Preface

About SunFounder

SunFounder is a technology company focused on Raspberry Pi and Arduino open source community development. Committed to the promotion of open source culture, we strive to bring the fun of electronics making to people all around the world and enable everyone to be a maker.

Our products include learning kits, development boards, robots, sensor modules and development tools. In addition to high quality products, SunFounder also offers video tutorials to help you make your own project. If you have interest in open source or making something cool, welcome to join us!

About This Kit

This cute learning kit focuses on the popular open source platform Arduino. You can learn the knowledge of the Arduino servo and ultrasonic ranging module by applying this kit.

In this book, we will show you how to build the biped robot via description, illustrations of physical components, in both hardware and software respects. You may visit our website www.sunfounder.com to download the related code and view the user manual on **LEARN -> Get Tutorials** and watch related videos under **VIDEO**.

Free Support

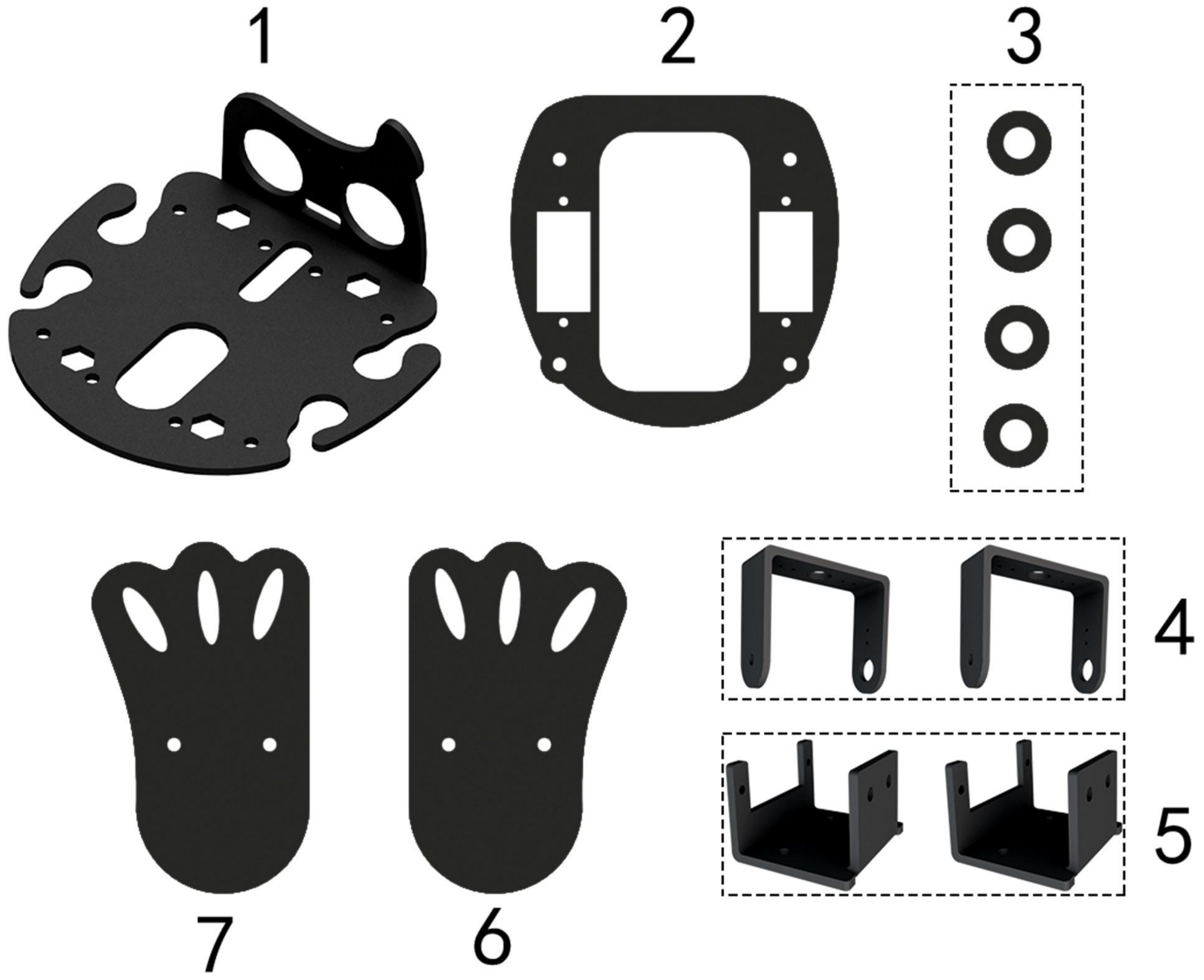
If you have any **TECHNICAL questions**, add a topic under **FORUM** section on our website and we'll reply as soon as possible.

For **NON-TECH questions** like order and shipment issues, please **send an email to service@sunfounder.com**. You're also welcomed to share your projects on FORUM.

Content

Components

i. Structural Plate



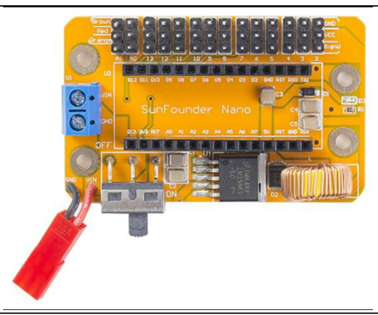
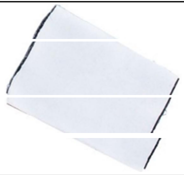
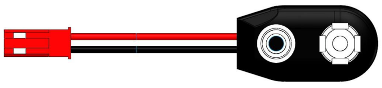
ii. Mechanical Fasteners

Parts	Name	Qty.

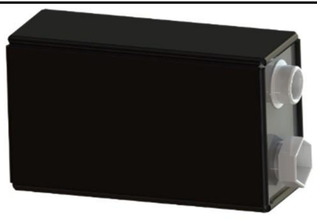
	M1.5*5 Self-tapping Screw	10
	M1.4*8 Screw	6
	M2*8 Screw	12
	M3*5 Screw	18
	M3*8 Countersunk Screw	8
	M1.4 Nut	6
	M2 Nut	12
	M3 Self-locking Nut	8
	M3*8 Bi-pass Copper Standoff	4
	M3*25 Bi-pass Copper Standoff	4

iii. Electrical Components

Parts	Name	Qty.
	SF006C Servo	4
	Ultrasonic Module	1
	SunFounder Nano Board	1

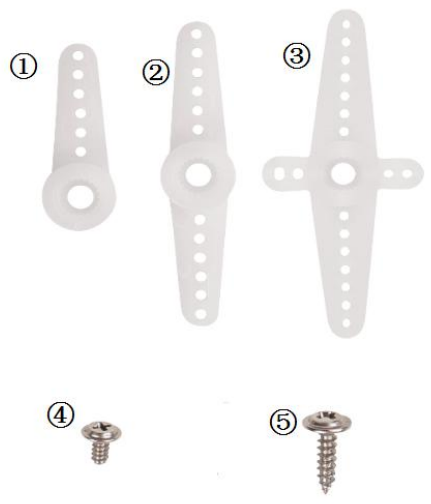
	Expansion board	1
	Velcro Tape	1
	Mini USB Cable	1
	Battery buckle wire	1
	4-Pin Anti-reverse Cable	1
	Phillips Screw Driver	1
	Cross Socket Wrench	1

iv. Battery（Not Included）

Parts	Name	Qty.
	9V battery	1

v. Servo Accessories

The steering gear package is divided into the following sections



Note：

Please check the above list and structural plate drawings against the product you received to check for missing components. If the above two situations occur, please take pictures of all the components you have received and inform us of the missing parts，and send E-mail to service@sunfounder.com

Introduction

This cute learning kit focuses on the popular open source platform Arduino. You can learn the knowledge of the Arduino servo and ultrasonic ranging module by applying this kit.

It is a new mobile robot called Sloth developed by SunFounder. Each leg has 2 joints driven by servo. One 9V chargeable lithium batteries are to supply the bot when the SunFounder Nano is used as the control board, compatible with the Arduino Nano. A servo control board connects with the batteries, servos, SunFounder Nano, and the HC-SR04 ultrasonic ranging module. Sloth can move forward and detect the range to make a turn when encountering an obstacle. In addition, when learning to program, you can also have the fun to build a pretty cool bio-robot by yourself.

Getting Started

Note:

Before starting your own project, you must download the file **DIY 4-DOF Robot Kit - Sloth.zip** on our official website by visiting LEARN -> Get Tutorials -> **DIY 4-DOF Robot Kit - Sloth** and unzip it.

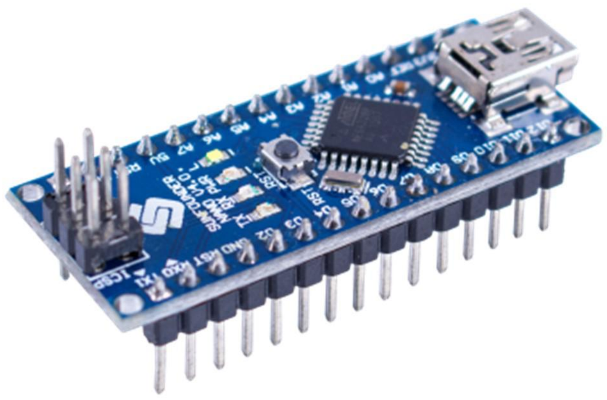
Arduino

Arduino is an open source platform that applies simple software and hardware. You can get it in a short even when you know little of it. It provides an integrated development environment (IDE) for code editing and compiling, compatible with multiple control boards. So you can just download the Arduino IDE, upload the sketches (i.e. the code files) to the board, and then you can see experimental phenomena. For more information, refer to http://www.arduino.cc.

Arduino Board – SunFounder Compatible

Arduino senses the environment by receiving inputs from many sensors, and affects its surroundings by controlling lights, motors, and other actuators.

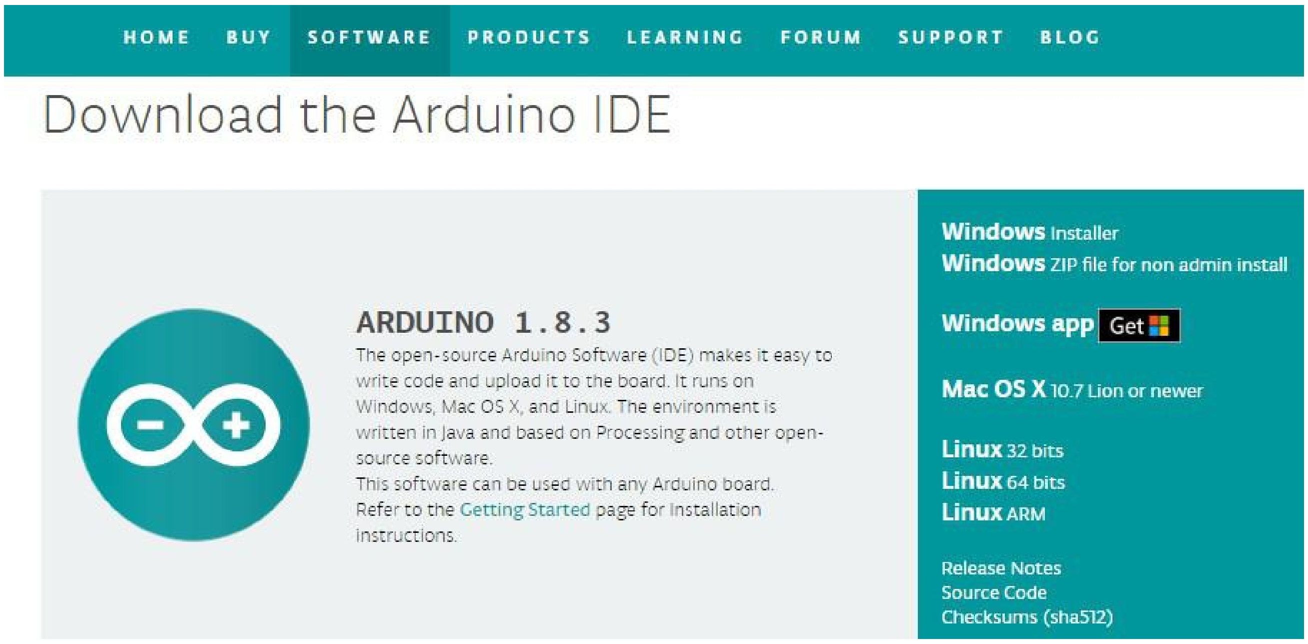
In this kit, SunFounder Nano board is used.



Install Arduino IDE

The code in this kit is written based on Arduino, so you need to install the IDE first. Skip it if you have done this.

Now go to the arduino.cc website and click **DOWNLOAD**. On the page, check the software list on the right side under **Download the Arduino Software**.



Download the Arduino IDE

ARDUINO 1.8.3

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.
This software can be used with any Arduino board.
Refer to the Getting Started page for Installation instructions.

Windows Installer
Windows ZIP file for non admin install

Windows app Get

Mac OS X 10.7 Lion or newer

Linux 32 bits
Linux 64 bits
Linux ARM

Release Notes
Source Code
Checksums (sha512)

Find the one that suits your operation system and click to download. There are two versions of Arduino for Windows: **Installer** or **ZIP file**. You're recommended to download the former. Just download the package, and run the executable file to start installation. It will download the driver needed to run Arduino IDE. After downloading, follow the prompts to install. For the details of installing steps, you can refer to the guide on **Learning**->Getting Started with Arduino, scroll down and see Install the Arduino Software.

After installing, you will see Arduino icon on your desk and double click to open it.

