2.4G GFSK 雙向透傳模組

BMC56M001
Arduino Library V1.0.2 說明

版本：V1.11　日期：2023-12-11

www.bestmodulescorp.com

目錄

簡介

BMC56M001 是倍創推出的一款 2.4G GFSK 雙向透傳模組，使用 UART 通訊方式。本文檔對 BMC56M001 的 Arduino Lib 函式、Arduino Lib 安裝方式進行說明；範例演示了模組搭配形成 Peer 網路拓撲或 Star 網路拓撲實現配對以及無線通訊等功能。

Arduino Lib 函式

Arduino Lib 名稱：BMC56M001		Lib 版本：V1.0.2
構造函式 & 初始化		
1	BMC56M001(HardwareSerial *theSerial=&Serial)	
	描述	構造函式，使用 HW Serial 介面
	參數	*theSerial：選擇 HW Serial 介面 (預設 Serial 介面)
	返回值	—
	備註	—
2	BMC56M001(uint8_t rxPin, uint8_t txPin)	
	描述	構造函式，使用 SW Serial 介面
	參數	rxPin：RX 腳位，連接 BMC56M001 的 TX 腳位 txPin：TX 腳位，連接 BMC56M001 的 RX 腳位
	返回值	—
	備註	—
3	void begin(uint8_t baud=BR_9600)	
	描述	模組初始化
	參數	baud：通訊速率選擇 0x00 (BR_9600)：9600bps (預設) 0x01 (BR_19200)：19200bps 0x02 (BR_38400)：38400bps
	返回值	void
	備註	—
功能函式		
4	bool isPaired()	
	描述	是否配對過
	參數	—
	返回值	是否配對過狀態 TRUE：已經配對過 FALSE：沒配對過
	備註	(1) 此函數僅支援硬體 V1.01 及以上的版本號，V1.00 不支援 (2) 如果模組已經配對過，則不需要重新配對。可直接獲取通訊短位址，進行通訊。 (3) 通訊短位址可通過下面方式獲取： Peer 模式：配對成功後，使用 getShortAddress 函式進行獲取 Star 模式： Concentrator：依據上次配對順序的短位址 (0x0001~0x0005) Node：配對成功後，使用 getShortAddress 函式進行獲取

5	uint8_t writePairPackage()	
	描述	發送配對包
	參數	—
	返回值	執行情況 *
	備註	—
6	uint8_t getPairStatus()	
	描述	獲取配對狀態
	參數	—
	返回值	配對狀態 0x00：配對中 0x01：配對成功 0x02：配對失敗 0x03：配對超時
	備註	—
7	uint8_t writeRFData(uint32_t shortAddr, uint8_t len, uint8_t data[])	
	描述	發送資料封包
	參數	shortAddr：短位址 len：需要發送資料的長度 data[]：需要發送的資料
	返回值	執行情況 *
	備註	發送資料的長度不能超過 32-byte
8	bool isInfoAvailable()	
	描述	判斷是否接收到資料
	參數	—
	返回值	資料接收情況 false：沒接收到資料 true：接收到資料
	備註	—
9	uint8_t readRFData(uint8_t rxData[], uint8_t &len)	
	描述	讀取 RF 封包裡面的資料
	參數	rxData[]：儲存接收到的資料 &len：儲存接收到資料的長度
	返回值	接收到的資料情況 0x00：不是資料封包 0x01：是資料封包
	備註	rxData[] 的長度建議設為 36-byte
10	uint8_t getShortAddress()	
	描述	獲取通訊短位址
	參數	—
	返回值	通訊短位址
	備註	在配對成功後 Peer 角色或 Node 角色可獲取通訊短位址，用於發送資料封包

11	uint8_t getRSSI()	
	描述	獲取當前訊號強度
	參數	—
	返回值	當前訊號強度
	備註	—
12	uint8_t getPktRSSI()	
	描述	獲取接收封包訊號強度
	參數	—
	返回值	接收封包訊號強度
	備註	—
13	uint8_t writeEEPROM(uint8_t len, uint8_t deviInfo[])	
	描述	向 EEPROM 寫入資料
	參數	len：需要寫入資料的長度 deviInfo[]：需要寫入的資料
	返回值	執行情況 *
	備註	deviInfo[] 的長度不能超過 32-byte
14	uint8_t readEEPROM(uint8_t deviInfo[], uint8_t &len)	
	描述	讀取 EEPROM 中的資料
	參數	deviInfo[]：儲存獲取的資料 &len：儲存獲取資料的長度
	返回值	執行情況 *
	備註	deviInfo[] 的長度建議設為 32-byte
15	uint8_t getFWVer(uint8_t number[])	
	描述	獲取版本號
	參數	number[]：儲存接收到的版本號
	返回值	執行情況 *
	備註	number[] 的長度建議設為 16-byte
16	uint8_t getSN(uint8_t id[])	
	描述	獲取序列號
	參數	id[]：儲存獲取的序列號
	返回值	執行情況 *
	備註	id[] 的長度建議設為 4-byte
設定 & 讀取函式		
17	uint8_t getDeviceRole()	
	描述	獲取設備角色
	參數	—
	返回值	設備角色 0x00：Peer 角色 0x01：Node 角色 0x02：Concentrator 角色
	備註	—

