Shenzhen Hiwonder Technology Co,Ltd

# HTD-35H Serial Bus Servo


## 1. Product Introduction

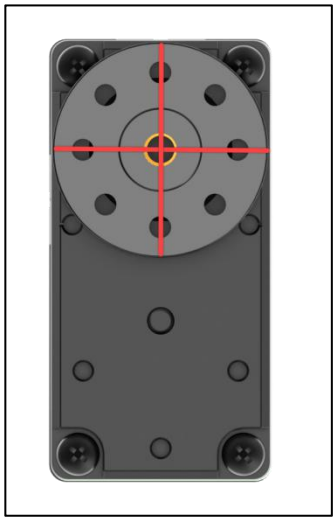
### 1.1 Instruction

HTD-35H 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 required 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 to different ID through the serial port, which controls servo individually. It is widely applicable in different robotic arm joints.

### 1.2 Servo Horn Installation and Port Instruction

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



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


| PIN | PIN Instruction |
|---|---|
| GND | GND |
| VIN | Power input |
| SIG | Signal terminal, half-duplex UART asynchronous serial interface |

## 1.3 Servo Feature

1) High voltage servo, more power saving:

Compared with the traditional 7.4V servo, the 11.1V high-voltage servo can reduce the current by more than 60%, which greatly improves the battery life of the robot.

2) Serial bus interface:

There is an I/O port on the controller board for connecting to the serial bus servo. The servos are connected through the three connectors, which makes the project have clean wiring and beautiful outlook.

3) 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 then perform actions according to the commands.

4) 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.

5) Strong Torque

35KG strong torque builds up your robot.

6) Position, temperature and voltage feedback:

With position, temperature and voltage feedback, you can get the internal data of the servo in real time to protect the servo.

7) Two Operation Modes:

Support servo mode and geared motor mode.

Under servo mode, servo can rotate to the specific angle within 240 ° .

Under geared motor mode, servo can rotate within 360° and the direction and speed of rotation can be controlled.

8) Metal Gear:

The high-precision inlay of the gears reduces the noise caused by the friction.

9) Metal Shell:

Green oxidation metal shell improves heat dissipation ability.

### 1.4 Communication Protocol

HTS-35H servo is compatible with Bus Servo Communication Protocol. You can view the specific protocol under the same directory.

## 2. Servo Parameter

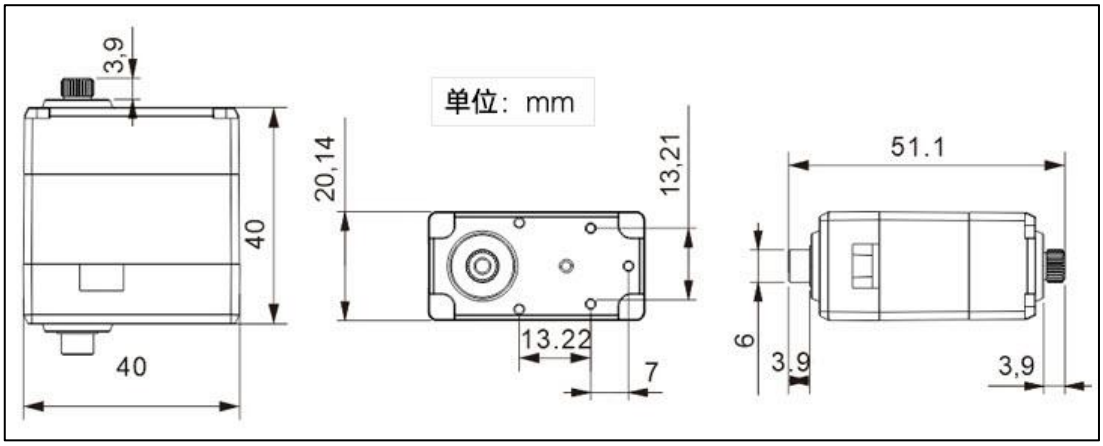
### 2.1 Specification

| | |
|---|---|
| Working voltage | DC 9-12.6V |
| Rotation speed | 0.18sec/60°（DC 11.1V） |
| Rotation torque | 35kg.cm（DC 11.1V） |
| Static maximum torque | 35kg.cm（DC 11.1V） |

