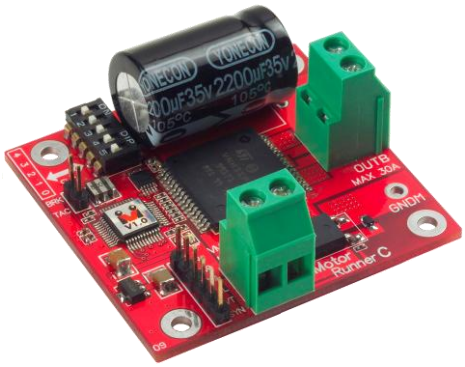
Motor Runner C

單直流馬達控制模組

版本: V2.0



產品介紹: 利基 Motor Runner C 模組可以達到透過簡易的指令設定，自由操控單顆直流馬達的需求。可以隨時動態的更改馬達轉速，並取得馬達現在的設定狀態包含轉速或是方向。提供 PID 參數可以控制馬達轉速。與 Motor Runner A，Motor Runner B 相比，能承受更高的電壓與電流值。

應用方向:

- 控制馬達趨動，設定模型車的前進與後退。
- 需要轉速回傳的設備，動態調整轉速。
- 可以直接加上小風扇，並操作風量強度。

產品特色:

- 以簡單指令控制單個馬達的轉向與轉速。
- 可承受最大±30A 的連續輸出電流。
- 輸入電壓最高可承受至 35V。
- 內部固定頻率 10KHz PWM 電流控制。
- 提供過熱自動斷電保護(150℃)。
- 提供過載電流保護。
- 提供 Crossover-Current Protection 與低壓閉鎖保護(UVLO)
- 透過 Brake 指令能快速停止馬達的動作。
- 可以設定 256 階不同轉速。
- 透過指令能隨時取得現在馬達轉速或轉向等各種設定。
- 提供外接停止訊號，接上簡單的外部按鈕，就能由按鈕停止馬達轉動。
- 馬達轉速連接腳位，可以與有提供轉速計的馬達輸出相連，就可以透過指令，及時取得更精準的馬達轉速。
- 轉速計可以設定 13 種不同偵測頻率。
- 在連接轉速計的狀態下，透過轉速控制指令，能直接設定轉速，由模組控制馬達加速(減速)到所要的轉速，並且維持在設定的轉速。
- 在連接轉速計的狀態下，可以使用 PID 相關參數，調整轉速變化的的效率。
- 在連接轉速計的狀態下，透過計數控制指令，能直接設定計數，由模組控制馬達在計數到設定值時，依設定方式停止。
- 可透過 cmdBUS 方式，下達指令。

- **連接方式:** 直接將 ID 開關撥至欲設定的編號，再將 cmdBUS 連接至 Arminno 上對應的腳位，就可透過 Arminno 執行操作。欲操作的直流馬達請依據腳位連接至對應的 OUTA 與 OUTB 馬達輸入接腳，並將模組的馬達電源 VM 與 GND 腳位，連接至能提供符合馬達需求的電源。指令操作時，如果馬達轉向與指令相反，代表 OUTA 與 OUTB 連接反向，此時可以將 OUTA 與 OUTB 對調，或是將程式中的前進與後退指令顛倒。