18	uint8_t getMode()	
	描述	獲取工作模式
	參數	—
	返回值	工作模式 0x00：深睡眠模式 0x01：睡眠模式 0x02：RX 模式 0x03：配對模式
	備註	—
19	uint8_t getChannelPtn()	
	描述	獲取跳頻頻道
	參數	—
	返回值	跳頻頻道 0x00：跳頻組 1 0x01：跳頻組 2 ..... 0x0F：跳頻組 16
	備註	—
20	uint8_t getRFPower()	
	描述	獲取發射功率
	參數	—
	返回值	發射功率 0x00：-3dBm 0x01：0dBm 0x02：5dBm 0x03：7dBm
	備註	—
21	uint8_t getDataRate()	
	描述	獲取空中通訊速率
	參數	—
	返回值	空中通訊速率 0x00：125kbps 0x01：250kbps 0x02：500kbps
	備註	—
22	uint8_t getHoppPeriod(uint8_t period[])	
	描述	獲取跳頻週期
	參數	period[]：儲存跳頻週期參數，範圍：0x0002~0xFFFE 跳頻週期 = 8μs* 跳頻週期參數
	返回值	執行情況 *
	備註	—

23	uint8_t getBaud()	
	描述	獲取通訊速率
	參數	—
	返回值	通訊速率 0x01：9600bps 0x02：19200bps 0x03：38400bps
	備註	—
24	uint8_t setDeviceRole(uint8_t role)	
	描述	設定設備角色
	參數	role：設備角色 0x00 (Peer)：Peer 角色 0x01 (Node_of_Star)：Node 角色 0x02 (Concentrator_of_Star)：Concentrator 角色
	返回值	執行情況 *
	備註	Peer 角色的設備可倆倆之間組成 Peer 網路拓撲進行資料交流。 Concentrator 角色的設備可與多個 (最多 5 個) Node 角色的設備組成 Star 網路拓撲進行資料交流。
25	uint8_t setMode(uint8_t mode)	
	描述	設定工作模式
	參數	mode：工作模式 0x00 (DeepSleep_Mode)：深睡眠模式 0x01 (Sleep_Mode)：睡眠模式 0x02 (Rx_Mode)：RX 模式 0x03 (Pairing_Mode)：配對模式
	返回值	執行情況 *
	備註	—
26	uint8_t setChannelPtn(uint8_t channel)	
	描述	設定跳頻頻道
	參數	channel：跳頻頻道 0x00 (ChannelG_1)：跳頻組 1 0x01 (ChannelG_2)：跳頻組 2 ..... 0x0f (ChannelG_16)：跳頻組 16
	返回值	執行情況 *
	備註	—
27	uint8_t setRFPower(uint8_t power)	
	描述	設定發射功率
	參數	power：發射功率 0x00 (N3dBm)：-3dBm 0x01 (P0dBm)：0dBm 0x02 (P5dBm)：5dBm 0x03 (P7dBm)：7dBm
	返回值	執行情況 *
	備註	—

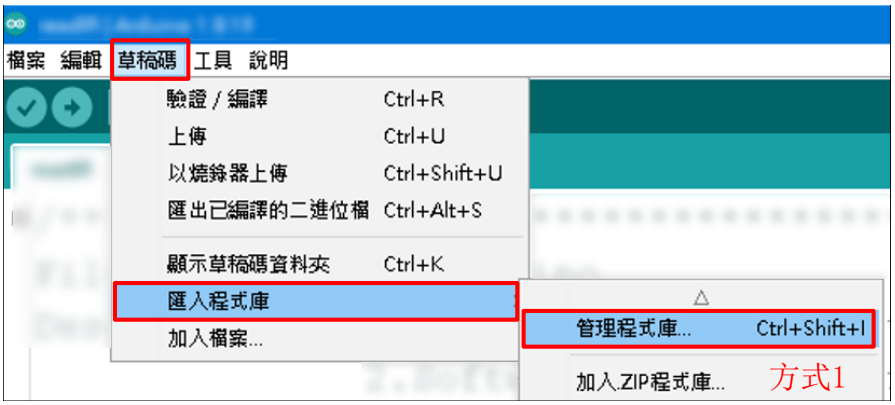
28	uint8_t setDataRate(uint8_t rate)	
	描述	設定空中通訊速率
	參數	rate：空中通訊速率 0x00 (DR125Kbps)：125kbps 0x01 (DR250Kbps)：250kbps 0x02 (DR500Kbps)：500kbps
	返回值	執行情況 *
	備註	—
29	uint8_t setHoppPeriod(uint8_t period[])	
	描述	設定跳頻週期
	參數	period[]：跳頻週期參數，範圍 0x0002~0xFFFE 跳頻週期 = 8μs* 跳頻週期參數
	返回值	執行情況 *
	備註	—
執行情況 *：0 – 指令執行成功；1 – 指令執行失敗；2 – 指令不支援；3 – 格式錯誤；4 – 資料太長；5 – 配對失敗；6 – 配對超時；7 – 發送失敗；8 – 發送成功		

Arduino Lib 下載及安裝

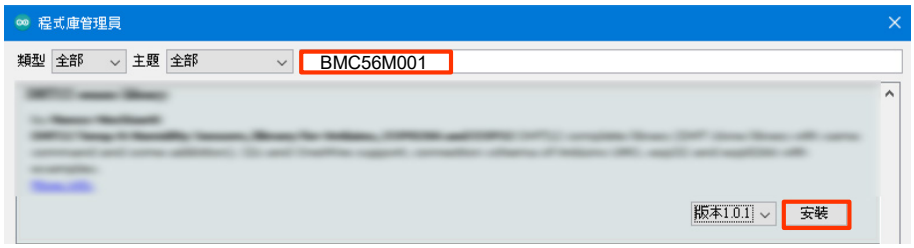
BMC56M001 Library：可參考下面兩種方法安裝 BMC56M001 的 Arduino Library