Install the Driver

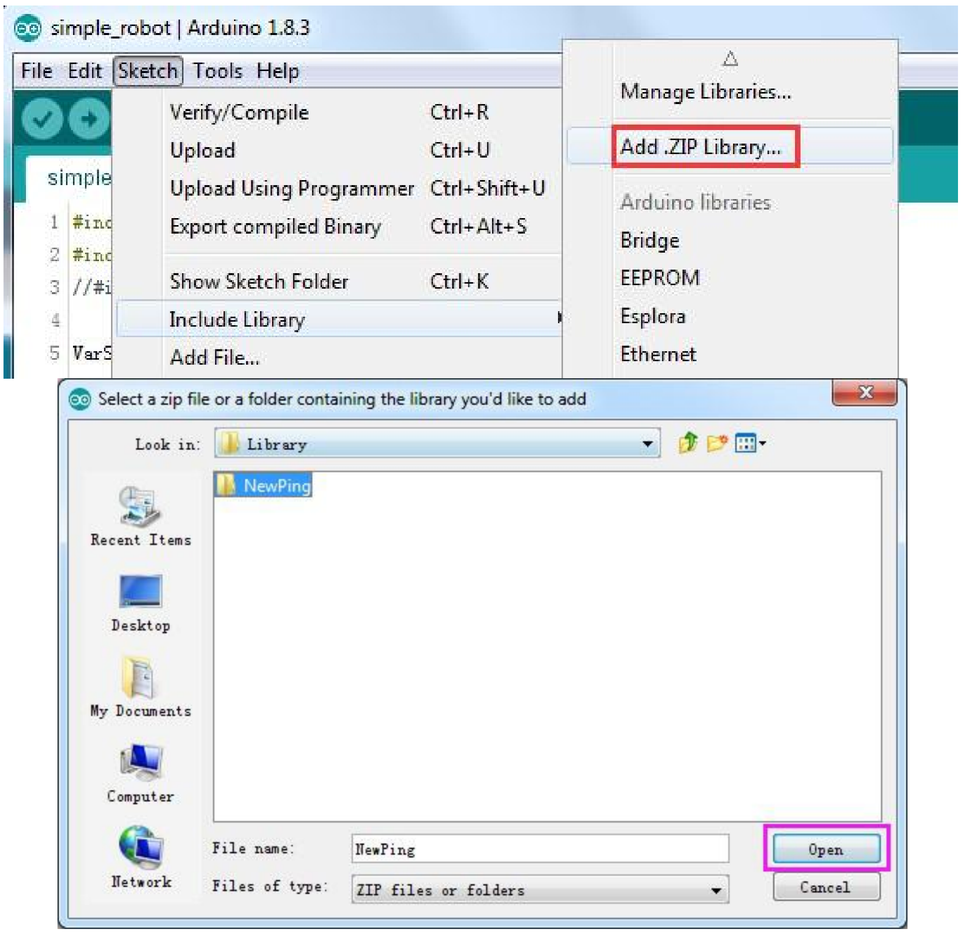
If the driver is not installed, the Nano board will not be able to be recognized by your computer. Therefore, before using it, please install appropriate driver.

For Windows users, run ***PL2303_Prolific_DriverInstaller_v1180B*** in the folder Driver.

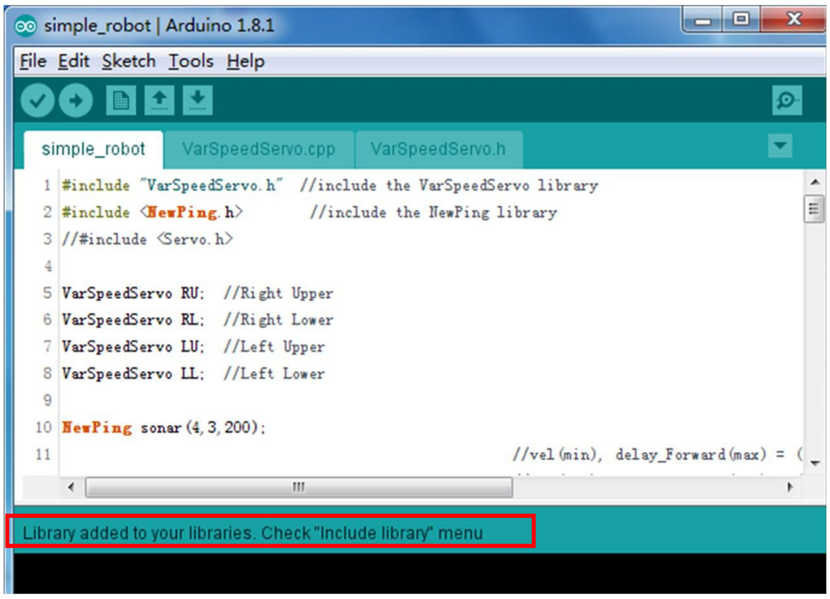
For Mac users, refer to the folder ***PL2303_MacOSX_1_6_1_20170620*** in the folder Driver.

Add Libraries

Libraries are a collection of code that makes it easy for you to connect to a sensor, display, module, etc. In this kit, you will need to include **NewPing library** under path ***DIY 4-DOF Robot Kit - Sloth\Library*** to **Arduino/libraries**.


simple_robot | Arduino 1.8.3
File Edit Sketch Tools Help
Verify/Compile Ctrl+R
Upload Ctrl+U
Upload Using Programmer Ctrl+Shift+U
Export compiled Binary Ctrl+Alt+S
Show Sketch Folder Ctrl+K
Include Library
Add File...
Manage Libraries...
Add .ZIP Library...
Arduino libraries
Bridge
EEPROM
Esplora
Ethernet
Select a zip file or a folder containing the library you'd like to add
Look in: Library
NewPing
Recent Items
Desktop
My Documents
Computer
Network
File name: NewPing
Files of type: ZIP files or folders
Open
Cancel

Then you will see "Library added to your libraries", indicating the library has been included successfully.



```
#include "VarSpeedServo.h"  //include the VarSpeedServo library
#include <NewPing.h>        //include the NewPing library
//#include <Servo.h>

VarSpeedServo RU;  //Right Upper
VarSpeedServo RL;  //Right Lower
VarSpeedServo LU;  //Left Upper
VarSpeedServo LL;  //Left Lower

NewPing sonar(4,3,200);
                                                          //vel(min), delay_Forward(max) = (
```

Library added to your libraries. Check "Include library" menu

Test for Servos and Ultrasonic Module

Before assembling, you need to test the servos and the ultrasonic module according to the following steps.

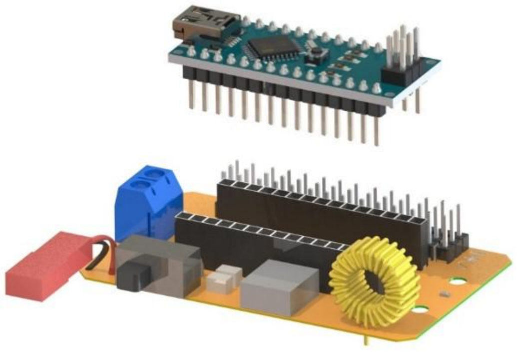
i. Servo Test

1. Physical Test

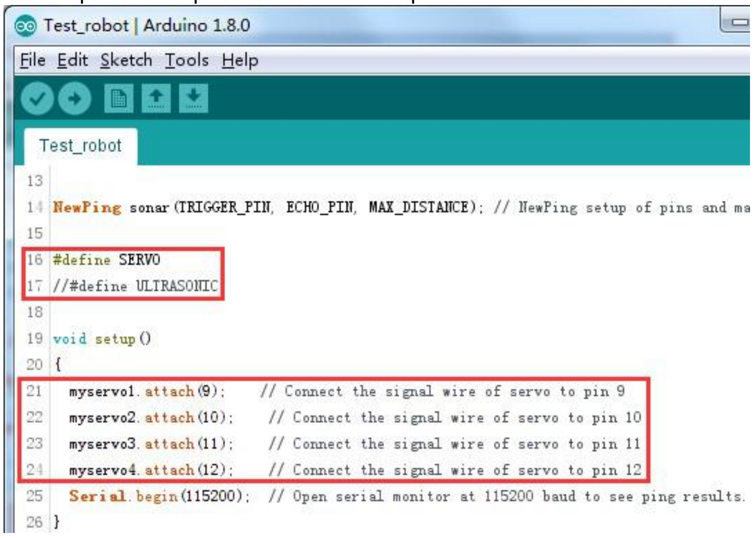
Please prepare four servos and four unilateral rocker arms and install the rocker arm on the servo. If you shake the rocker arm by hand, if the sound is inconsistent and the rotation is not smooth, the servo may be damaged. It is recommended that the following software tests be performed. If the servo does not work properly, we believe that the servo has been damaged.

2. Software Test

Insert SunFounder Nano board into the Servo Control Board. Note: The USB port should be at the same side with blue power supply terminal.



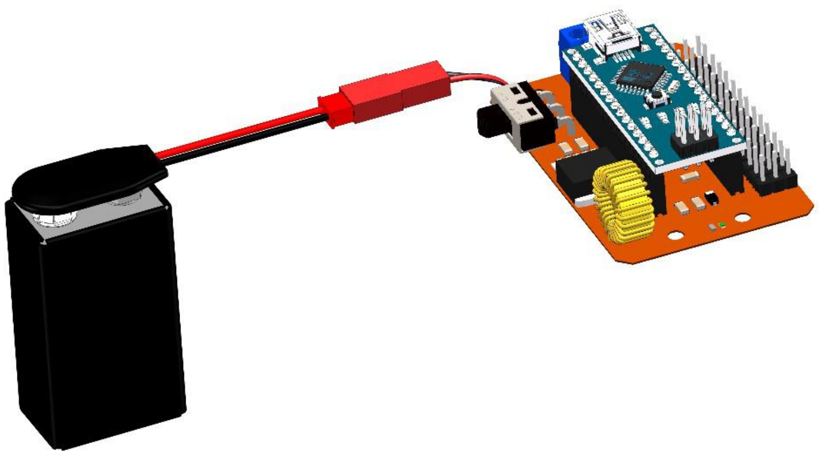
Connect the Nano board to computer with a USB cable, you can see a notification in system tray. Then open the program **Test_robot.ino and go to line 21**. In the code, four servos are connected to pin 9 to pin 12 of the expansion board.



```
13
14 NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE); // NewPing setup of pins and ma
15
16 #define SERVO
17 //#define ULTRASONIC
18
19 void setup()
20 {
21   myservo1.attach(9);    // Connect the signal wire of servo to pin 9
22   myservo2.attach(10);    // Connect the signal wire of servo to pin 10
23   myservo3.attach(11);    // Connect the signal wire of servo to pin 11
24   myservo4.attach(12);    // Connect the signal wire of servo to pin 12
25   Serial.begin(115200);   // Open serial monitor at 115200 baud to see ping results.
26 }
```

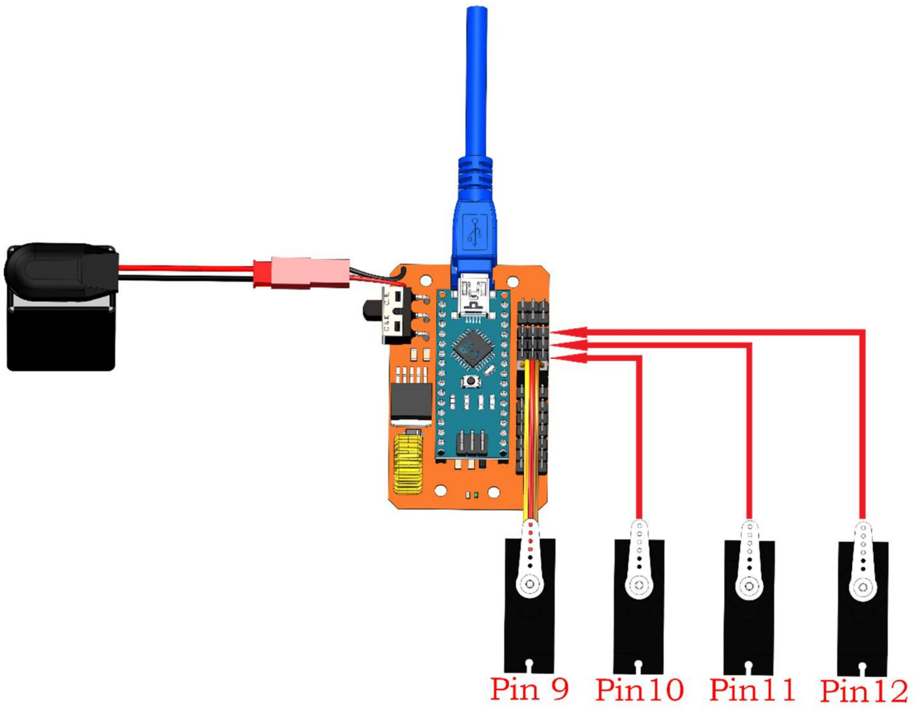
Insert the battery to the battery cable.

And connect the battery cable to the expansion board.

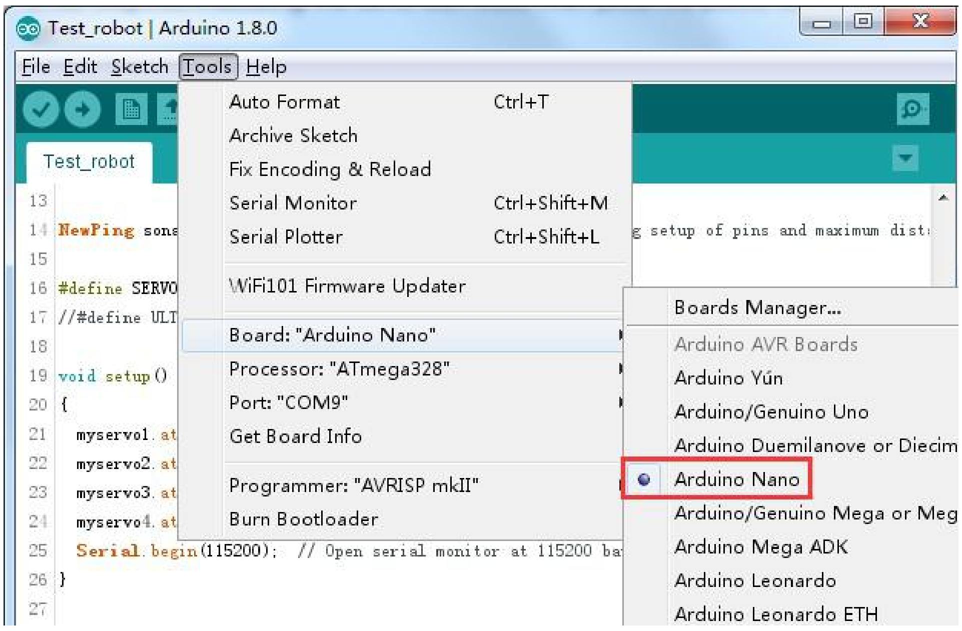


Connect four servos to pin 9 to pin 12 of the expansion board.

Note: The yellow, **red**, and **brown** wires connect to **Signal**, **VCC**, and **GND** on the expansion board, respectively.

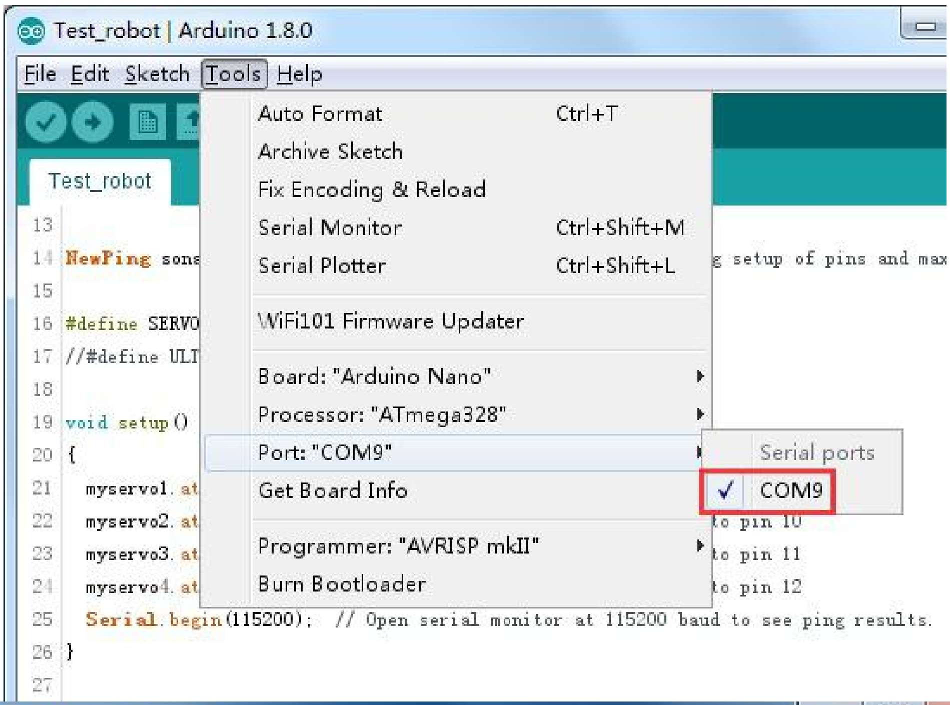


Select the right **board** to Nano board in IDE.



