UART 擴充模組

BMB22M210
Arduino Library V1.0.2 說明

版本：V1.10　日期：2023-10-10

www.bestmodulescorp.com

目錄

簡介

BMB22M210 是倍創推出的 UART 擴充模組，使用 UART 通訊方式。本文檔對 BMB22M210 的 Arduino Lib 函式、Arduino Lib 安裝方式進行說明；範例 1 使用 BMB22M210，演示了子序列埠通訊的功能。範例 2 使用 BMB22T101，演示了通過觸控按鍵控制舵機旋轉的功能。

適用型號：

型號	說明
BMB22M210	UART 擴充模組
BMB22T101	通訊擴充板

Arduino Lib 函式

Arduino Lib 名稱：BMB22M210		Lib 版本：V1.0.2
構造函式 & 初始化		
1	BMB22M210(uint8_t intPin, HardwareSerial *theSerial=&Serial)	
	描述	構造函式，使用 HW Serial 介面
	參數	intPin：INT 連接的腳位 *theSerial：選擇 HW Serial 介面
	返回值	—
	備註	—
2	void begin(uint32_t baud)	
	描述	模組初始化
	參數	baud：M-UART 的鮑率
	返回值	void
	備註	由於模組的 M-UART 的鮑率自適應，因此多個產品在同一個 UART 介面使用時，請優先進行本模組的初始化
3	void begin(uint32_t baud, uint8_t rstPin);	
	描述	模組初始化
	參數	baud：M-UART 的鮑率 rstPin：RST 腳位，BMB22T101 擴充板連到 D2
	返回值	void
	備註	此函式僅可用在 BMB22T101 擴充板 (1)

4	void beginPort(uint8_t port, uint32_t baud, uint8_t config=MYSERIAL_8N1)	
	描述	初始化 port
	參數	port：port 選擇 1：子序列埠 1 2：子序列埠 2 3：子序列埠 3 4：子序列埠 4 Baud：對應 port 的鮑率設定 (4800/9600/14400/19200/38400/ 57600/115200/230400 可設定) config：UART 格式設定 0 (MYSERIAL_8N1)：8 個資料位、無校驗、1 個停止位 1 (MYSERIAL_8N2)：8 個資料位、無校驗、2 個停止位 8 (MYSERIAL_801)：8 個資料位、0 校驗、1 個停止位 9 (MYSERIAL_802)：8 個資料位、0 校驗、2 個停止位 10 (MYSERIAL_8O1)：8 個資料位、奇校驗、1 個停止位 11 (MYSERIAL_8O2)：8 個資料位、奇校驗、2 個停止位 12 (MYSERIAL_8E1)：8 個資料位、偶校驗、1 個停止位 13 (MYSERIAL_8E2)：8 個資料位、偶校驗、2 個停止位 14 (MYSERIAL_811)：8 個資料位、1 校驗、1 個停止位 15 (MYSERIAL_812)：8 個資料位、1 校驗、2 個停止位
	返回值	—
	備註	—
功能函式		
5	uint8_t getINT()	
	描述	獲取 INT 腳位準位
	參數	—
	返回值	INT 腳位準位 1：HIGH 0：LOW
	備註	中斷腳位，任何一個子序列埠的 INTn 腳位為低準位，均能拉低主介面的 INT 腳位
6	void setPortStatus(uint8_t port, PORTSTATUS status)	
	描述	設定子序列埠的 Pin 狀態
	參數	port：port 選擇 status： 0 (TXRX_DISABLE)：TX、RX 全停用 1 (TX_DISABLE)：TX 停用，RX 啟用 2 (RX_DISABLE)：RX 停用，TX 啟用 3 (TXRX_ENABLE)：TX、RX 全啟用
	返回值	void
	備註	—