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 BMC56M001 → 安裝



搜索安裝流程 1

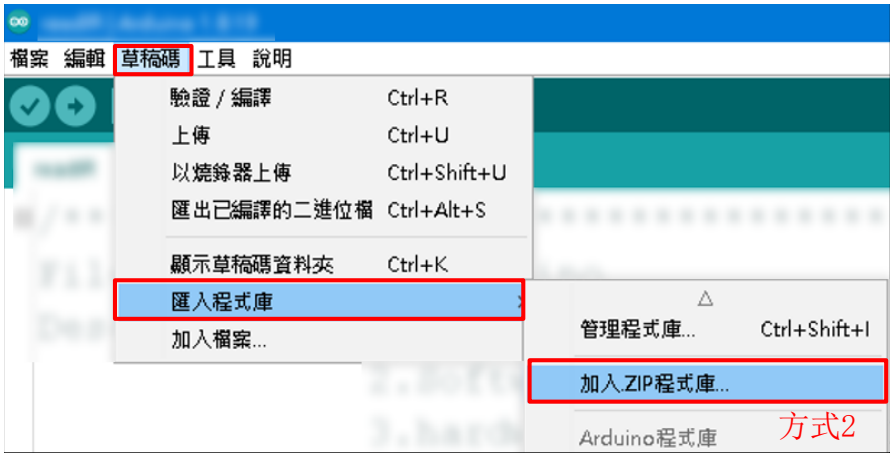


搜索安裝流程 2

方式 2：添加 .ZIP 庫，需提前下載 .ZIP 庫

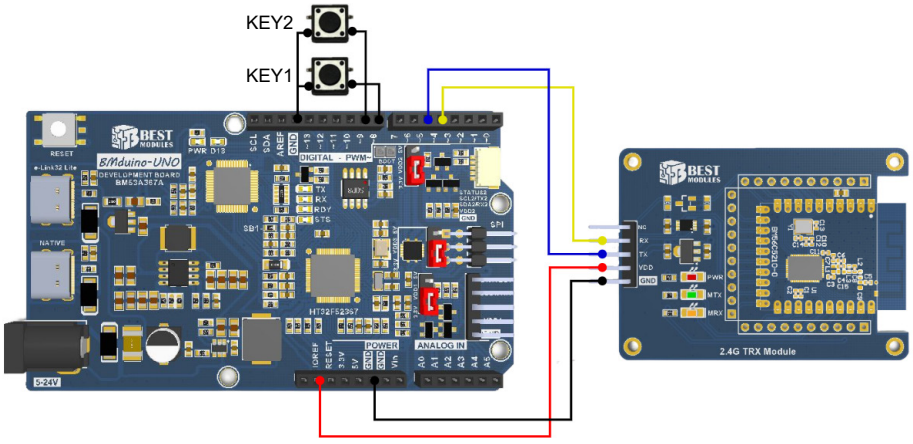
下載方法：打開倍創官方網站 (https://www.bestmodulescorp.com/bmc56m001.html)，下載“文件”菜單下的 Arduino 範例程式 (BMC56M001 Library)。

添加 .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：Peer



實物連接示意圖

範例實現功能：設定模組角色為 Peer 並且與同為 Peer 角色的模組搭配形成 Peer 網路拓撲並且進行配對以及資料交流。

兩個模組可同時運用此範例搭配形成 Peer 網路拓撲並且進行配對以及資料交流。

步驟如下：

(1). Peer 端與 Peer 端配對

兩個模組皆按下 KEY1 (D8)，進入配對模式，配對時間為 8 秒，此時 MRX 指示燈閃爍；任意一個模組，短按 KEY2 (D9) 會發送配對請求封包，配對成功 MRX 指示燈熄滅，並且在序列埠監視視窗上顯示配對成功，若 8 秒內沒有配對成功 MRX 指示燈也會熄滅，用戶可再次重新配對。

(2). 接收與發送資料切換

兩個模組配對成功後，都進入接收模式，用戶可短按任意一模組 KEY2 (D9) 即可發送資料封包。

(3). Peer 端給 Peer 端發送資料

用戶可短按任意一模組的 KEY2 (D9) 來發送資料封包，此時 MTX 指示燈會閃爍一次指示有資料封包發出。另一模組收到發送的資料時，MRX 指示燈會閃爍一次指示有接收到資料封包，並在序列埠監視視窗上顯示接收到的資料。

1. 範例打開方式：Arduino IDE → 文件 → 範例 → Lib 選擇 (BMC56M001) → 選擇範例 (Peer)
2. 範例說明：

a. 建立對象 & 模組初始化及設定

```
#include  "BMC56M001.h"
BMC56M001      BMC56(5,4);  // TX 腳位連接開發板 D5，RX 腳位連接開發板 D4
#define   KEY1_Pin  (8)      // 將 D8 與 GND 通過按鍵連接，此按鍵為 KEY1
#define   KEY2_Pin  (9)      // 將 D9 與 GND 通過按鍵連接，此按鍵為 KEY2
uint8_t   Message_ShortAddr;
bool Flag_Pairing,Flag_PairSuccess;
// 發送配對包中的資料部分
uint8_t TXDATA[16] = {0},RXDATA[32] = {0};
uint8_t DATA,STATUS,len;
/******* 函式宣告 *******/
uint8_t Sys_KEY(void);          // 獲取按鍵狀態
void RFMessage_Process();       // 根據按鍵狀態執行相關動作
void Handle_RFPacket_Process();// 獲取資料封包 & 獲取配對回復包
void setup()
{
  /***** 按鍵初始化 *****/
  pinMode(KEY1_Pin, INPUT_PULLUP);
  pinMode(KEY2_Pin, INPUT_PULLUP);
  Serial.begin(115200);         // 設定序列埠監視視窗
  BMC56.begin(BR_38400);        // 初始化模組，設定通訊速率
  BMC56.setDeviceRole(Peer);    // 選擇設備角色
}
```

