LX-16A Serial Bus Servo

1. Product Introduction

1.1 Introduction

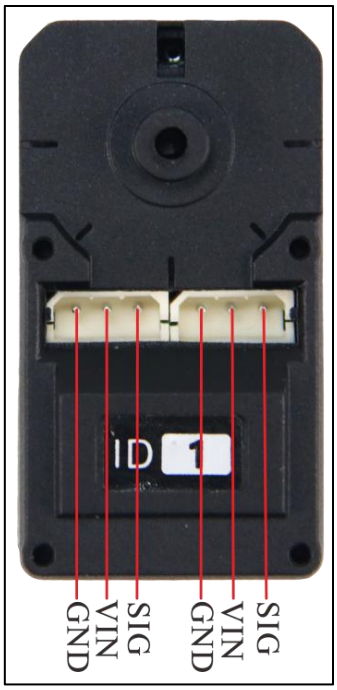
LX-16A Serial Bus Servo is controlled by serial port commands. The serial port baud rate is 115200. According to the provided communication protocol, user can send corresponding commands to servo to control servo rotation or read servo information. Servo parameters and ID are require to be set before controlling.

The interface of this servo is a half-duplex UART asynchronous serial interface so that the signal terminal can send and receive signals. When in use, we can send different commands through the serial port according to different ID to control servo individually. It is widely applicable in different bionic robot joints.

1.2 Servo Horn Installation and Port Instruction

Servo horn aims red “+” to install, please refer to the following picture.

Please refer to the following picture and form for interface distribution and instruction.



PIN	PIN Instruction
GND	GND
VIN	Power input

SIG	Signal terminal, half-duplex UART asynchronous serial interface

1.3 Servo Feature

1. Serial bus interface

The controller uses I/O port to connect the serial servo and servos are cascaded through a dual interface, which reduces the use of the serial ports. Simple wiring makes the product more exquisite and attractive.

2. ID identification and Bus communication:

Each servo can be set the ID number for the identification. The default ID number for each servo is 1 which is modifiable. The communication method of controller and servo is single-bus communication and its baud rate is 115200. User can set a corresponding ID number to each servo. The command from controller includes ID information so that only the servo matching the ID number can receive the corresponding command completely and perform actions according to the commands.

3. High-precision potentiometer:

The servo uses imported high-precision potentiometer as angle feedback. Excellent precision and linearity of the servo make robot run more table and greatly extend the service life of servo.

4. Angle readback:

LX-16A servo has angle feedback and supports angle readback. It can quickly read the angle of the servo and capture the position of robot joints, which greatly simplify the motion design of the robot. There are tick marks on the servo to facilitate the angle adjustment of the servo.

5. Temperature/Voltage Feedback/RGB Indicator：

LX-16A servo with temperature and voltage feedback can get the real-time internal data of the servo to protect the servo. There is an alarm indicator on the top of the servo. If the servo is abnormal internally, the RGB light will flash.

6. Two Operation Modes:

Support servo mode and geared motor mode.

1) Servo Mode: Able to rotate 240 degrees

2) Gear Motor Mode: Able to rotate 360 degrees, control the direction and speed.

7. Tiny Body with Unique Shell Design:

Compared to other bus servos on the market, LX-16A servo adopts tiny body with unique shell design. Compact structure make the design of robot more exquisite and do make robot with higher bionic degree.

8. Fixed with Hiwonder biaxial bracket：

Fixed with Hiwonder biaxial bracket, LX-16A servo can drive the performance of the joint of robot stably.

9. Metal Gear

The high-speed output of Internal DC motor obtains bigger torque through 5-level reduction ratio while the high precision gear inlay reduces the noise caused by gear friction.

1.4 Communication Protocol

LX-16A servo is compatible with Bus Servo Communication Protocol.