7	void write(uint8_t port, uint8_t data)	
	描述	子序列埠發送 1 字節資料
	參數	port：選擇要發送資料的 Port 1：子序列埠 1 2：子序列埠 2 3：子序列埠 3 4：子序列埠 4 data：要發送的資料
	返回值	void
	備註	—
8	int read(uint8_t port)	
	描述	子序列埠讀取 1 字節資料
	參數	port：選擇要讀取的 Port 1：子序列埠 1 2：子序列埠 2 3：子序列埠 3 4：子序列埠 4
	返回值	讀取的資料
	備註	資料為空時，返回 -1
9	void writeBytes(uint8_t port, uint8_t buff[], uint8_t num)	
	描述	子序列埠發送 num 字節資料
	參數	port：選擇要發送資料的 port 1：子序列埠 1 2：子序列埠 2 3：子序列埠 3 4：子序列埠 4 buff[]：發送的資料存儲陣列 num：要發送的資料個數
	返回值	void
	備註	—
10	void readBytes(uint8_t port, int buff[], uint8_t num)	
	描述	子序列埠讀取 num 字節資料
	參數	port：選擇要讀取資料的 port 1：子序列埠 1 2：子序列埠 2 3：子序列埠 3 4：子序列埠 4 buff[]：讀取的資料存儲陣列 num：要讀取的資料個數
	返回值	void
	備註	—

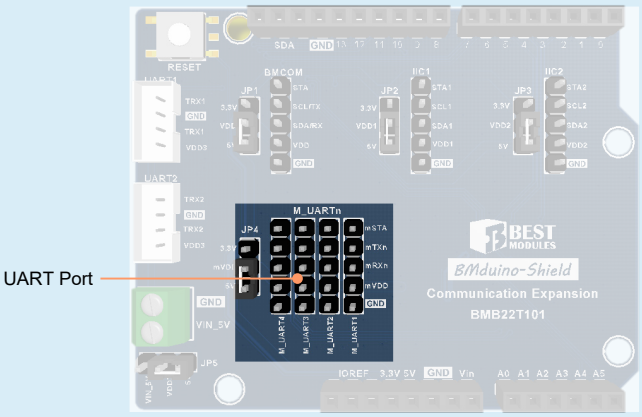
11	uint8_t available(uint8_t port)	
	描述	讀取子序列埠讀取 FIFO 中的資料個數
	參數	port：port 選擇 1：子序列埠 1 2：子序列埠 2 3：子序列埠 3 4：子序列埠 4
	返回值	子序列埠讀取 FIFO 中的資料個數
	備註	—
12	void resetPort(uint8_t port)	
	描述	重置子序列埠
	參數	port：選擇需要重置的子序列埠 1：子序列埠 1 2：子序列埠 2 3：子序列埠 3 4：子序列埠 4
	返回值	void
	備註	—
13	uint8_t getTxFifoNum(uint8_t port)	
	描述	讀取子序列埠發送 FIFO 中的資料個數
	參數	port：port 選擇 1：子序列埠 1 2：子序列埠 2 3：子序列埠 3 4：子序列埠 4
	返回值	子序列埠發送 FIFO 中的資料個數
	備註	—

14	uint8_t getInterruptFlag(uint8_t port)	
	描述	獲取子序列埠中斷標誌
	參數	port：port 選擇 1：子序列埠 1 2：子序列埠 2 3：子序列埠 3 4：子序列埠 4
	返回值	Bit0：接收 FIFO 觸點中斷 (2) 標誌位 0：無接收 FIFO 觸點中斷 1：有接收 FIFO 觸點中斷 Bit1：接收 FIFO 超時中斷 (3) 標誌位 0：無接收 FIFO 超時中斷 1：有接收 FIFO 超時中斷 Bit2：發送 FIFO 觸點中斷 (4) 標誌位 0：無發送 FIFO 觸點中斷 1：有發送 FIFO 觸點中斷 Bit3：發送 FIFO 空中斷 (5) 標誌位 0：無發送 FIFO 空中斷 1：有發送 FIFO 空中斷 Bit4~6：預留 Bit7：接收 FIFO 資料錯誤中斷 (6) 標誌位 0：無接收 FIFO 資料錯誤中斷 1：有接收 FIFO 資料錯誤中斷
	備註	—

