UART Expansion Module

BMB22M210
Arduino Library V1.0.3 Description

Revision: V1.20 Date: August 28, 2024

www.bestmodulescorp.com

Contents

Introduction

The Best Modules BMB22M210 is an UART expansion module, which uses the UART communication method. This document provides the description of the BMB22M210 Arduino Lib functions and how to install the Arduino Lib. The example 1 demonstrates the function of sub-serial port communication. The example 2 demostrates the function of controlling the servo moving using the touch keys on the BMB22T101.

Applicable types:

Part No.	Description
BMB22M210	UART expansion module
BMB22T101	Communication expansion board

Arduino Lib Functions

Arduino Lib Name: BMB22M210		Lib Version: V1.0.3
Constructor & Initialisation		
1	BMB22M210(uint8_t intPin, HardwareSerial *theSerial=&Serial)	
	Description	Constructor, uses the hardware serial port
	Parameter	intPin: INT connection pin *theSerial: Select hardware serial port
	Return Value	—
	Note	—
2	void begin(uint32_t baud)	
	Description	Module initialisation
	Parameter	baud: M-UART baud rate
	Return Value	void
	Note	Because the M-UART baud rate of the module is adaptive, initialise this module first if multiple products use the same UART port
3	void begin(uint32_t baud, uint8_t rstPin);	
	Description	Module initialisation
	Parameter	baud: M-UART baud rate rstPin: RST pin, the BMB22T101 expansion board connects to the D2
	Return Value	void
	Note	This function is only available for the BMB22T101 extension board(1)

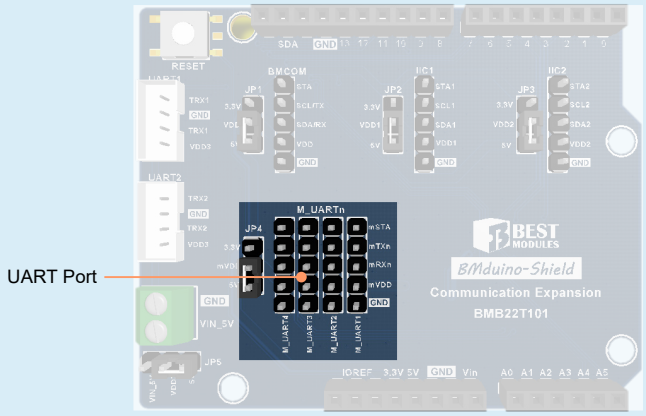
4	void beginPort(uint8_t port, uint32_t baud, uint8_t config=MYSERIAL_8N1)	
	Description	Port initialisation
	Parameter	port: Port selection 1: Sub-serial port 1 2: Sub-serial port 2 3: Sub-serial port 3 4: Sub-serial port 4 baud: Set the corresponding port baud rate(4800/9600/14400/19200/38400/57600/115200/230400) config: Configure UART format 0 (MYSERIAL_8N1): 8 bits of data, no parity, 1 stop bit 1 (MYSERIAL_8N2): 8 bits of data, no parity, 2 stop bits 8 (MYSERIAL_801): 8 bits of data, 0 parity, 1 stop bit 9 (MYSERIAL_802): 8 bits of data, 0 parity, 2 stop bits 10 (MYSERIAL_8O1): 8 bits of data, odd parity, 1 stop bit 11 (MYSERIAL_8O2): 8 bits of data, odd parity, 2 stop bits 12 (MYSERIAL_8E1): 8 bits of data, even parity, 1 stop bit 13 (MYSERIAL_8E2): 8 bits of data, even parity, 2 stop bits 14 (MYSERIAL_811): 8 bits of data, 1 parity, 1 stop bit 15 (MYSERIAL_812): 8 bits of data, 1 parity, 2 stop bits
	Return Value	—
	Note	—
Performance Functions		
5	uint8_t getINT()	
	Description	Gets INT pin level
	Parameter	—
	Return Value	INT pin voltage voltage level 1: HIGH 0: LOW
	Note	Interrupt pin. If the INTn pin status of any sub-serial port is low, the INT pin of the main port will be pulled low.
6	void setPortStatus(uint8_t port, PORTSTATUS status)	
	Description	Configure the sub-serial port pin status
	Parameter	port: Port selection status: 0 (TXRX_DISABLE): Both TX and RX are disabled 1 (TX_DISABLE): TX is disabled, RX is enabled 2 (RX_DISABLE): RX is disabled, TX is enabled 3 (TXRX_ENABLE): Both TX and RX are enabled
	Return Value	void
	Note	—
7	void write(uint8_t port, uint8_t data)	
	Description	Sub-serial port transmits 1 byte data
	Parameter	port: Select a port to transmit the data 1: Sub-serial port 1 2: Sub-serial port 2 3: Sub-serial port 3 4: Sub-serial port 4 data: The data to be transmitted
	Return Value	void
	Note	—