b. 根據按鍵狀態執行配對、發送配對包、發送資料封包等操作，有接收到資料時獲取資料並在序列埠監視視窗上顯示

```
void loop()
{
  RFMessage_Process();          // 掃描按鍵
  Handle_RFPacket_Process();    // 掃描是否接收到資料
}
```

c. 獲取按鍵狀態函式

```
uint8_t Sys_KEY(void)
{
  if(!digitalRead(KEY1_Pin))
  {
    delay(60);
    if(!digitalRead(KEY1_Pin))
    {
      return 0x01;
    }
  }
  if(!digitalRead(KEY2_Pin))
  {
    delay(60);
    if(!digitalRead(KEY2_Pin))
    {
      return 0x02;
    }
  }
  return 0x00;
 }
```

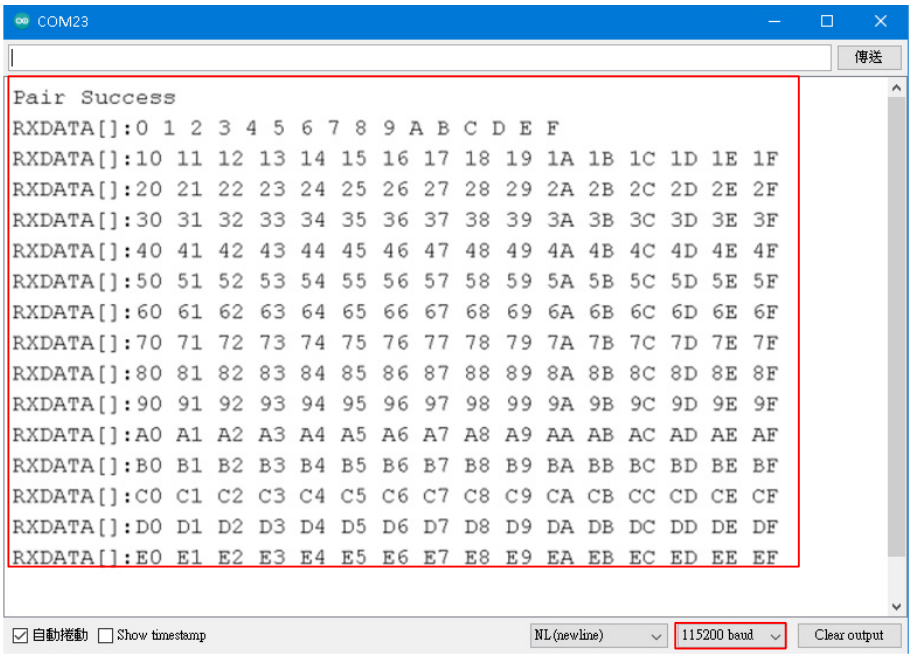
d. 根據按鍵狀態執行相關程式

```
void RFMessage_Process()
{
  switch(Sys_KEY())
  {
    case 0x01:
    /****KEY1 按鍵有按下 ****/
    BMC56.setMode(Pairing_Mode); // 進入配對模式
    Flag_Pairing = TRUE;
    Flag_PairSuccess = FALSE;
    break;
    case 0x02:
    /****KEY2 按鍵有按下 ****/
    if(Flag_Pairing)              // 判斷是否正在配對中
    {
      BMC56.writePairPackage();  // 發送配對請求包
    }
    if(Flag_PairSuccess)          // 判斷是否已配對成功
    {
      for(uint8_t temp=0;temp<16;temp++)
      {
        TXDATA[temp] = DATA++;// 發送的資料從 0x00~0x0f、0x10~0x1f……、
                              // 0xf0~0xff 循環
      }
      BMC56.writeRFData(Message_ShortAddr,16,TXDATA); // 發送資料封包
    }
    break;
  }
}
```

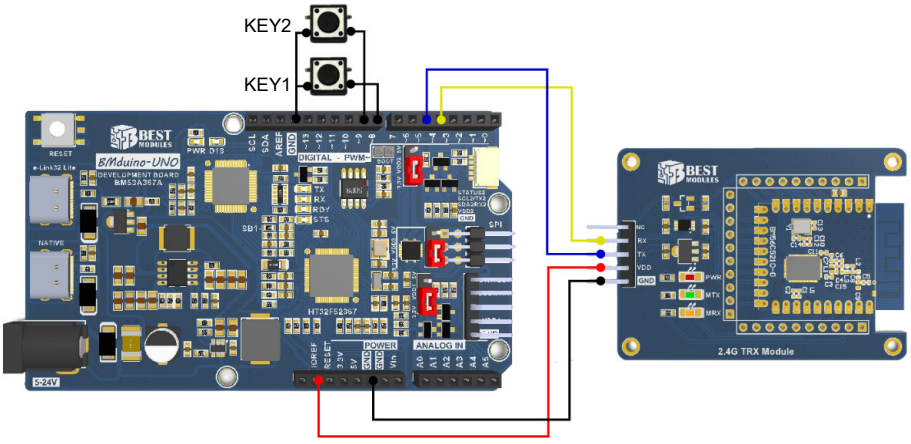
e. 接收到資料時在序列埠監視視窗上顯示

```
void Handle_RFPacket_Process()
{
  if(Flag_Pairing)                          // 判斷是否在配對中
  {
    STATUS = BMC56.getPairStatus();      // 獲取配對情況
    if(STATUS == 1) // pairing success  // 判斷是否配對成功
    {
      Flag_Pairing = FALSE;
      Flag_PairSuccess = TRUE;
      Message_ShortAddr = BMC56.getShortAddress(); // 獲取短位址
      BMC56.setMode(Rx_Mode);        // 進入 RX 模式
      Serial.println("Pair Success");// 在序列埠監視視窗上顯示 "Pair
Success"
    }
    if(STATUS == 2)      // 判斷是否配對失敗
    {
      Flag_Pairing = FALSE;
      Flag_PairSuccess = FALSE;
    }
    if(STATUS == 3)      // 判斷是否配對超時
    {
      Flag_Pairing = FALSE;
      Flag_PairSuccess = FALSE;
    }
  }
  if(Flag_PairSuccess)  // 判斷是否已經配對成功
  {
    if(BMC56.isInfoAvailable())          // 判斷是否有資料待讀取
    {
      STATUS = BMC56.readRFData(RXDATA,len);  // 讀取資料
      if(STATUS == 1)                    // 判斷讀取的資料是否為資料封包
      {
        Serial.print("RXDATA[]:");       // 序列埠監視視窗上顯示讀取的資料
        for(uint8_t temp=0;temp<len;temp++)
        {
          Serial.print(RXDATA[temp],HEX);
          Serial.print(" ");
        }
        Serial.println(" ");
      }
    }
  }
}
```