15	void getFifoStatus (uint8_t port)	
	描述	獲取子序列埠 FIFO 狀態
	參數	port：port 選擇 1：子序列埠 1 2：子序列埠 2 3：子序列埠 3 4：子序列埠 4
	返回值	Bit0：子序列埠發送 TX 忙標誌位 0：子序列埠發送 TX 空 1：子序列埠發送 TX 忙 Bit1：子序列埠發送 FIFO 滿標誌位 0：子序列埠發送 FIFO 未滿 1：子序列埠發送 FIFO 滿 Bit2：子序列埠發送 FIFO 空標誌位 0：子序列埠發送 FIFO 空 1：子序列埠發送 FIFO 未空 Bit3：子序列埠接收 FIFO 空標誌位 0：子序列埠接收 FIFO 空 1：子序列埠接收 FIFO 未空 Bit4：子序列埠接收 FIFO 中資料校驗錯誤標誌位 0：無 PE 錯誤 1：有 PE 錯誤 Bit5：子序列埠接收 FIFO 中資料幀錯誤標誌位 0：無 FE 錯誤 1：有 FE 錯誤 Bit6：子序列埠接收 FIFO 中資料有 Line-Break 錯誤 0：無 Line-Break 錯誤 1：有 Line-Break 錯誤（Rx 訊號一直為 0 的狀態，包括校驗位和停止位在內） Bit7：接收 FIFO 資料錯誤中斷 (6) 標誌位 0：無接收 FIFO 資料錯誤中斷 1：有接收 FIFO 資料錯誤中斷
	備註	—
16	void setTxFifoInterrupt(uint8_t port,uint8_t data)	
	描述	設定子序列埠的發送 FIFO 觸發點
	參數	port：port 選擇 1：子序列埠 1 2：子序列埠 2 3：子序列埠 3 4：子序列埠 4 data：發送 FIFO 觸發點，範圍：0~255
	返回值	—
	備註	—

17	void setRxFifoInterrupt(uint8_t port,uint8_t data)	
	描述	設定子序列埠的接收 FIFO 觸發點
	參數	port：port 選擇 1：子序列埠 1 2：子序列埠 2 3：子序列埠 3 4：子序列埠 4 data：接收 FIFO 觸發點，範圍：0~255
	返回值	—
	備註	—

BMB22M210 Library 也可應用於 BMB22T101 擴充板的 UART Port。BMB22T101 擴充板直插於 BMduino 開發板使用時：UART 介面是 Serial4，INT 腳位連接 D10，RST 腳位連接 D2。



註 1：此函式僅可在 BMB22T101 擴充板上使用。拉低 RST 腳位實現 UART Port 重置，並可以重新設定其 M-UART 的鮑率。

註 2：當接收 FIFO 中的資料個數大於設定的發送 FIFO 觸發點時，產生該中斷。當接收 FIFO 中的資料個數小於設定的發送 FIFO 觸發點時，該中斷被清除。

註 3：當接收 FIFO 中資料個數小於設定的接收 FIFO 觸發點並且 RX 腳位 4 個字節之內沒有資料，產生該中斷。當接收 FIFO 中的資料被讀走或者 RX 繼續接收資料時，該中斷消失。

註 4：當發送 FIFO 中的資料個數小於設定的發送 FIFO 觸發點時，產生該中斷。當發送 FIFO 中的資料個數大於設定的發送 FIFO 觸發點時，該中斷被清除。

註 5：當發送 FIFO 中沒有資料，產生該中斷。當發送 FIFO 中的資料個數大於設定的發送 FIFO 觸發點時，該中斷被清除。

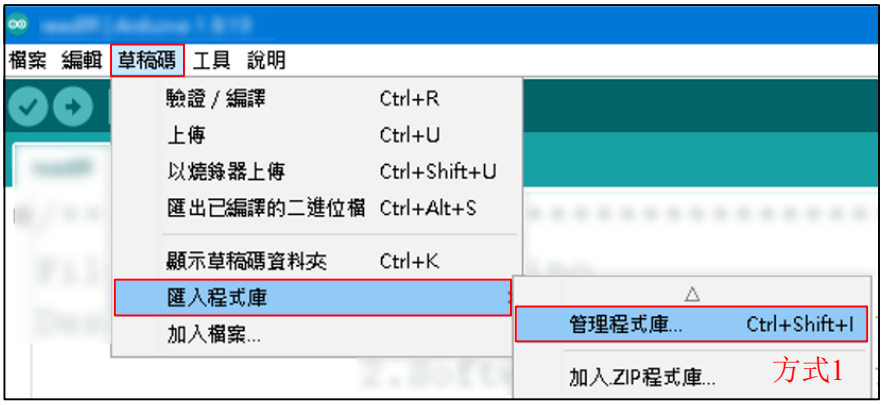
註 6：FIFO 資料錯誤中斷表明當前接收 FIFO 中有一個或以上的資料錯誤，產生錯誤的條件包括 OE（資料溢出錯誤），FE（資料幀錯誤），PE（奇偶校驗錯誤）和 BE（Line-Break 錯誤）。一旦有接收 FIFO 中有出錯資料，當讀取 FSR 寄存器後，清除相應的中斷。