8	int read(uint8_t port)	
	Description	Sub-serial port reads 1 byte data
	Parameter	port: Select a port to read the data 1: Sub-serial port 1 2: Sub-serial port 2 3: Sub-serial port 3 4: Sub-serial port 4
	Return Value	Read data
	Note	When the data is null, return -1
9	void writeBytes(uint8_t port, uint8_t buff[], uint8_t num)	
	Description	Sub-serial port transmits num bytes data
	Parameter	port: Select a port to transmit the data 1: Sub-serial port 1 2: Sub-serial port 2 3: Sub-serial port 3 4: Sub-serial port 4 buff[]:The data array to be transmitted num: The data byte number to be transmitted
	Return Value	void
	Note	—
10	void readBytes(uint8_t port, int buff[], uint8_t num)	
	Description	Sub-serial port reads num bytes data
	Parameter	port: Select a port to read the data 1: Sub-serial port 1 2: Sub-serial port 2 3: Sub-serial port 3 4: Sub-serial port 4 buff[]: The read data array num: The data byte number to be read
	Return Value	void
	Note	—
11	uint8_t available(uint8_t port)	
	Description	Read the number of data in the FIFO read by the sub-serial port
	Parameter	port: Port selection 1: Sub-serial port 1 2: Sub-serial port 2 3: Sub-serial port 3 4: Sub-serial port 4
	Return Value	The number of data in the FIFO read by the sub-serial port
	Note	—
12	void resetPort(uint8_t port)	
	Description	Reset sub-serial port
	Parameter	port: Select a sub-serial port to reset 1: Sub-serial port 1 2: Sub-serial port 2 3: Sub-serial port 3 4: Sub-serial port 4
	Return Value	void
	Note	—

13	uint8_t getTxFifoNum(uint8_t port)	
	Description	Read the number of data in the FIFO sent by the sub-serial port
	Parameter	port: Port selection 1: Sub-serial port 1 2: Sub-serial port 2 3: Sub-serial port 3 4: Sub-serial port 4
	Return Value	The number of data in the FIFO sent by the sub-serial port
	Note	—
14	uint8_t getInterruptFlag(uint8_t port)	
	Description	Obtain the sub-serial port interrupt flag
	Parameter	port: Port selection 1: Sub-serial port 1 2: Sub-serial port 2 3: Sub-serial port 3 4: Sub-serial port 4
	Return Value	Bit0: Receiving FIFO trigger point interrupt(2) flag bit 0: No receiving FIFO trigger point interrupt 1: Receiving FIFO trigger point interrupt Bit1: Receiving FIFO timeout interrupt(3) flag bit 0: No receiving FIFO timeout interrupt 1: Receiving FIFO timeout interrupt Bit2: Transmitting FIFO trigger interrupt(4) flag bit 0: No transmitting FIFO trigger interrupt 1: Transmitting FIFO trigger interrupt Bit3: Transmitting FIFO empty interrupt(5) flag bit 0: No transmitting FIFO empty interrupt 1: Transmitting FIFO empty interrupt Bit4~6: Reserved Bit7: Receiving FIFO data error interrupt(6) flag bit 0: No receiving FIFO data error interrupt 1: Receiving FIFO data error interrupt
	Note	—