| | |
|---|---|
| Rotation range | 0~ 240° |
| No-load current | 100mA |
| Stall current | 3A |
| Servo accuracy | 0.2° |
| Control angle range | 0-1000 (0°~240°) |
| Control method | UART serial command |
| 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 |
| Parameter Feedback | Temperature, voltage and position |
| Working mode | Servo mode and deceleration motor mode |
| Gear type | Metal Gear |
| Servo Wire | 20cm, other line length can be selected |
| Plug-in model | PH2.0-3P |
| Weight | 64g |

| | |
|---|---|
| Size | 54.38mm*20.14mm*45.5mm |
| Application | All kinds of bionic robot joints |

## 2.2 Servo Dimension Drawing




# 3. Project

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

## 3.1 Getting Ready

### 3.1.1 Hardware Preparation

① Arduino UNO Board *1

② HTS-35H Bus Servo *2

③ USB Cable *1

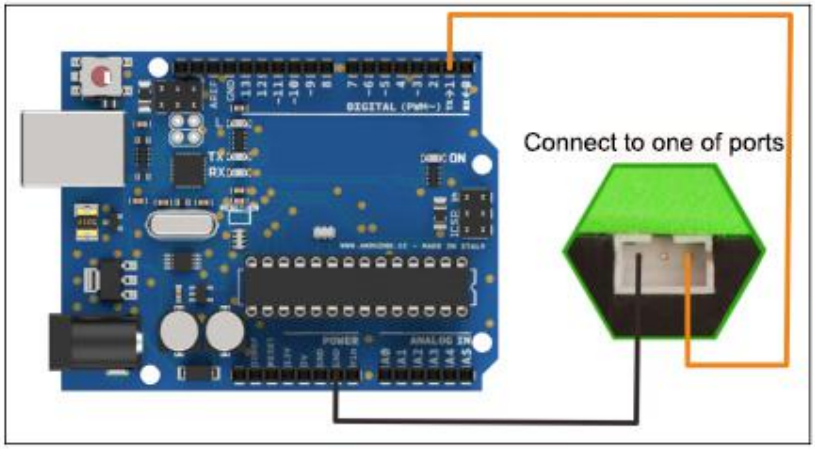
④ Male to Male Dupont Line *3

⑤ Servo Wire *2

## 3.1.2 Wiring Preparation

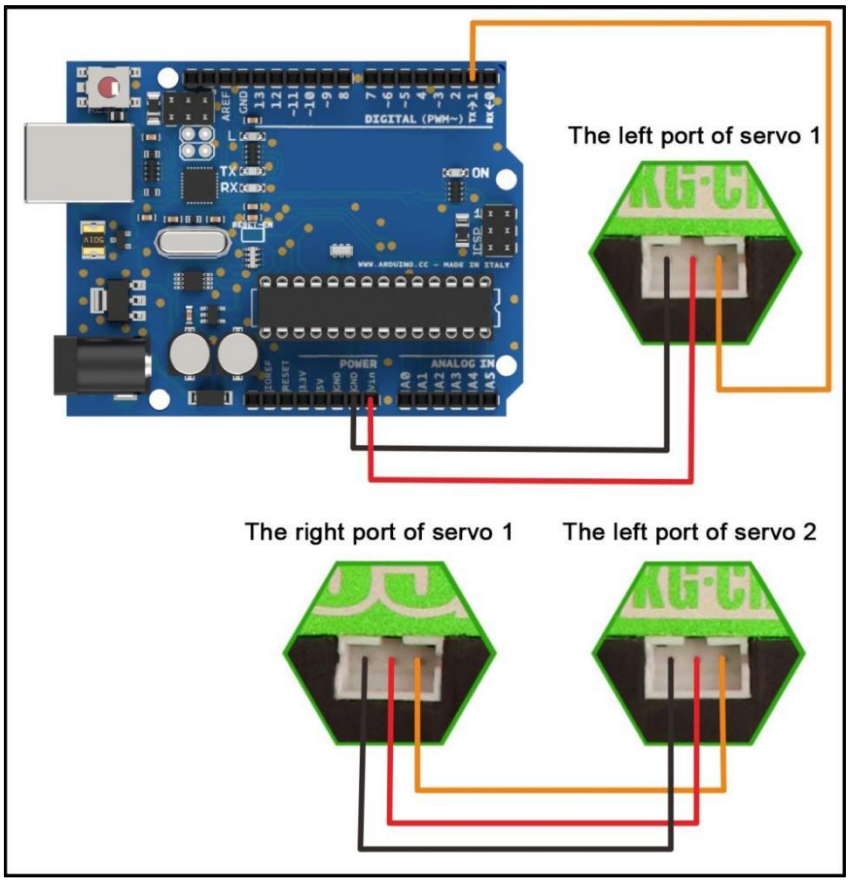
Wiring diagram for project 1(ID setting)