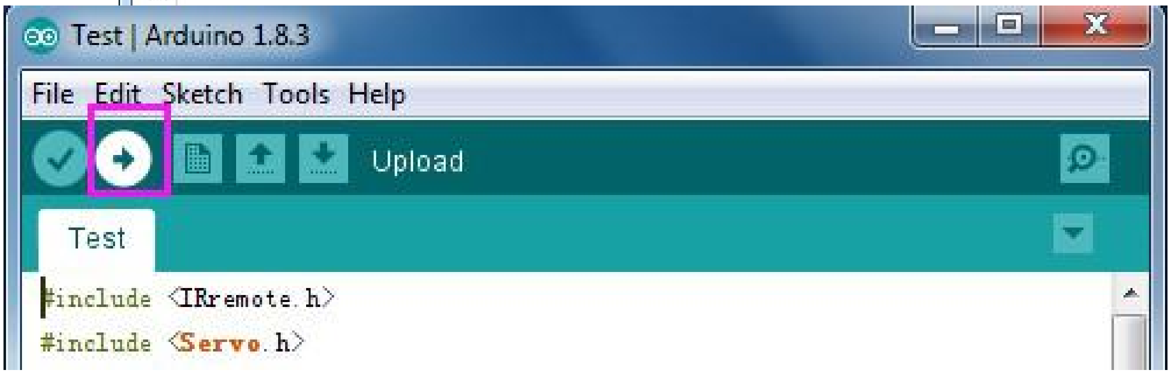
```
NewPing sona                                    g setup of pins and maximum dist

#define SERVO
//#define ULT

void setup()
{
  myservo1.at
  myservo2.at
  myservo3.at
  myservo4.at
  Serial.begin(115200);  // Open serial monitor at 115200 ba
}
```

Then select the right **port** and upload the sketch.



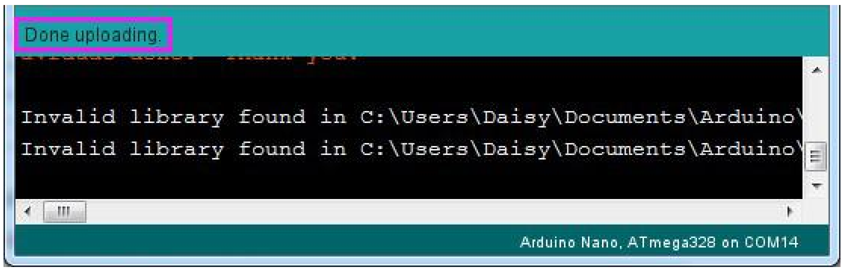
Board: "Arduino Nano"
Processor: "ATmega328"
Port: "COM9"
✓ COM9
Programmer: "AVRISP mkII"

```
NewPing sona                                       g setup of pins and max

#define SERVO
//#define ULT

void setup()
{
  myservo1.at
  myservo2.at                                      to pin 10
  myservo3.at                                      to pin 11
  myservo4.at                                      to pin 12
  Serial.begin(115200);  // Open serial monitor at 115200 baud to see ping results.
}
```



```
#include <IRremote.h>
#include <Servo.h>
```

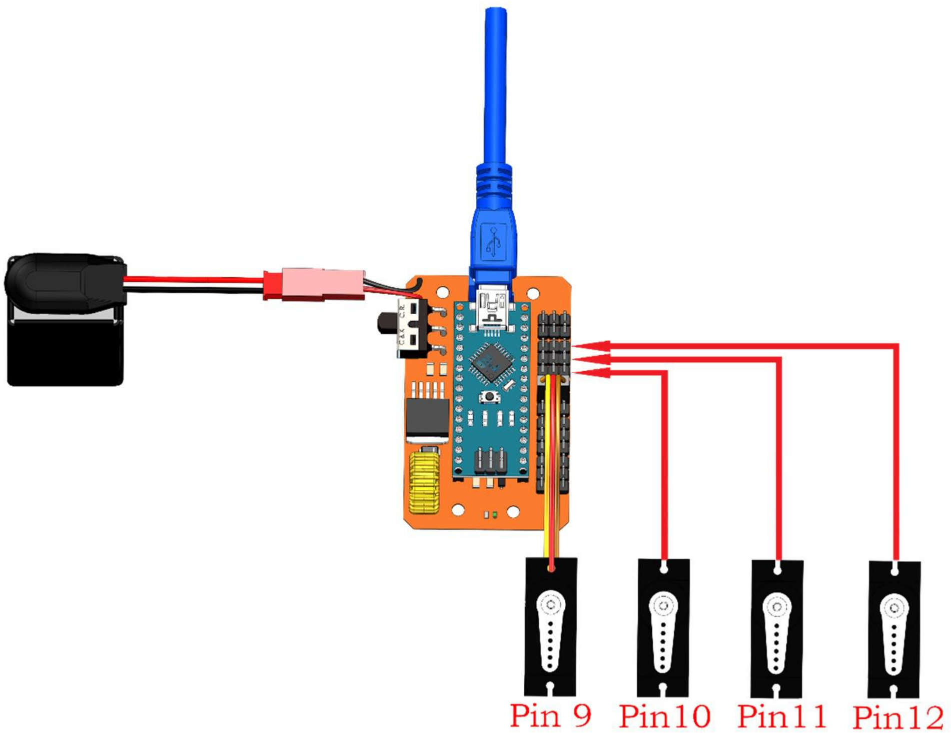
After waiting for a few seconds, the download process is successful. The following window will prompt "**Done uploading**".



```
Invalid library found in C:\Users\Daisy\Documents\Arduino\
Invalid library found in C:\Users\Daisy\Documents\Arduino\
```

Arduino Nano, ATmega328 on COM14

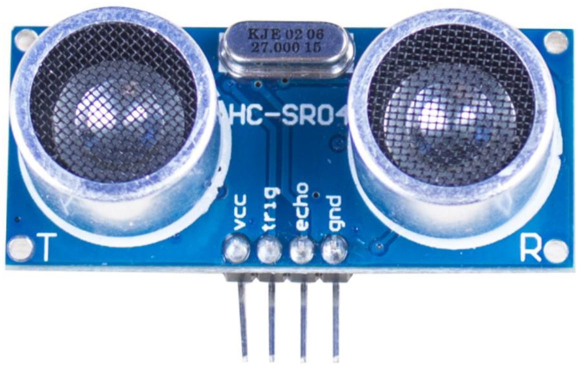
Press the power button on the servo control board. You will see the rocker arm rotates within 0-180 degrees, indicating the servo can work. Then press the power button again to turn it off.



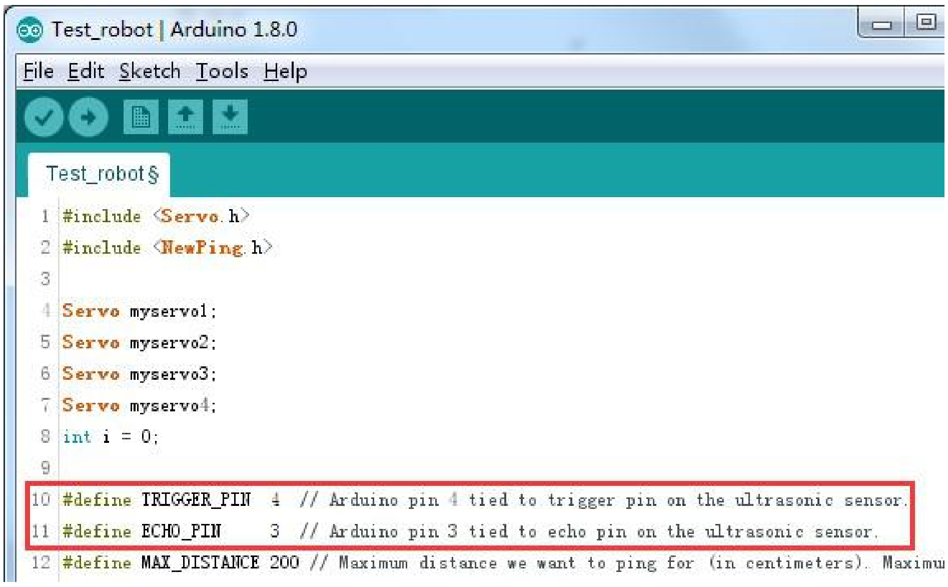
Keep the **Test_robot.ino** file open, and **ii. Ultrasionic Test** still needs to use this file in the next step.

ii. Ultrasonic Test

Connect the ultrasonic connecting cable to the ultrasonic module, it has an anti-reversing port. Here the 4 pins are marked with labels on the modules.

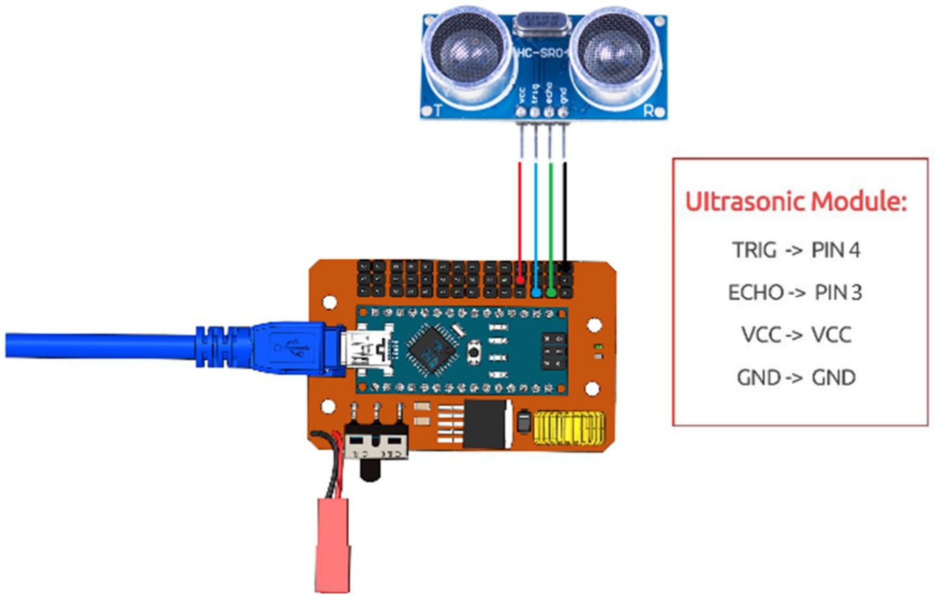


Go to line 10 and 11, you can see that **TRIGGER_PIN** is defined as **5** and **ECHO_PIN** is defined as **4**.

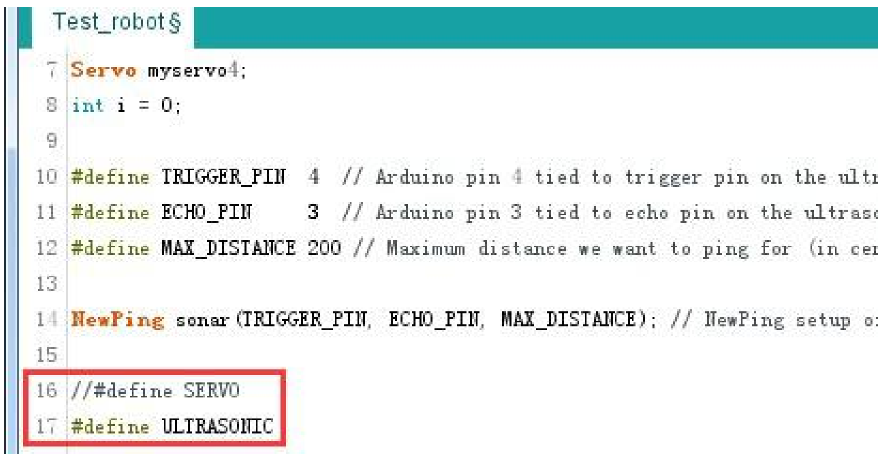


```
1  #include <Servo.h>
2  #include <NewPing.h>
3
4  Servo myservo1;
5  Servo myservo2;
6  Servo myservo3;
7  Servo myservo4;
8  int i = 0;
9
10 #define TRIGGER_PIN  4  // Arduino pin 4 tied to trigger pin on the ultrasonic sensor.
11 #define ECHO_PIN     3  // Arduino pin 3 tied to echo pin on the ultrasonic sensor.
12 #define MAX_DISTANCE 200 // Maximum distance we want to ping for (in centimeters). Maximu
```

And connect the pin **TRIG** to pin **4** of the servo control board. **ECHO** to pin **3**, **VCC** to **V** and **GND** to **G**.