15	void getFifoStatus (uint8_t port)	
	Description	Obtain the sub-serial port FIFO status
	Parameter	port: Port selection 1: Sub-serial port 1 2: Sub-serial port 2 3: Sub-serial port 3 4: Sub-serial port 4
	Return Value	Bit0: Sub-serial port transmitting TX busy flag bit 0: Sub-serial port transmitting TX is empty 1: Sub-serial port transmitting TX is busy Bit1: Sub-serial port transmitting FIFO full flag bit 0: Sub-serial port transmitting FIFO is not full 1: Sub-serial port transmitting FIFO is full Bit2: Sub-serial port transmitting FIFO empty flag bit 0: Sub-serial port transmitting FIFO is empty 1: Sub-serial port transmitting FIFO is not empty Bit3: Sub-serial port receiving FIFO empty flag bit 0: Sub-serial port receiving FIFO is empty 1: Sub-serial port receiving FIFO is not empty Bit4: Sub-serial port receiving FIFO data check error flag bit 0: No PE error 1: PE error Bit5: Sub-serial port receiving FIFO data frame error flag bit 0: No FE error 1: FE error Bit6: Sub-serial port receiving FIFO data with the Line-Break error 0: No Line-Break error 1: Line-Break error (RX signal is always 0, including parity bit and stop bit) Bit7: Receiving FIFO data error interrupt(6) flag bit 0: No receiving FIFO data error interrupt 1: Receiving FIFO data error interrupt
	Note	—
16	void setTxFifoInterrupt(uint8_t port,uint8_t data)	
	Description	Set the sub-serial port transmitting FIFO trigger point
	Parameter	port: Port selection 1: Sub-serial port 1 2: Sub-serial port 2 3: Sub-serial port 3 4: Sub-serial port 4 data: Transmitting FIFO trigger point, range: 0~255
	Return Value	—
	Note	—
17	void setRxFifoInterrupt(uint8_t port,uint8_t data)	
	Description	Set the sub-serial port receiving FIFO trigger point
	Parameter	port: Port selection 1: Sub-serial port 1 2: Sub-serial port 2 3: Sub-serial port 3 4: Sub-serial port 4 data: Receiving FIFO trigger point, range:0~255
	Return Value	—
	Note	—

BMB22M210 Library can also be applied to BMB22T101 expansion board UART Port.When the BMB22T101 expansion board is directly inserted into the BMduino development board, the UART port is Serial4, the INT pin is connected to D10, and the RST pin is connected to D2.



Note 1: This function is only available for the BMB22T101 extension board. Pull low the RST pin to reset UART Port, which can re-configure the M-UART baud rate.

Note 2: When the number of data in the receiving FIFO is more than the preset transmitting FIFO trigger point, this interrupt will be generated. When the number of data in the receiving FIFO is less than the preset transmitting FIFO trigger point, this interrupt will be cleared.

Note 3 : When the number of data in the receiving FIFO is less than the preset receiving FIFO trigger point and there is no data within 4 bytes on the RX pin, this interrupt will be generated. When the data in the receiving FIFO is read or the RX pin need continue to receive data, this interrupt will be cleared.

Note 4 : When the number of data in the transmitting FIFO is less than the preset transmitting FIFO trigger point, this interrupt will be generated. When the number of data in the transmitting FIFO is more than the preset transmitting FIFO trigger point, this interrupt will be cleared.

Note 5: When there is no data in the transmitting FIFO, this interrupt will be generated. When the number of data in the transmitting FIFO is more than the preset transmitting FIFO trigger, this interrupt will be cleared.