Connect a servo to UNO board through male to male dupont line as follow:




Wiring diagram for project 2 (servo rotation)

Connect two servos to UNO board through male to male Dupont line as follow:



## 3.2 Project 1 - ID setting

The default ID of HTS-35H servo is 1. Take modifying ID number 1 to 2 as an example.

### 3.2.1 Project Process

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

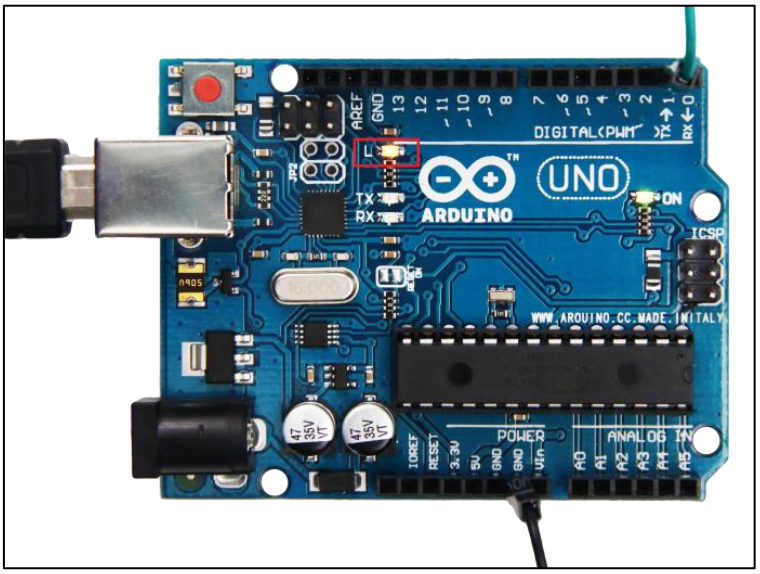
Step 2: Connect HTD-35H bus serial servo to UNO board as Wiring diagram for project 1.

Step3: Connect UNO board to your computer through USB cable. After opening Arduino IDE, please paste “3.3.2 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 and the D13 indicator on UNO board starts to flash.



### 3.2.3 Sample Code

```
/*******HTD-35H serial port servo test program 1*******

  * Arduino Model：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 rate 115200

    pinMode(13, OUTPUT);

    delay(1000);

}

void loop() {

    // put your main code here, to run repeatedly:

    delay(500);

    digitalWrite(13,HIGH);    //Indicator, running indication
```

```
        LobotSerialServoSetID(Serial, 1, 2); // The first parameter is serial port for communication and the second parameter is old ID
      // ID The third parameter is new ID
        delay(500);
        digitalWrite(13,LOW);
    }
```

## 3.3 Project 2 - Servo Rotation

In project 2, control two servos with different ID to rotate.

### 3.3.1 Project Process

1) Refer to 3.1.2 Wiring Diagram for project 2 to connect two HTD-35H bus servos to UNO board.
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 angles 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 Sample Code

```
/*******HTD-35H serial port servo test program 2*******
  * Arduino Model：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 perform after uploading?

A:Please check the wiring first to confirm whether the servo signal end is connected to the 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 I power the UNO board when testing project 2?

A: We can power the UNO board and servo through DC power port. The power voltage

The power supply voltage of UNO board ranges from DC 7 to 12V. The working voltage of the servo ranges from DC 9 to 12.6V. The power supply needs to meet two requirements, so the voltage range of the external power supply we need to provide is DC 9-12V.

Q4: Why the two servos perform the same outcome when testing project 2?

A: The default ID number of two servos is 1. Please follow the method of project 1 to modify one of the servo ID to number 2.

When test case 1, please note that:

1) 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.