2. Servo Parameters

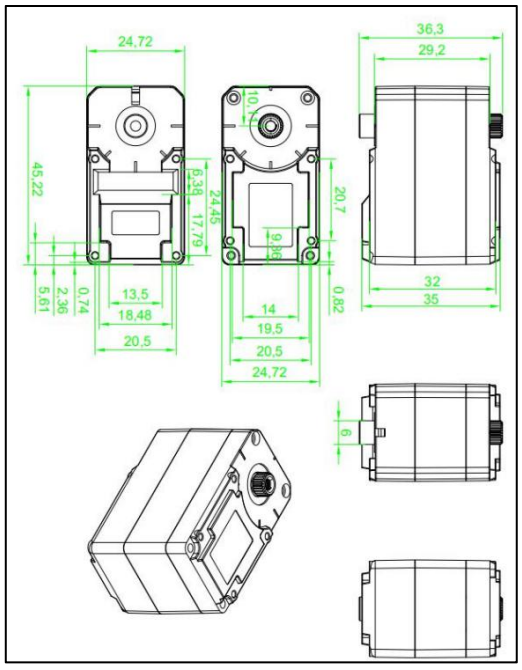
2.1 Specification

Working Voltage	DC 6-8.4V
Rotation speed	0.19sec/60°（DC 7.4V）
Stall torque	17kg.cm（DC 6V） 19.5kg.cm（DC 7.4V）
Rotation range	0°~240°
No-load current	100mA
Stall current	2.4~3A
Servo accuracy	0.3°
Control angle range	0-1000 (0°~240°)
Control method	UART serial port command
Communication	TTL
Communication baud rate	115200
Storage	Servo settings are automatically saved when power off
Servo ID	0~253 for user setting, ID 1 is default
Read back function	Support angle read back function
Protection	Avoid stalling and overheat

Date feedback	Temperature, voltage and position
Working mode	Servo mode and geared motor mode
Indicator	Blue LED
Gear type	Metal
Wire	20cm, can select other line length
Connector	5264-3P
Weight	54g
Size	45.22mm*24.72mm*36.3mm
Application	All kinds of bionic robot joints

2.2 Servo Dimension Drawing

Unit：mm



3. Project

Using servo with UNO controller to help you get quick experience. Project 1 is to set the servo ID through the UNO controller. Project 2 is to control two different servos to rotate through the UNO controller.