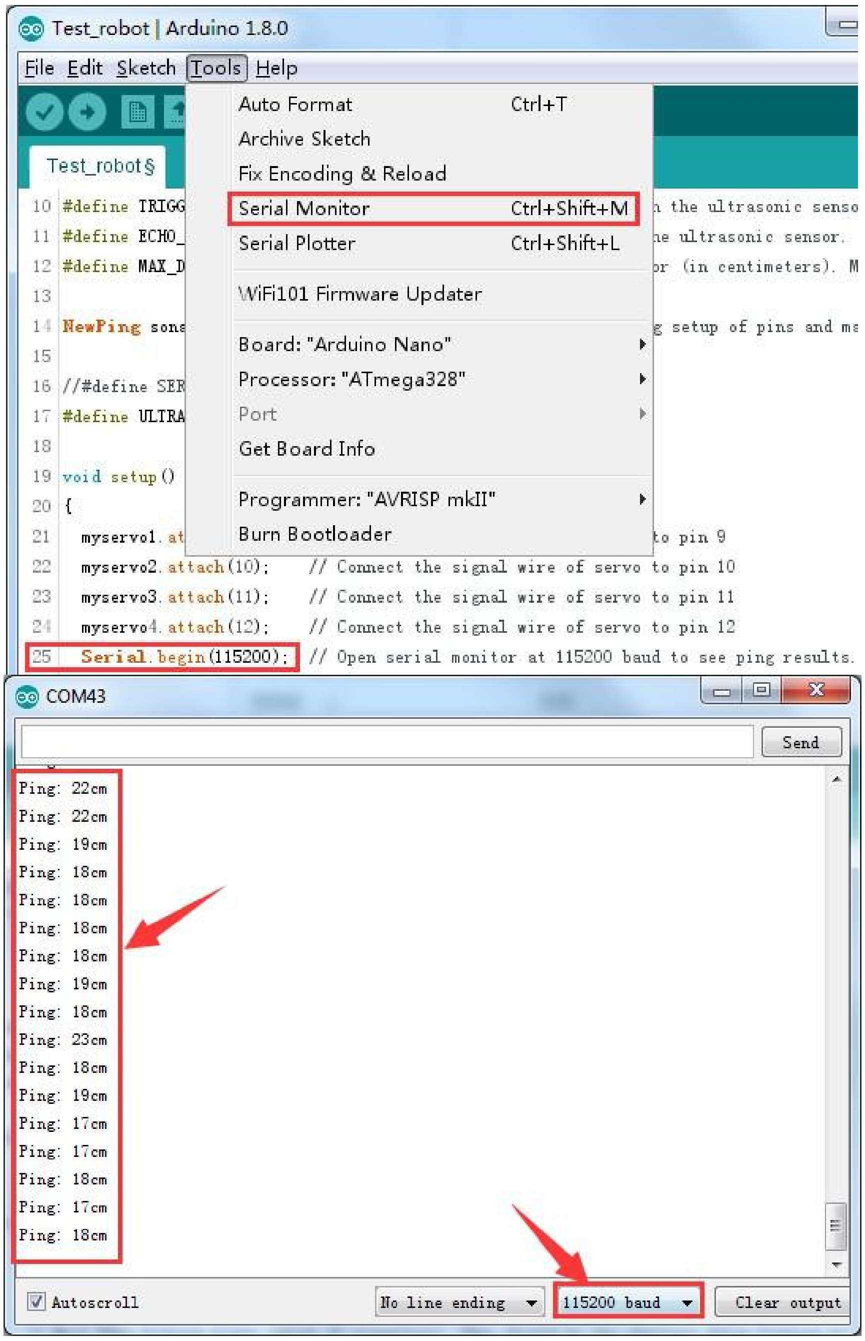


Go to **Line 16**, disable the **SERVO** and activate the **ULTRASONIC**.



```
7  Servo myservo4;
8  int i = 0;
9
10 #define TRIGGER_PIN  4  // Arduino pin 4 tied to trigger pin on the ultr
11 #define ECHO_PIN     3  // Arduino pin 3 tied to echo pin on the ultrasc
12 #define MAX_DISTANCE 200 // Maximum distance we want to ping for (in cen
13
14 NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE); // NewPing setup o:
15
16 //#define SERVO
17 #define ULTRASONIC
```

Upload the program to the Nano board, open the serial monitor. Then press the power button on the servo control board. Set the **baud rate** as **115200**, hold the ultrasonic module and make the two ultrasonic "eyes" facing an obstacle, move the robot back and forth, and observe the data shown on serial monitor. If the data changes with the robot's moving, it proves the ultrasonic module is good. At last, turn it off, and unwire the battery。



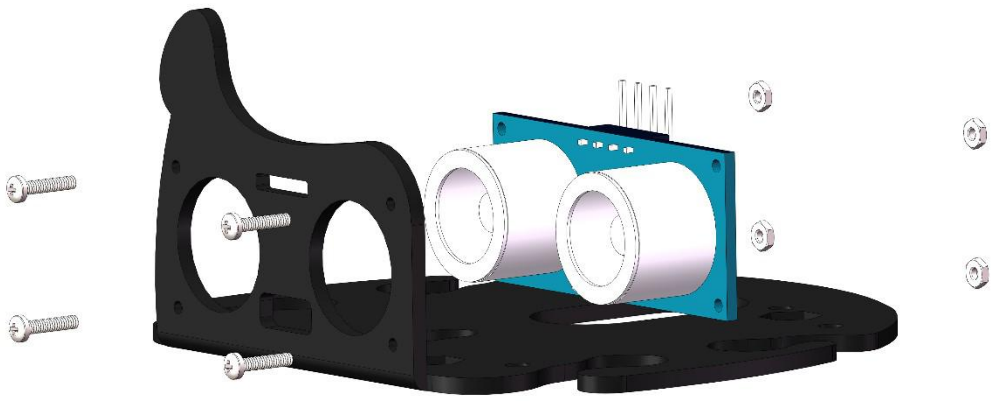
```
#define TRIGG                          the ultrasonic senso
#define ECHO_                          he ultrasonic sensor.
#define MAX_D                          or (in centimeters). M

NewPing sona                           g setup of pins and ma

//#define SER
#define ULTRA

void setup()
{
  myservo1.at                          to pin 9
  myservo2.attach(10);    // Connect the signal wire of servo to pin 10
  myservo3.attach(11);    // Connect the signal wire of servo to pin 11
  myservo4.attach(12);    // Connect the signal wire of servo to pin 12
  Serial.begin(115200);   // Open serial monitor at 115200 baud to see ping results.
```

```
Ping: 22cm
Ping: 22cm
Ping: 19cm
Ping: 18cm
Ping: 18cm
Ping: 18cm
Ping: 18cm
Ping: 19cm
Ping: 18cm
Ping: 23cm
Ping: 18cm
Ping: 19cm
Ping: 17cm
Ping: 17cm
Ping: 18cm
Ping: 17cm
Ping: 18cm
```

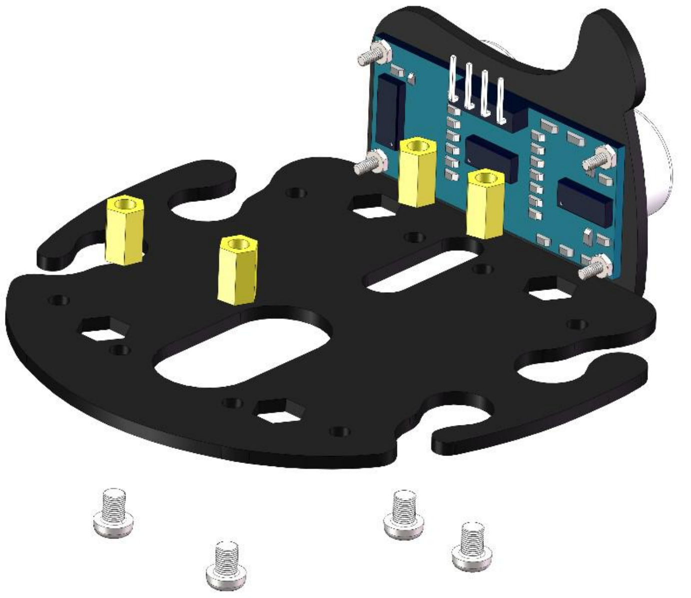
Assembly

i. Head Assembly

Insert the ultrasonic module into No. 1 board and secure it with M1.4*8 screws and M1.4 nuts.

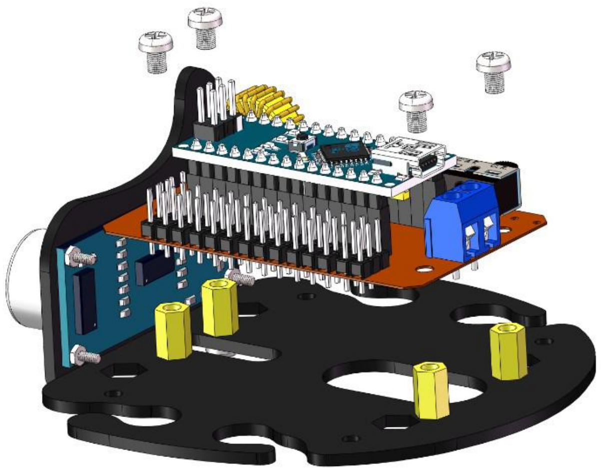


Use a M3*5 screw to secure the M3*8 two-way hexagonal copper post on No. 1 board.

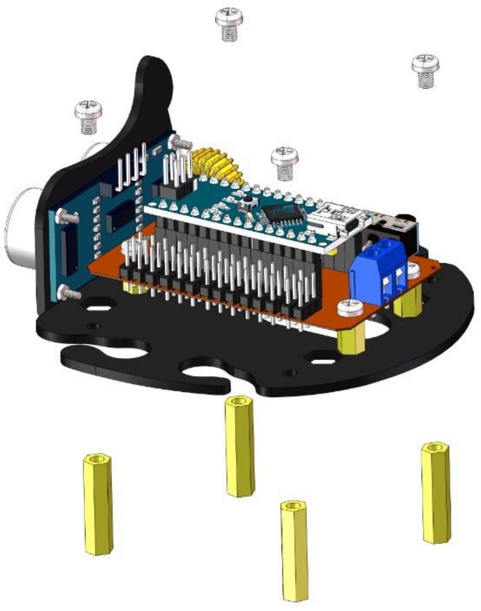


ii. Electrical Module Assembly

Use a M3*5mm screw to mount the previously installed circuit board on No. 1 board.

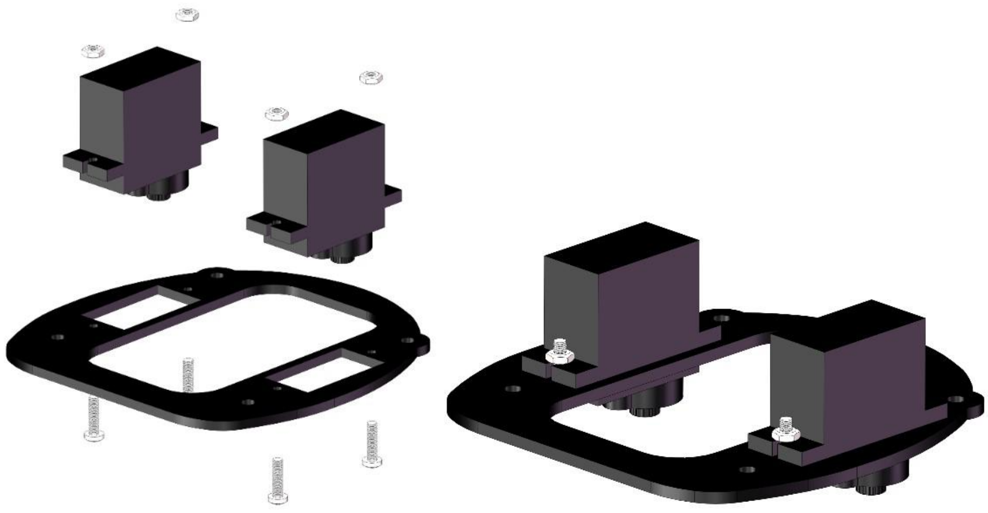


Use M3*5 screws to fix M3*25 Bi-pass Copper Standoff under the No. 1 board.

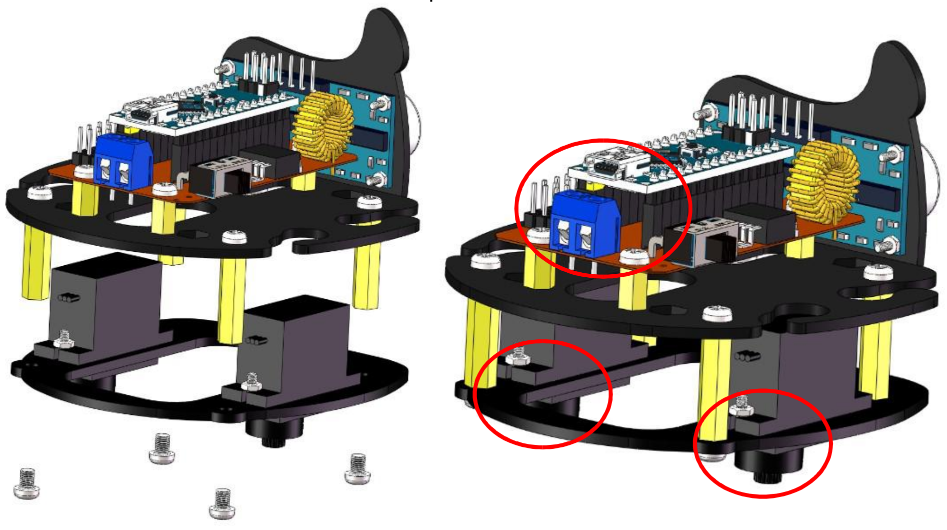


iii. Servo Assembly

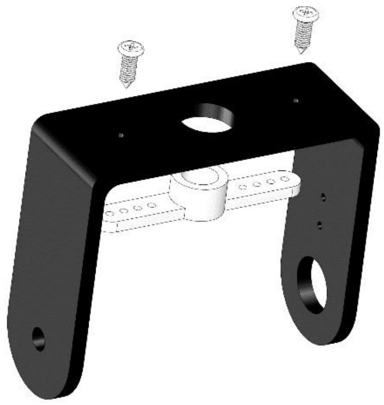
Use M2*8 screws and M2 nuts to mount the servo on the corresponding position on the No. 2 board. (Note the direction of the servo installation)



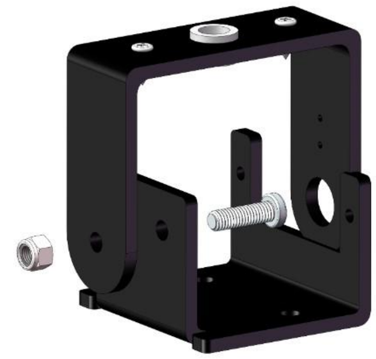
Secure the No. 1 and No. 2 boards with M3*5 screws. Note that the side of the servo shaft should be mounted on the side of the USB port.



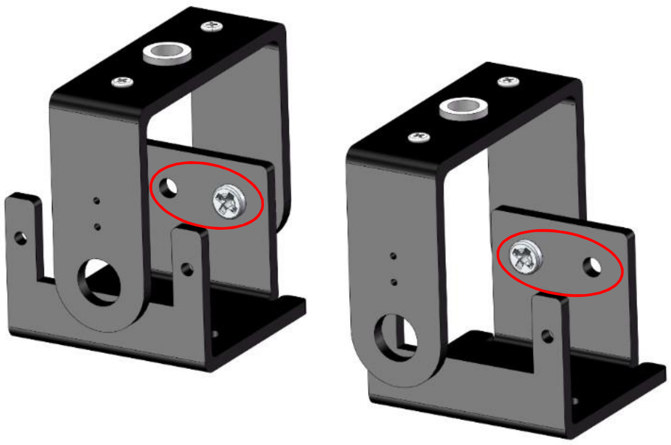
Use two M1.5*5 self-tapping screws to fix the 2-arm rocker arm to the No. 4 board and use the same method to install another No. 4 board.



