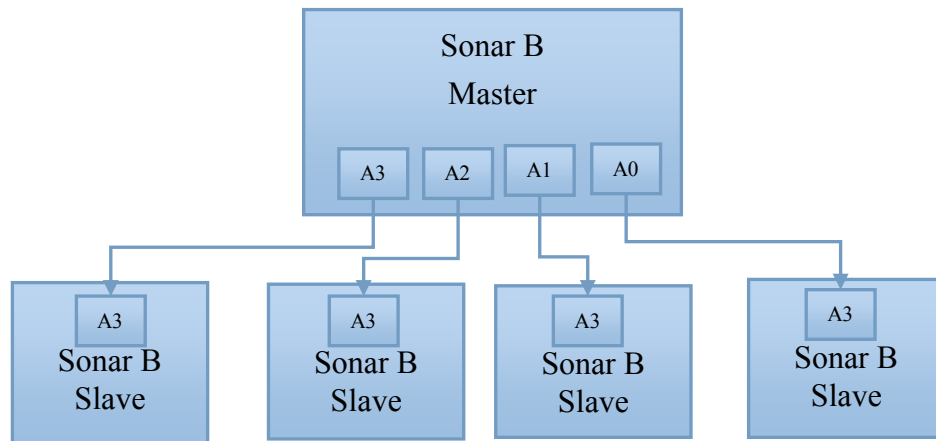
1. 一般模式 - 四組 A/D 輸入
2. Master 模式 - 四組同步輸出訊號
3. Slave 模式 - 三組 A/D 輸入加一組同步輸入訊號



【動作原理示意圖】

連接方式: 直接將ID開關撥至欲設定的編號，再將cmdBUS連接至Ozone上對應的腳位，就可透過Ozone執行操作





【Master Slave 連接示意圖】

1. Master 和 Slave 模式可由程式設定。
2. 一個 Master SonarB 可接一到四個 Slave SonarB，連接時請注意各模組間的共地問題。若各模組使用 cmdBUS 連接，則不需再另外連接 GND。
3. Slave 端 A3 固定為脈波輸入。其他沒使用到的接腳(A0, A1 和 A2)仍可做為 A/D 使用。

操作注意事項：

- ✓ 操作溫度 0°C~70°C
- ✓ 儲存溫度 -30°C~80°C
- ✓ 偵測物體的表面平滑度與吸音材質，將影響測量準確率。

| 指令格式 | 指令功能 |
|---|---|
| 超音波測距設定相關指令 | |
| SetRangingTime (uint8_t RangingTime) | 設定超音波發射後等待接收時間，可定義範圍 0~63。(單位 ms，預設值 40ms，設為 0 時定義為 1ms。) |
| GetRangingTime (uint8_t RangingTime) | 取得目前超音波發射後等待接收的時間設定，存入 RangingTime 中。回傳值範圍為 0~63。 |
| Ranging (void) | 執行超音波發射與接收。 |
| RepeatRanging (void) | 重複(週期性)執行超音波偵測。 |
| SetRepeatTime (uint8_t RepeatTime) | 設定重複偵測間隔時間，範圍 0~255。(單位 10ms，設為 0 時定義為 10ms。) |
| GetRepeatTime (uint8_t& RepeatTime) | 取得重複偵測的間隔時間設定，存入 RepeatTime 中。回傳值範圍為 0~255。 |
| SetRepeatCount (uint8_t RepeatCount) | 設定重複偵測次數。可定義範圍 1~255，0=無限次數。 |
| GetRepeatCount (uint8_t& RepeatCount) | 取得重複偵測的次數設定，存入 RepeatCount 中。回傳值範圍為 0~255。 |
| StopRanging (void) | 停止發射超音波。 |
| uint8_t Status = GetDistance (uint8_t Type, uint16_t& Distance) | 取得接收狀態存放於 Status ，並以 Type 設定取得資料型態，存入 Distance 中。 (Status : 0=not ready, 1=ready, 2=time out ; Type : 0 = us , 1=cm , 2=inch。) |
| GetADValue (uint8_t Ch, uint16_t & Value) | 讀取 A0~A3 的 A/D 值。 Ch = 0, A0 = 1, A1 = 2, A2 = 3, A3 Value 讀回的 A/D 值，範圍 0~4095。 請注意，當使用 SetMode 的 Slave 模式時 (=1)， Ch=3 讀回的值為無效。 使用 Master 模式時(=2)，此命令讀回的值皆無效。 |
| SetMode (uint8_t Mode) | Mode = 0 ，一般模式 A0~A3 為 A/D 輸入。 |

| | |
|--|--|
| | <p>= 1, Slave 模式 A3 做為脈波輸入, A0~A2 仍為 A/D 輸入。</p> <p>= 2, Master 模式 A0 ~A3 都做為脈波輸出。</p> <p>預設為 0</p> |
|--|--|