3. 打開序列埠監視視窗，鮑率選擇 115200；序列埠監視視窗顯示如下



範例 2：Node



實物連接示意圖

範例實現功能：設定模組角色為 Node 並且與 Concentrator 角色的模組搭配形成 Star 網路拓撲並且進行配對以及資料交流。

此範例需搭配範例 Concentrator 使用。

步驟如下：

(1). Node 端與 Concentrator 端配對

兩個模組皆按下 KEY1 (D8)，進入配對模式：配對時間為 8 秒，此時 MRX 指示燈閃爍；Node 端短按 KEY2 (D9) 會發送配對請求封包，配對成功 MRX 指示燈熄滅，並且在 Node 端與 Concentrator 端的序列埠監視視窗上顯示配對成功，若 8 秒內沒有配對成功 MRX 指示燈也會熄滅，用戶可再次重新配對。

(2). 接收與發送資料切換

Node 端與 Concentrator 端配對成功後，都進入接收模式，用戶可短按任意 Node 端 KEY2 (D9) 來發送資料封包給配對好的 Concentrator 端；或短按 Concentrator 端 KEY2~KEY6 (D9~D13) 來發送資料封包給配對好之不同 Node。

(3). Node 端給 Concentrator 端發送資料

用戶可短按 Node 端 KEY2 (D9) 來發送資料封包，此時 Node 端 MTX 指示燈會閃爍一次指示有資料封包發出。Concentrator 端接收到資料時 MRX 指示燈會閃爍一次指示有接收到資料封包，並在序列埠監視視窗上顯示接收到的資料。

(4). Concentrator 端給 Node 端發送資料

用戶可短按 Concentrator 端 KEY2 (D9) 來發送資料封包，此時 Concentrator 端 MTX 指示燈會閃爍一次，指示有資料封包發出。Node 端接收到資料時 MRX 指示燈會閃爍一次指示有接收到資料封包，並在序列埠監視視窗上顯示接收到的資料。若有超過一個以上 Node，可按照配對順序短按 Concentrator 端 KEY2~KEY6 (D9~D13) 分別對不同 Node 發送資料。

1. 範例打開方式：Arduino IDE → 文件 → 範例 → Lib 選擇 (BMC56M001) → 選擇 (Star) → 選擇範例 (Node)
2. 範例說明：

a. 建立對象 & 模組初始化及設定

```
#include  "BMC56M001.h"
BMC56M001      BMC56(5,4); // TX 腳位連接開發板 D5，RX 腳位連接開發板 D4
#define   KEY1_Pin (8)      // 將 D8 與 GND 通過按鍵連接，此按鍵為 KEY1
#define   KEY2_Pin (9)      // 將 D9 與 GND 通過按鍵連接，此按鍵為 KEY2
uint8_t   Message_ShortAddr;
bool Flag_Pairing,Flag_PairSuccess;
uint8_t TXDATA[16] = {0},RXDATA[32] = {0};
uint8_t DATA,STATUS,len;
/******* 函式宣告 *******/
uint8_t Sys_KEY(void);           // 獲取按鍵狀態
void RFMessage_Process();        // 根據按鍵狀態執行相關動作
void Handle_RFPacket_Process(); // 獲取資料封包 & 獲取對方設備 ID
void setup()
{
  /***** 按鍵初始化 *****/
  pinMode(KEY1_Pin, INPUT_PULLUP);
  pinMode(KEY2_Pin, INPUT_PULLUP);
  Serial.begin(115200);                 // 設定序列埠監視視窗
  BMC56.begin(BR_38400);                // 初始化模組，設定通訊速率
  BMC56.setDeviceRole(Node_of_Star); // 選擇設備角色
}
```

b. 根據按鍵狀態執行配對、發送配對包、發送資料封包等操作，有接收到資料時獲取資料並在序列埠監視視窗上顯示

```
void loop()
{
  RFMessage_Process();          // 掃描按鍵
  Handle_RFPacket_Process();    // 掃描是否接收到資料
}
```

c. 獲取按鍵狀態函式

```
uint8_t Sys_KEY(void)
{
  if(!digitalRead(KEY1_Pin))
  {
    delay(60);
    if(!digitalRead(KEY1_Pin))
    {
      return 0x01;
    }
  }
  if(!digitalRead(KEY2_Pin))
  {
    delay(60);
    if(!digitalRead(KEY2_Pin))
    {
      return 0x02;
    }
  }
  return 0x00;
}
```