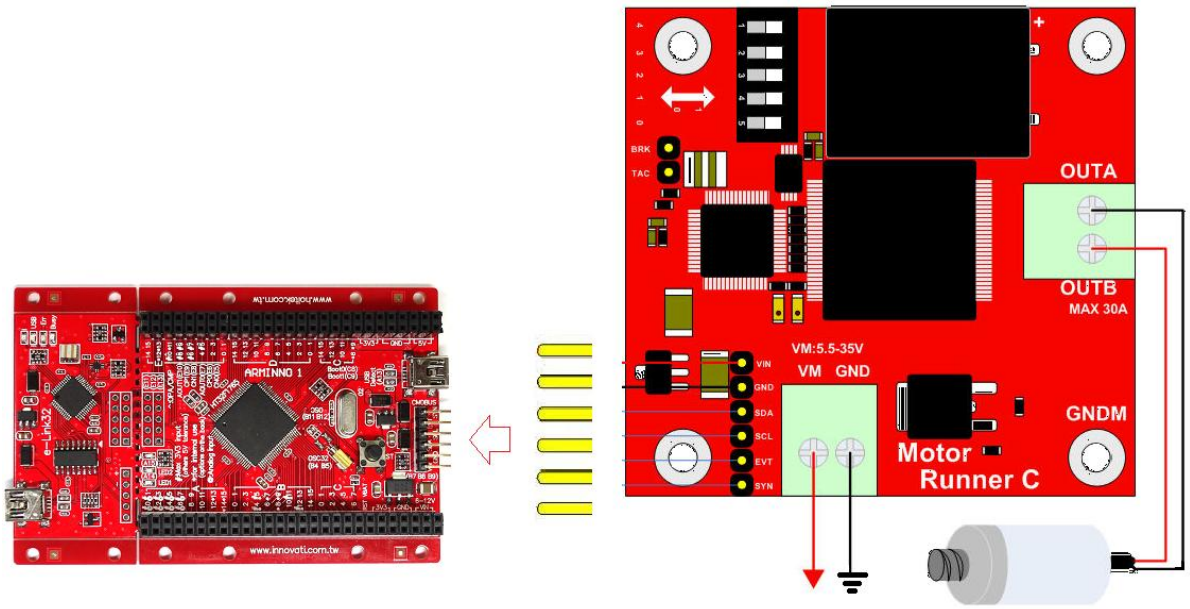
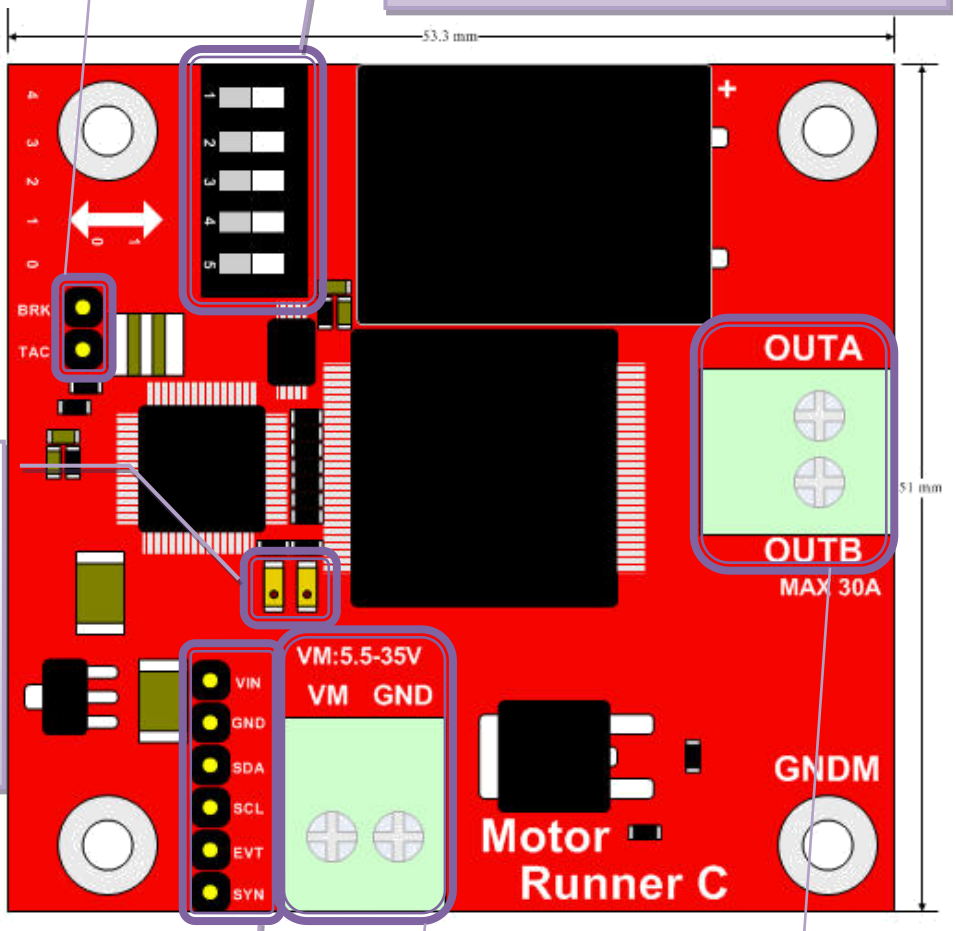


圖 1: 馬達模組連接範例

產品規格:

BRK與按鈕連接，可以用按鈕停止馬達轉動，按鈕另一端請接地
TAC是轉速計接腳，連接轉速計後，能使用轉速計控制相關指令

模組編號設定開關，由右至左以二進制設定馬達模組的模組編號，編號可以讓Arminno操控時，判斷想要控制的模組。



靠右的橙色燈號是資料傳輸指示燈，閃爍代表模組正在傳收資料，靠左的綠色燈號是事件指示燈，未使用

cmdBUS接腳，將此處腳位與Arminno對應腳位相接，就能透過Arminno操控馬達模組(連接時請注意腳位對應，將Vin對接Arminno上的Vin腳位，腳位錯誤會造成模組的毀損)