註 1：離地高度會受地面的材質影響。

註 2：當事件被啟動並進入時，必需將 Status 狀態值讀出，否則事件將重複被觸發。

範例程式：

範例 A：單次偵測

```
#include <ozone.h>
SonarB mySonar (0);          // 設定模組編號為 0

uint8_t Status;              // 儲取得得的偵測結果狀態
uint16_t Distance;          // 儲取得得的偵測結果
void setup(void)
{
}
void loop(void)
{
  do {
    mySonar.Ranging();        //執行偵測
    delay (100);              //等待 100ms
    Status = mySonar.GetDistance(1,Distance); //取得偵測結果
  }while(Status != 1);        //迴圈停止條件(Status = 1)
  Serial.print(Distance);     //顯示偵測結果
  Serial.println("cm");
}
```

範例 B：重複偵測

```
#include <ozone.h>
SonarB mySonar (0);          // 設定模組編號為 0
uint8_t Status;              // 儲取得得的偵測結果狀態
uint16_t Distance;          // 儲取得得的偵測結果
void setup(void)
{
  mySonar.RepeatRanging() ;
}
void loop(void)
{
  do {
    delay (5);                //等待 5 ms
    Status = mySonar.GetDistance(1,Distance); //取得偵測結果
  }while(Status != 1);        //迴圈停止條件(Status = 1)
  Serial.print(Distance);     //顯示偵測結果
  Serial.println("cm");
}
```

範例 C：A/D 範例

```
#include <ozone.h>
SonarB mySonar (0);          // 設定模組編號為 0

uint16_t Value;              // 儲存取得的結果
void setup(void)
{
}
void loop(void)
{
    mySonar.GetADValue(0, Value);
    Serial.print("A0= ");    //顯示結果
    Serial.println(Value);
    mySonar.GetADValue(1, Value);
    Serial.print("A1= ");    //顯示結果
    Serial.println(Value);
    mySonar.GetADValue(2, Value);
    Serial.print("A2= ");    //顯示結果
    Serial.println(Value);
    mySonar.GetADValue(3, Value);
    Serial.print("A3= ");    //顯示結果
    Serial.println(Value);
    delay(200);
}
```

範例 D：Sonar Master/Slave 範例

Master

```
#include <ozone.h>
SonarB mySonar (0);          // 設定模組編號為 0

uint8_t Status;              // 儲存取得的偵測結果狀態
uint16_t Distance;           // 儲存取得的偵測結果
void setup(void)
{
    mySonar.SetMode(2);
    mySonar.RepeatRanging() ;
}
void loop(void)
{
```



```

do {
    delay (5);                //等待 5 ms
    Status = mySonar.GetDistance(1,Distance); //取得偵測結果
}while(Status != 1);        //迴圈停止條件(Status = 1)
Serial.print("Distance="); //顯示偵測結果
Serial.println(Distance);
}

```

Slave

```

#include <ozone.h>
SonarB mySonar (1);        // 設定模組編號為 1

uint8_t Status;           // 儲存取得的偵測結果狀態
uint16_t Distance;       // 儲存取得的偵測結果

































void setup(void)
{
    mySonar. SetMode(1);
}

void loop(void)
{
    do {
        delay (5);                //等待 5 ms
        Status = mySonar.GetDistance(1,Distance); //取得偵測結果
    }while(Status != 1);        //迴圈停止條件(Status = 1)
    Serial.print("Distance="); //顯示偵測結果
    Serial.println(Distance);
}

```

附錄：

1 · 模組編號開關對應編號表：

| | | | | | | | |
|--|---|--|----|--|----|--|----|
|  | 0 |  | 8 |  | 16 |  | 24 |
|  | 1 |  | 9 |  | 17 |  | 25 |
|  | 2 |  | 10 |  | 18 |  | 26 |
|  | 3 |  | 11 |  | 19 |  | 27 |
|  | 4 |  | 12 |  | 20 |  | 28 |
|  | 5 |  | 13 |  | 21 |  | 29 |
|  | 6 |  | 14 |  | 22 |  | 30 |
|  | 7 |  | 15 |  | 23 |  | 31 |