3.1 Getting Ready

3.1.1 Preparation

① UNO Controller *1

② LX-16A Bus Serial Servo *2

③ USB Cable *1

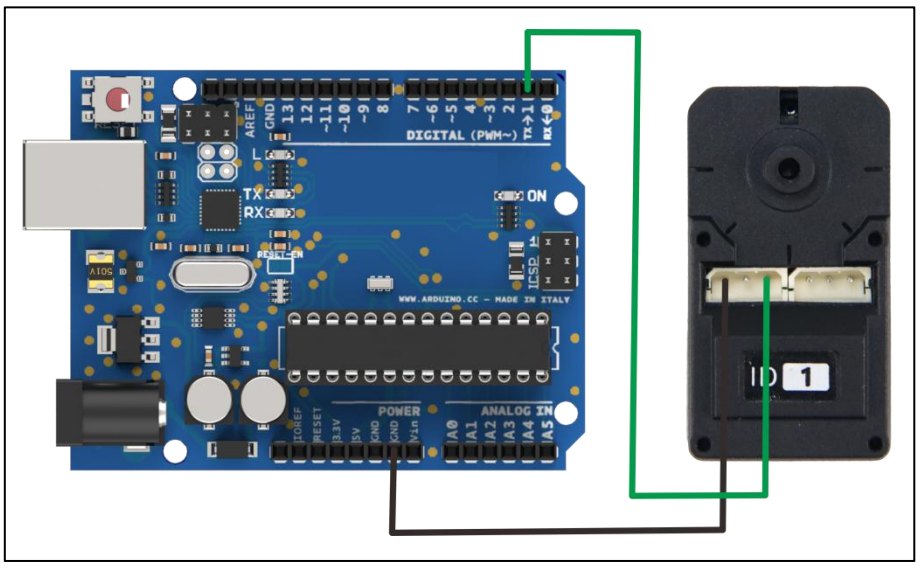
④ Male to Male Dupont line *3

⑤ Matching Wire *2

3.1.2 Wiring diagram

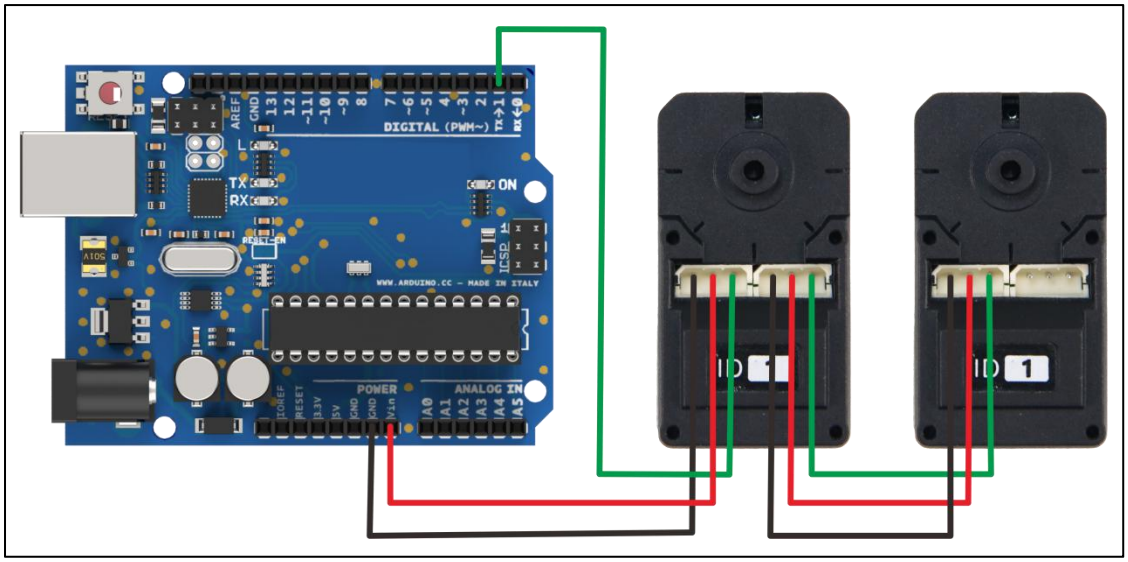
Wiring diagram for project 1 (setting ID)

Take connecting servo to UNO controller with male-to-male Dupond line as example.



wiring diagram for project 2 (servo rotation)

Take connecting two servos to UNO controller with male-to-male Dupond line as example.



3.2 Case 1-ID Setting

The default ID of LX-16A servo is 1. Take modifying ID number 1 to 2 as example.

3.2.1 Project Process

Step 1: Download and install Arduino IDE on your computer.

Step 2: Connect LX-16A bus serial servo to UNO controller as the picture shown above.

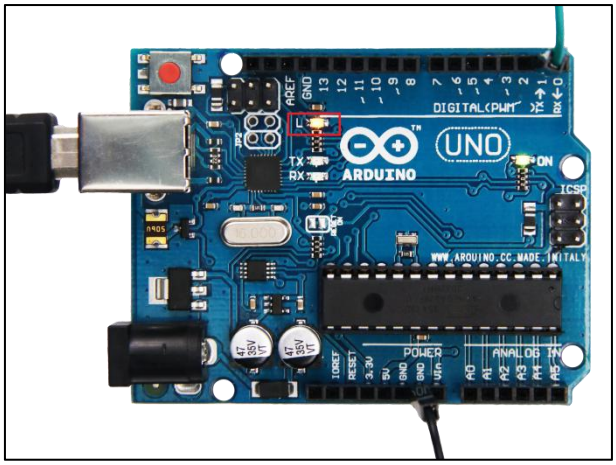
Step 3: Connect UNO controller to computer with USB cable. After opening Arduino IDE, please paste “3.5 Sample code” in “File/New”.

Step 4: Select suitable demo board and port, then compile and upload the program.

3.2.2 Project Outcome

After the code is uploaded successfully, The default ID number 1 is modified to

2 andD13 indicator on UNO board starts to blink.



3.2.3 Example Code

```
/*******LX-16A Serial Servo Testing Program 1*******

  * Arduino Type：Arduino UNO

  ************************/

#define GET_LOW_BYTE(A) (uint8_t)((A))

//Macro functions get low byte of A

#define GET_HIGH_BYTE(A) (uint8_t)((A) >> 8)

//Macro functions get high byte of A

#define BYTE_TO_HW(A, B) ((((uint16_t)(A)) << 8) | (uint8_t)(B))

// Macro functions define A as the high byte and B as the low byte, which unit to 16-bit.

  #define LOBOT_SERVO_FRAME_HEADER             0x55

#define LOBOT_SERVO_ID_WRITE                   13

byte LobotCheckSum(byte buf[])

{
```

```
    byte i;
    uint16_t temp = 0;
    for (i = 2; i < buf[3] + 2; i++) {
        temp += buf[i];
    }
    temp = ~temp;
    i = (byte)temp;
    return i;
}
void LobotSerialServoSetID(HardwareSerial &SerialX, uint8_t oldID, uint8_t newID)
{
    byte buf[7];
    buf[0] = buf[1] = LOBOT_SERVO_FRAME_HEADER;
    buf[2] = oldID;
    buf[3] = 4;
    buf[4] = LOBOT_SERVO_ID_WRITE;
    buf[5] = newID;
    buf[6] = LobotCheckSum(buf);
    SerialX.write(buf, 7);
}
void setup() {
    // put your setup code here, to run once:
```

```
    Serial.begin(115200); //baud rate115200
    pinMode(13, OUTPUT);
    delay(1000);
}
void loop() {
    // put your main code here, to run repeatedly:
    delay(500);
    digitalWrite(13,HIGH);    //Indicator, operating indicate
  LobotSerialServoSetID(Serial, 1, 2); // The first parameter is the serial port for communication, the second parameter is the old ID.
 (The old ID 1 is to broadcast this command to all online servos, valid for all online servos)
     // The third parameter is the new ID.
    delay(500);
    digitalWrite(13,LOW);
}
```

3.3 Case 2-Servo Rotation

Control two servos with different ID to rotate in case 2.

