Gamepad BT

藍牙搖桿控制模組

版本： V2.0

產品介紹：

GamepadBT 模組提供簡易的設定與位置取得指令，搭配 12 個按鈕，讓使用者規劃符合自己需求的操作模式。透過 cmdBUS 與 BASIC Commander 連接，可以用簡單的指令與藍牙搖桿溝通，取得按鍵資訊製作專屬的應用指令。

應用方向：

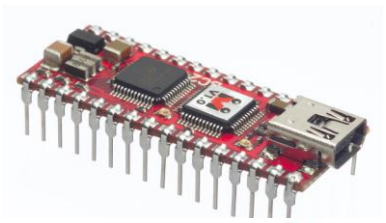
- 連結機器人，設定按鍵達成控制動作與行進等行為。
- 各種測試機具的操作。
- 與無線搖桿結合，控制各種搖控車、飛機等應用。
- 控制利基應用科技的各項應用套件。

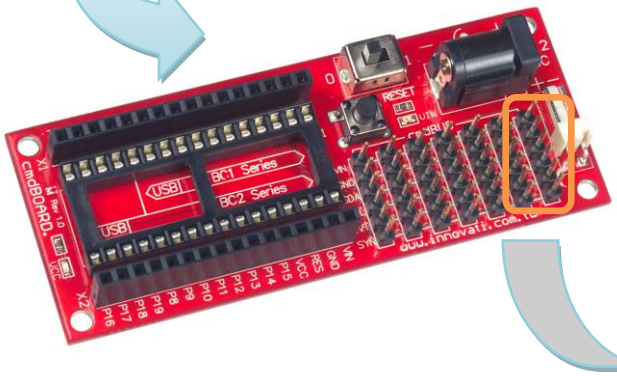
產品特色：

- 設定容易，只要使用 cmdBUS 連接 BASIC Commander，就可以用專屬的指令做各種應用。
- 操縱桿部份，可設定類比回傳、四向或八向操縱桿位置回傳。
- 方向鍵，可設定四向或八向位置回傳。
- 十二個功能鍵，可單獨控制或組合控制。
- 可自行定義按鈕功能，包括按鍵的連續觸發起動時間，以及連續觸發的速率，都可透過指令設定。
- 可透過 I2C 方式，下達指令。

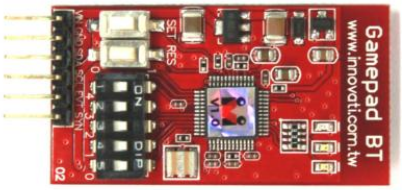
連接方式：直接將 ID 開關撥至欲設定的編號，再將 cmdBUS 連接至 BASIC Commander 上對應的腳位，連接上藍牙搖桿後，就可透過 BASIC Commander 執行操作。

可以先將 BASIC Commander 放在有提供 cmdBUS 接出腳位的板上，較容易連接模組。





連接所選用的藍牙搖桿



以 cmdBUS 連接模組，請注意腳位的方向性。

產品規格：

功能鍵：L1、L2

左邊操縱桿
(L Joystick)
功能鍵：L3

方向鍵

功能鍵：
Select、Start、Home

功能鍵：R1、R2

功能鍵：
A、B、X、Y

右邊操縱桿
(R Joystick)
功能鍵：R3

操作注意事項:

模組操作溫度　　-40 ℃~ 123.8 ℃
模組儲存溫度　　-40 ℃~125℃
本模組適用於原廠藍牙搖桿，副廠藍牙搖桿不在保證範圍內。

如何配對

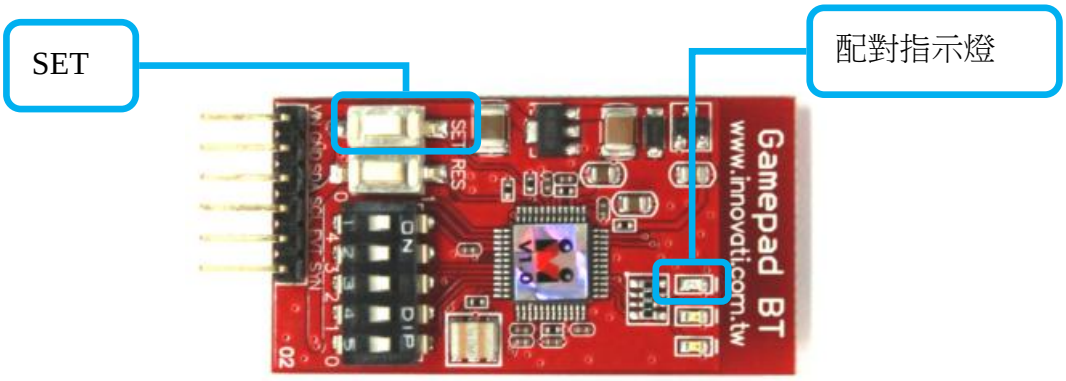
1. 模組接上 CmdBUS 並提供電源，請確認模組的**配對指示燈**保持閃爍。
2. 打開手把中間的蓋子，將電源開關撥至 ON。此時手把上的所有 LED 燈仍保持熄滅。
   如未熄滅，請重新 OFF-ON 一次，或是長按住 HOME 讓手機進入待機模式。
3. 在待機模式下先按住手把 X 並保持，再按住 HOME，待手把上**連線指示燈**開始閃爍即可全部放開。**連線指示燈**會閃爍一段時間後保持一顆恒亮，即表示配對成功

註 : 1) 如果手把進入睡眠模式(LED 燈全滅) -可按住 X，再按住 HOME 來喚醒手把。

2) 如果模組進入睡眠模式(**配對燈**滅)，或是一直無法成功配對-(二擇一)

2-1 可重新給 CmdBUS 上電

2-2 或是按住 RES 鍵約 6 秒。放開 RES 後再快速按模組上的 **SET 鍵**兩次。
此時模組**配對指示燈**會快速閃爍並**重新**進入配對模式。



模組下達指令的方式可分為兩種：cmdBUS、I2C 控制方式

cmdBUS 指令表:

下面的指令表是專供控制 GamepadBT 模組的各種指令，必要輸入的指令名稱與參數，以粗底或粗斜體表示，粗體的文字在輸入時請不要更改，粗斜體的文字請自行定義適當格式的參數填入。輸入時請注意 innoBASIC Workshop 大寫與小寫會視為相同字。在執行 GamepadBT 指令前，請先於程式開頭定義對應參數與編號，例:

Peripheral *ModuleName* As GamepadBT @ *ModuleID*

I2C 通訊協議(Protocol):

為了使更廣泛的使用者能控制模組，提供了部份指令的通訊協議讓使用者應用。透過通訊規格，使用者可使用 I2C 通訊協議為模組下達命令。

通訊協議常見的封包如下：

MID：模組 ID 編號，空間大小為 Byte 的變數。對應於硬體的指播開關。

CID：命令 ID 編號，空間大小為 Byte 的變數。依不同命令而改變。

CheckSum1：驗證位元_1，空間大小為 Byte 的變數。

定義方式：255 – (MID * 2) – CID

CheckSum2：驗證位元_2，空間大小為 Byte 的變數。

定義方式：255 – (CheckSum1~CheckSum2 之間的變數總和)

CheckSum3：驗證位元_3，空間大小為 Byte 的變數。

定義方式：255 – MID-(MID~CheckSum3 之間的變數總和)

Dummy：虛設位元，可為任意變數。空間大小為 Byte 的變數。

於通訊規格每筆資料空間大小階為 Byte，若資料空間大小超過一個 Byte 時，需將資料拆開，並由 Low Byte 開始傳送。

Ex:傳送資料 Temp 為一筆空間大小為 Word 的資料,則需將 Temp 拆開,分為 Temp_L、Temp_H，並且先傳送 Temp_L。

Ex1 模組編號為 2，命令編號為 153，傳送參數 Byte 為 100，通訊協議為 MID+CID+CheckSum1+Byte+CheckSum2+Dummy 則：