d. 根據按鍵狀態執行相關程式

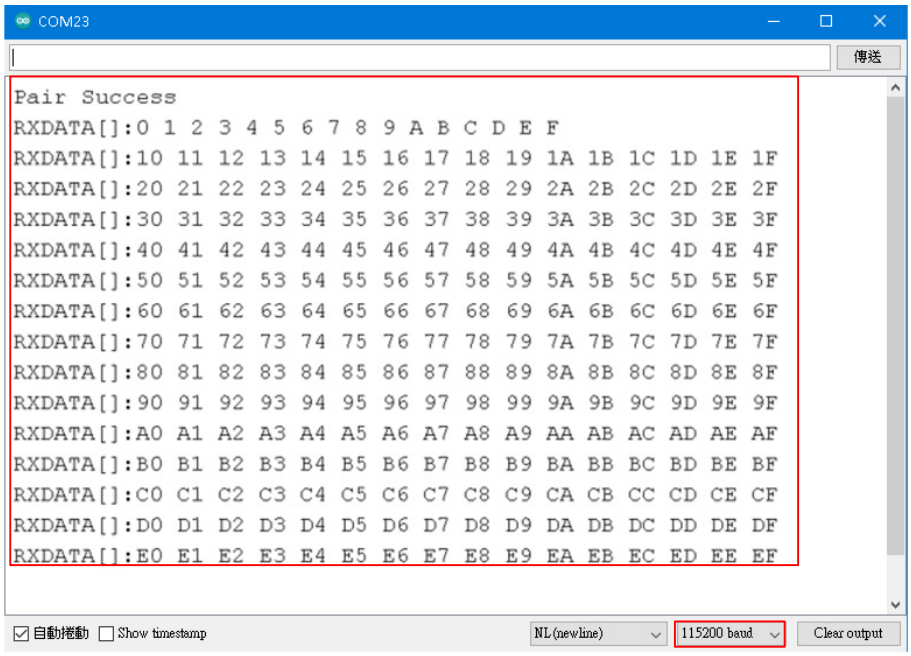
```
void RFMessage_Process()
{
  switch(Sys_KEY())
  {
    case 0x01:
    /****KEY1 按鍵有按下 ****/
    BMC56.setMode(Pairing_Mode); // 進入配對模式
    Flag_Pairing = TRUE;
    Flag_PairSuccess = FALSE;
    break;
    case 0x02:
    /****KEY2 按鍵有按下 ****/
    if(Flag_Pairing)              // 判斷是否正在配對中
    {
      BMC56.writePairPackage();  // 發送配對請求包
    }
    if(Flag_PairSuccess)          // 判斷是否已配對成功
    {
      for(uint8_t temp=0;temp<16;temp++)
      {
        TXDATA[temp] = DATA++; // 發送的資料從 0x00~0x0f、0x10~0x1f.....、
                               // 0xf0~0xff 循環
      }
```

```
      BMC56.writeRFData(Message_ShortAddr,16,TXDATA); // 發送資料封包
    }
    break;
  }
}
```

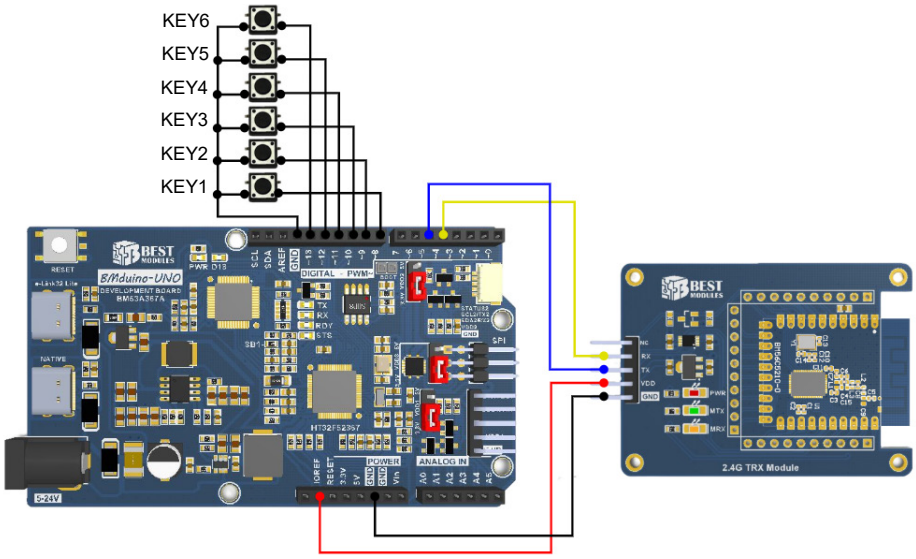
e. 接收到資料時在序列埠監視視窗上顯示

```
void Handle_RFPacket_Process()
{
  if(Flag_Pairing)                          // 判斷是否在配對中
  {
    STATUS = BMC56.getPairStatus();         // 獲取配對情況
    if(STATUS == 1) // pairing success      // 判斷是否配對成功
    {
      Flag_Pairing = FALSE;
      Flag_PairSuccess = TRUE;
      Message_ShortAddr = BMC56.getShortAddress(); // 獲取短位址
      BMC56.setMode(Rx_Mode);               // 進入 RX 模式
      Serial.println("Pair Success");// 在序列埠監視視窗上顯示 "Pair
Success"
    }
    if(STATUS == 2)                         // 判斷是否配對失敗
    {
      Flag_Pairing = FALSE;
      Flag_PairSuccess = FALSE;
    }
    if(STATUS == 3)                         // 判斷是否配對超時
    {
      Flag_Pairing = FALSE;
      Flag_PairSuccess = FALSE;
    }
  }
  if(Flag_PairSuccess)                      // 判斷是否已經配對成功
  {
    if(BMC56.isInfoAvailable())             // 判斷是否有資料待讀取
    {
      STATUS = BMC56.readRFData(RXDATA,len);        // 讀取資料
      if(STATUS == 1)                       // 判斷讀取的資料是否為資料封包
      {
        Serial.print("RXDATA[]:");   // 序列埠監視視窗上顯示讀取的資料
        for(uint8_t temp=0;temp<len;temp++)
        {
          Serial.print(RXDATA[temp],HEX);
          Serial.print(" ");
        }
        Serial.println(" ");
      }
    }
  }
}
```