3.3.1 Project Process

1) Refer to 3.1.2 Wiring Diagram for project 2 to connect two LX-16A bus serial servos to UNO controller.
2) Refer to 3.2.1 Project Process to upload “3.3.3 Example Code” to UNO controller.

3.3.2 Project Outcome

After the code is uploaded successfully, the angle of servo ID1 and servo ID2 keep changing every 1 second. The angle change of servo ID1 is significantly larger than that of servo ID2.

3.3.3 Example Code

```
/*******LX-16A Serial Servo Testing Program 2*******
  * Arduino Type：Arduino UNO
  ************************/
#define GET_LOW_BYTE(A) (uint8_t)((A))
//Macro functions get low byte of A
#define GET_HIGH_BYTE(A) (uint8_t)((A) >> 8)
//Macro functions get high byte of A
#define BYTE_TO_HW(A, B) ((((uint16_t)(A)) << 8) | (uint8_t)(B))
//Macro functions define A as the high byte and B as the low byte, which unit to 16-bit.
#define LOBOT_SERVO_FRAME_HEADER         0x55
#define LOBOT_SERVO_MOVE_TIME_WRITE      1
byte LobotCheckSum(byte buf[])
{
   byte i;
   uint16_t temp = 0;
   for (i = 2; i < buf[3] + 2; i++) {
      temp += buf[i];
```

```
        }
        temp = ~temp;
        i = (byte)temp;
        return i;
    }
    void LobotSerialServoMove(HardwareSerial &SerialX, uint8_t id, int16_t position, uint16_t time)
    {
        byte buf[10];
        if(position < 0)
            position = 0;
        if(position > 1000)
            position = 1000;
        buf[0] = buf[1] = LOBOT_SERVO_FRAME_HEADER;
        buf[2] = id;
        buf[3] = 7;
        buf[4] = LOBOT_SERVO_MOVE_TIME_WRITE;
        buf[5] = GET_LOW_BYTE(position);
        buf[6] = GET_HIGH_BYTE(position);
        buf[7] = GET_LOW_BYTE(time);
        buf[8] = GET_HIGH_BYTE(time);
        buf[9] = LobotCheckSum(buf);
```

```
    SerialX.write(buf, 10);
}
void setup() {
    // put your setup code here, to run once:
    Serial.begin(115200);
    delay(1000);
}
#define ID1      1
#define ID2      2
void loop() {
    // put your main code here, to run repeatedly:
    LobotSerialServoMove(Serial, ID1, 100, 500);
    LobotSerialServoMove(Serial, ID2, 500, 500);
    delay(1000);
    LobotSerialServoMove(Serial, ID1, 500, 500);
    LobotSerialServoMove(Serial, ID2, 600, 500);
    delay(1000);
    LobotSerialServoMove(Serial, ID1, 900, 500);
    LobotSerialServoMove(Serial, ID2, 700, 500);
    delay(1000);
    LobotSerialServoMove(Serial, ID1, 500, 500);
    LobotSerialServoMove(Serial, ID2, 600, 500);
```

```
    delay(1000);

}
```

4. Q&A

Q1: Why servo failed to display after uploading?

A: Please check the wiring. Signal terminal of the servo needs to be connected to D1 port of the controller.

Q2: How many servos can be cascaded?

A: Theoretically, up to 253 servos can be cascaded at the same time.

Q3: How should we supply the UNO controller power when test case 2?

A: We connect black DC plug to a external power as well as supply UNO controller and servo power in case 2. The power voltage of UNO controller ranges from DC 7 to 12V and the working voltage of UNO controller ranges from DC 6 to 8.4V. Therefore, the provided power voltage is required to meet two demands above so that the voltage should range from DC 7 to 8.4V.

Q4：Why the two servos perform the same outcome when test case 2?

A：The default ID number of two servos is 1. Please follow the method of case 1 to modify one of the servo ID number to number 2. When test case 1, please note that:

1. Please use the dupont wire with suitable length.
2. After wiring and uploading successfully, D13 on the UNO controller starts to blink. Please do not disconnect the wire until the servo ID is modified successfully.