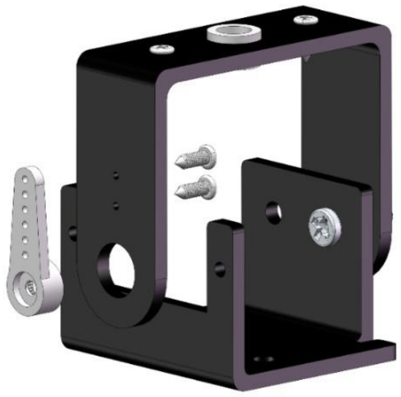
Secure one of the round holes on the 4th and 5th boards with M3*8 Countersunk screws



Use the same method to secure the other round hole on the 4th and 5th boards, as shown



Use two M1.5*5 self-tapping screws to secure the 1-arm rocker arm on the No.4 board.

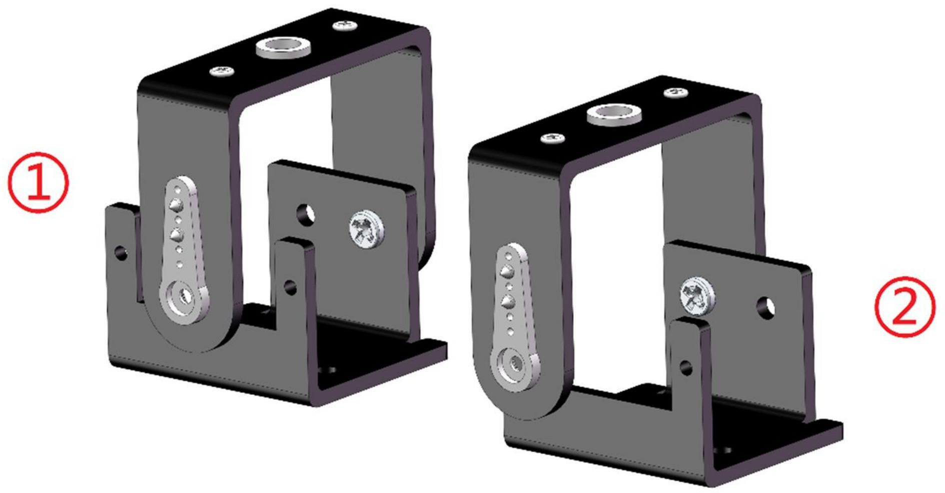


a

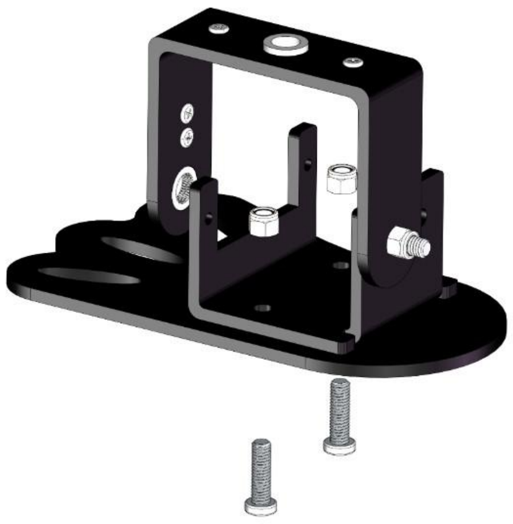
nd M3 self-locking nuts. in

the following figure:

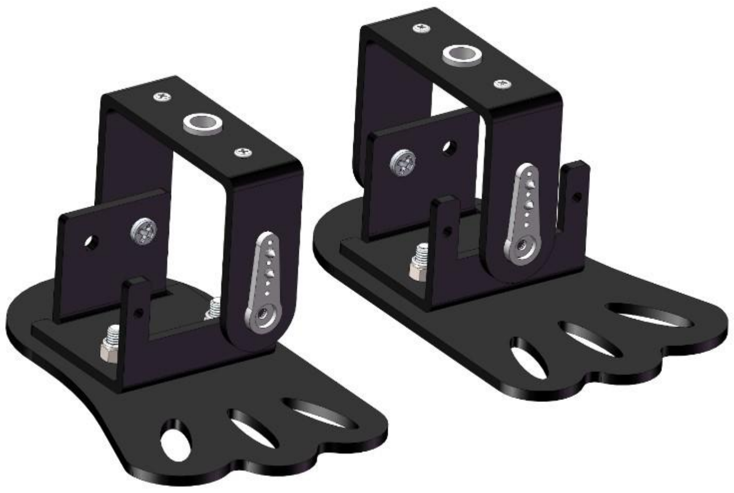
Install another No.4 board in the same way.



Turn the No. 6 board with the countersunk side down and secure the No. 6 board to the ① **module (right leg)** described above with the M3*8 countersunk screw and the M3 selflocking nut.

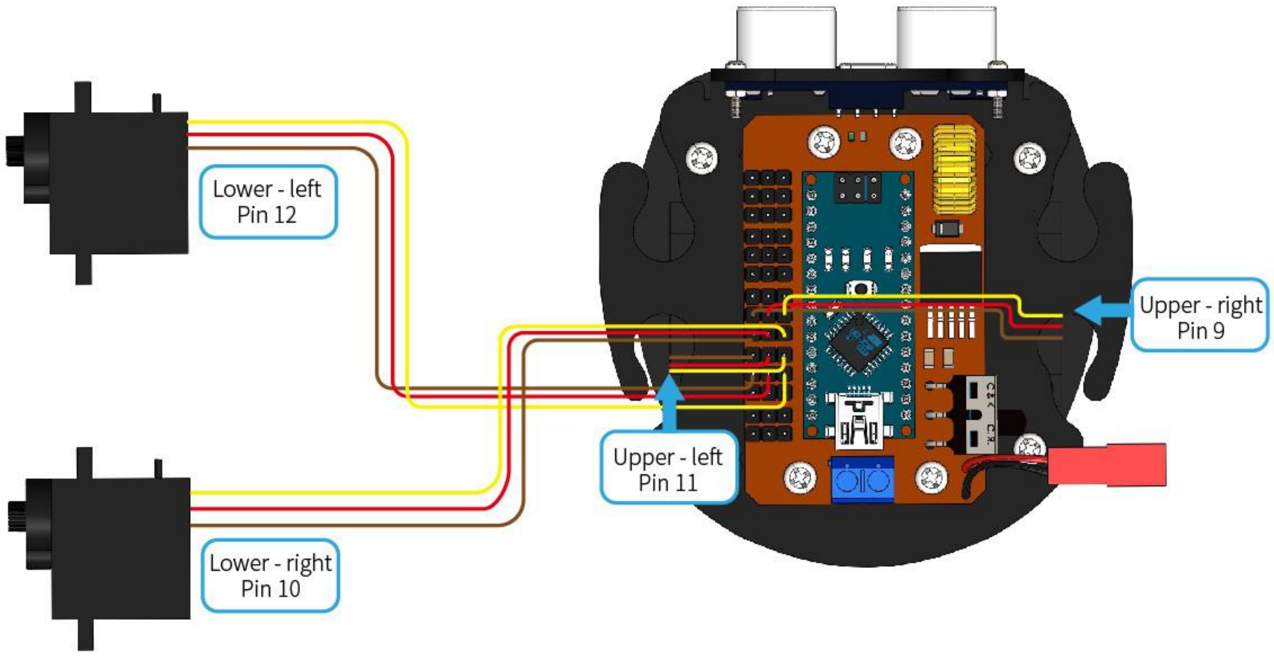


The same method can be used to secure the No.7 and the ② **module (left leg)**.

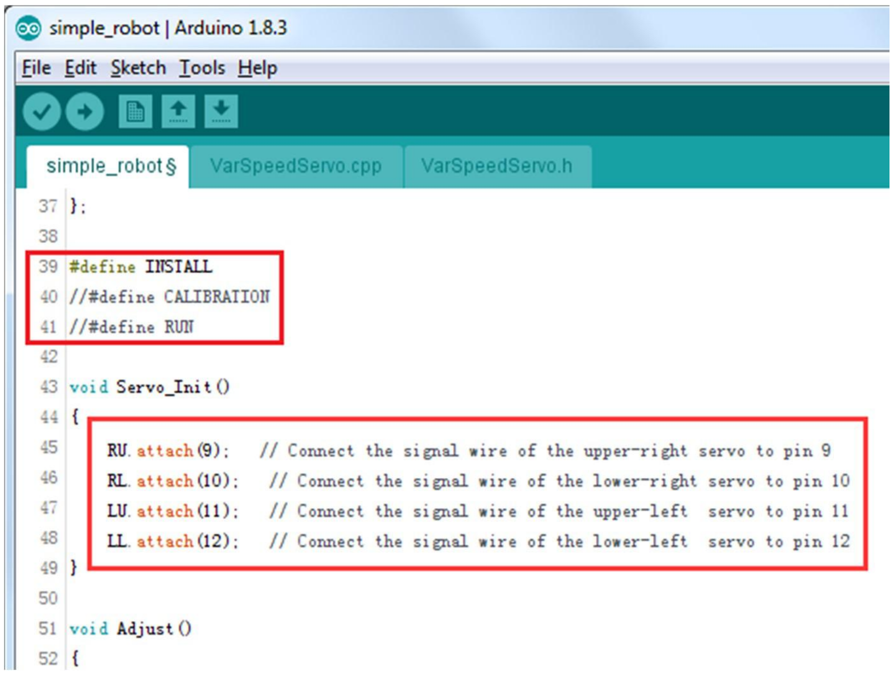


iv. Servo INSTALL Test

Connect the **upper-right** servo to **port 9**, the **yellow** cable to the **signal pin**, **red** to the **positive** pole and **brown** to the **negative** pole. Then connect the **lower-right** servo, the **upper-left** one and the **lower-left** servo to pin **10**, **11** and **12** respectively in the same way.



Connect the Nano board and the computer with a USB cable. Open the program **simple_robot.ino**, and upload the program to the board. Now, it's in install process.

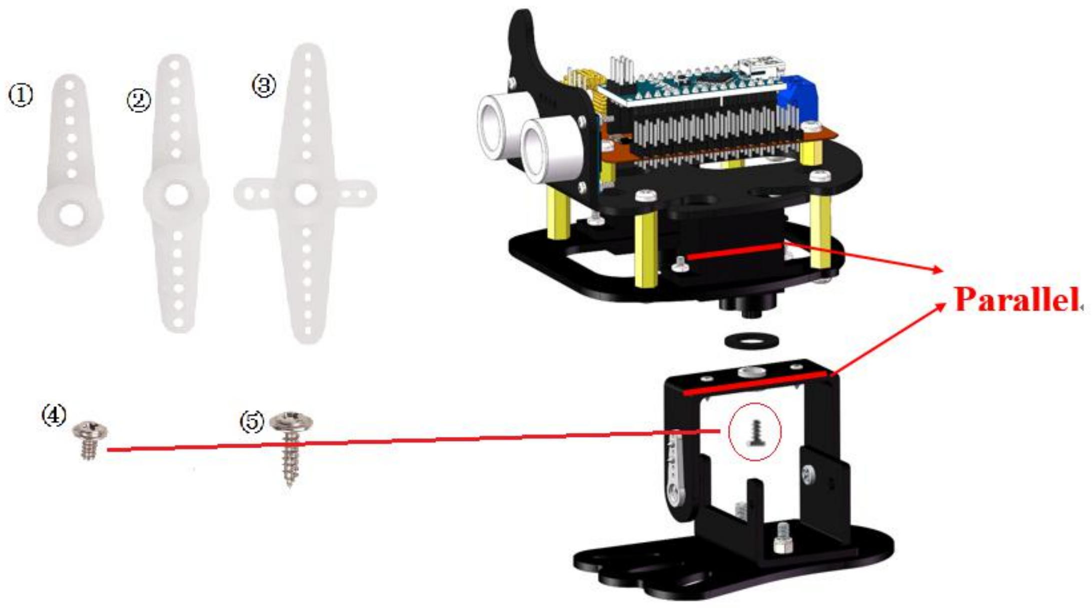


```
};

#define INSTALL
//#define CALIBRATION
//#define RUN

void Servo_Init()
{
   RU.attach(9);   // Connect the signal wire of the upper-right servo to pin 9
   RL.attach(10);   // Connect the signal wire of the lower-right servo to pin 10
   LU.attach(11);   // Connect the signal wire of the upper-left  servo to pin 11
   LL.attach(12);   // Connect the signal wire of the lower-left  servo to pin 12
}

void Adjust()
{
```

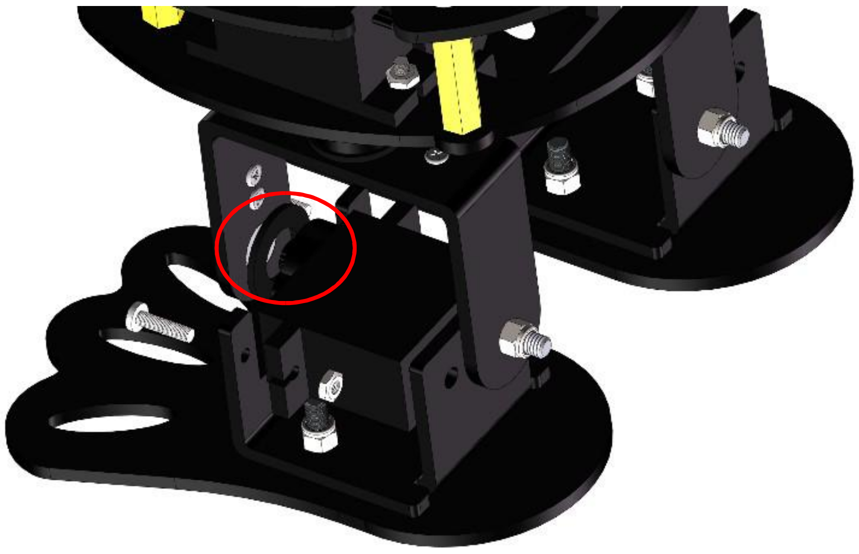
Then power on the robot, keep the power on and the servo connect to the board.

v. Foot Assembly

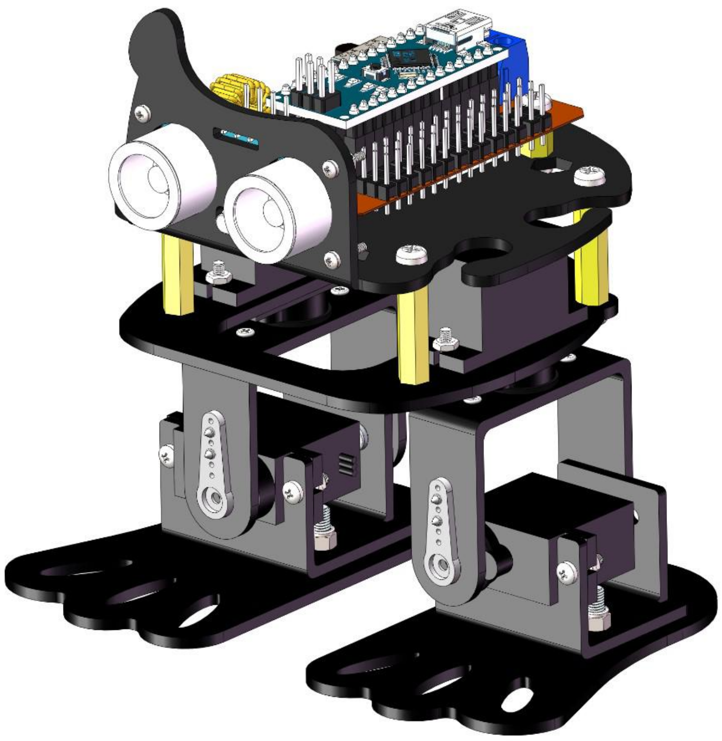
Assemble the No.4 board on the left leg and the No.3 board on the cover together with the upper-left servo through the servo's matching screws (**No.4 screw in servo package**). Try to keep the edges of the 4th board and the servo parallel to each other. They need to be reinstalled if not parallel to each other.