3. 打開序列埠監視視窗，鮑率選擇 115200；序列埠監視視窗顯示如下



範例 3：Concentrator



實物連接示意圖

範例實現功能：設定模組角色為 Concentrator 並且與 Node 角色的模組搭配形成 Star 網路拓撲並且進行配對以及資料交流。

此範例需搭配範例 Node 使用。

步驟如下：

(1). Node 端與 Concentrator 端配對

兩個模組皆按下 KEY1 (D8)，進入配對模式：配對時間為 8 秒，此時 MRX 指示燈閃爍；Node 端短按 KEY2 (D9) 會發送配對請求封包，配對成功 MRX 指示燈熄滅，並且在 Node 端與 Concentrator 端的序列埠監視視窗上顯示配對成功，若 8 秒內沒有配對成功 MRX 指示燈也會熄滅，用戶可再次重新配對。

(2). 接收與發送資料切換

Node 端與 Concentrator 端配對成功後，都進入接收模式，用戶可短按任意 Node 端 KEY2 (D9) 來發送資料封包給配對好的 Concentrator 端；或短按 Concentrator 端 KEY2~KEY6 (D9~D13) 來發送資料封包給配對好之不同 Node。

(3). Node 端給 Concentrator 端發送資料

用戶可短按 Node 端 KEY2 (D9) 來發送資料封包，此時 Node 端 MTX 指示燈會閃爍一次指示有資料封包發出。Concentrator 端接收到資料時 MRX 指示燈會閃爍一次指示有接收到資料封包，並在序列埠監視視窗上顯示接收到的資料。

(4). Concentrator 端給 Node 端發送資料

用戶可短按 Concentrator 端 KEY2 (D9) 來發送資料封包，此時 Concentrator 端 MTX 指示燈會閃爍一次，指示有資料封包發出。Node 端接收到資料時 MRX 指示燈會閃爍一次指示有接收到資料封包，並在序列埠監視視窗上顯示接收到的資料。若有超過一個以上 Node，可按照配對順序短按 Concentrator 端 KEY2~KEY6 (D9~D13) 分別對不同 Node 發送資料。

1. 範例打開方式： Arduino IDE → 文件 → 範例 → Lib 選擇 (BMC56M001) → 選擇 (Star) → 選擇範例 (Concentrator)
2. 範例說明：

a. 建立對象 & 模組初始化及設定

```
#include  "BMC56M001.h"
BMC56M001     BMC56(5,4); // TX 腳位連接開發板 D5，RX 腳位連接開發板 D4
#define   KEY1_Pin  (8)     // 將 D8 與 GND 通過按鍵連接，此按鍵為 KEY1
#define   KEY2_Pin  (9)     // 將 D9 與 GND 通過按鍵連接，此按鍵為 KEY2
#define   KEY3_Pin  (10)    // 將 D10 與 GND 通過按鍵連接，此按鍵為 KEY3
#define   KEY4_Pin  (11)    // 將 D11 與 GND 通過按鍵連接，此按鍵為 KEY4
#define   KEY5_Pin  (12)    // 將 D12 與 GND 通過按鍵連接，此按鍵為 KEY5
#define   KEY6_Pin  (13)    // 將 D13 與 GND 通過按鍵連接，此按鍵為 KEY6
bool Flag_Pairing,Flag_PairSuccess;
uint8_t TXDATA[16] = {0},RXDATA[32] = {0};
uint8_t DATA,STATUS,len;
/******* 函式宣告 *******/
uint8_t Sys_KEY(void);           // 獲取按鍵狀態
void RFMessage_Process();        // 根據按鍵狀態執行相關動作
void Handle_RFPacket_Process(); // 獲取資料封包 & 獲取對方設備 ID
void setup()
{
  /***** 按鍵初始化 *****/
  pinMode(KEY1_Pin, INPUT_PULLUP);
  pinMode(KEY2_Pin, INPUT_PULLUP);
  pinMode(KEY3_Pin, INPUT_PULLUP);
  pinMode(KEY4_Pin, INPUT_PULLUP);
```

```
  pinMode(KEY5_Pin, INPUT_PULLUP);
  pinMode(KEY6_Pin, INPUT_PULLUP);
  Serial.begin(115200);   // 設定序列埠監視視窗
  BMC56.begin(BR_38400);  // 初始化模組，設定通訊速率
  BMC56.setDeviceRole(Concentrator_of_Star); // 選擇設備角色
}
```

b. 根據按鍵狀態執行配對、發送配對包、發送資料封包等操作，有接收到資料時獲取資料並在序列埠監視視窗上顯示

```
void loop()
{
  RFMessage_Process();        // 掃描按鍵
  Handle_RFPacket_Process(); // 掃描是否接收到資料
}
```