MID = 2

CID = 153

CheckSum1 = 255 – (2*2) – 153 = 98

Byte = 100

CheckSum2 = 255 – 100

Dummy = 0~255 之間的任意數

Ex2 模組編號為 2，命令編號為 153，傳送參數 Temp 為 511，通訊協議為 MID+CID+CheckSum1+Temp_L+Temp_H+CheckSum2+Dummy 則：

MID = 2

CID = 153

CheckSum1 = 255 – (2*2) – 153 = 98

Temp_L = 255，Temp_H = 1

CheckSum2 = 255 – Temp_L–Temp_H = 255

Dummy = 0~255 之間的任意數

<table>
<tr><th>指令格式</th><th>指令功能</th></tr>
<tr><td colspan="2">設定相關指令</td></tr>
<tr><td>SetKeyRepeatFunc(Key_ID)</td><td>設定是否啟動重複輸入判定。
啟動 = 1，關閉 = 0
Key_ID<table><tr><th>Bit</th><th>對應按鈕</th><th>十進制</th></tr><tr><td>0</td><td>Y</td><td>1</td></tr><tr><td>1</td><td>B</td><td>2</td></tr><tr><td>2</td><td>A</td><td>4</td></tr><tr><td>3</td><td>X</td><td>8</td></tr><tr><td>4</td><td>L1</td><td>16</td></tr><tr><td>5</td><td>R1</td><td>32</td></tr><tr><td>6</td><td>L2</td><td>64</td></tr><tr><td>7</td><td>R2</td><td>128</td></tr><tr><td>8</td><td>Select</td><td>256</td></tr><tr><td>9</td><td>Start</td><td>512</td></tr><tr><td>10</td><td>L3</td><td>1024</td></tr><tr><td>11</td><td>R3</td><td>2048</td></tr></table>EX：若想啟動 Y、B 則 Key_ID 可設定為：&B11(二進制)，或 3(十進制)。</td></tr>
<tr><td>GetKeyRepeatFunc(Key_ID)</td><td>取得是否啟動重複輸入判定設定。
啟動 = 1，關閉 = 0
Key_ID<table><tr><th>Bit</th><th>對應按</th><th>十進制</th></tr><tr><td>0</td><td>Y</td><td>1</td></tr><tr><td>1</td><td>B</td><td>2</td></tr><tr><td>2</td><td>A</td><td>4</td></tr><tr><td>3</td><td>X</td><td>8</td></tr><tr><td>4</td><td>L1</td><td>16</td></tr><tr><td>5</td><td>R1</td><td>32</td></tr><tr><td>6</td><td>L2</td><td>64</td></tr><tr><td>7</td><td>R2</td><td>128</td></tr><tr><td>8</td><td>Select</td><td>256</td></tr><tr><td>9</td><td>Start</td><td>512</td></tr><tr><td>10</td><td>L3</td><td>1024</td></tr><tr><td>11</td><td>R3</td><td>2048</td></tr></table></td></tr>
<tr><td>SetRepeatTime(Time)</td><td>設定重複輸入判定時間值。
經由 Time 設定，可輸入範圍為 0~255 之間的整數值，單位為 10 ㎳。</td></tr>
<tr><td>GetRepeatTime(Time)</td><td>取得重複輸入判定時間設定值。
回傳值存放於 Time。回傳範圍為 0~255 之間的整數值，單位為 10 ㎳。</td></tr>
</table>