馬達電源接腳，請將要提供馬達的電源連接到此對應的腳位

馬達接腳，連接馬達提供馬達的轉向與轉速判定，請注意OUTA與OUTB若顛倒，會造成馬達轉向與指令相反

圖 2: 模組腳位與開關介紹

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{DD}	Operating Current	7.5	No I/O	—	5.4	—	mA
f_{pwm}	PWM Output frequency	—	—	0	—	10	kHz

表 1: 工作電流特性 (於 25℃之環境)

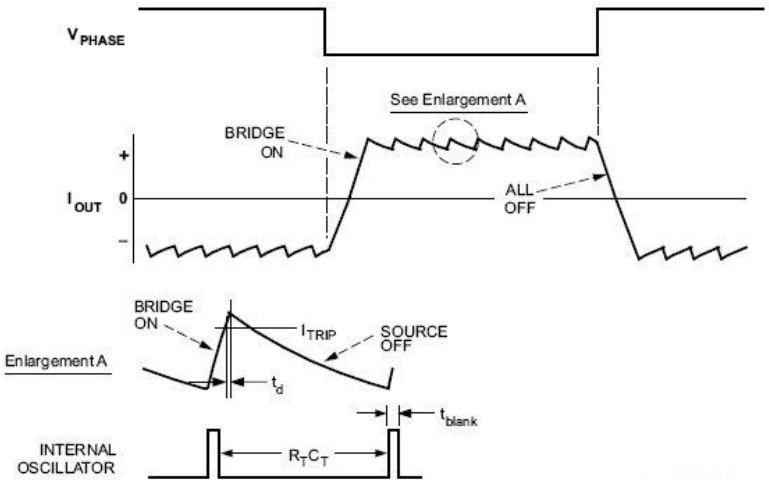
Test Condition: T_A=25℃，V_M=5V-35V

Characteristic	Symbol	Test Conditions	Limits			
			Min.	Typ.	Max.	Units
Load Supply Voltage Range	V_M	Operating	5.5	-	35	V
Thermal Shutdown Temp.	T_J	V_{IN} = 3.25V	150	170	200	℃
Thermal Shutdown Hysteresis.	△T_J		7	15	-	℃

表 2: 馬達相關電氣特性

過熱保護動作: 過熱保護電路在感測到驅動 IC 內部溫度到達 165℃時，將自動斷路，此時馬達即停止動作，當溫度下降 8℃後，保護電路自動回復導通，馬達就繼續先前的動作。

電流限流保護動作: 請參照右圖，在 H-bridge 開始輸出時，電流隨著馬達轉動增加，當電流值超過 I_{TRIP} (如右下圖 Enlargement A 中之指示)，就會停止 H-bridge 的輸出，直到內部震盪器下一個時脈傳送出(如右下圖 INTERNAL OSCILLATOR)，又會開始電流的傳送，如此反覆，電流會被固定在如圖的範圍內。



操作注意事項:

馬達模組提供一組馬達連接腳位，請確認所連接的馬達為直流馬達。

模組出廠時並未加上散熱片，在低電流熱對流良好環境，模組可以正常操作，但在大電流通過，或是於高熱無法靠一般對流散除時，建議將所附的散熱片加上。下表是在室溫(25℃)，熱對流良好環境下，以較大電流測試，部安裝散熱片的情況下，模組可以正常

運作的約略時間:

Current (A)	Time to overheat protection (Sec)
10	>300
12	~107
15	~40
18	~20

表 3: 電流與過熱保護啟動時間(未安裝散熱片)