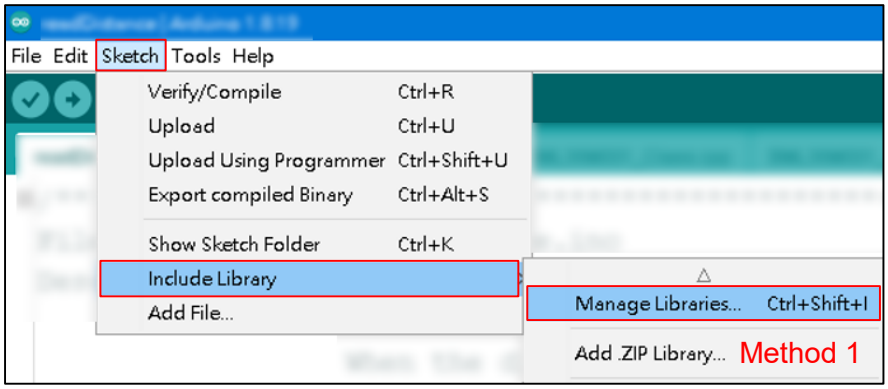
Note 6: The FIFO data error interrup indicates that there is one or more data errors in the current receiving FIFO. Conditions that generate errors include OE(data overflow error), FE(data frame error), PE(parity check error) and BE(Line-Break error). Once there is error data in the receiving FIFO, clear the corresponding interrupt after reading the FSR register.

Arduino Lib Download and Installation

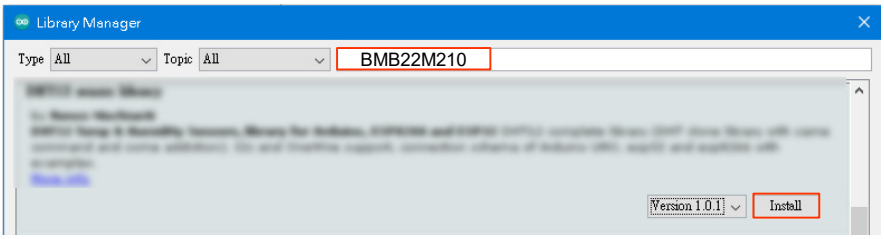
BMB22M210 Library: Refer to the following two methods to install the BMB22M210 Arduino Library.

Method 1: Search for installation

Arduino IDE → Sketch → Include Library → Manage Libraries... → Search BMB22M210 → Install