SetRepeatRate(*Rate*)	設定重複輸入判定速率值。 經由 ***Rate*** 設定，可輸入範圍為 0~255 之間的整數值，單位為 10 ms。
GetRepeatRate(*Rate*)	取得重複輸入判定速率設定值。 回傳值存放於 ***Rate***。回傳範圍為 0~255 之間的整數值，單位為 10 ms。
應用相關指令	
CmdBUS： **GetLXYPos(*POSx,POSy*)** **I2C：** **Out：MID+113+CheckSum1+Dummy** **In：MID+PosX+PosY+CheckSum3**	取得左邊操縱桿座標值。 回傳 XY 座標，分別儲存於 ***POSx，POSy*** 中，預設範圍為 -127~+127。
CmdBUS： **GetRXYPos(*POSx,POSy*)** **I2C：** **Out：MID+114+CheckSum1+Dummy** **In：MID+ PosX + PosY +CheckSum3**	取得右邊操縱桿座標值。 回傳 XY 座標，分別儲存於 ***POSx，POSy*** 中，預設範圍為 -127~+127。
CmdBUS： **GetL4WayValue(*Dir*)** **I2C：** **Out：MID+115+CheckSum1+Dummy** **In：MID+Dir+CheckSum3**	以四向表示方式，取得左邊操縱桿位置。 回傳值存放於 ***Dir*** 是方向值，只會有 0~4 的回傳值，分別代表： **0**：操縱桿位於中心點　**1**：操縱桿位於右方→ **2**：操縱桿位於下方↓　**3**：操縱桿位於左方← **4**：操縱桿位於上方↑
CmdBUS： **GetR4WayValue(*Dir*)** **I2C：** **Out：MID+116+CheckSum1+Dummy** **In：MID+Dir+CheckSum3**	以四向表示方式，取得右邊操縱桿位置。 回傳值存放於 ***Dir*** 是方向值，只會有 0~4 的回傳值，分別代表： **0**：操縱桿位於中心點　**1**：操縱桿位於右方→ **2**：操縱桿位於下方↓　**3**：操縱桿位於左方← **4**：操縱桿位於上方↑
CmdBUS： **GetL8WayValue(*Dir*)** **I2C：** **Out：MID+117+CheckSum1+Dummy** **In：MID+Dir+CheckSum3**	以八向表示方式，取得左邊操縱桿位置。 回傳值存放於 ***Dir*** 是方向值，只會有 0~8 的回傳值，分別代表： **0**：操縱桿位於中心點　**1**：操縱桿位於右方→ **2**：操縱桿位於右下方↘**3**：操縱桿位於下方↓ **4**：操縱桿位於左下方↙**5**：操縱桿位於左方← **6**：操縱桿位於左上方↖**7**：操縱桿位於上方↑