模組操作溫度　　0 ℃~ 70 ℃ (馬達之操作溫度請另行確認)
模組儲存溫度　　-50 ℃~125℃

指令格式	指令功能
設定馬達速度	
Forward(uint8_t *Speed*)	命令馬達執行向前轉的動作，並且根據 ***Speed*** 所給的值，決定馬達的轉速，***Speed*** 可以輸入的範圍為 0~255 (***Speed*** 值越高，轉速越快)
Backward(uint8_t *Speed*)	命令馬達執行向後轉的動作，並且根據 ***Speed*** 所給的值，決定馬達的轉速，***Speed*** 可以輸入的範圍為 0~255 (***Speed*** 值越高，轉速越快)
SetSpdDC(uint8_t *Speed)*	命令馬達模組以 ***Speed*** 的值，設定為轉速，馬達模組仍以原先設定的轉向動作，***Speed*** 可以輸入的範圍為 0~255 (***Speed*** 值越高，轉速越快)
設定馬達轉向	
SetDir(uint8_t *Dir*)	命令馬達以 ***Dir*** 指定的方向轉動，***Dir*** 輸入 0 為向前，1 為向後
停止馬達轉動	
Brake(*void*)	命令馬達模組快速停止轉動
Stop(*void*)	命令馬達模組停止轉動
取得設定狀態	
GetDir(uint8_t &*Dir*)	取得馬達設定的轉向值，存放於 ***Dir*** 參數中 (0 表示向前轉，1 表示向後轉)
設定轉速計與取得轉速	
SetTACHPeriod(uint8_t *Period*)	命令馬達模組以 ***Period*** 設定的時間，計數轉速值，輸入的 ***Period*** 值定義如下: 0: 4ms　　1: 8 ms 2: 16 ms　　3: 32ms 4: 64 ms　　5: 125 ms 6: 250 ms　　7: 500 ms

	8: 1 s　　9: 2 s 10: 4 s　　11: 8 s 12: 15 s　　13: 30 s 14: 60 s 馬達模組會在設定的時間內量測轉速計回傳的脈波數，時間越短則更新越快，但是誤差也越大，時間越長則更新越慢，但計數的誤差也越小
uint8_t *Status* = TACHIn(uint32_t &*Speed*)	取得轉速計更新狀態，存放於 ***Status*** 中(0 表示從上次更新後，尚未取得新的計數值，1 表示轉速值已經更新)，同時取得轉速計每分鐘回傳的脈波數，存放於 ***Speed*** 參數中，***Speed*** 回傳值範圍為 0~4294836225，請注意回傳值的更新時間，與準確度會因設定的計數時間而不同＊1
設定轉速與取得設定 (此部分指令需要在接上轉速計後，才能正常動作)	
SetSpdCtrl(uint8_t *Dir*, uint32_t *Speed*)	設定轉速控制的動作，***Dir*** 輸入 0 為向前，1 為向後，***Speed*** 可以輸入的範圍為 0~245756250 (數字越大速度越快)，若馬達無法達到設定的轉速，會以馬達最大轉速動作，請注意設定後不會啟動轉速控制，需要再執行 **SpdCtrlOn**()啟動轉速控制，達到的轉速值，會根據 **SetTACHPeriod** 設定的量測時間影響，有不同程度的誤差，達到所需轉速的時間也會不同＊1
SpdCtrlOn(*void*)	依轉速控制設定，啟動轉速控制，請先執行 **SetSpdCtrl** 指令，設定相關轉速控制後，再執行此命令
SpdCtrlOff(*void*)	關閉轉速控制
uint8_t *Status* = GetSpdCtrlStatus(*void*)	取得轉速控制狀態，存放於 ***Status*** 中(0 表示尚未達到設定的轉速，或是沒有啟動轉速控制，1 表示已經達到設定的轉速)
SetP(uint8_t *ValP*)	以 ***ValP*** 設定 PID 轉速控制的 P 值，***ValP*** 可以輸入 0~255 間的整數，預設為 0
SetI(uint8_t *ValI*)	以 ***ValI*** 設定 PID 轉速控制的 I 值，***ValI*** 可以輸入 0~255 間的整數，預設為 0
SetD(uint8_t *ValD*)	以 ***ValD*** 設定 PID 轉速控制的 D 值，***ValD*** 可以輸入 0~255 間的整數，預設為 0
SetScalar(uint8_t *Scalar*)	以 ***Scalar*** 設定 PID 轉速控制的比例值，***Scalar*** 可以輸入 0~255 間的整數，比例設定如下: 0: PID 計算結果速度值/1 1: PID 計算結果速度值/2 2: PID 計算結果速度值/4 3: PID 計算結果速度值/8 …