Arduino Lib 下載及安裝

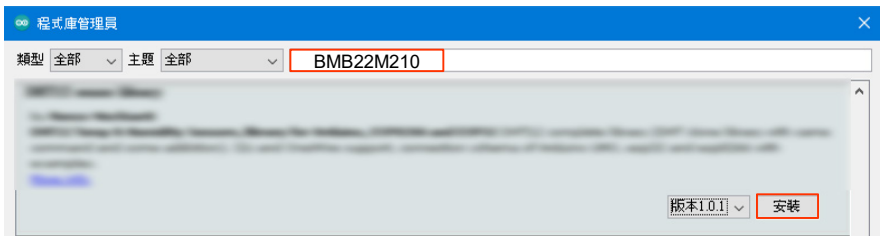
BMB22M210 Library：可參考下面兩種方法安裝 BMB22M210 的 Arduino Library

方式 1：搜索安裝

搜索安裝：Arduino IDE → 草稿碼 → 匯入程式庫 → 管理程式庫 → 搜索 BMB22M210 → 安裝



搜索安裝流程 1

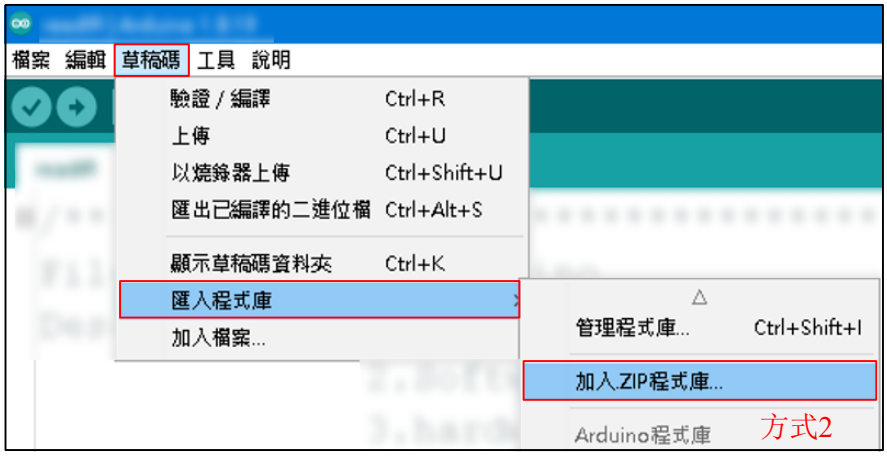


搜索安裝流程 2

方式 2：加入 .ZIP 程式庫，需提前下載 .ZIP 程式庫

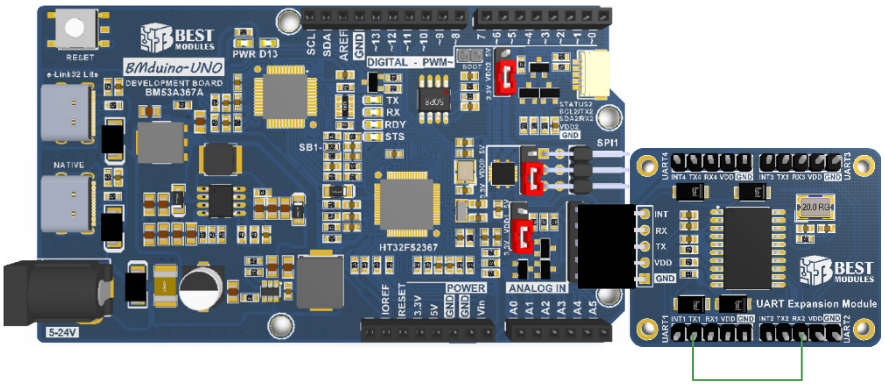
下載方法：打開倍創官方網站 (https://www.bestmodulescorp.com/ bmb22m210.html) 文件目錄下的 Arduino 範例程式 (BMB22M210 Library)。

加入 . ZIP 程式庫：Arduino IDE → 草稿碼 → 匯入程式庫 → 加入 .ZIP 程式庫 ...