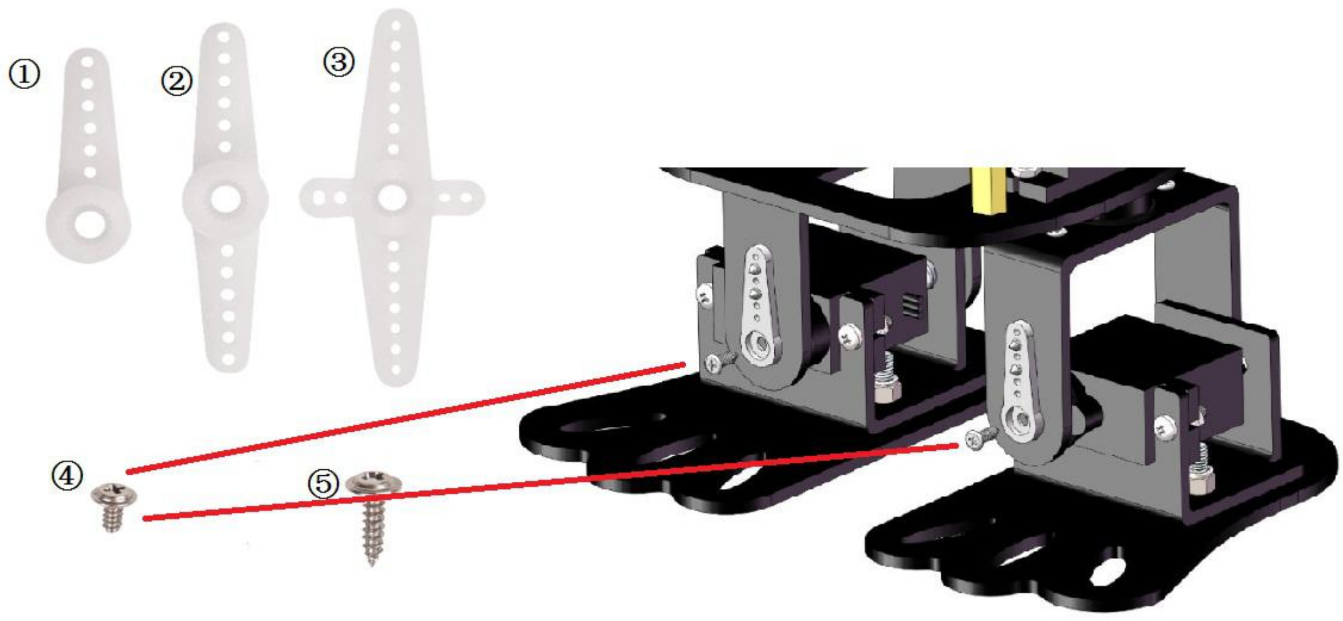
Put a servo shaft on No.3 board, and then secure the lower-left servo (pin 12) to the No.5 board with M2*8 screws and M2 nuts. **Note that the servo shaft should be close to the inside of No. 7 board**.



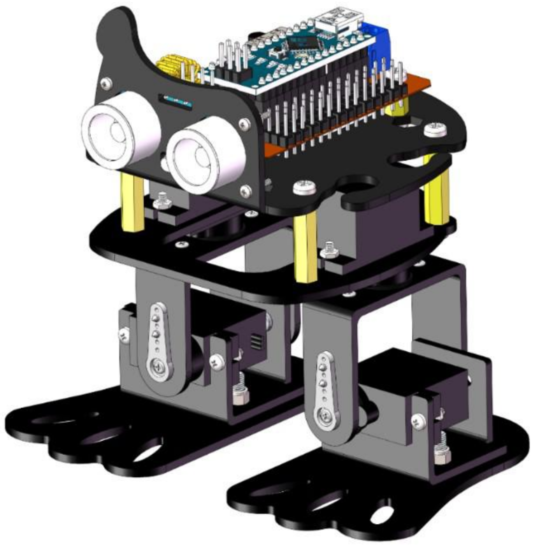
The **lower-right servo (pin 10)** is installed in the same way.



Insert the 1-arm rocker arm into the two servos and fix it with the No. 4 screw in the servo package.

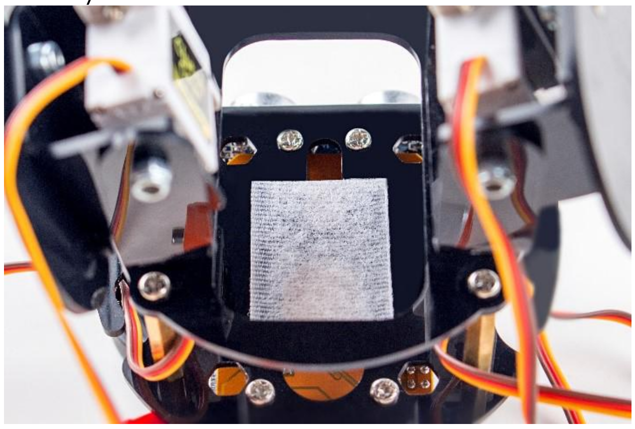


The effect chart is as follows:

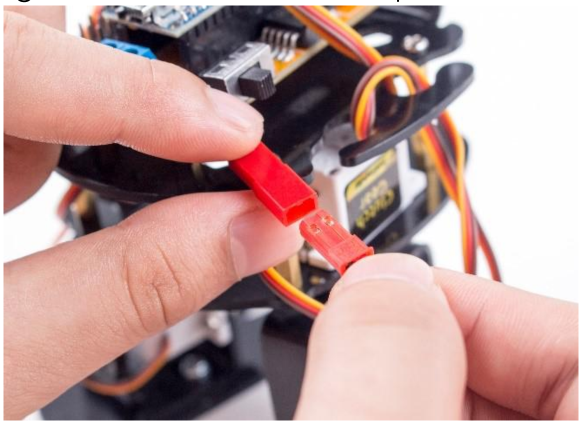


vi. Battery Assembly

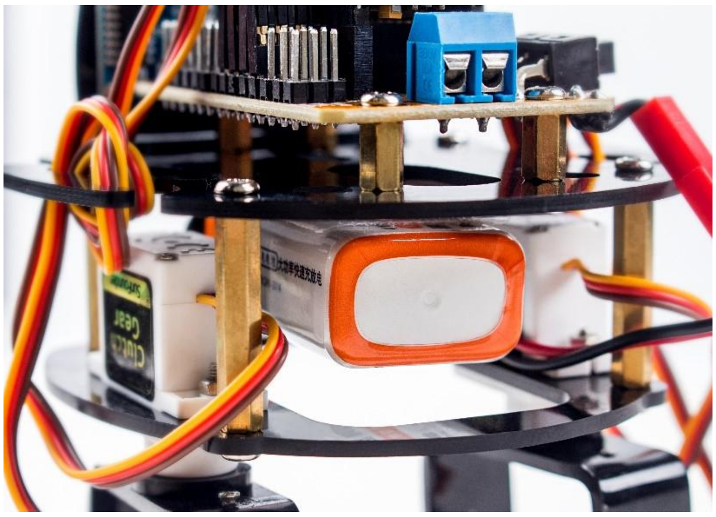
Attach one side of velcro tape to the bottom of the No. 1 board and the other side to the battery.



Insert the battery into the battery cable and plug the other end into the expansion board.

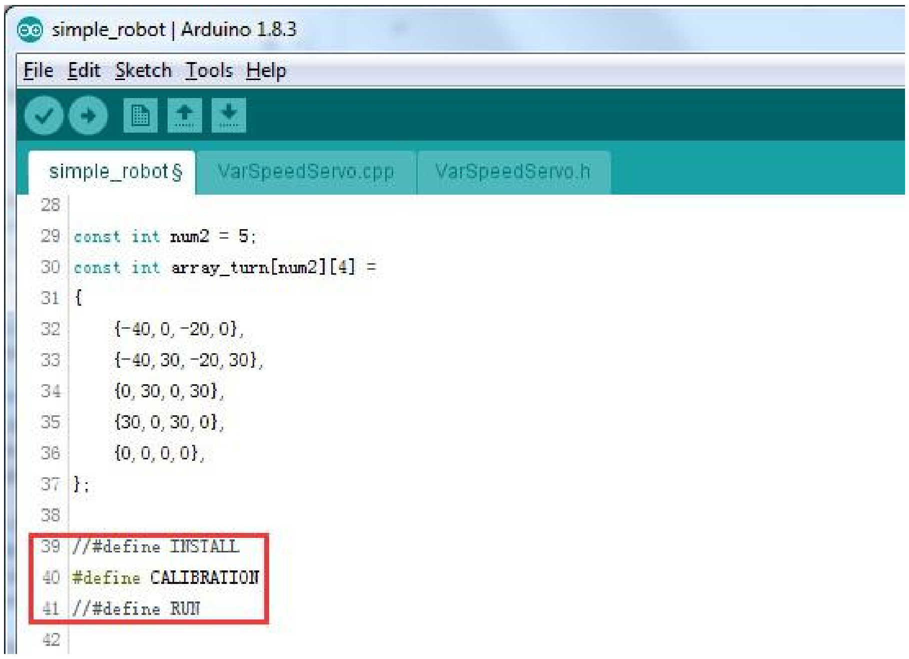


Lastly, paste the battery on the No. 1 board



vii. Servo CALIBRATION Test

Open the program and go to **Line 39**, disable the **INSTALL** and activate the **CALIBRATION**. Select the correct board and port, then upload the sketch. If the robot is not set right, change the angle and upload the code until it is.



```
28
29 const int num2 = 5;
30 const int array_turn[num2][4] =
31 {
32     {-40, 0, -20, 0},
33     {-40, 30, -20, 30},
34     {0, 30, 0, 30},
35     {30, 0, 30, 0},
36     {0, 0, 0, 0},
37 };
38
39 //#define INSTALL
40 #define CALIBRATION
41 //#define RUN
42
```

Tips for calibration:

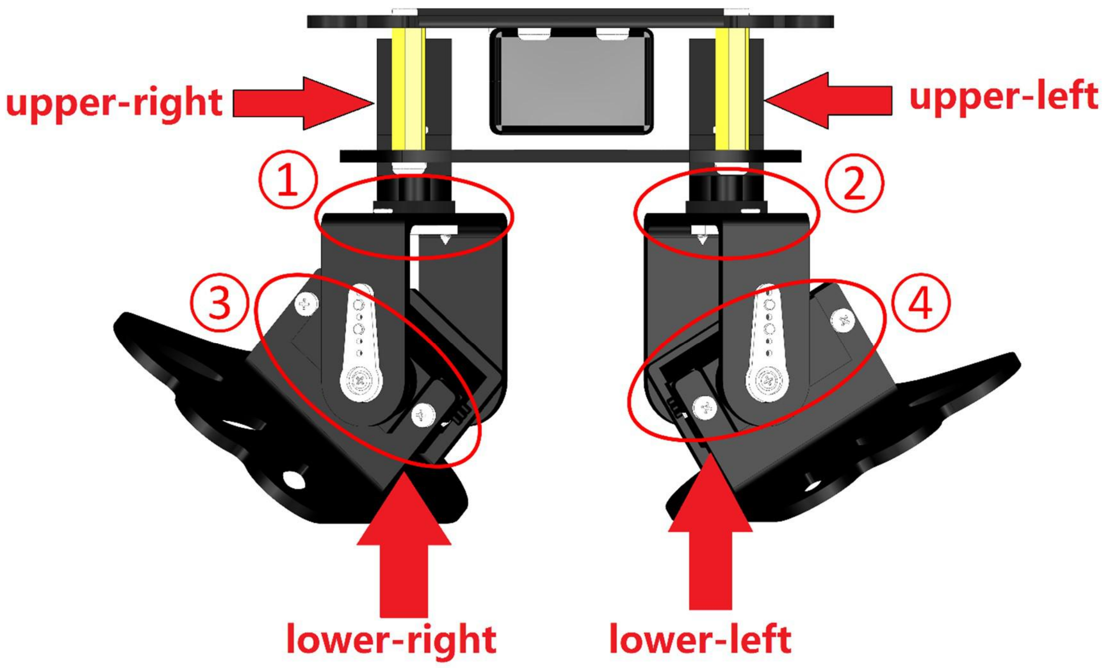
① If the right leg is toe out, you need to decrease the upper-right servo's angle; if it is toe in, you need to increase the angle.

② The calibration method for the left leg works the opposite way for right leg.

③ If the right foot's sole faces outward, you need to decrease the lower-right servo's angle; if its sole faces inward, you need to increase the angle.

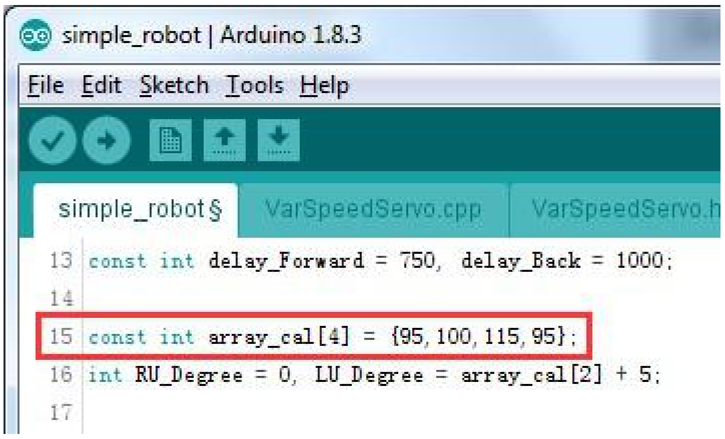
④ The calibration method for the left foot works the opposite way for right foot.

Here we take this robot as an example. After uploading the code, press thepower button on the servo control board, pick up the robot and you will see the slight change of the robot legs and feet:



Observe the robot above we can know: ① right leg is toe out, ② left leg is toe out, ③ right foot's sole faces outward, ④ left foot's sole faves outward.