<table>
<tr><td></td><td>8：操縱桿位於右上方↗</td></tr>
<tr><td>CmdBUS：
GetR8WayValue(Dir)
I2C：
Out：MID+118+CheckSum1+Dummy
In：MID+Dir+CheckSum3</td><td>以八向表示方式，取得右邊操縱桿位置。
回傳值存放於 Dir 是方向值，只會有 0~8 的回傳值，分別代表：
0：操縱桿位於中心點 1：操縱桿位於右方→
2：操縱桿位於右下方↘3：操縱桿位於下方↓
4：操縱桿位於左下方↙5：操縱桿位於左方←
6：操縱桿位於左上方↖7：操縱桿位於上方↑
8：操縱桿位於右上方↗</td></tr>
<tr><td>CmdBUS：
Status = GetKeyStatus( )
I2C：
Out：MID+138+CheckSum1+Dummy
In：
MID+Status_L+Status_H+CheckSum3</td><td>取得按鈕狀態存放於 Status 中。
啟動 = 1，關閉 = 0
<table>
<tr><td rowspan="13">Status</td><td>Bit</td><td>對應按鈕</td><td>十進制</td></tr>
<tr><td>0</td><td>Y</td><td>1</td></tr>
<tr><td>1</td><td>B</td><td>2</td></tr>
<tr><td>2</td><td>A</td><td>4</td></tr>
<tr><td>3</td><td>X</td><td>8</td></tr>
<tr><td>4</td><td>L1</td><td>16</td></tr>
<tr><td>5</td><td>R1</td><td>32</td></tr>
<tr><td>6</td><td>L2</td><td>64</td></tr>
<tr><td>7</td><td>R2</td><td>128</td></tr>
<tr><td>8</td><td>Select</td><td>256</td></tr>
<tr><td>9</td><td>Start</td><td>512</td></tr>
<tr><td>10</td><td>L3</td><td>1024</td></tr>
<tr><td>11</td><td>R3</td><td>2048</td></tr>
</table>
EX：若 Status = 3 則 Y、B 被啟動。</td></tr>
<tr><td>CmdBUS：
GetDir4Way(Dir)
I2C：
Out：MID+141+CheckSum1+Dummy
In：MID+Dir+CheckSum3</td><td>取得方向鍵狀態，以四向方式回傳。
回傳值存放於 Dir，只會有 0~4 的回傳值，分別代表：
0：無方向 1：右方→ 2：下方↓
3：左方← 4：上方↑</td></tr>
<tr><td>CmdBUS：
GetDir8Way(Dir)
I2C：
Out：MID+142+CheckSum1+Dummy
In：MID+Dir+CheckSum3</td><td>取得方向鍵狀態，以八向方式回傳。
回傳值存放於 Dir，只會有 0~8 的回傳值，分別代表：
0：無方向 1：右方→ 2：右下方↘
3：下方↓ 4：左下方↙ 5：左方←
6：左上方↖ 7：上方↑ 8：右上方↗</td></tr>
<tr><td>GetConnect(Status)</td><td>取得控制器連接狀態。
存放於 Status 中，回傳值為 0、1。</td></tr>
</table>

	分別代表： 0：沒有偵測到搖桿，1：搖桿正常連接
應用事件相關指令	
SetStickRefreshRate(*Rate*)	設定操縱桿連續變動時，最快產生 EVENT 速率。 ***Rate*** 輸入數值 1~255, 單位 10ms, 其餘數值無效 0=1 皆為 10ms。
GetStickRefreshRate(*Rate*)	取得操縱桿連續變動時，最快產生 EVENT 速率。 回傳值存入 ***Rate***，範圍 1~255, 單位 10ms。
EnableLStickEvent( )	啟動**左邊**操縱桿 **StickEvent** 事件。 經由 **SetStickRefreshRate** 指令決定產生速率。
DisableLStickEvetn( )	關閉**左邊**操縱桿 **StickEvent** 事件。
EnableRStickEvent( )	啟動**右邊**操縱桿 **StickEvent** 事件。 經由 **SetStickRefreshRate** 指令決定產生速率。
DisableRStickEvetn( )	關閉**右邊**操縱桿 **StickEvent** 事件。
EnableL4WayEvent( )	啟動**左邊**操縱桿 **4WayEvent** 事件。
DisableL4WayEvent( )	關閉**左邊**操縱桿 **4WayEvent** 事件。
EnableR4WayEvent( )	啟動**右邊**操縱桿 **4WayEvent** 事件。
DisableR4WayEvent( )	關閉**右邊**操縱桿 **4WayEvent** 事件。
EnableL8WayEvent( )	啟動**左邊**操縱桿 **8WayEvent** 事件。
DisableL8WayEvent( )	關閉**左邊**操縱桿 **8WayEvent** 事件。
EnableR8WayEvent( )	啟動**右邊**操縱桿 **8WayEvent** 事件。
DisableR8WayEvent( )	關閉**右邊**操縱桿 **8WayEvent** 事件。
EnableKeyPressedEvent( )	啟動 **KeyPressedEvent** 事件。
DisableKeyPressedEvent( )	關閉 **KeyPressedEvent** 事件。
EnableKeyRelesedEvent( )	啟動 **KeyRelesedEvent** 事件。
DisableKeyRelesedEvent( )	關閉 **KeyRelesedEvent** 事件。
EnableDir4WayEvent( )	啟動 **Dir4WayEvetn** 事件。
DisableDir4WayEvent( )	關閉 **Dir4WayEvetn** 事件。
EnableDir8WayEvent( )	啟動 **Dir8WayEvetn** 事件。
DisableDir8WayEvent( )	關閉 **Dir8WayEvetn** 事件。