Arduino 範例

範例 1：write_read



實物連接示意圖

範例實現功能：子序列埠 1 和 2 之間進行通訊，子序列埠 1 發送 1 字節資料，子序列埠 2 進行讀取並在序列埠監視視窗上顯示；子序列埠 1 連續發送 16 字節資料，子序列埠 2 進行讀取並在序列埠監視視窗上顯示。

1. 範例打開方式：Arduino IDE → 檔案 → 範例 → Lib 選擇 → 選擇範例 (write_read)
2. 範例说明：F
   a. 建立對象 & 模組初始化及設定

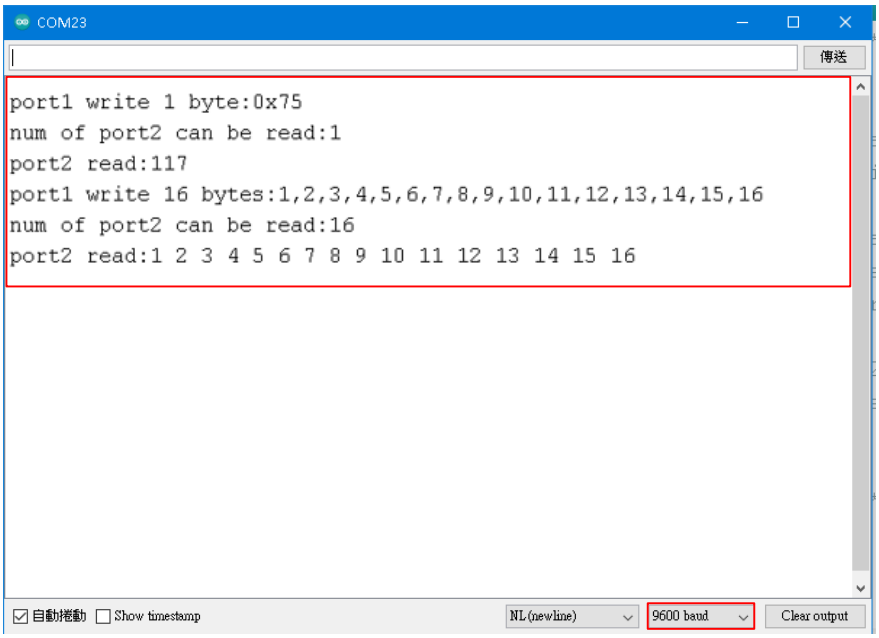
```
#include  "BMB22M210.h"
BMB22M210 mySerial(22,&Serial1);          // 建立對象
uint8_t sendbuff[15]={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15};
int receivebuff[15]={0};
uint8_t i=0;
int data = 0;
void setup()
{
  Serial.begin(9600);                     // 初始化序列埠監視視窗
  mySerial.begin(9600);                   // 初始化模組，設定主序列埠鮑率
  mySerial.beginPort(1,9600);             // 設定子序列埠 1 鮑率
  mySerial.beginPort(2,9600);             // 設定子序列埠 2 鮑率
}
```

   b. 子序列埠 1 發送 1 字節資料，子序列埠 2 進行讀取並在序列埠監視視窗上顯示；子序列埠 1 連續發送 16 字節資料，子序列埠 2 進行讀取並在序列埠監視視窗上顯示