Then go back to **Line 15**.



```
13 const int delay_Forward = 750, delay_Back = 1000;
14
15 const int array_cal[4] = {95, 100, 115, 95};
16 int RU_Degree = 0, LU_Degree = array_cal[2] + 5;
17
```

Thus we can calibrate as follows:

① **Decrease the upper-right servo's angle**

Change 95 degrees to 90 (array_cal[0]: is the upper-right servo's rotation angle);

② **Increase the upper-left servo's angle**

Change 115 degrees to 120 (array_cal[2]: is the upper-left servo's rotation angle);

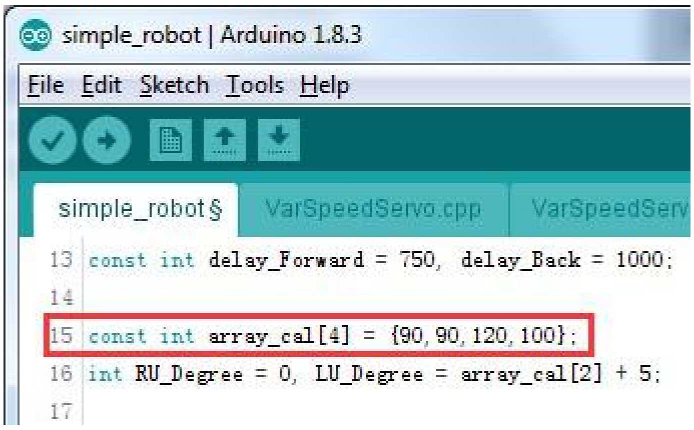
③ **Decrease the lower-right servo's angle**

Change 100 degrees to 90 (array_cal[1]: is the lower-right servo's rotation angle);

④ **Increase the lower-left servo's angle**

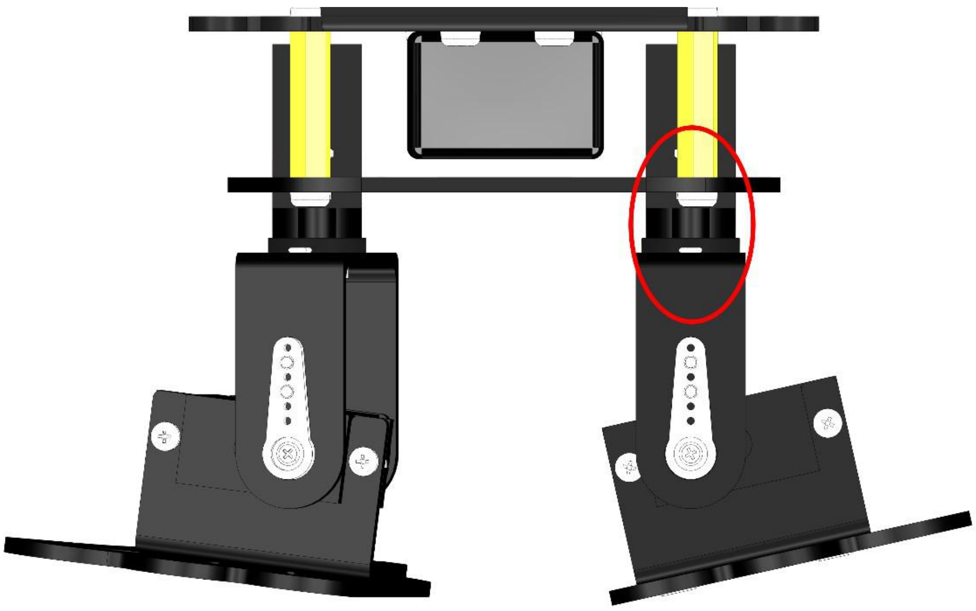
Change 95 degrees to 100 (array_cal[3]: is the lower-left servo's rotation angle);

i.e.: In line 15, change code to array_cal[4] = {90, 90, 120, 100}; Then click **Upload**.

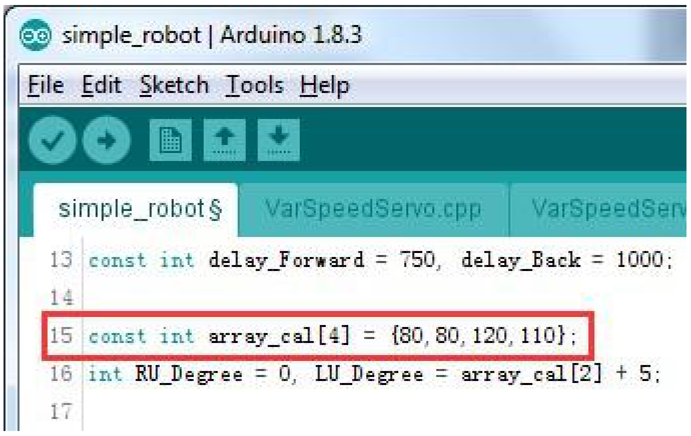


```
13 const int delay_Forward = 750, delay_Back = 1000;
14
15 const int array_cal[4] = {90, 90, 120, 100};
16 int RU_Degree = 0, LU_Degree = array_cal[2] + 5;
17
```

The edge of upper-left plate and upper plate are parallel with each other, but upper-right is not parallel to the upper one, the deviation angles of lower-left and lower-right servos are decreased.

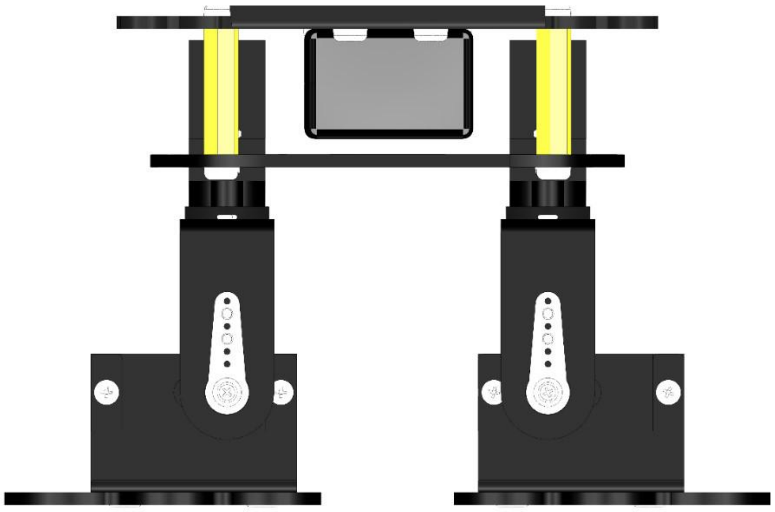


Change code in line 15 to array_cal[4] = {80, 80, 120, 110}; Then click **Upload**.



```
const int delay_Forward = 750,  delay_Back = 1000;

const int array_cal[4] = {80,80,120,110};
int RU_Degree = 0,  LU_Degree = array_cal[2] + 5;

```

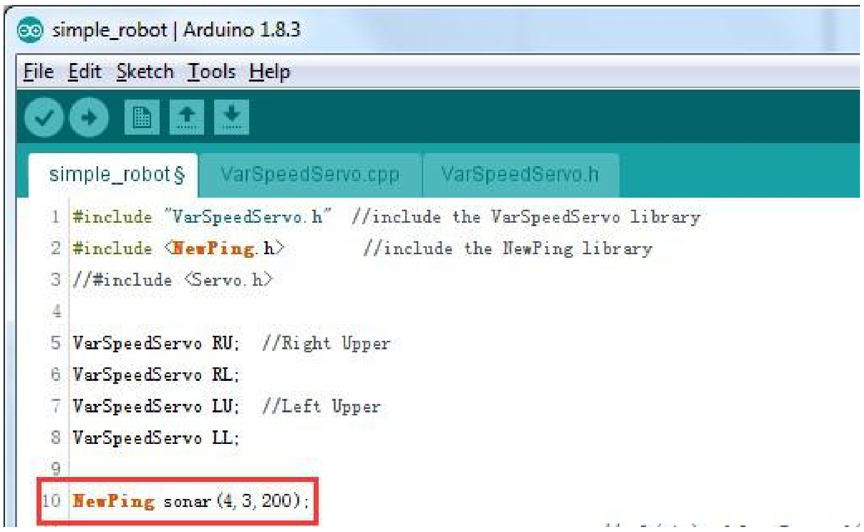
Observe the four servos to make sure they are in proper angle, then the servo calibration is completed. You can do fine tuning with value changing of “1” each time, if there is a little deviation.



Since the servo angles on legs and feet differ, the final calibrated angles (array_cal[4]) will be different too. It will take multiple times of calibration, you should adjust in patience.

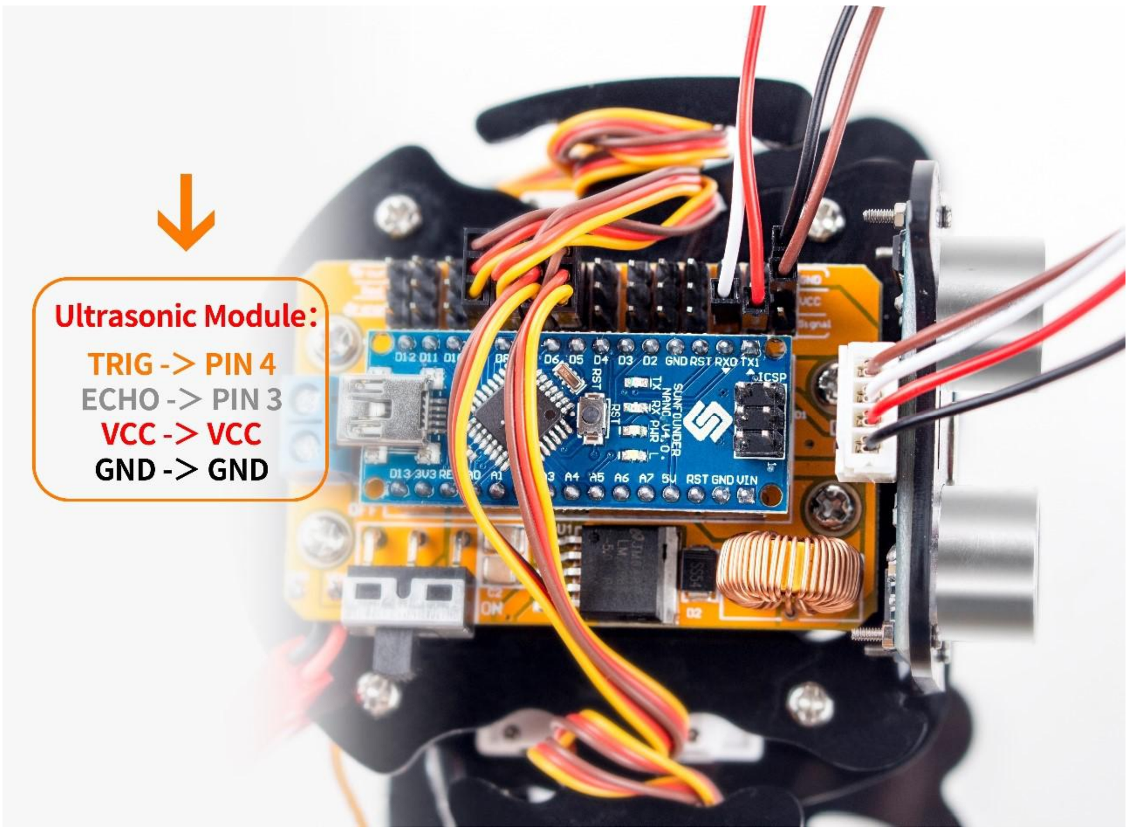
viii. Ultrasonic Connecting

Plug one end of the ultrasonic connecting cable into the ultrasonic module, and the other to the servo control board.



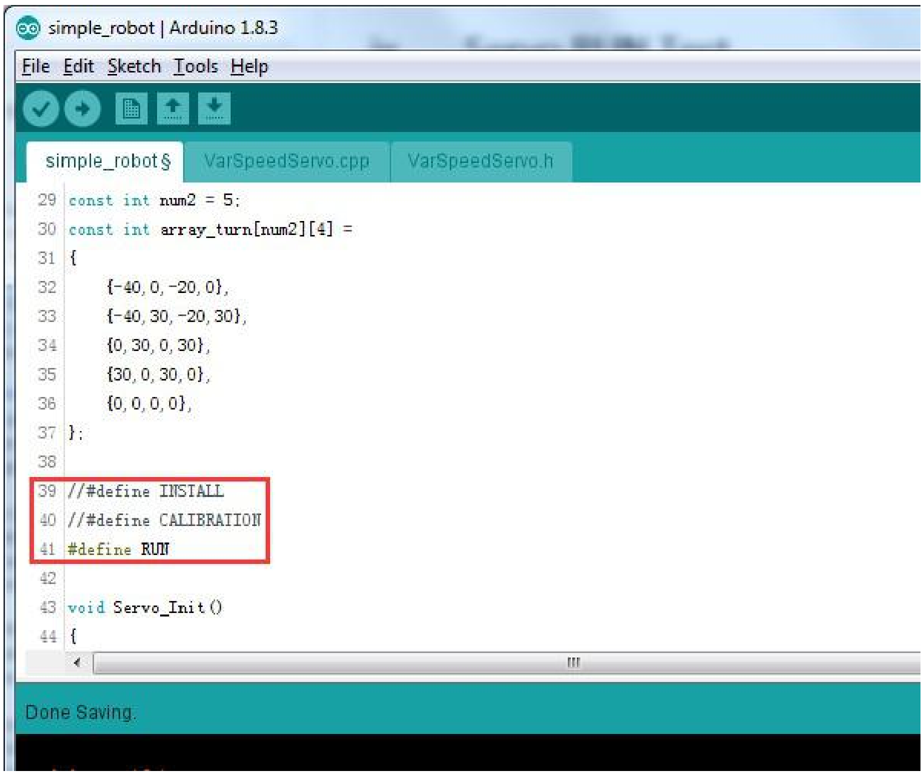
```
#include "VarSpeedServo.h"  //include the VarSpeedServo library
#include <NewPing.h>         //include the NewPing library
//#include <Servo.h>

VarSpeedServo RU;  //Right Upper
VarSpeedServo RL;
VarSpeedServo LU;  //Left Upper
VarSpeedServo LL;

NewPing sonar(4,3,200);
```

Connect **pin TRIG** of the ultrasonic to **pin 4** of the board, **ECHO** to **pin 3**, **VCC** to **VCC** and **GND** to **GND**.



ix. Servo RUN Test

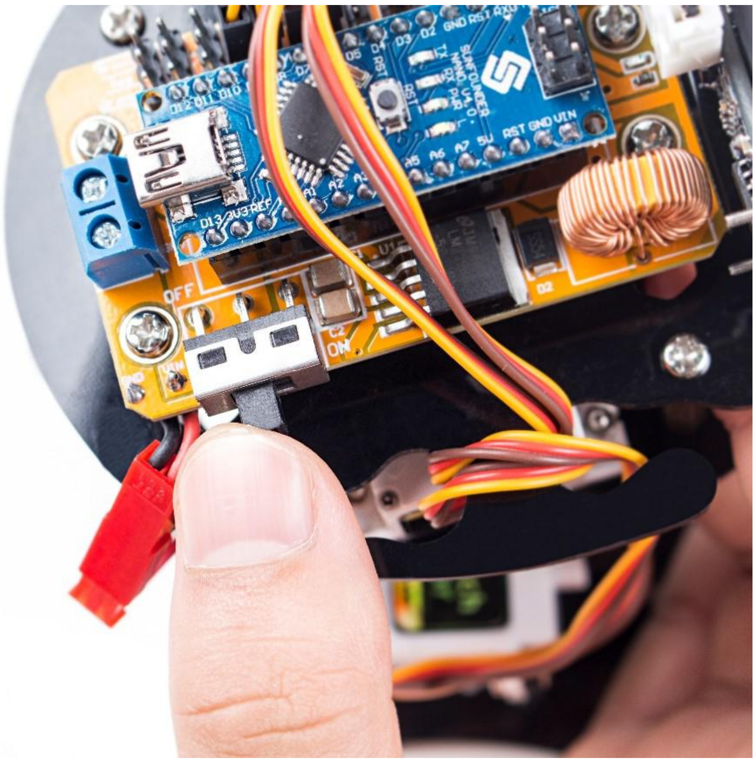
Go to **Line 39** again, disable the **CALIBRATION** line. Activate the operating programe "**run**" and burn the program to the board.



```
const int num2 = 5;
const int array_turn[num2][4] =
{
     {-40, 0, -20, 0},
     {-40, 30, -20, 30},
     {0, 30, 0, 30},
     {30, 0, 30, 0},
     {0, 0, 0, 0},
};

//#define INSTALL
//#define CALIBRATION
#define RUN

void Servo_Init()
{
```

Done Saving.