	32: PID 計算結果速度值/2^32 33~255: 關閉 PID 控制改以 1 為單位改變速度值 預設值為 255
設定計數與取得設定 (此部分指令需要在接上轉速計後，才能正常動作)	
SetCount(uint8_t *Mode*, uint16_t *Count*)	設定計數控制的動作，***Mode*** 輸入 0 會在計數到達後，執行 **Stop** 指令停止馬達,1 會在計數到達後,執行 **Brake** 指令停止馬達，***Count*** 可以輸入的範圍為 0~65535，代表所要計數的次數，請注意設定後不會開始計數，需要再執行 **CountOn ()**啟動計數動作
***Status*= GetCount(uint8_t &*Mode*, uint16_t &*Count*)**	取得計數控制設定,狀態存放於 ***Status*** 中(0 表示沒有啟動計數控制，1 表示已經啟動計數控制)，結束模式存放於 ***Mode*** 參數中，***Mode*** 回傳 0 會在計數到達後，執行 **Stop** 指令停止馬達，1 會在計數到達後，執行 **Brake** 指令停止馬達，計數值存放於 ***Count*** 參數中，***Count*** 回傳值範圍為 0~65535
CountOn(*void*)	依計數控制設定，啟動計數控制，請先執行 **SetCount** 指令，設定相關計數控制後，再執行此命令
CountOff(*void*)	關閉計數控制
uint8_t *Status* = GetCountStatus(*void*)	取得計數控制狀態,存放於 ***Status*** 中(0 表示尚未達到設定的計數值，或是沒有啟動計數控制，1 表示已經達到設定的計數值)
取得錯誤狀態與回復原設定值	
uint8_t *Status* = GetBrakeButStatus(*void*)	取得 Brake 按鍵的狀態值,存放於 ***Status*** 中,0 表示 Brake 按鍵沒有被啟動，1 表示 Brake 按鍵被啟動過 請注意在 Brake 按鍵被啟動過後，此回傳值就會一直保持為 1，直到執行 **ClrBrakeButStatus** 指令 在狀態為 1 時，所有啟動馬達的指令都無效
ClrBrakeButStatus(*void*)	清除 Brake 按鍵的狀態值
uint8_t *Status* = GetFaultStatus(*void*)	取得錯誤檢測狀態,存放於 ***Status*** 中(0 表示沒有檢測到錯誤狀態，1 表示檢測到錯誤狀態) ＊2
EnFaultStop(*void*)	啟動偵測到錯誤狀態，自動停止馬達動作的功能，錯誤狀態包含偵測到 IC 溫度過熱停止馬達，或瞬間電流過大進行限流等動作，執行此指令後，當錯誤狀態發生，必須執行 **ClearFault()**才能再啟動馬達動作，預設為 **DisFaultStop()**
DisFaultStop(*void*)	關閉偵測到錯誤狀態，自動停止馬達動作的功能，執行此指令後，模組若偵測到錯誤狀態，會自動執行 **ClearFault()**指令，並繼續原先設定的馬達動作，不會

	強迫停止馬達，使用者必須自己判斷是否要停止馬達，預設為 **DisFaultStop()**，可以根據 **FaultProtectionEvent** 判斷是否錯誤狀態仍不斷產生
ClearFault(*void*)	確認清除錯誤狀態，當執行 **EnFaultStop()**指令後，請在確認排除產生錯誤的原因後，再執行此指令，才可繼續操作馬達
RestoreStatus(*void*)	回復錯誤狀態產生時的設定，當模組產生錯誤狀態時，會自動儲存當時的所有設定，在執行這個指令後，可以再將所有設定重設為當時的狀態

＊1 請注意設定值與回傳值會因為 Period 的設定，而有不同的最大值限制

Period=0 ➔ 983025000　　Period=8 ➔ 3932100
Period=1 ➔ 491512500　　Period=9 ➔ 1966050
Period=2 ➔ 245756250　　Period=10 ➔ 983025
Period=3 ➔ 122878125　　Period=11 ➔ 491513
Period=4 ➔ 61439063　　Period=12 ➔ 262140
Period=5 ➔ 31456800　　Period=13 ➔ 131070
Period=6 ➔ 15728400　　Period=14 ➔ 65535
Period=7 ➔ 7864200

＊2 在 DisFaultStop()狀態下，收到 Fault 後，系統就會自動進行 ClearFault()的動作，執行 GetFaultStatus()仍會得到 0

範例程式**:**

```
#include "arminno.h"
MotorRunnerC myMotor(0);        //      設定模組編號為 0
int main(void)
{
        myMotor.Forward(200);                   //      讓馬達以 200 的速度向前轉動
        Pause(30000);
        myMotor.Stop();                         //      停止馬達轉動
        Pause(30000);
        myMotor.Backward(200) ;                 //      讓馬達以 200 的速度向後轉動
        Pause(30000);
        myMotor.SetDir(0) ;                     //      設定馬達轉向改為向前轉動
        Pause(30000);
        myMotor.SetSpdDC(150) ;                 //      將馬達轉速改為 150
        Pause(3000);
        myMotor.Brake()    ;                    //      快速停止馬達
}
```

附錄

1. 已知問題:
   - V1.0 版本在快速連傳送速度設定指令時，可能造成模組停止動作。
   - V1.1 版本(含)之前版本，GetCountStatus 回傳值只會回傳 0。

2. 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