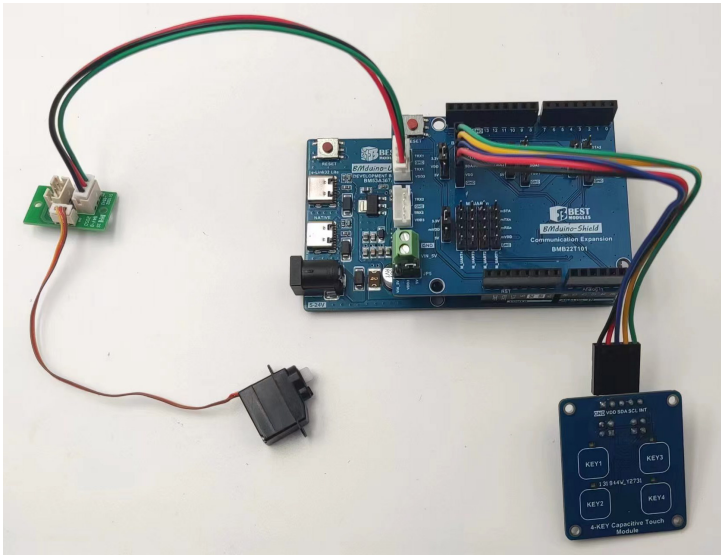
```
void loop()
{
  Serial.println( "port1 write 1 byte:0x75" );
  mySerial.write(1,0x75);                 // 寫 1 字節資料
  delay(2);
  Serial.print( "num of port2 can be read:" );
  Serial.println(mySerial.available(2)); // 判斷子序列埠 2 可讀取的資料數量
  Serial.print( "port2 read:" );
  if(mySerial.available(2)>=1)
```

```
  {
    Serial.print(mySerial.read(2));       // 讀 1 字節資料
    Serial.print( " " );
  }
  Serial.println( "" );
  Serial.println( "port1 write 16 bytes:1,2,3,4,5,6,7,8,9,10,11,12,13,
                                        14,15,16" );
  mySerial.writeBytes(1,sendbuff,16);     // 寫 16 字節資料，
                                          // 最大不超過 16 字節
  delay(20);
  Serial.print( "num of port2 can be read:" );
  Serial.println(mySerial.available(2)); // 判斷子序列埠 2 可讀取的資料數量
  Serial.print( "port2 read:" );
  if(mySerial.available(2)>=16)
  {
    mySerial.readBytes(2,receivebuff,16); // 連續讀 16 字節資料，
                                          // 最大不超過 16 字節
    for(i=0;i<16;i++)
    {
      Serial.print(receivebuff[i]);
      Serial.print( " " );
    }
  }
  Serial.println( "" );
  delay(1000);
}
```

3. 打開序列埠監視視窗，鮑率選擇 9600；序列埠監視視窗顯示如下：



範例 2：servoRun



实物连接示意图

實物連接示意圖

範例實現功能：將小型數字舵機 (BM22O2221-A) 與 4KEY 觸控按鍵模組 (BMK52M134) 分別連接在通訊擴充板的 UART1 介面與 BMCOM 介面上。通過觸控 4 個按鍵來控制舵機旋轉到 4 個對應的角度：KEY1 對應 0°，KEY2 對應 45°，KEY3 對應 90°，KEY4 對應 140°。

1. 範例打開方式：Arduino IDE→ 文件 → 示例 →Lib 選擇 → 選擇範例 (servoRun)

注意：使用該範例需先安裝 BMK52M134 Arduino Library。

2. 示例說明：

a. 建立對象 & 模組初始化及設定

```
#include "BMK52M134.h"
#include <SoftwareSerial.h>
/***One-Wire UART Port(UART1)****/
#define TX1   6
#define RX1   7
SoftwareSerial mySerial(RX1,TX1);          // 建立對象，小型數字舵機連接在通
                                           // 訊擴充板的 UART1 介面
/***BMCOM****/
BMK52M134     BMK52(3,&Wire);              // 建立對象，4KEY 觸控按鍵模組直
                                           // 插與通訊擴充板的 BMCOM 介面
uint8_t angle = 45;          // 小型數字舵機需旋轉到的角度參數，單位：度
uint8_t KEYValue = 0;        // 儲存按鍵觸控情況
void setup()
{
  Serial.begin(115200);        // 設定序列埠監控視窗
  BMK52.begin();             // 初始化 4KEY 觸控按鍵模組
  mySerial.begin(115200, SERIAL_9N1);      // 初始化小型數字舵機
  setEID(1);                // 設定小型數字舵機的 EID 為 1
  setPosTime(1,angle,0);      // 設定小型數字舵機旋轉到初始角度 45°
```

```
  Serial.print("Angle:");
  Serial.print(angle);          // 在序列埠監控視窗上顯示初始角度
  Serial.println("°");
}
```