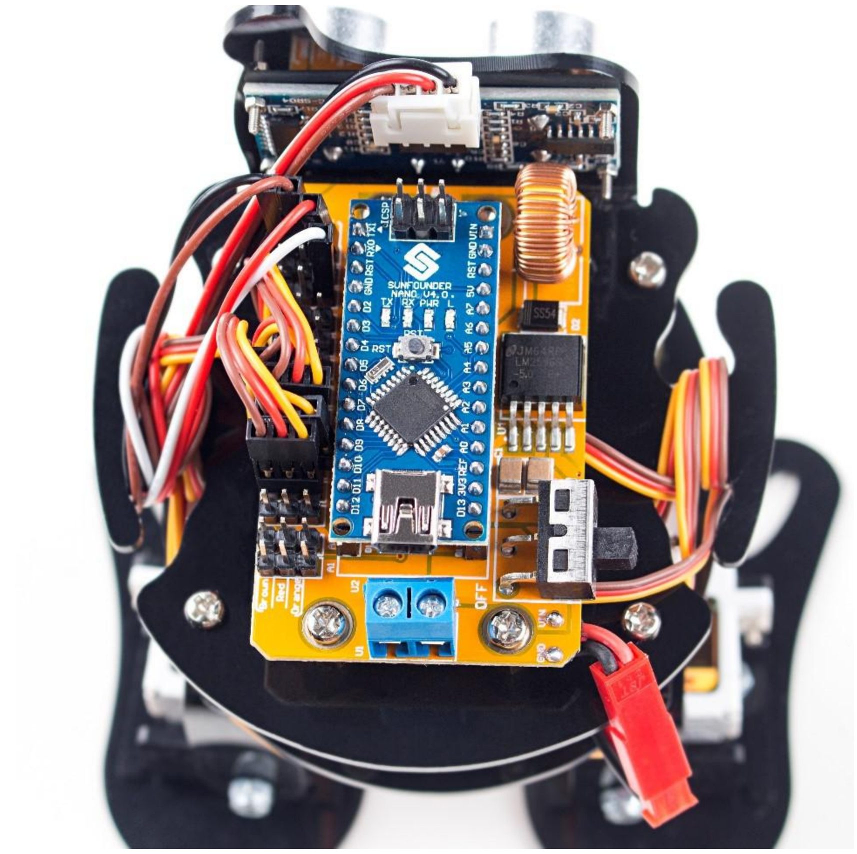
After burning successfully, unplug the USB cable and press the power button on the servo control board.



You will see the robot moving forward. When encountering an obstacle, it will make a turn and then go forward again.

x. Wire Arrangement

Twine the servo wire and 4-Pin anti-reverse cable on the No. 1 board.



So far the robot has been assembled successfully, it's easy if you follow our steps closely.

Hope you enjoy the fun of the bot, thanks for watching.

Q&A

Q1: How can we know the servo is damaged?

A1: In Servo Test step, if the servo rocker arm shake, get stuck or can not rotate smoothly, with an abnormal sound, we can judge it as a damaged one.

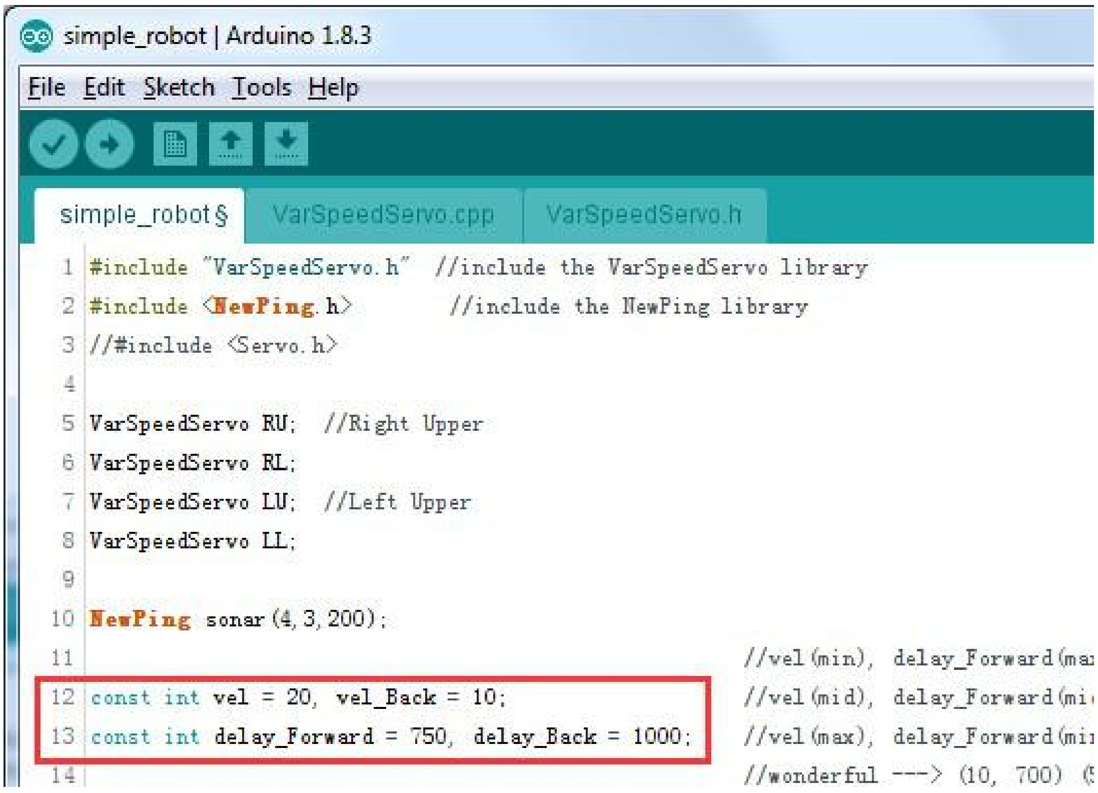
Q2: Why the Sloth reboots in running?

A2:

1) If the Sloth is in lower power, rebooting will happen ,please charge the battery in time.
2) It could be the servos are lacking for power. Open the program and go to Line 12, 13. "**vel**" is the servos rotating speed in "initialization or moving forward"; "**vel_Back**" is the servos rotating speed in "moving backward"; "**delay_Forward**", "**delay_Back**" are the delays between two moving forward loops and moving backward loops.
    a) If rebooting happens in moving forward actions, you can decrease the value of "vel" or/ and increase the value of "delay_Forward". For example, decrease "vel" value to 10, and increase "delay_Forward" to 1500.

b) If rebooting happens in moving backward actions, you can decrease “vel_Back” or/ and increase “delay_Backward”. For instance, decrease “vel_Back” to 8, and increase “delay_Backward” to 1500. You can adjust to a proper value as you want.

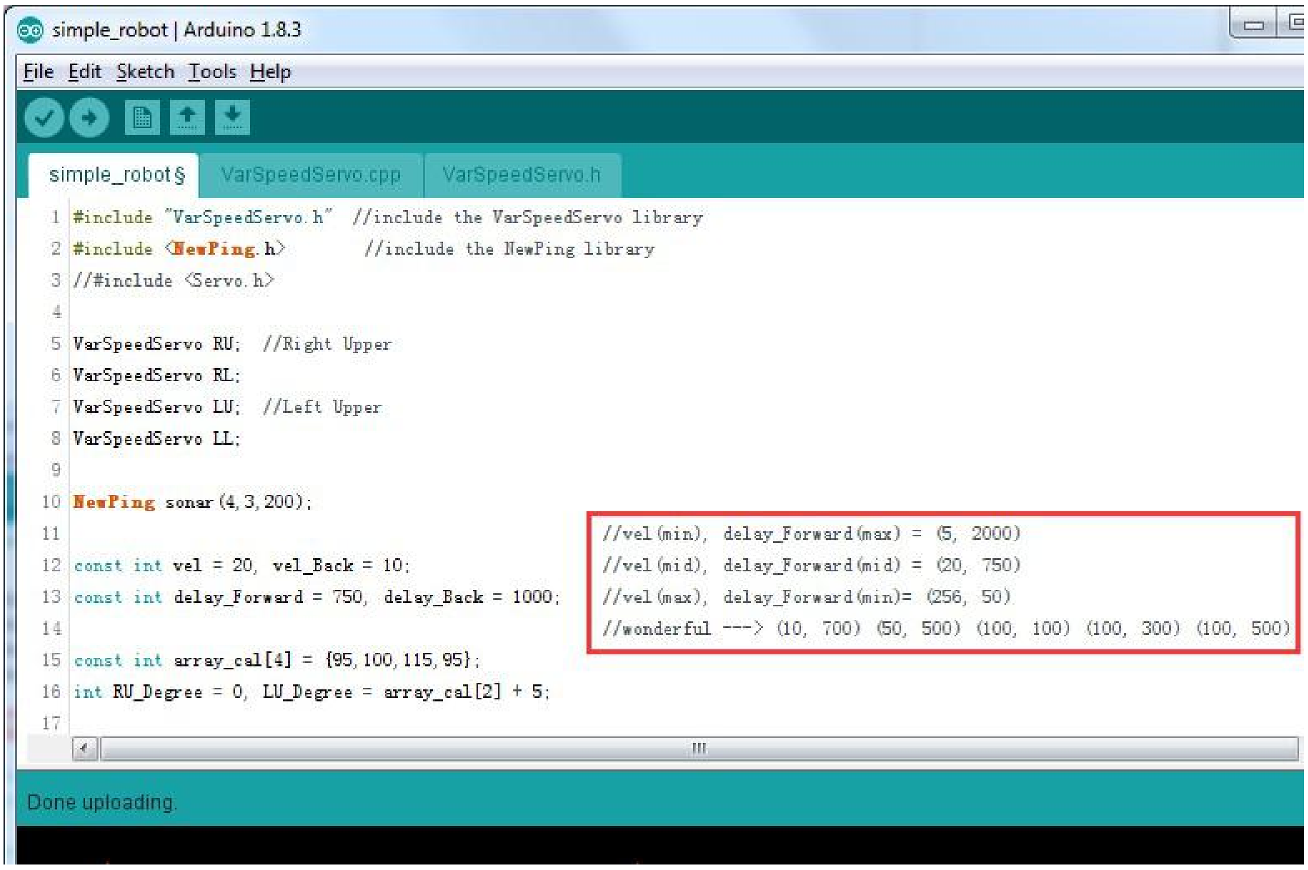
Then click **Upload**.



```
#include "VarSpeedServo.h"  //include the VarSpeedServo library
#include <NewPing.h>        //include the NewPing library
//#include <Servo.h>

VarSpeedServo RU;  //Right Upper
VarSpeedServo RL;
VarSpeedServo LU;  //Left Upper
VarSpeedServo LL;

NewPing sonar(4,3,200);
                                                     //vel(min), delay_Forward(ma
const int vel = 20, vel_Back = 10;                   //vel(mid), delay_Forward(mi
const int delay_Forward = 750, delay_Back = 1000;    //vel(max), delay_Forward(mi
                                                     //wonderful ---> (10, 700) (
```

Q3: Sloth walks too slowly when it moves forward. How to solve this?

A3: Sloth’s default speed is middle speed, the related sketch is “vel(mid), delay_Forward(mid) = (20, 750)”. You can change the speed value as shown below to adjust the walking speed.



```
 1 #include "VarSpeedServo.h"  //include the VarSpeedServo library
 2 #include <NewPing.h>         //include the NewPing library
 3 //#include <Servo.h>
 4
 5 VarSpeedServo RU;  //Right Upper
 6 VarSpeedServo RL;
 7 VarSpeedServo LU;  //Left Upper
 8 VarSpeedServo LL;
 9
10 NewPing sonar(4,3,200);
11                                                  //vel(min), delay_Forward(max) = (5, 2000)
12 const int vel = 20, vel_Back = 10;               //vel(mid), delay_Forward(mid) = (20, 750)
13 const int delay_Forward = 750, delay_Back = 1000;  //vel(max), delay_Forward(min)= (256, 50)
14                                                  //wonderful ---> (10, 700) (50, 500) (100, 100) (100, 300) (100, 500)
15 const int array_cal[4] = {95,100,115,95};
16 int RU_Degree = 0, LU_Degree = array_cal[2] + 5;
17
```

```
Done uploading.
```

change the value of vel and delay_Forward in line12 and 13 to as shown:

vel = 50, delay_Forward = 500

Then click **Upload**.

Note: If you adjust the robot to a high walking speed, it may fall down and break. Thus it's better to do some protection for the Sloth.

Q4: Sloth walks too slowly when it moves backward. How to solve this?

A4: Considering the structure of Sloth, it's better do adjust a slow speed for backward walking. If you want to adjust the walking speed, refer to Q3 to adjust the value. DO NOT adjust a high speed for walking backward to avoid possible falling down.

Q5: How to make the sloth more stable in walking?

A5: Cut to get two paper cushion for the robot feet, and stick them on the Sloth soles to maintain enough friction for a stable walking.

Summary

In this manual, having learned the related components for building the robot kit, you've gone through the assembly of the mechanical parts and electrical modules with the knowledge of Arduino as well as a brief introduction of the key parts like servo, ultrasonic, etc. Also you've got a lot of software and coding, which lays a solid foundation for your futrue journey of exloring open-source field.

The SunFounder DIY 4-DOF Robot Kit is not only a toy, but more a meaningful development kit for Arduino. After all the study and hands-on practice of the kit, you should have a better understanding of Aduino. Now, get started to make better work!

Copyright Notice