Search for Installation Step 1

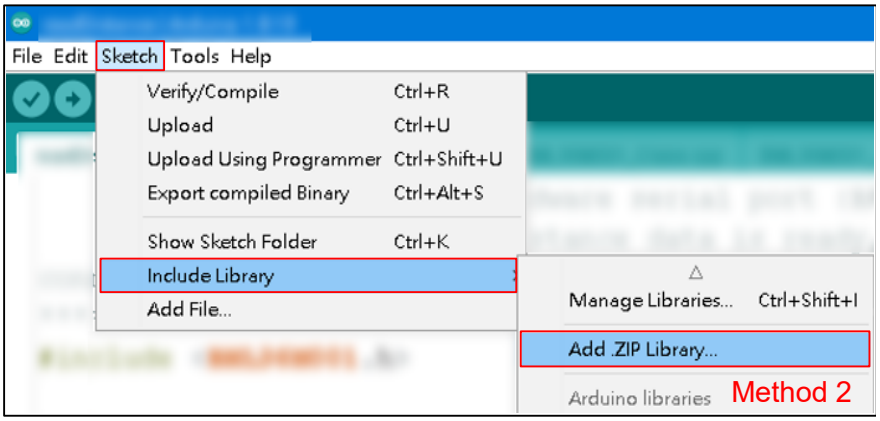


Search for Installation Step 2

Method 2: Download the .ZIP library before adding it

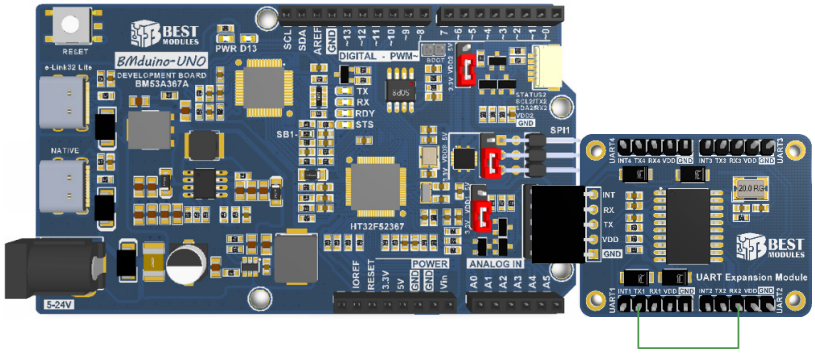
Download the Arduino example (BMB22M210 Library) under the DOCUMENTS menu from the Best Modules website (https://www.bestmodulescorp.com/ bmb22m210.html).

Add .ZIP library: Arduino IDE → Sketch → Include Library → Add .ZIP Library...



Arduino Examples

Example1: write_read



Physical Connection Diagram

Example function: The communication between the sub-serial port1 and the sub-serial port2 is that the sub-serial port1 transmits 1 byte data and the sub-serial port2 reads and displays on the serial monitor. The sub-serial port1 transmits 16 bytes data continuously, then the sub-serial port2 reads and displays on the serial monitor.

1. Open the example: Arduino IDE → File → Examples → Select Lib → Select example (write_read)

2. Example Description:

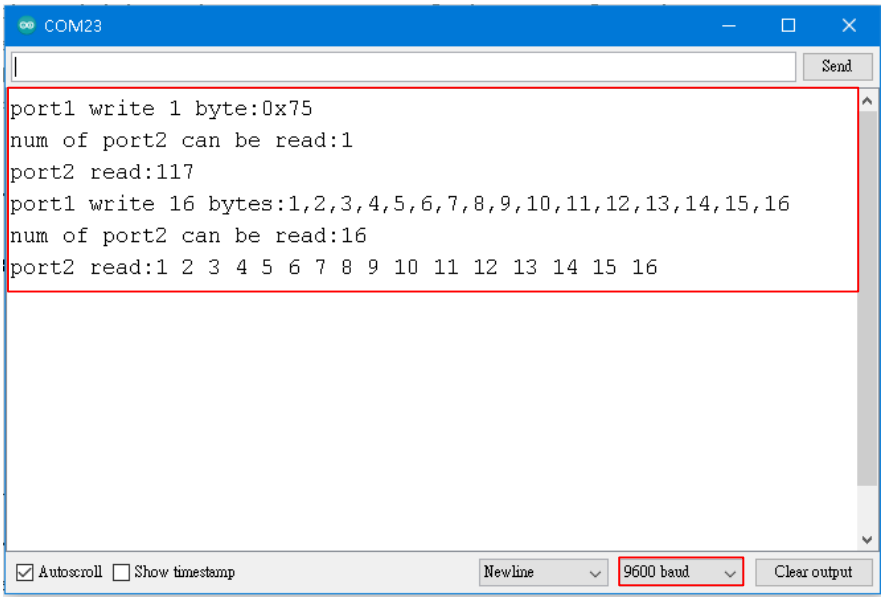
   a. Create object & Initialise and configure object

```
#include "BMB22M210.h"
BMB22M210 mySerial(22,&Serial1); // Create object
uint8_t sendbuff[16]={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16};
int receivebuff[16]={0};
uint8_t i=0;
int data = 0;
void setup()
{
  Serial.begin(9600);             // Serial monitor initialisation
  mySerial.begin(9600);           // Module initialisation and set the
                                  // main serial port baud rate
  mySerial.beginPort(1,9600);     // Configure the sub-serial
                                  // port1 baud rate
  mySerial.beginPort(2,9600);     // Configure the sub-serial
                                  // port2 baud rate
}
```