b. 根據按鍵觸控情況設定舵機旋轉到相應的角度，並在序列埠監控視窗上顯示

```
void loop()
{
    if(!BMK52.getINT()) // 判斷按鍵是否有按下
   {
    KEYValue = BMK52.getKeyValue(); // 獲取按鍵觸控情況
    switch (KEYValue)
    {
    case 1: angle = 0; break;  //KEY1 按下，設定舵機需旋轉到的角度參數為 0
    case 2: angle = 45; break; //KEY2 按下，設定舵機需旋轉到的角度參數為 45
    case 3: angle = 90; break; //KEY3 按下，設定舵機需旋轉到的角度參數為 90
    case 4: angle = 140; break;//KEY4 按下，設定舵機需旋轉到的角度參數為
                               //140
    }
    setPosTime(1,angle,0);    // 根據按鍵觸控情況設定舵機旋轉到相應的角度
    Serial.print("Angle:");
    Serial.print(angle);       // 在序列埠監控視窗上顯示當前角度
    Serial.println("°");
   }
}
```

c. 設定小型數字舵機的 EID

```
uint8_t setEID(uint8_t EID)    // EID：需要設定的 EID，範圍：0~15
{
  uint16_t rBuf[4]={0};
  uint16_t mid = 0x100;
  uint8_t Tlen_eid = (0X02<<4)+0x00;
  uint8_t instr = 0x80 + (EID & 0x0f);
  uint8_t check_sum = (uint8_t)~(mid + Tlen_eid + instr);
  mySerial.SetRxStatus(DISABLE);  //RXPin 停用
  mySerial.SetTxStatus(ENABLE);   //TXPin 啟用
  /***** 发送指令 *****/
  mySerial.write(mid);
  mySerial.write(Tlen_eid);
  mySerial.write(instr);
  mySerial.write(check_sum);
  mySerial.flush();

  mySerial.SetRxStatus(ENABLE);  //RXPin 啟用
  mySerial.SetTxStatus(DISABLE);  //TXPin 停用
  delay(13);
/***** 讀取小型數字舵機回復的資料 *****/
  rBuf[0] = mySerial.read();
  rBuf[1] = mySerial.read();
  rBuf[2] = mySerial.read();
  rBuf[3] = mySerial.read();
  delay(20);
  return  rBuf[2];
}
```

d. 設定小型數字舵機旋轉角度位置

```
uint8_t setPosTime(uint8_t EID,int16_t Position, uint16_t Moving_
Time)
// EID：模組 EID，範圍：0~15
// Position：轉動角度位置，範圍：0~140，單位 ：°
// Moving_Time：轉動時間，範圍：0~5000，單位：ms
{
  uint8_t rBuf[4]={0};
  uint16_t mid = 0x125;
  uint8_t Tlen_eid = (0x06<<4)+EID;
  uint8_t instr = 0x09;
  uint8_t DATA1 = Position;
  uint8_t DATA2 = Position >> 8;
  uint8_t DATA3 = Moving_Time;
  uint8_t DATA4 = Moving_Time >> 8;
  uint8_t check_sum = (uint8_t)~(mid + Tlen_eid + instr + DATA1 +
DATA2 + DATA3 + DATA4);
  mySerial.SetRxStatus(DISABLE);  //RXPin 停用
  mySerial.SetTxStatus(ENABLE);   //TXPin 啟用
  /***** 發送指令 *****/
  mySerial.write(mid);
  mySerial.write(Tlen_eid);
  mySerial.write(instr);
  mySerial.write(DATA1);
  mySerial.write(DATA2);
  mySerial.write(DATA3);
  mySerial.write(DATA4);
  mySerial.write(check_sum);
  mySerial.flush();
  mySerial.SetRxStatus(ENABLE);   //RXPin 啟用
  mySerial.SetTxStatus(DISABLE);  //TXPin 停用
  delay(13);
/***** 讀取小型數字舵機回復的資料 *****/
  rBuf[0] = mySerial.read();
  rBuf[1] = mySerial.read();
  rBuf[2] = mySerial.read();
  rBuf[3] = mySerial.read();
  delay(20);
  return  rBuf[2];
}
```

3. 打開序列埠監控視窗，鮑率選擇 115200，觸控按鍵，序列埠監控視窗顯示如下：