c. 獲取按鍵狀態函式

```
uint8_t Sys_KEY(void)
{
  if(!digitalRead(KEY1_Pin))
  {
    delay(60);
    if(!digitalRead(KEY1_Pin))
    {return 0x01;}
  }
  if(!digitalRead(KEY2_Pin))
  {
    delay(60);
    if(!digitalRead(KEY2_Pin))
    {return 0x02;}
  }
  if(!digitalRead(KEY3_Pin))
  {
    delay(60);
    if(!digitalRead(KEY3_Pin))
    {return 0x03;}
  }
  if(!digitalRead(KEY4_Pin))
  {
    delay(60);
    if(!digitalRead(KEY4_Pin))
    {return 0x04;}
  }
  if(!digitalRead(KEY5_Pin))
  {
    delay(60);
    if(!digitalRead(KEY5_Pin))
    {return 0x05;}
  }
  if(!digitalRead(KEY6_Pin))
  {
    delay(60);
    if(!digitalRead(KEY6_Pin))
    {return 0x06;}
  }
  return 0x00;
 }
```

d. 根據按鍵狀態執行相關程式

```
void RFMessage_Process()
{
  switch(Sys_KEY())
  {
    case 0x01:
    /****KEY1 按鍵有按下 ****/
    if(BMC56.setMode(Pairing_Mode))  // 進入配對模式
    {
      Flag_Pairing = TRUE;
      Flag_PairSuccess = FALSE;
    }
    break;
    case 0x02:
    /****KEY2 按鍵有按下 ****/
    if(Flag_PairSuccess)        // 判斷是否已配對成功
    {
      for(uint8_t temp=0;temp<16;temp++)
      {
        TXDATA[temp] = DATA++; // 發送的資料從 0x00~0x0f、0x10~0x1f.....、
                               // 0xf0~0xff 循環
      }
      BMC56.writeRFData(Node1_ShortAddr,16,TXDATA);// 向 Node1 發送資料封
包
    }
    break;
    case 0x03:
    /****KEY3 按鍵有按下 ****/
    if(Flag_PairSuccess)        // 判斷是否已配對成功
    {
      for(uint8_t temp=0;temp<16;temp++)
      {
        TXDATA[temp] = DATA++; // 發送的資料從 0x00~0x0f、0x10~0x1f.....、
                               // 0xf0~0xff 循環
      }
      BMC56.writeRFData(Node2_ShortAddr,16,TXDATA);// 向 Node2 發送資料封
包
    }
    break;
    case 0x04:
    /****KEY4 按鍵有按下 ****/
    if(Flag_PairSuccess)        // 判斷是否已配對成功
    {
      for(uint8_t temp=0;temp<16;temp++)
      {
        TXDATA[temp] = DATA++; // 發送的資料從 0x00~0x0f、0x10~0x1f.....、
                               // 0xf0~0xff 循環
      }
      BMC56.writeRFData(Node3_ShortAddr,16,TXDATA);// 向 Node3 發送資料封
包
    }
    break;
    case 0x05:
```

```
    /****KEY5 按鍵有按下 ****/
    if(Flag_PairSuccess)        // 判斷是否已配對成功
    {
      for(uint8_t temp=0;temp<16;temp++)
      {
        TXDATA[temp] = DATA++; // 發送的資料從 0x00~0x0f、0x10~0x1f……、
                               // 0xf0~0xff 循環
      }
      BMC56.writeRFData(Node4_ShortAddr,16,TXDATA);// 向 Node4 發送資料封
包
    }
    break;
    case 0x06:
    /****KEY6 按鍵有按下 ****/
    if(Flag_PairSuccess)        // 判斷是否已配對成功
    {
      for(uint8_t temp=0;temp<16;temp++)
      {
        TXDATA[temp] = DATA++; // 發送的資料從 0x00~0x0f、0x10~0x1f……、
                               // 0xf0~0xff 循環
      }
      BMC56.writeRFData(Node5_ShortAddr,16,TXDATA);// 向 Node5 發送資料封
包
    }
    break;
  }
}
```

e. 接收到資料時在序列埠監視視窗上顯示

```
void Handle_RFPacket_Process()
{
  if(Flag_Pairing)                    // 判斷是否在配對中
  {
    STATUS = BMC56.getPairStatus();   // 獲取配對情況
    if(STATUS == 1)                   // 判斷是否配對成功
    {
      Flag_Pairing = FALSE;
      Flag_PairSuccess = TRUE;
      BMC56.setMode(Rx_Mode);         // 進入 RX 模式
      Serial.println("Pair Success");// 在序列埠監視視窗上顯示 "Pair
Success"
    }
    if(STATUS == 2)                   // 判斷是否配對失敗
    {
      Flag_Pairing = FALSE;
      Flag_PairSuccess = FALSE;
    }
    if(STATUS == 3)                   // 判斷是否配對超時
    {
      Flag_Pairing = FALSE;
      Flag_PairSuccess = FALSE;
    }
  }
```

```
  if(Flag_PairSuccess)                  // 判斷是否已經配對成功
  {
    if(BMC56.isInfoAvailable())         // 判斷是否有資料待讀取
    {
      STATUS = BMC56.readRFData(RXDATA,len);  // 讀取資料
      if(STATUS == 1)                   // 判斷讀取的資料是否為資料封包
      {
        Serial.print("RXDATA[]:");     // 序列埠監視視窗上顯示讀取的資料
        for(uint8_t temp=0;temp<len;temp++)
        {
          Serial.print(RXDATA[temp],HEX);
          Serial.print(" ");
        }
        Serial.println(" ");
      }
    }
  }
}
```

3. 打開序列埠監視視窗，鮑率選擇 115200；序列埠監視視窗顯示如下