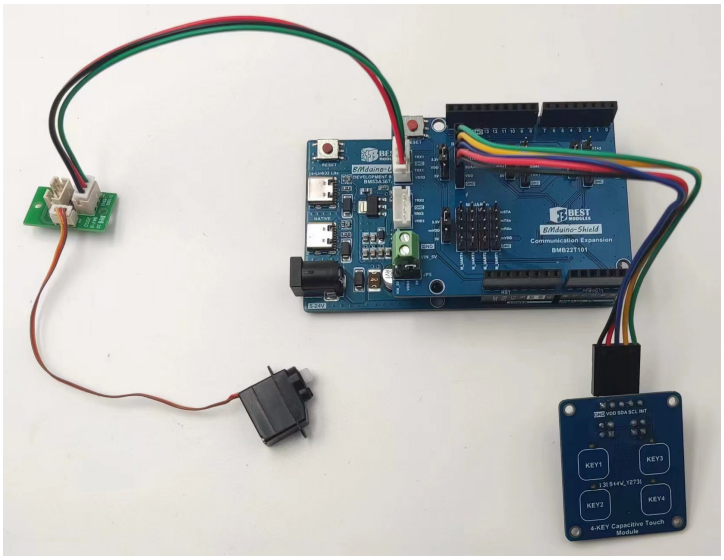
b. The sub-serial port1 transmits 1 byte data, then the sub-serial port2 reads and display on the serial monitor. The sub-serial port1 transmits 16 bytes data continuously, then the sub-serial port2 reads and display on the serial monitor.

```
void loop()
{
  Serial.println("port1 write 1 byte:0x75");
  mySerial.write(1,0x75);                // Write 1 byte data
  delay(2);
  Serial.print("num of port2 can be read:");
  Serial.println(mySerial.available(2)); // Determine the data byte
                        // that can be read by the sub-serial port2
  Serial.print("port2 read:");
  if(mySerial.available(2)>=1)
  {
    Serial.print(mySerial.read(2));      // Read 1 byte data
    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);    // Write 16 bytes data,
                                         // not exceed 16 bytes
  delay(20);
  Serial.print("num of port2 can be read:");
  Serial.println(mySerial.available(2)); // Determine the data byte
                        // that can be read by the sub-serial port2
  Serial.print("port2 read:");
  if(mySerial.available(2)>=16)
  {
    mySerial.readBytes(2,receivebuff,16); // Read 16 bytes data
                          // continuously, not exceed 16 bytes
    for(i=0;i<16;i++)
{
      Serial.print(receivebuff[i]);
      Serial.print(" ");
    }
  }
  Serial.println("");
  Serial.println("");
  delay(1000);
}
```

3. Open the serial monitor and select the baud rate to be 9600. The serial monitor will display as follows.



Example2: servoRun



Physical Connection Diagram

Example function: Connect the small digital servo(BM22O2221-A) and the 4-key touch module(BMK52M134) to the UART1 interface and the BMCOM interface in the communication shield respectively. Control the servo to move to four corresponding angles by using the four touch keys. Here, KEY1 corresponds to 0°, KEY2 corresponds to 45°, KEY3 corresponds to 90° and KEY4 corresponds to 140°.

1. Open the example: Arduino IDE → File → Examples → Select Lib → Select example (servoRun)

Note: The BMK52M134 Arduino Library should be installed before using this example.