模組提供應用事件**:**

事件名稱 **(Event)**	啟動條件
LStickEvent	當**左邊**操縱桿開始移動時產生事件。 依照 **SetStickEvent** 所設定的時間頻率回傳。
RStickEvent	當**右邊**操縱桿開始移動時產生事件。 依照 **SetStickEvent** 所設定的時間頻率回傳。
L4WayEvent	當**左邊**操縱桿方向改變時產生事件。與 **SetStickEvent** 無關。

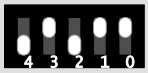
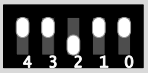
R4WayEvent	當右邊操縱桿方向改變時產生事件。與 **SetStickEvent** 無關。
L8WayEvent	當左邊操縱桿方向改變時產生事件。與 **SetStickEvent** 無關。
R8WayEvent	當右邊操縱桿方向改變時產生事件。與 **SetStickEvent** 無關。
KeyPressedEvent	所有按鈕共用。 當 **RepeatKey** 關閉時，按下按鈕即產生事件。 當 **RepeatKey** 啟動時，按下按鈕，到達 **RepeatTime** 設定時間，以及每隔 **RepeatRate** 設定的時間就會產生事件。
KeyReleasedEvent	所有按鈕共用。 每當偵測到 KeyRelese 的動作就產生事件。
Dir4WayEvent	當方向鍵改變時產生事件。
Dir8WayEvent	當方向鍵改變時產生事件。
ConChangeEvent	偵測到控制器接上或拔除時產生事件。Always Enable

範例程式:

```
Peripheral BT As GamepadBT @ 31          '設定模組編號
Dim b4Dir As Byte                        '儲存取得的方向值
Dim b8WayL,b8WayR As Byte                '儲存取得的操縱桿方向值
Dim wStatus As Word                      '儲存取得的按鈕狀態值

Sub Main()
    Debug CLS
    Debug "GamepadBT Demo"               '終端視窗顯示規劃
    Debug CSRXY(1,2),"Direction:"
    Debug CSRXY(1,3),"RStick8Way:"
    Debug CSRXY(1,4),"LStick8Way:"
    Debug CSRXY(1,5),"GetKeyStatus:"
    BT.EnableKeyPressedEvent()           '啟動按鈕按下事件
    BT.EnableKeyReleasedEvent()          '啟動按鈕放開事件

    Do
        BT.GetDir4Way(b4Dir)             '以四向回傳方式，取得方向鍵狀態
        Debug CSRXY(11,2),b4Dir          '於終端視窗(第 11 行，第 2 列)顯示

        BT.GetR8WayValue(b8WayR)         '以八向回傳方式，取得右邊操縱桿狀態
        Debug CSRXY(12,3),b8WayR         '於終端視窗(第 12 行，第 3 列)顯示

        BT.GetL8WayValue(b8WayL)         '以八向回傳方式，取得左邊操縱桿狀態
        Debug CSRXY(12,4),b8WayL         '於終端視窗(第 12 行，第 4 列)顯示

        Debug CSRXY(15,5),%BIN12 wStatus '於終端視窗(第 15 行，第 5 列)，以二進制顯示
    Loop

End Sub

Event BT.KeyPressedEvent()               '按鈕按下事件
    wStatus = BT.GetKeyStatus            '取得當下按鈕狀態，存放於 wStatus 中
End Event

Event BT.KeyReleasedEvent()              '按鈕放開事件
    wStatus = BT.GetKeyStatus            '更新當下按鈕狀態，存放於 wStatus 中
End  Event
```

附錄

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