2. Example Description:

a. Create object & Initialise and configure object

```
#include "BMK52M134.h"
#include <SoftwareSerial.h>
/***One-Wire UART Port(UART1)****/
#define TX1   6
#define RX1   7
SoftwareSerial mySerial(RX1,TX1);//Create the object and connect the
                                 //small digital servo to the UART1
                                 //interface of the communication
                                 //shield

/***BMCOM****/
BMK52M134     BMK52(3,&Wire); //Create the object and straightly
                              //connect the 4-key touch module to the
                              //BMCOM interface of the communication
                              //shield
uint8_t angle = 45; // The angle parameter used to set the moving
                    //angle of the small digital servo, unit: degree
uint8_t KEYValue = 0; //Store the touch key status

void setup()
{
  Serial.begin(115200);       //Configure the serial monitor
  BMK52.begin();            //Initialise the 4-key touch module
  mySerial.begin(115200, SERIAL_9N1);     //Initialise the small
                                          //digital servo
  setEID(1);            //Set the EID of the small digital servo to 1
  setPosTime(1,angle,0);     //Set the initial moving angle of the
                             //small digital servo to 45°
  Serial.print("Angle:");
  Serial.print(angle);         //Display the initial angle on the
                               //serial monitor
Serial.println("°");
}
```

b. According to the touch key status, set the servo to move to the corresponding angle and display the current angle on the serial monitor.

```
void loop()
{
    if(!BMK52.getINT()) //Determine whether the key is pressed
   {
    KEYValue = BMK52.getKeyValue(); //Obtain the touch key status
    switch (KEYValue)
    {
    case 1: angle = 0; break;    //If the KEY1 is pressed, set the
                                 //servo moving angle to 0
case 2: angle = 45; break;   //If the KEY2 is pressed, set the servo
                             //moving angle to 45
    case 3: angle = 90; break;   //If the KEY3 is pressed, set the
                                 //servo moving angle to 90
    case 4: angle = 140; break;  //If the KEY4 is pressed, set the
                                 //servo moving angle to 140
    }
    setPosTime(1,angle,0);//According to the touch key status, set the
                          //servo to move to the corresponding angle
```

```
    Serial.print("Angle:");
    Serial.print(angle);      //Display the current angle on the
                              //serial monitor
    Serial.println("°");
   }
}
```

c.Set the EID of the small digital servo

```
uint8_t setEID(uint8_t EID)   // EID, range: 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 is disabled
  mySerial.SetTxStatus(ENABLE);   //TXPin is enabled
  /*****Transmit instructions*****/
  mySerial.write(mid);
  mySerial.write(Tlen_eid);
  mySerial.write(instr);
  mySerial.write(check_sum);
  mySerial.flush();

  mySerial.SetRxStatus(ENABLE);  //RXPin is enabled
  mySerial.SetTxStatus(DISABLE);  //TXPin is disabled
  delay(13);
/*****Read the data responeded from the small digital servo*****/
  rBuf[0] = mySerial.read();
  rBuf[1] = mySerial.read();
  rBuf[2] = mySerial.read();
  rBuf[3] = mySerial.read();
  delay(20);
  return  rBuf[2];
}
```

d. Set the moving angle position of the small digital servo

```
uint8_t setPosTime(uint8_t EID,int16_t Position, uint16_t Moving_Time)
// EID: Module EID, range: 0~15
// Position: Moving angle position, range: 0~140, unit: °
// Moving_Time: Moving time, range: 0~5000, unit: 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 is disabled
  mySerial.SetTxStatus(ENABLE);   //TXPin is enabled
```

```
/*****Transmit instructions*****/
  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 is enabled
  mySerial.SetTxStatus(DISABLE);  //TXPin is disabled
  delay(13);
/*****Read the data responded from the small digital servo****/
  rBuf[0] = mySerial.read();
  rBuf[1] = mySerial.read();
  rBuf[2] = mySerial.read();
  rBuf[3] = mySerial.read();
  delay(20);
  return  rBuf[2];
}
```

3. Open the serial monitor, select the baud rate to be 115200 and use the touch keys. The serial monitor will display as: