

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.

Online-tutorials



There is an online tutorial available at <https://sloth-kit.rtfid.io>, which contains all the contents of this manual, and it is updated in real-time, so we recommend that you use it.

Free Support



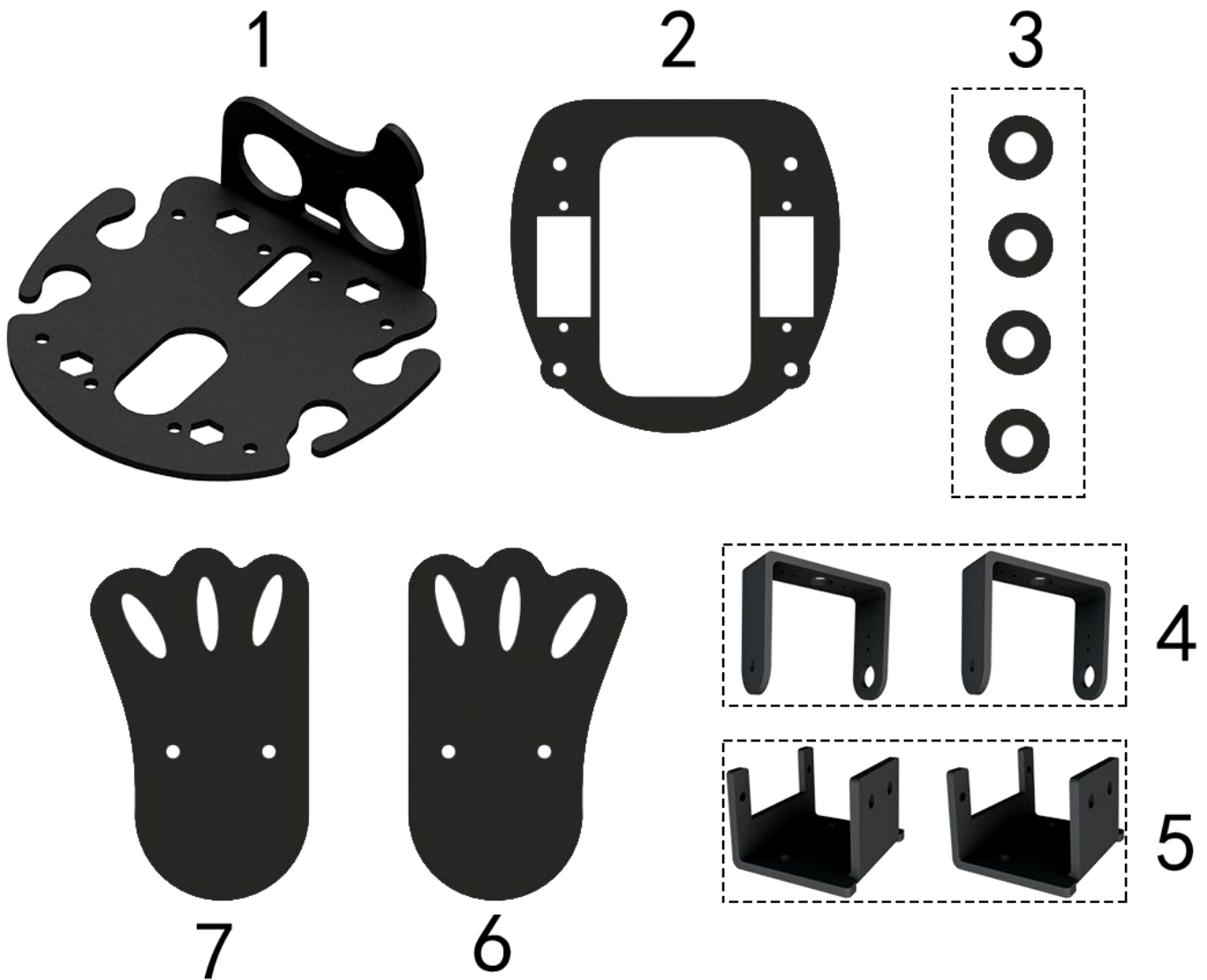
If you have any **questions**, please **send an email to** service@sunfounder.com and we'll reply as soon as possible.

Content

Components	1
Structural Plate	1
Mechanical Fasteners	2
Electrical Components	3
Battery (Not Included)	4
Introduction	5
Download the Code	6
Install Arduino IDE and Add Libraries	8
Download Arduino IDE	8
Installation	9
➤ Windows	9
➤ macOS	11
➤ Linux	11
Open the IDE	11
Add Libraries	13
Test Servos and the Ultrasonic Module	14
Test the Servo	14
Test the Ultrasonic Module	18
Assembly	20
Head Assembly	20
Electrical Module Assembly	21
Servo Assembly	22
Servo INSTALL Test	25
Foot Assembly	27
Battery Assembly	29
Servo CALIBRATION Test	31
Ultrasonic Connecting	32
Wire Arrangement	33
Example	34
Simple Robot	34
Dancing	35
Q&A	36











Components

Structural Plate






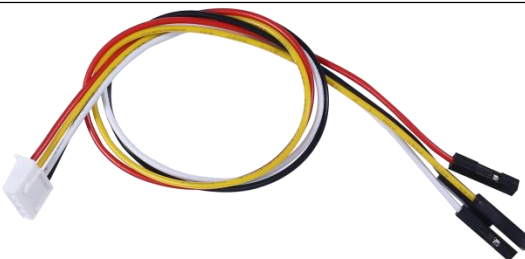


Note: Please check the structural plate and component list, if there misses any components, please take pictures of all the components you have received and inform us of the missing parts, and send E-mail to service@sunfounder.com.

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

Electrical Components

Parts	Name	Qty.
	SF006C Servo	4
	Ultrasonic Module	1
	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

Battery (Not Included)

Parts	Name	Qty.
	9V battery	1

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.

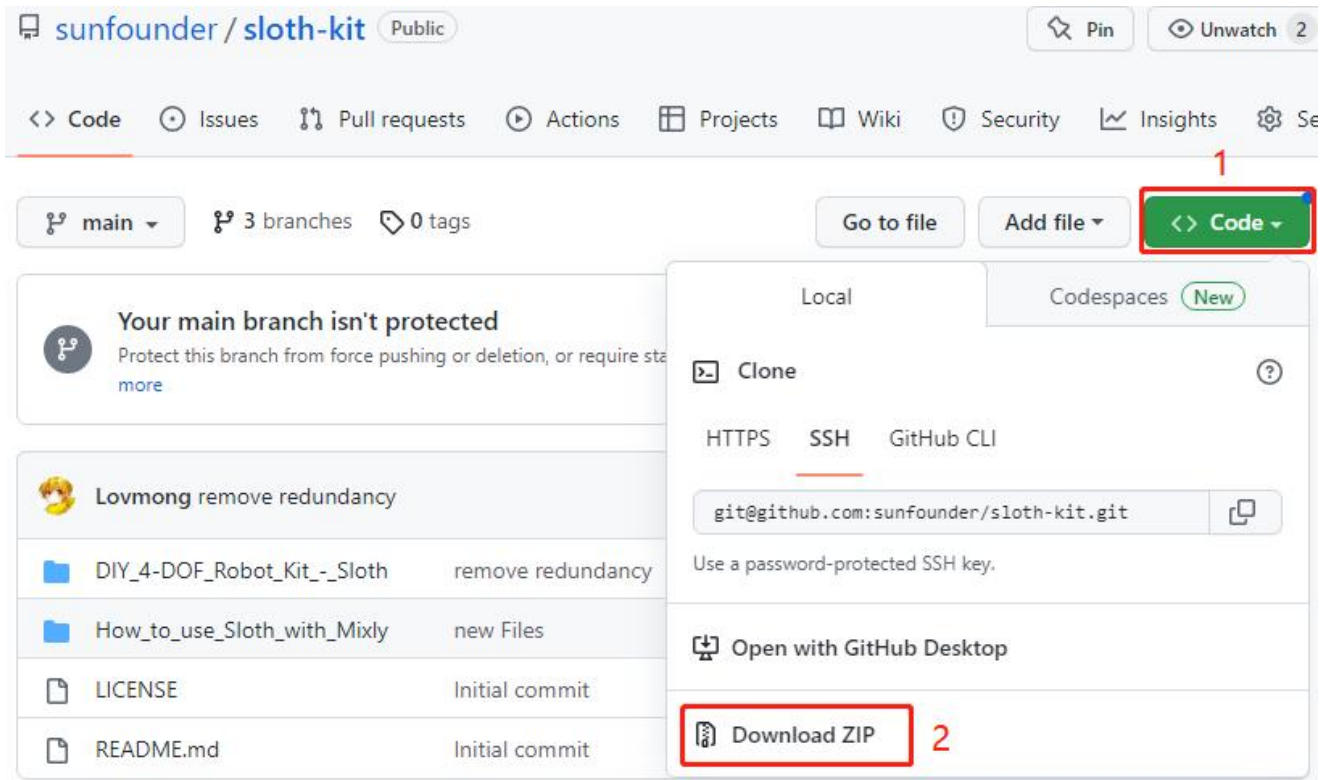
It is a new mobile robot called Sloth developed by SunFounder. Each leg has 2 joints driven by servo. One 9V chargeable lithium batteries are to supply the bot when the Nano board is used as the control board. A servo control board connects with the batteries, servos, Nano board, and the HC-SR04 ultrasonic 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.



Download the Code

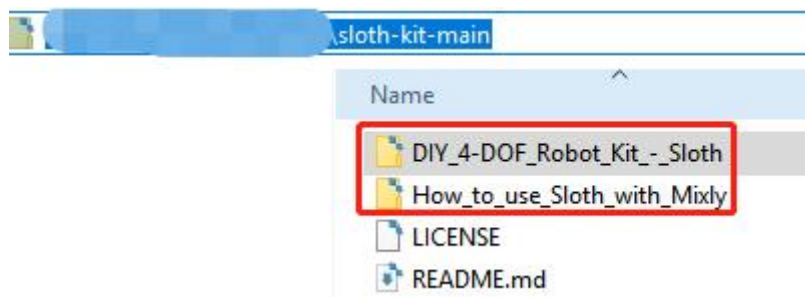
Step 1

You can download the code from SunFounder's github repository. Once there, click on **Code** and then select **Download ZIP**: <https://github.com/sunfounder/sloth-kit>

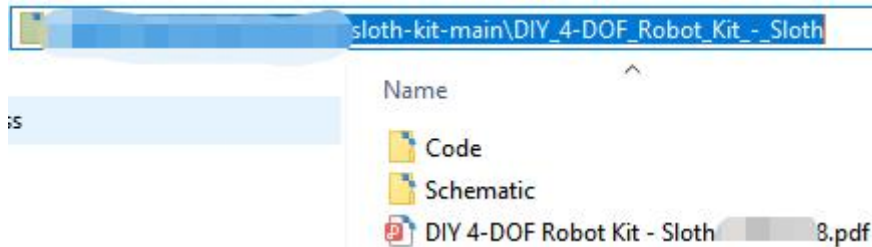


Step 2

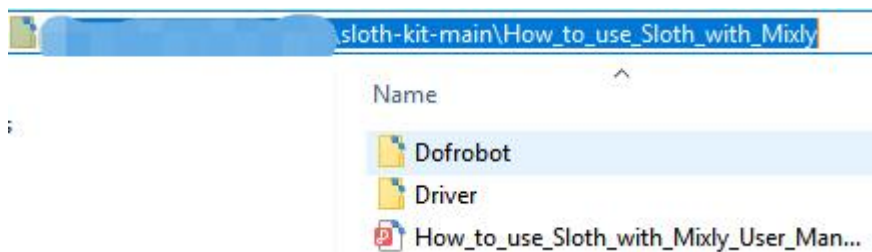
Once downloaded and unpacked, you will see 2 folders.



- **DIY_4-DOF_Robot_Kit_-_Sloth:** Here is the folder you must see, it contains code, libraries, schematics, and a PDF manual.



- **Code:** The Arduino code to make Rollarm work.
 - **Schematic:** The schematic of the Servo Control board.
 - **..User Manual.pdf:** Sloth's complete manual, including Arduino installation and basic use, Sloth's hardware assembly and code use.
- **How_to_use_Sloth_with_Mixly:** This is an extended use of Sloth via graphical programming software - Mixly to program. The manual contains only a basic introduction to Mixly and how to program in Mixly, but does not include hardware assembly part, you need to download DIY_4-DOF_Robot_Kit_-_Sloth.zip first, and after Sloth is completely assembled, you can use Mixly for programming.



- **Dofrobot:** The library that you need to add in the Mixly.
- **Driver:** The Nano board driver software for Windows and Mac OS X system.
- **..User_Manual.pdf:** The user manual of how to program in Mixly.

Install Arduino IDE and Add Libraries

The Arduino IDE, known as Arduino Integrated Development Environment, provides all the software support needed to complete an Arduino project. It is a programming software specifically designed for Arduino, provided by the Arduino team, that allows us to write programs and upload them to the Arduino board.

The Arduino IDE 2.0 is an open-source project. It is a big step from its sturdy predecessor, Arduino IDE 1.x, and comes with revamped UI, improved board & library manager, debugger, auto complete feature and much more.

In this tutorial, we will show how to download and install the Arduino IDE 2.0 on your Windows, Mac, or Linux computer.

Download 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 website: <https://www.arduino.cc/en/software>, find the one that suits your operation system and click to download.

[HARDWARE](#) [SOFTWARE](#) [CLOUD](#) [DOCUMENTATION](#) [COMMUNITY](#) [BLOG](#) [ABOUT](#)



Arduino IDE 2.0.3

The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger.

For more details, please refer to the [Arduino IDE 2.0 documentation](#).

Nightly builds with the latest bugfixes are available through the section below.

DOWNLOAD OPTIONS

Windows Win 10 and newer, 64 bits

Windows MSI installer

Windows ZIP file

Linux AppImage 64 bits (X86-64)

Linux ZIP file 64 bits (X86-64)

macOS Intel, 10.14: "Mojave" or newer, 64 bits

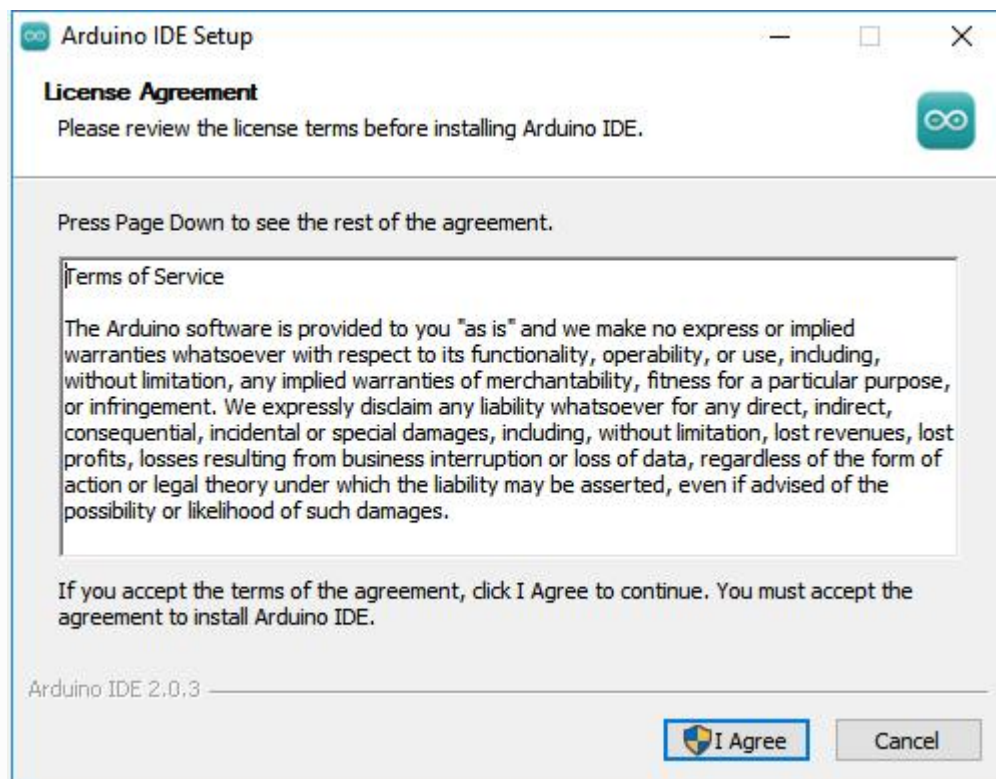
macOS Apple Silicon, 11: "Big Sur" or newer, 64 bits

Installation

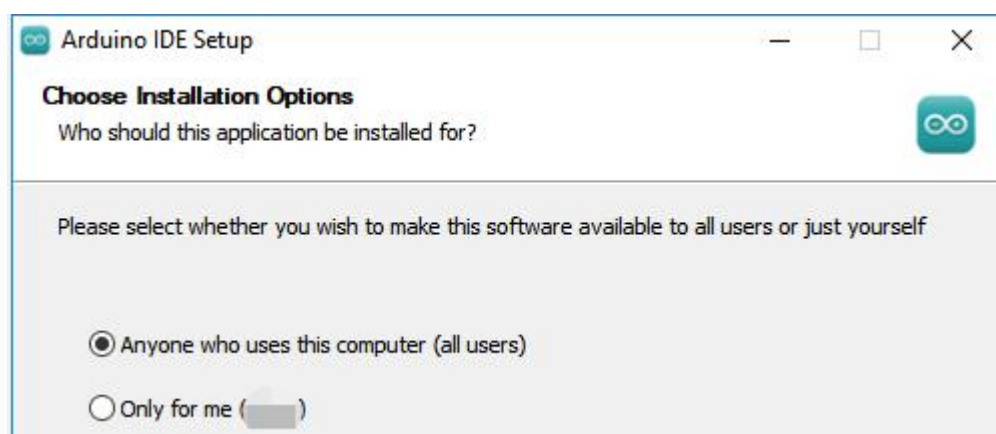
- Windows - Win 10 and newer, 64 bits
- Linux - 64 bits
- Mac OS X - Version 10.14: "Mojave" or newer, 64 bits

➤ Windows

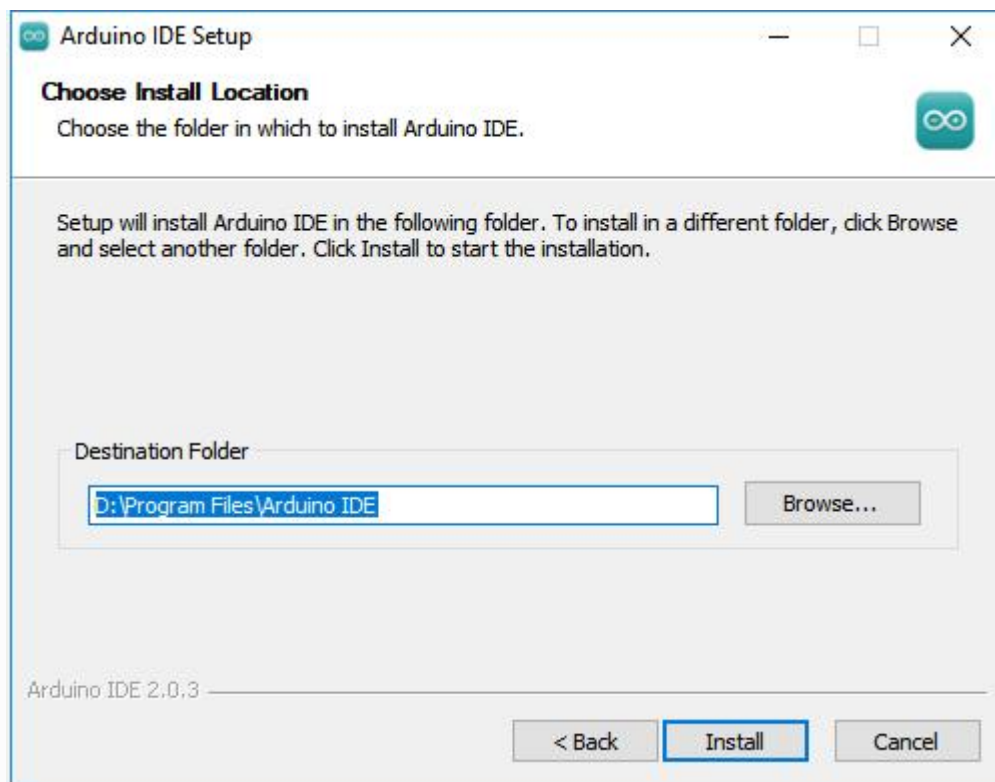
Double click the *arduino-ide_xxxx.exe* file to run the downloaded file.
Read the License Agreement and agree it.



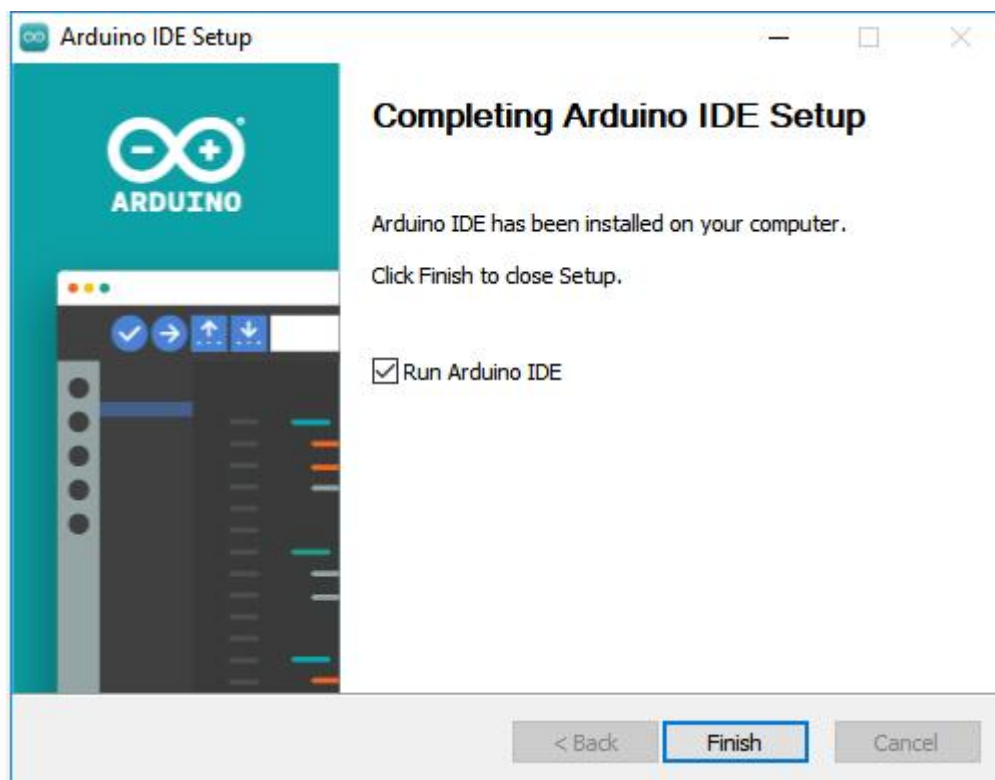
Choose installation options.



Choose install location. It is recommended that the software be installed on a drive other than the system drive.

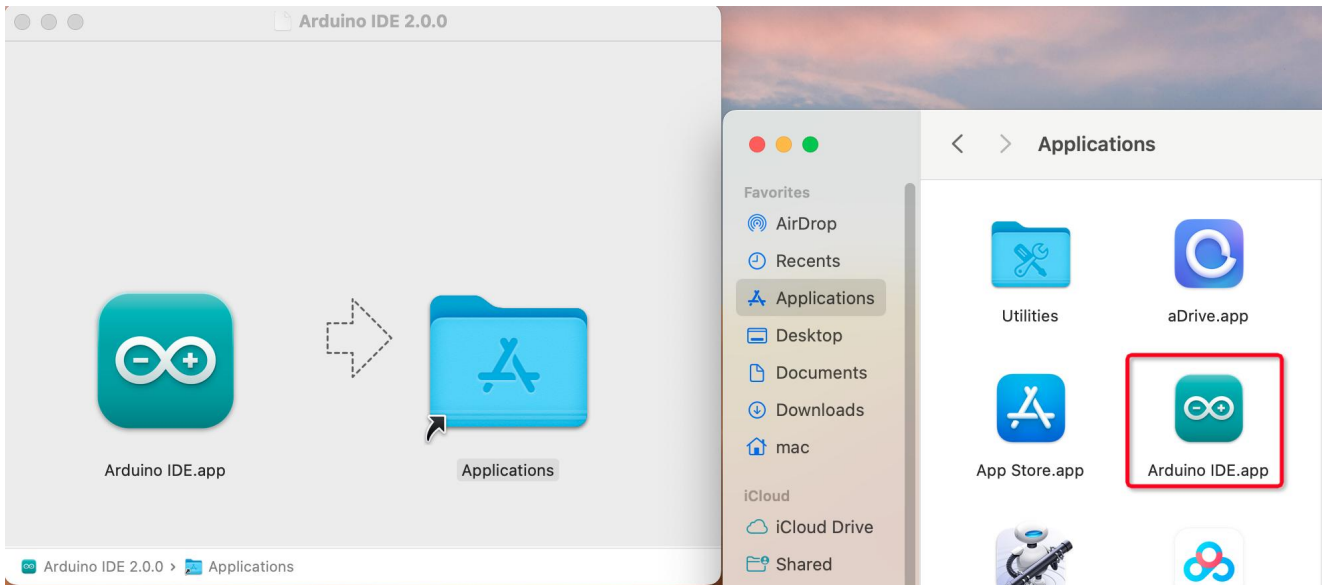


Then Finish.



➤ macOS

Double click on the downloaded *arduino_ide_xxxx.dmg* file and follow the instructions to copy the **Arduino IDE.app** to the **Applications** folder, you will see the Arduino IDE installed successfully after a few seconds.



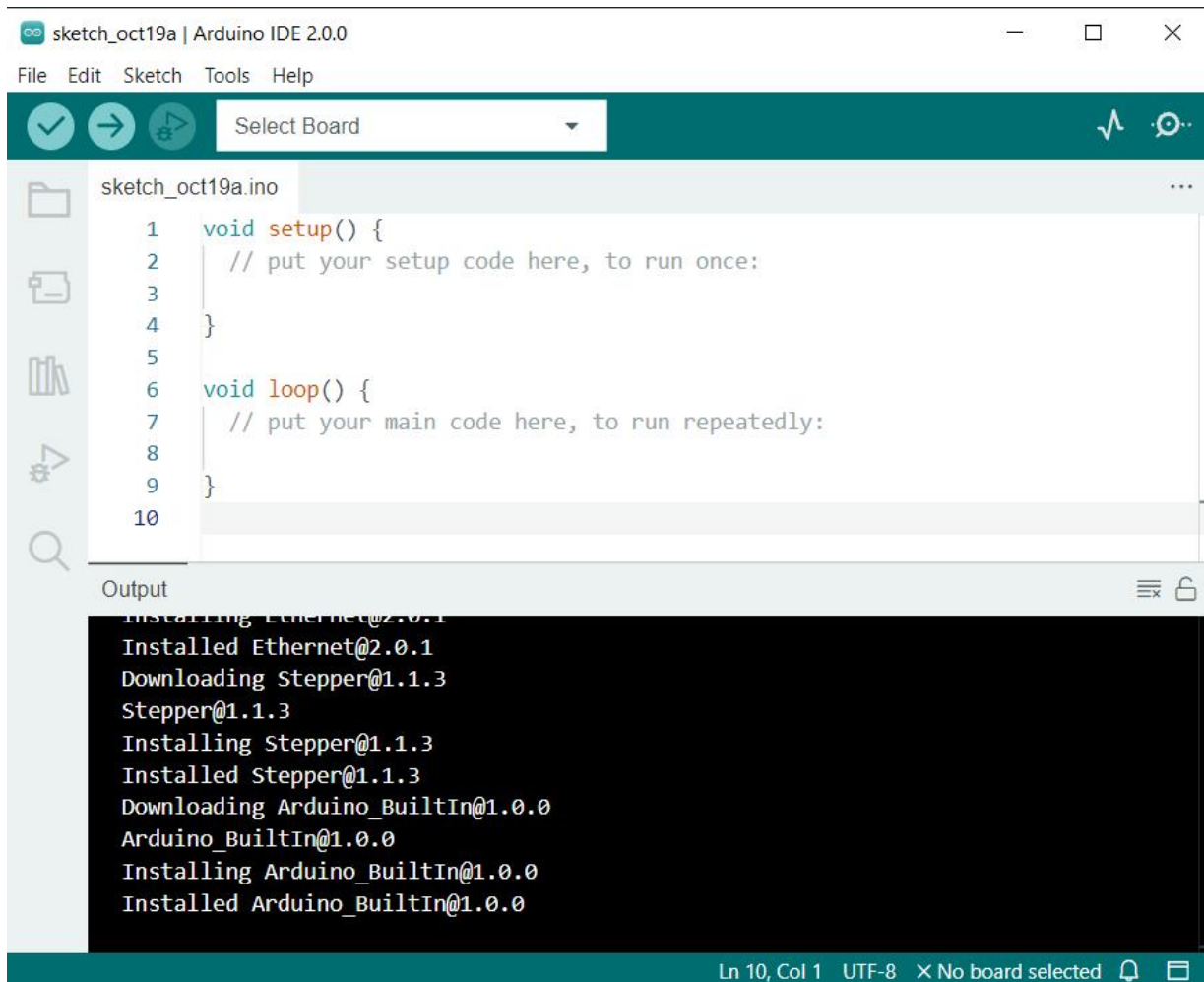
➤ Linux

For the tutorial on installing the Arduino IDE 2.0 on a Linux system, please refer to:

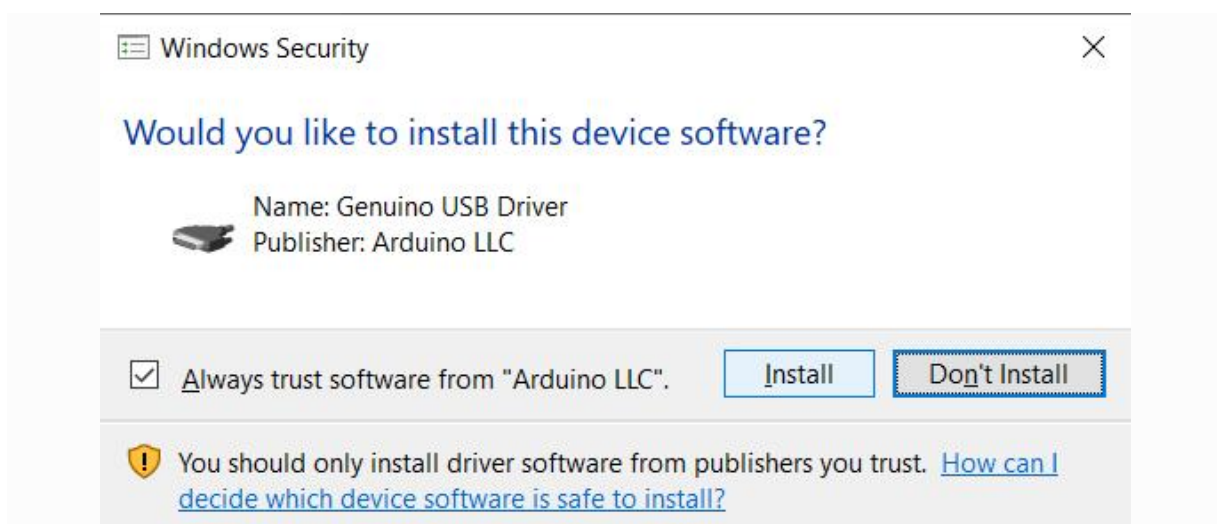
<https://docs.arduino.cc/software/ide-v2/tutorials/getting-started/ide-v2-downloading-and-installing#linux>

Open the IDE

When you first open Arduino IDE 2.0, it automatically installs the Arduino AVR Boards, built-in libraries, and other required files.



In addition, your firewall or security center may pop up a few times asking you if you want to install some device driver. Please install all of them.



Now your Arduino IDE is ready!

Note: In the event that some installations didn't work due to network issues or other reasons, you can reopen the Arduino IDE and it will finish the rest of the installation.

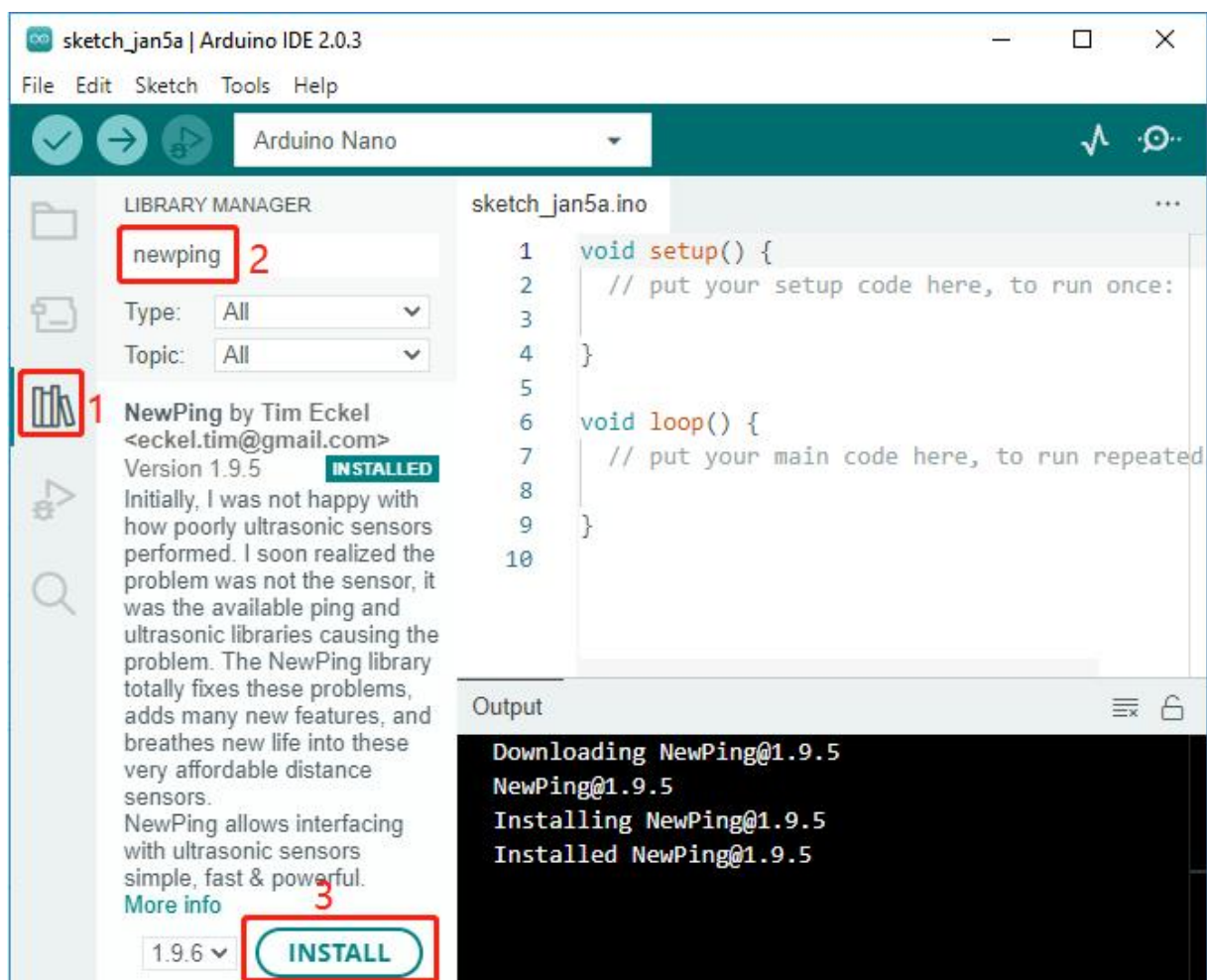
Add Libraries

You will get the following error when trying to run the code that contains the ultrasonic module:

NewPing.h: no such file or directory

Therefore, it needs to be manually added. Here are the steps.

Open the Arduino IDE, click on the **Library Manager** icon to open it, search for *newping* and click on **INSTALL** in the options that appear. Wait for the prompt to complete the installation and you're done!

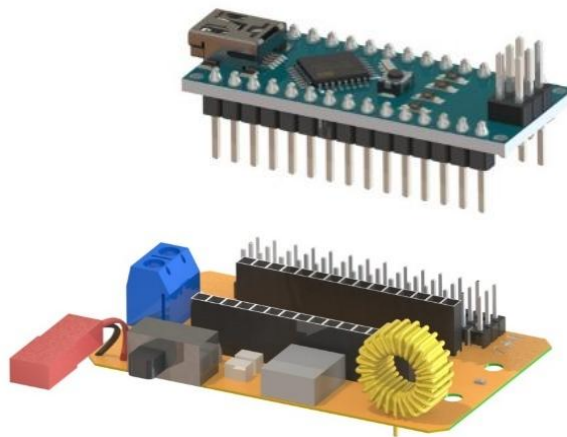


Test Servos and the Ultrasonic Module

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

Test the Servo

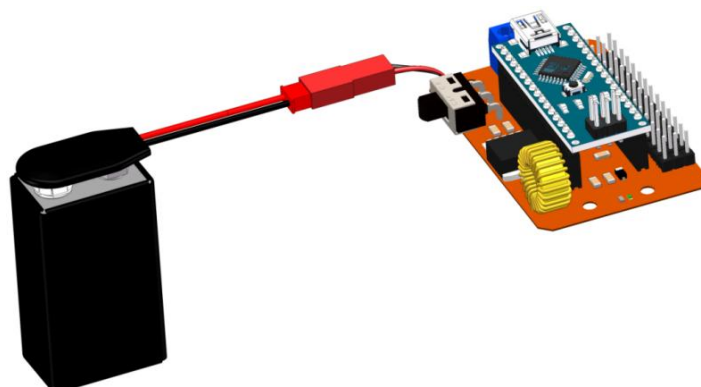
Step 1: Insert the Nano board into the Servo Control Board. **Note:** The USB port should be at the same side with blue power supply terminal.



Step 2: Insert the battery to the battery cable.

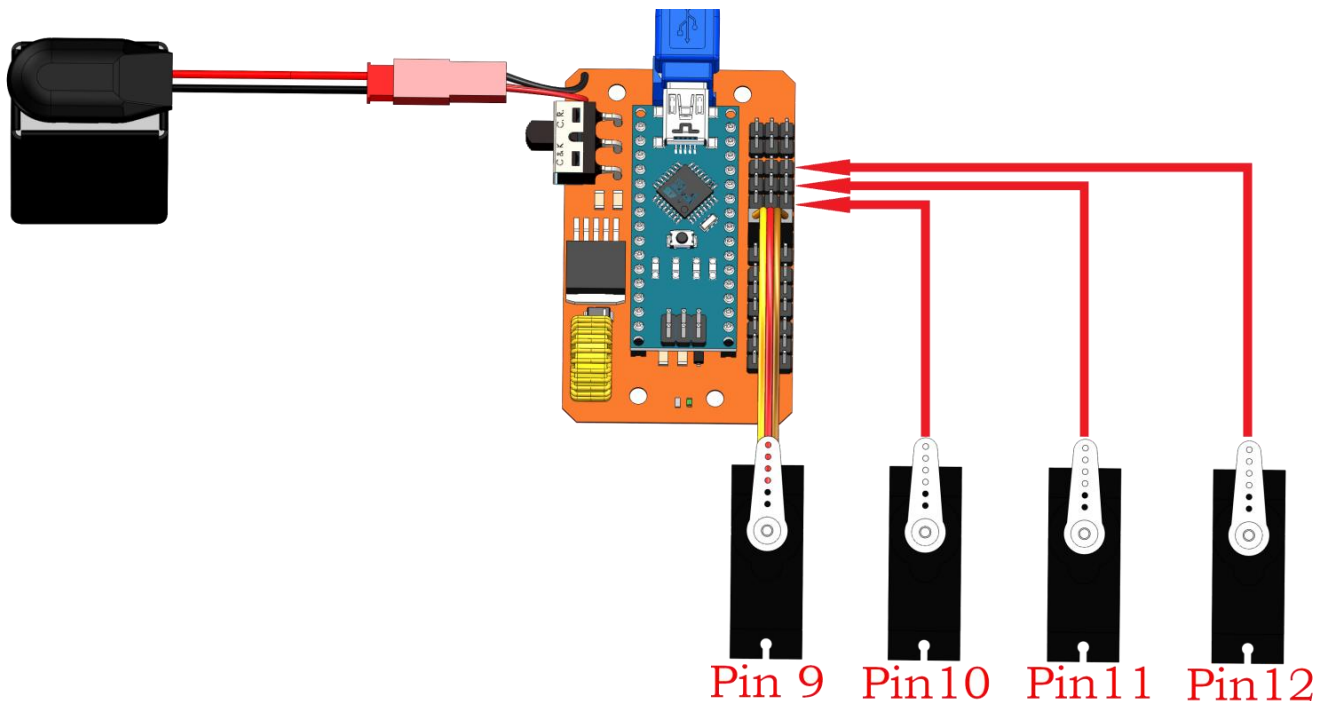


And connect the battery cable to the expansion board.



Step 3: Connect four servos to pin 9 to pin 12 of the expansion board.

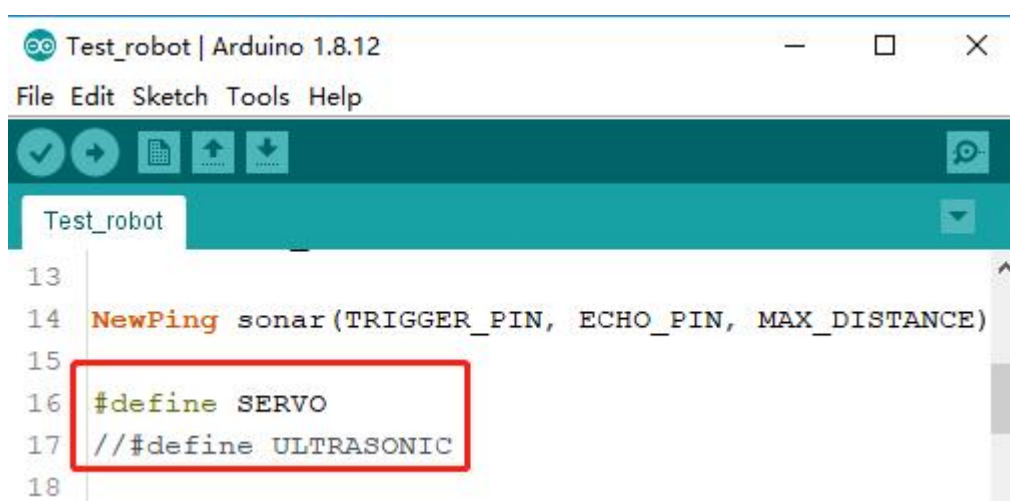
Note: The **yellow**, **red**, and **brown** wires connect to **Signal**, **VCC**, and **GND** on the expansion board, respectively.



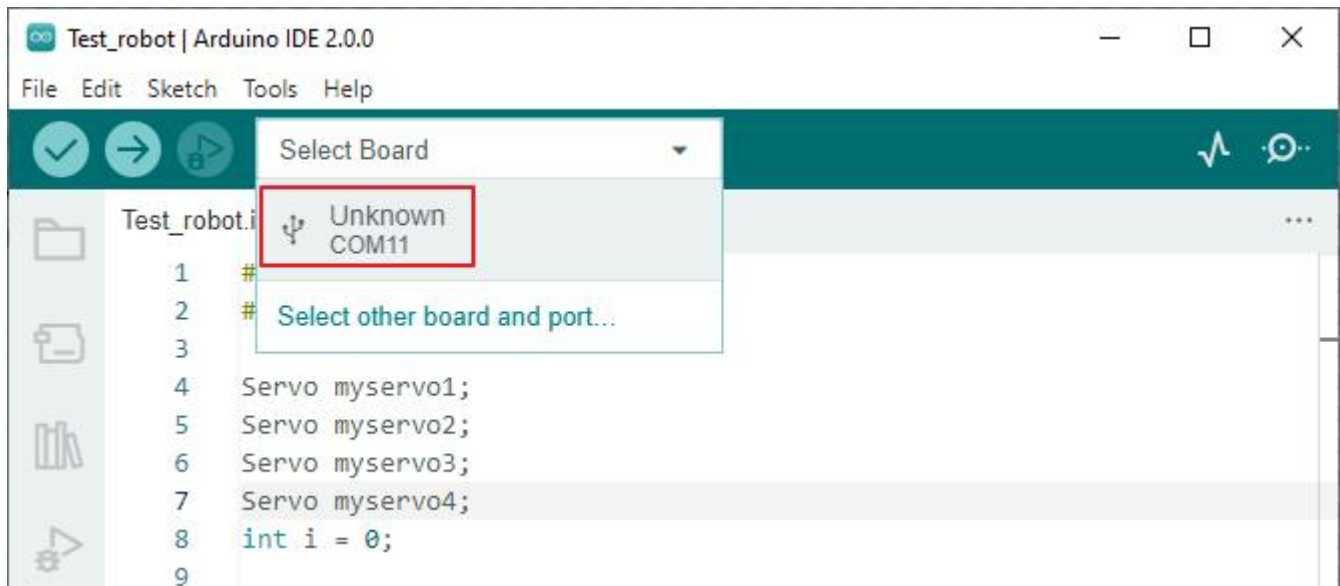
Step 4: Open the *Test_robot.ino* under this path of *DIY_4-DOF_Robot_Kit_-_Sloth\Code\Test_robot*.

Uncomment the line 16(delete sign "//" to start the corresponding servo test code; the comment the line: *//#define ULTRASONIC*.

Note: It is not recommended to uncomment both lines at the same time.

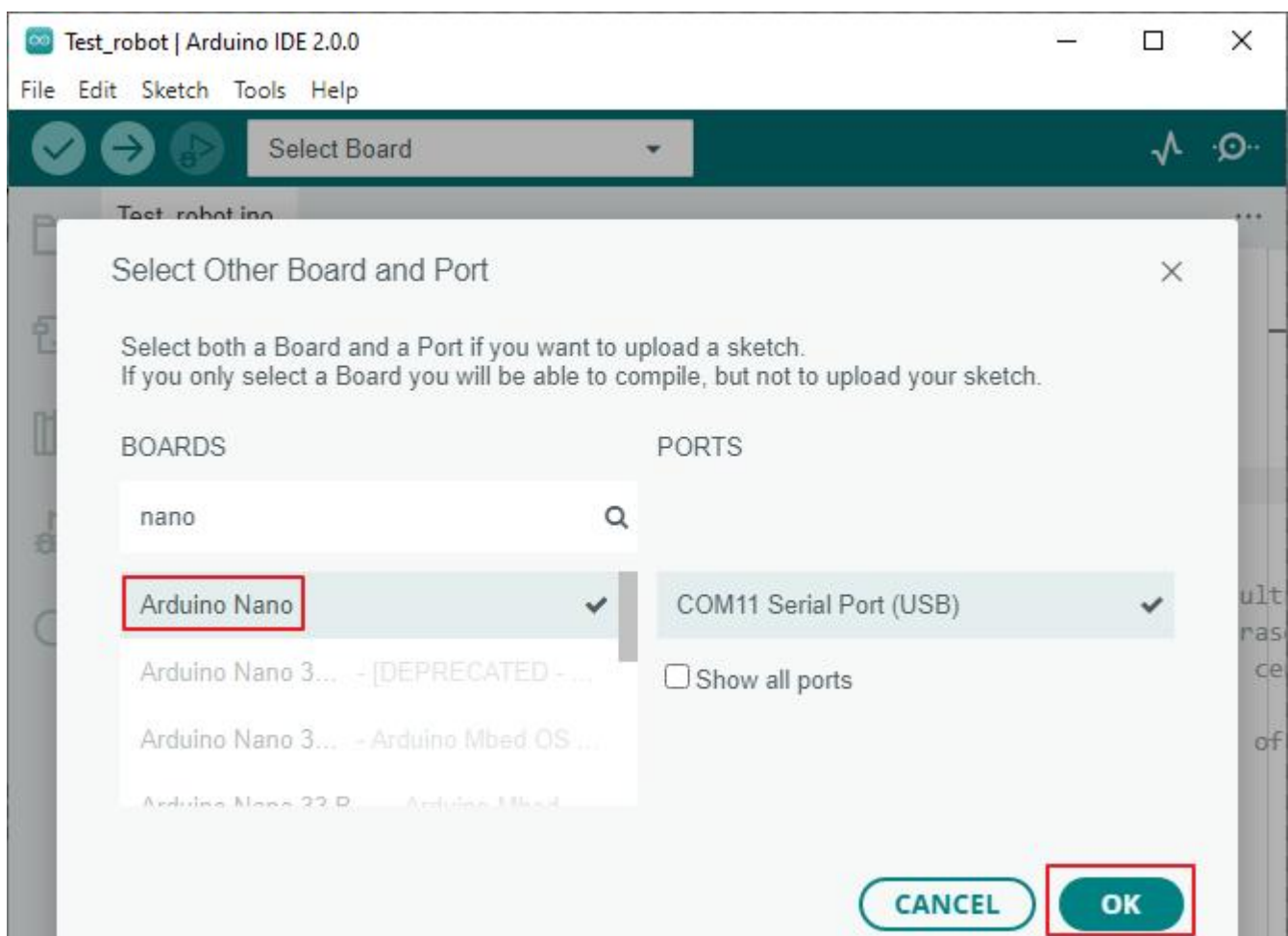


Step 5: The Arduino IDE may not recognize your board if you see **Unknown COMXX** when selecting a board here.

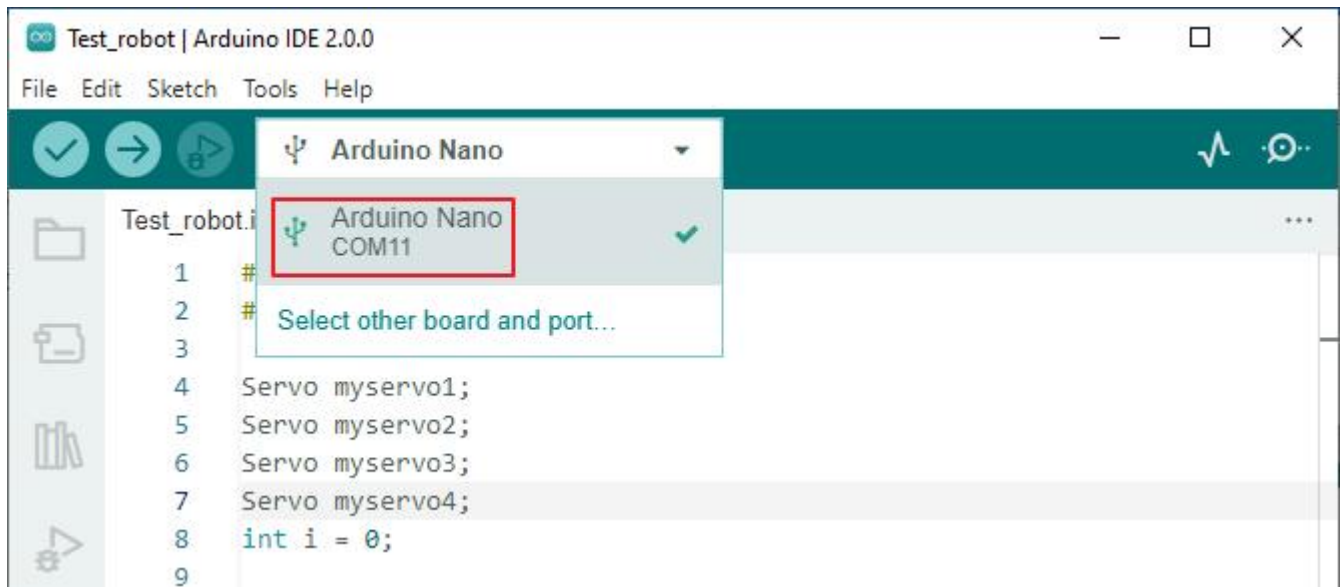


Step 6: After clicking on **Unknown COMXX**, the **Select Other Board and Port** pop-up will appear. Type *nano* in the search box, then select **Arduino Nano** and click **OK** to confirm.

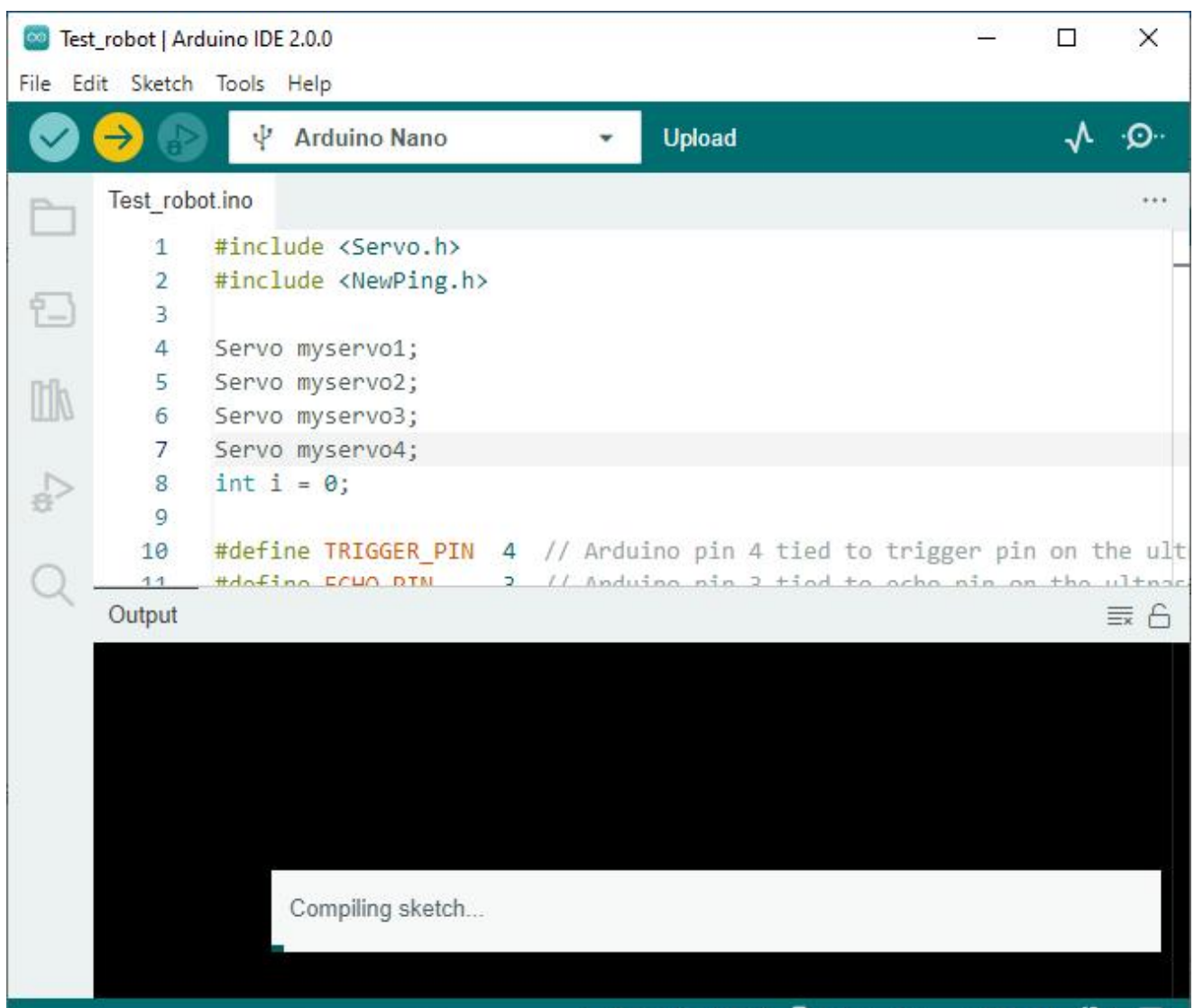
Note: It doesn't matter if your Arduino Nano is greyed out, it means you don't have the Arduino AVR Boards core installed, just select it here and click OK to install it automatically.



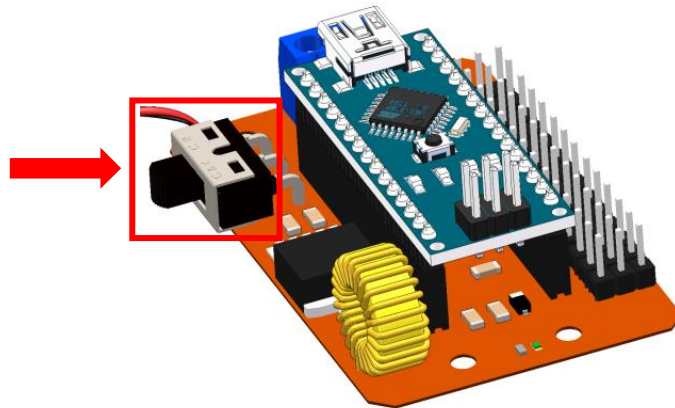
Step 7: Now you will be able to see the Arduino Nano board being recognized.



Step 8: Upload codes to the Nano board. After a few seconds, the "Done uploading" message will appear.

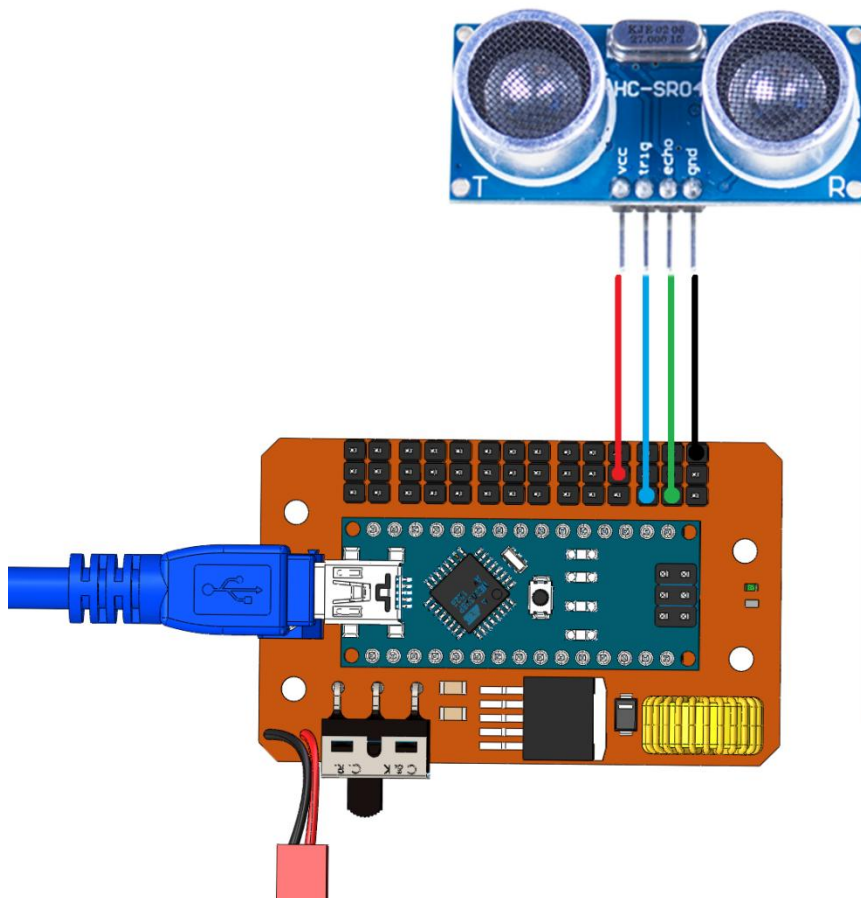


Step 9: Slide the power switch to ON. You will see the rocker arm rotates within 0-180 degrees, indicating the servo can work.



Test the Ultrasonic Module

Step 1: Connect Ultrasonic module to Servo Control Board via 4-Pin Anti-reverse Cable.



Ultrasonic Module:

TRIG -> PIN 4

ECHO -> PIN 3

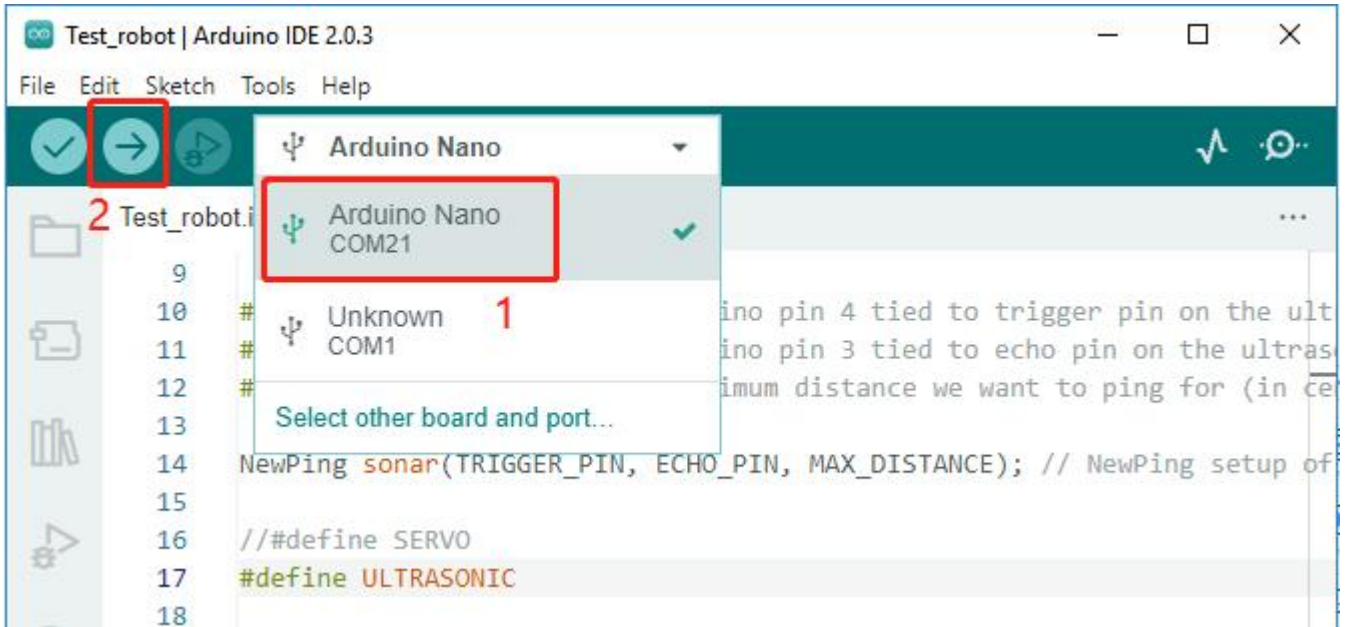
VCC -> VCC

GND -> GND

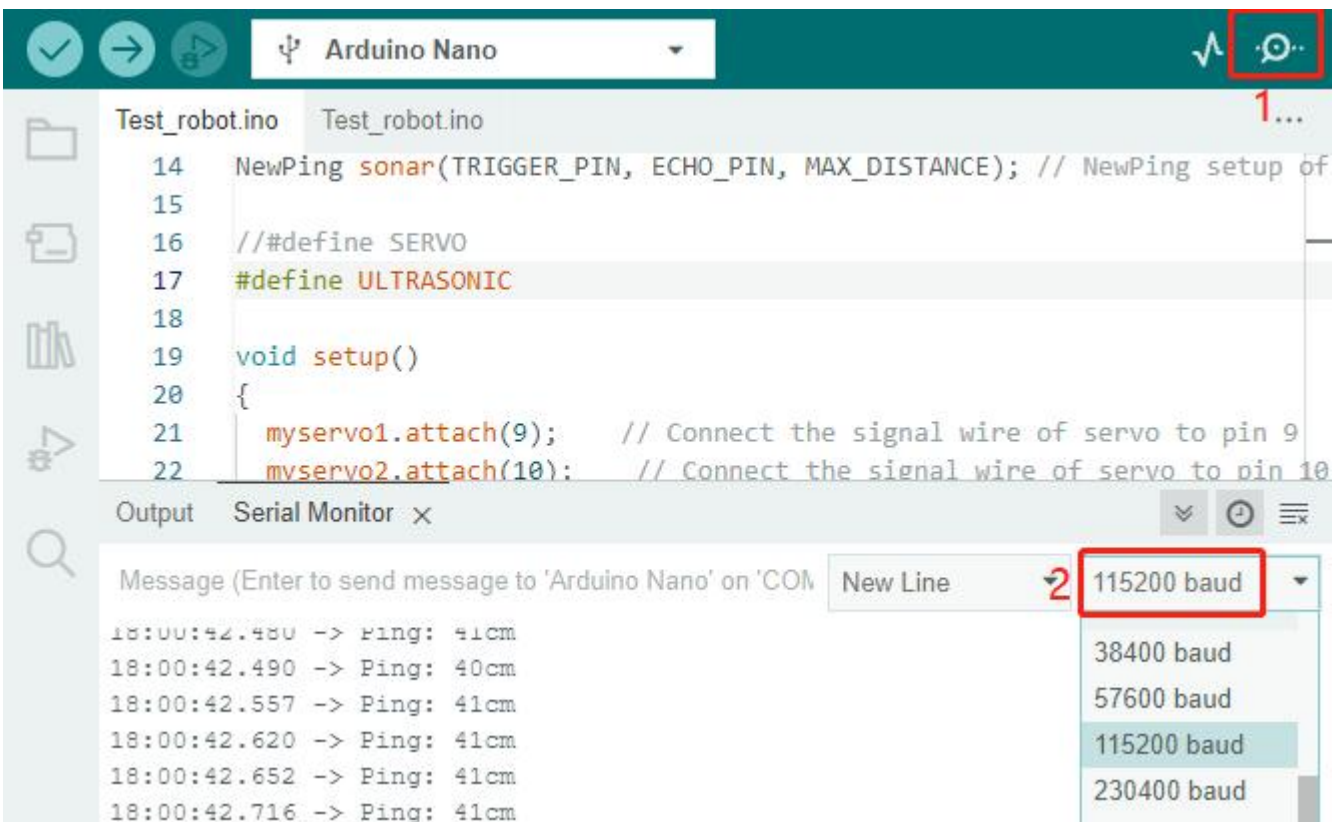
Step 2: Open the *Test_robot.ino* and select Board, Processor and Port.

Step 3: Comment out line 16 by prefixing *#define SERVO* with *"/"*; then uncomment *#define ULTRASONIC*.

Step 4: Open the serial monitor after uploading the code.



Step 5: Set the baud rate to 115200 (started by line 25 *serial.begin (115200)*).



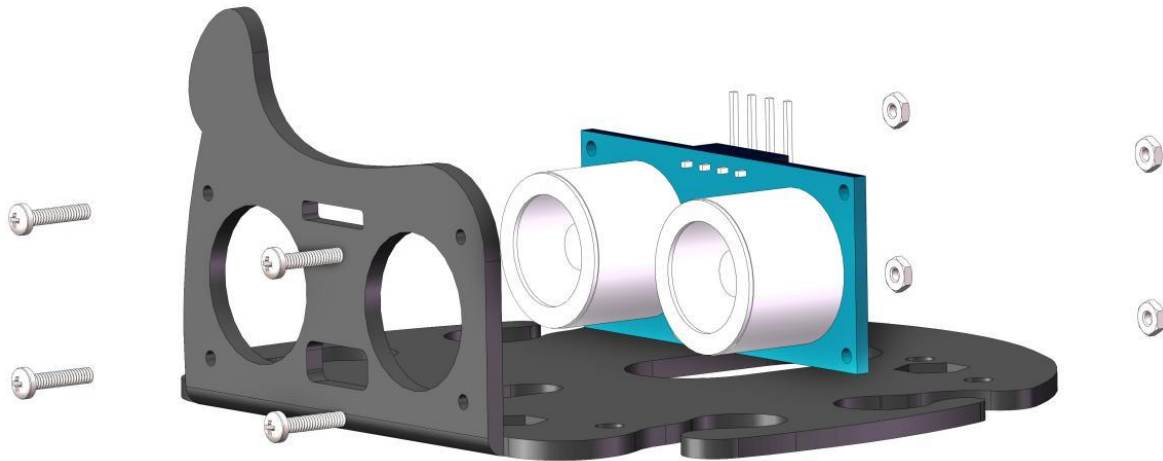
Step 6: Turn the power switch to ON you can see the detected distance.

Note: The detection distance of ultrasonic module is 2-400cm, if the data is 0 or a few thousand, it means that it is invalid data need to be ignored.

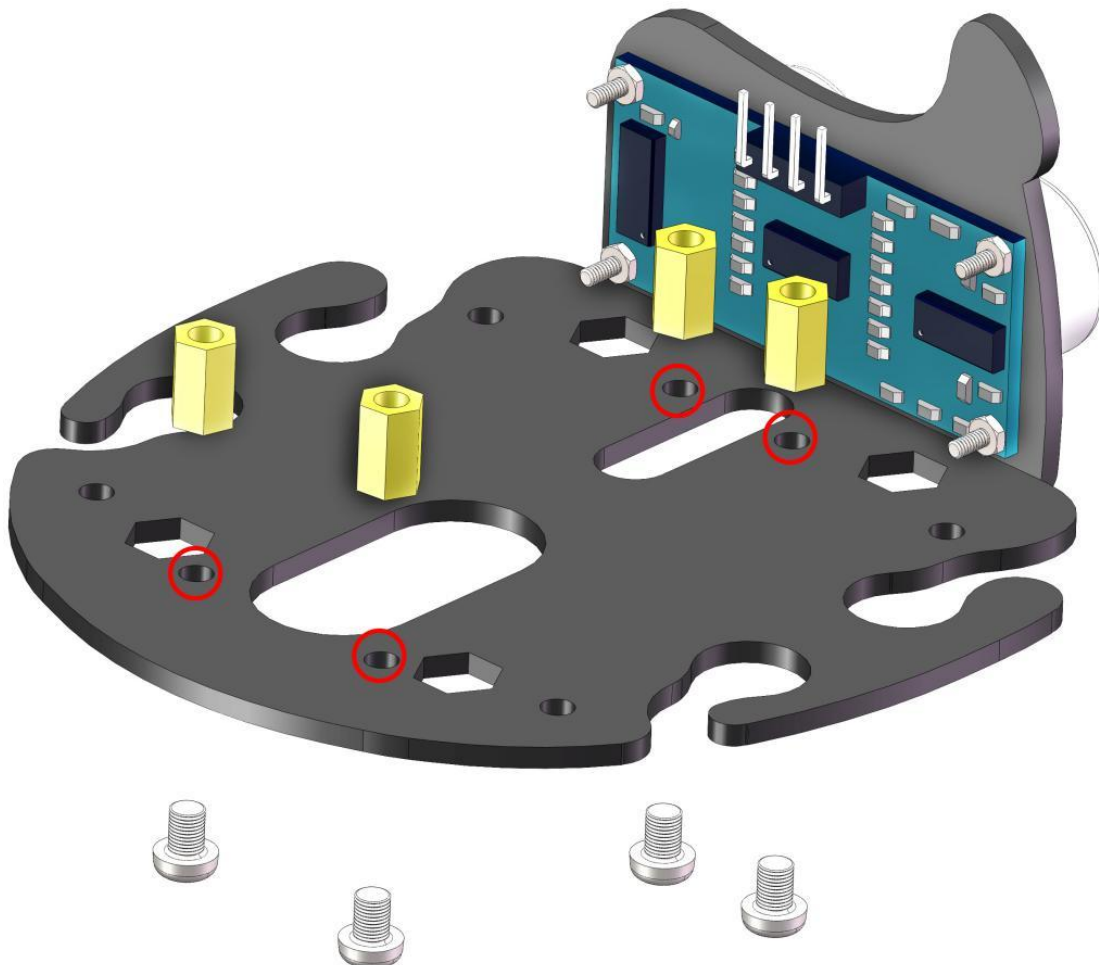
Assembly

Head Assembly

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

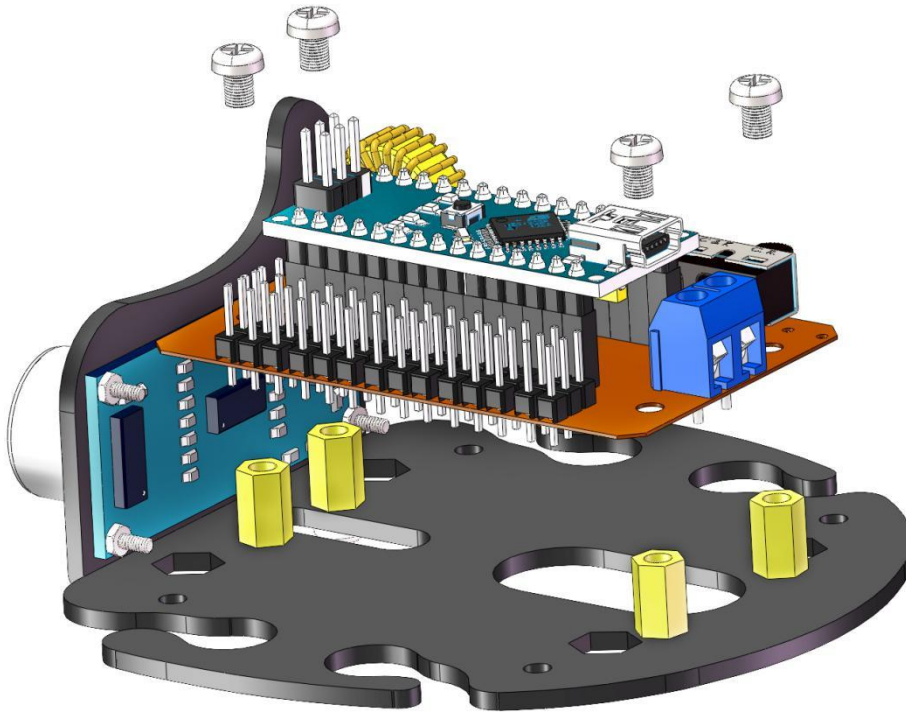


Use a **M3*5 screw** to secure the **M3*8 Bi-pass Copper Standoff** post on No. 1 board.

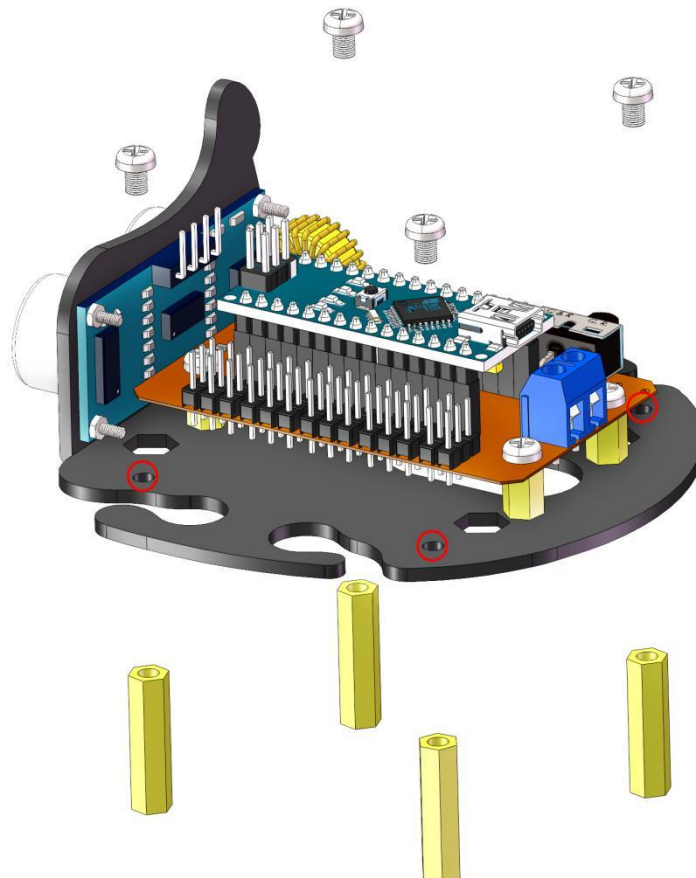


Electrical Module Assembly

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

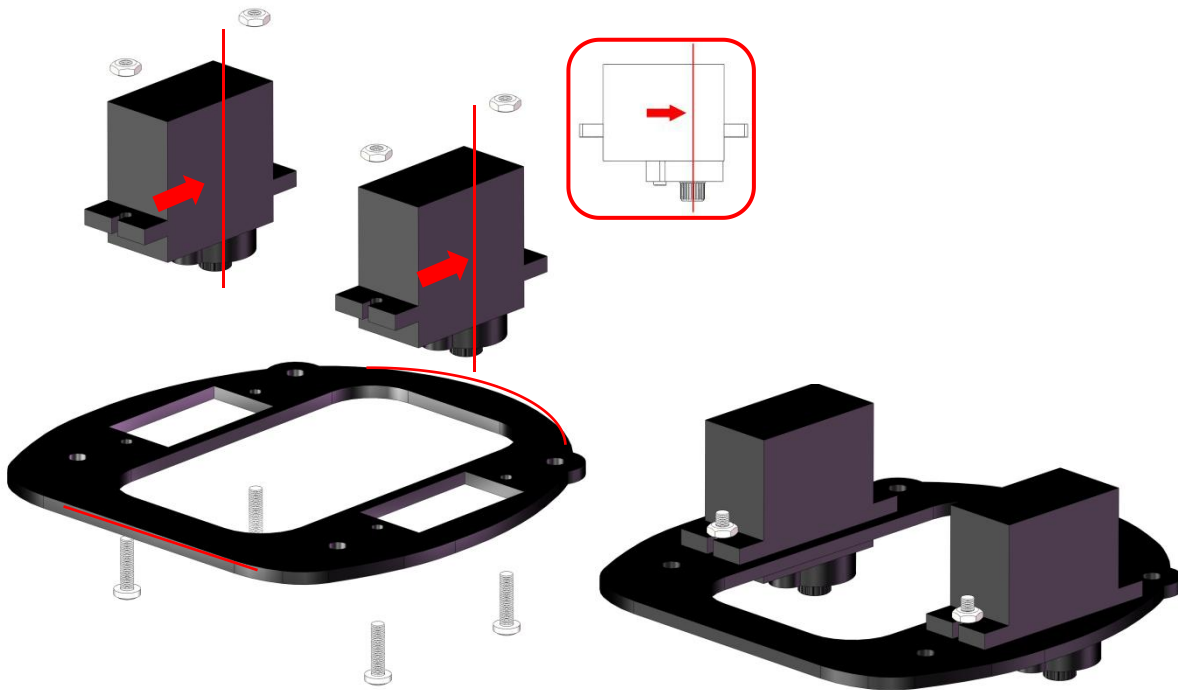


Use **M3*5 screws** to fix **M3*25 Bi-pass Copper Standoff** under the No. 1 board.

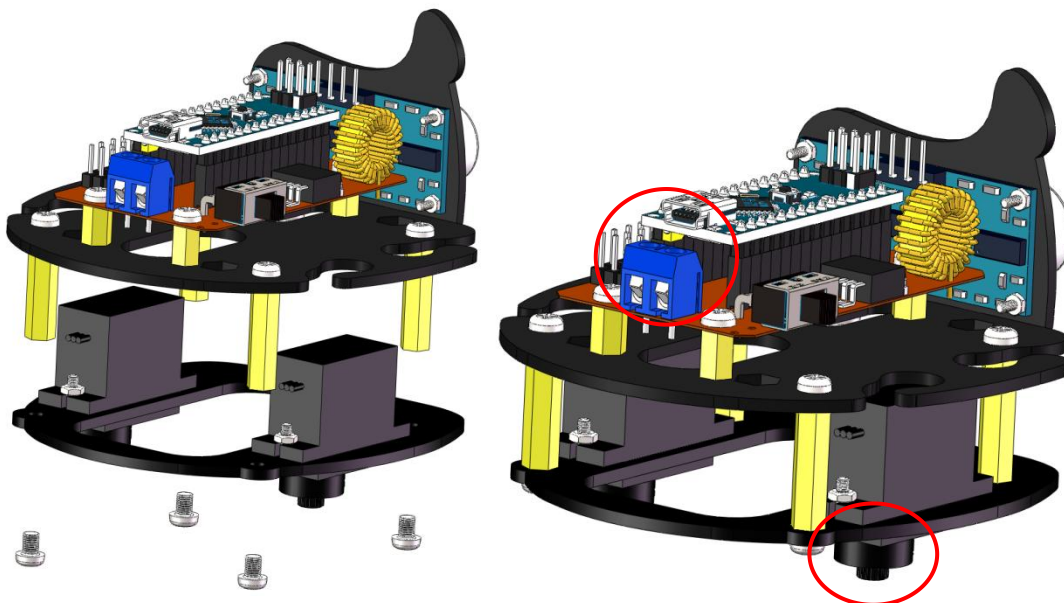


Servo Assembly

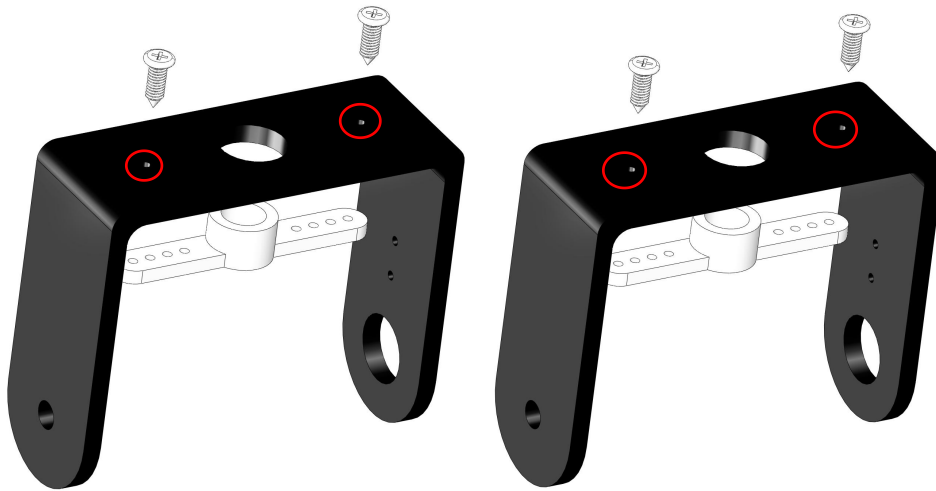
Use **M2*8 screws** and **M2 nuts** to mount the servo on the corresponding position on the No. 2 board. (Note the direction of the servo installation)



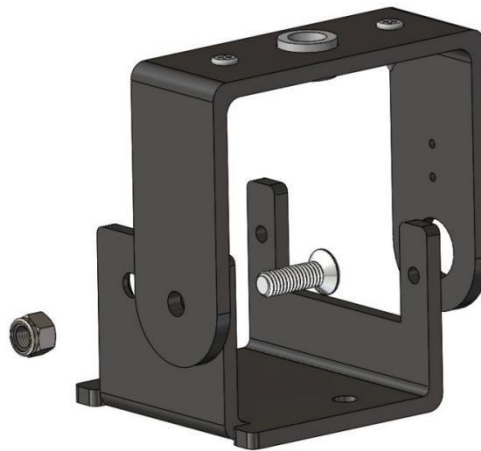
Secure the No. 1 and No. 2 boards with **M3*5 screws**. Note that the side of the servo shaft should be mounted on the side of the USB port.



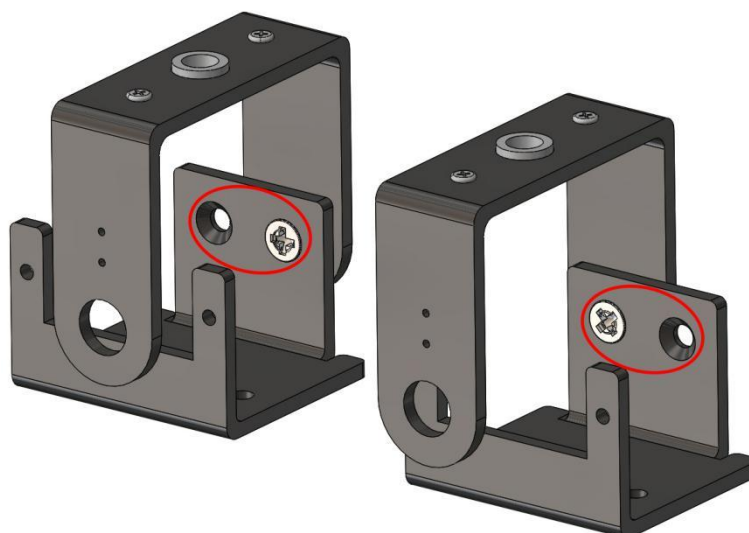
Use two **M1.5*5 self-tapping screws** to fix the 2-arm rocker arm to the No. 4 board and use the same method to install another No. 4 board.



Secure one of the round holes on the 4th and 5th boards with **M3*8 Countersunk screws** and **M3 self-locking nuts**.



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



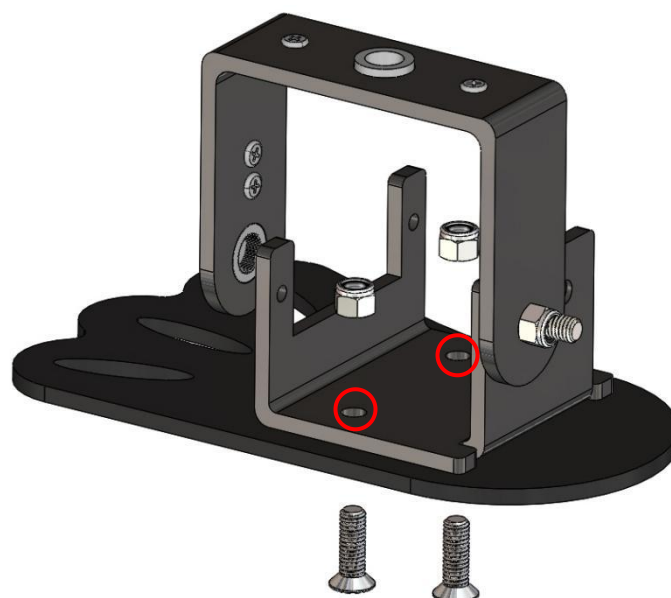
Use two **M1.5*5 self-tapping screws** to secure the 1-arm rocker arm on the No.4 board.



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 **right leg** described above with the **M3*8 countersunk screw** and the **M3 self-locking nut**.

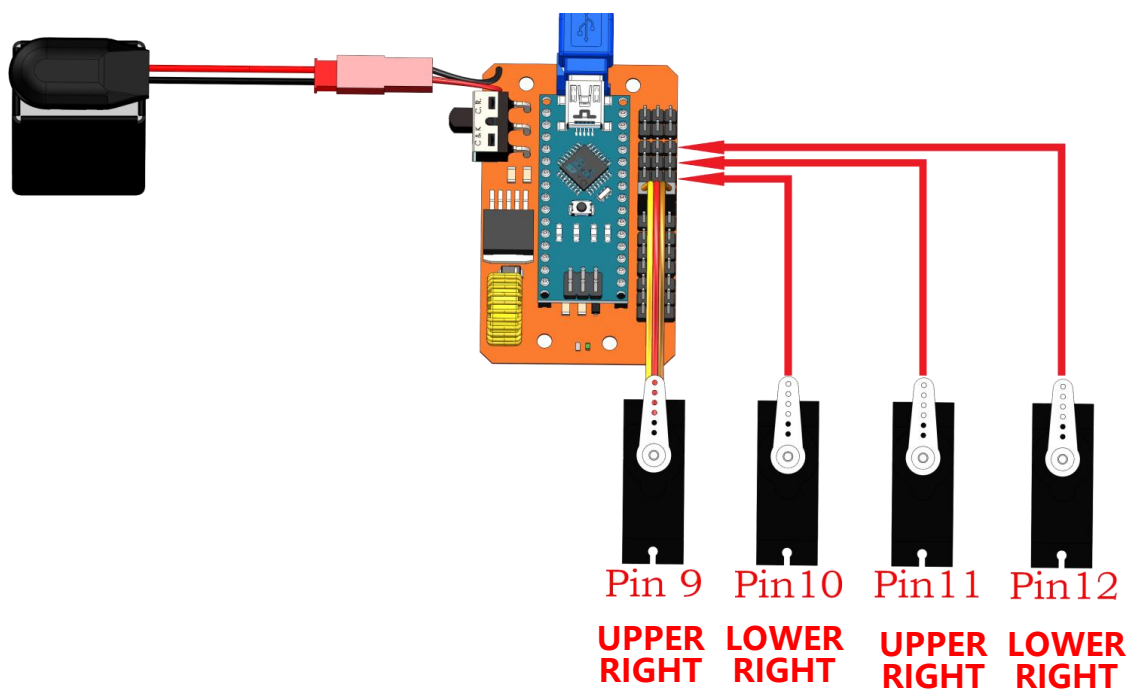


The same method can be used to secure the No.7 and the **left leg**. Observe the picture carefully. The left and right feet you have installed need to be exactly the same as that in the picture. Otherwise, the robot won't walk properly.

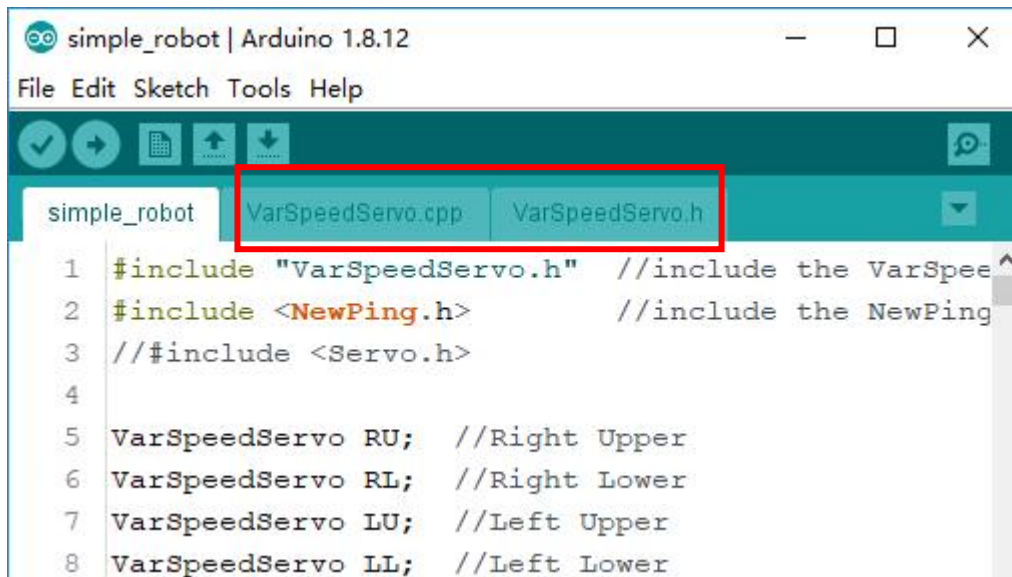


Servo INSTALL Test

Connect the 4 servos to pin 9, 10, 11 and 12 respectively again. This is designed to keep the servo angle of the upload code at 90°(internal angle) before the servo shaft is installed, in order to let the **Sloth** remain upright after assembly.



Open the program *simple_robot.ino* under the path of *DIY_4-DOF_Robot_Kit_-_Sloth\Code\simple_robot*. After opening, you can see the other 2 files: *VarSpeedServo.cpp* and *VarSpeedServo.h* are opened at the same time. This two files are set to adjust the angle of the servo.

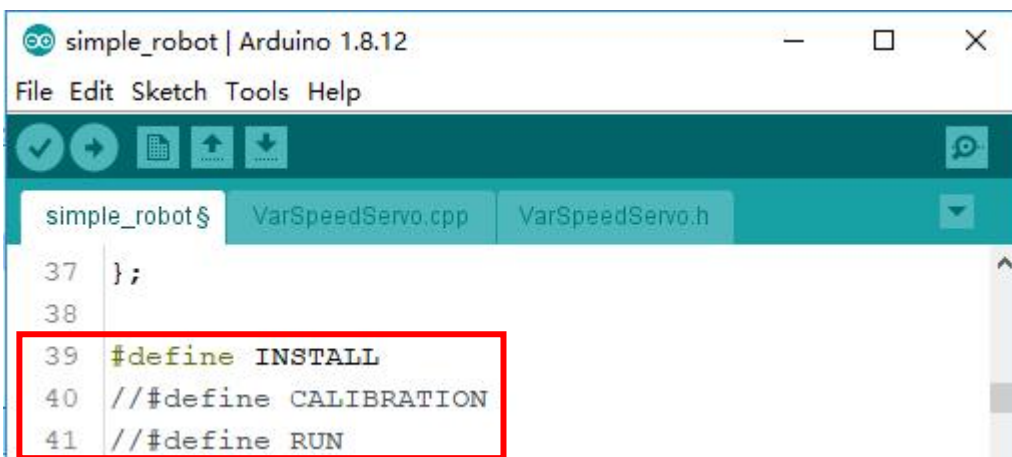


There are three **#define** statements in line 39-41. Removing the respective comment signs `"/"` enables you to start their functions as shown.

#define INSTALL: Start the INSTALL mode, in which 4 servos will be fixed at 90° for assembly.

#define CALIBRATION: Start the calibration mode, in which the angles of 4 servos can be adjusted.

#define RUN: Start the RUN mode, in which the robot can go ahead and get round if it meets obstacles.



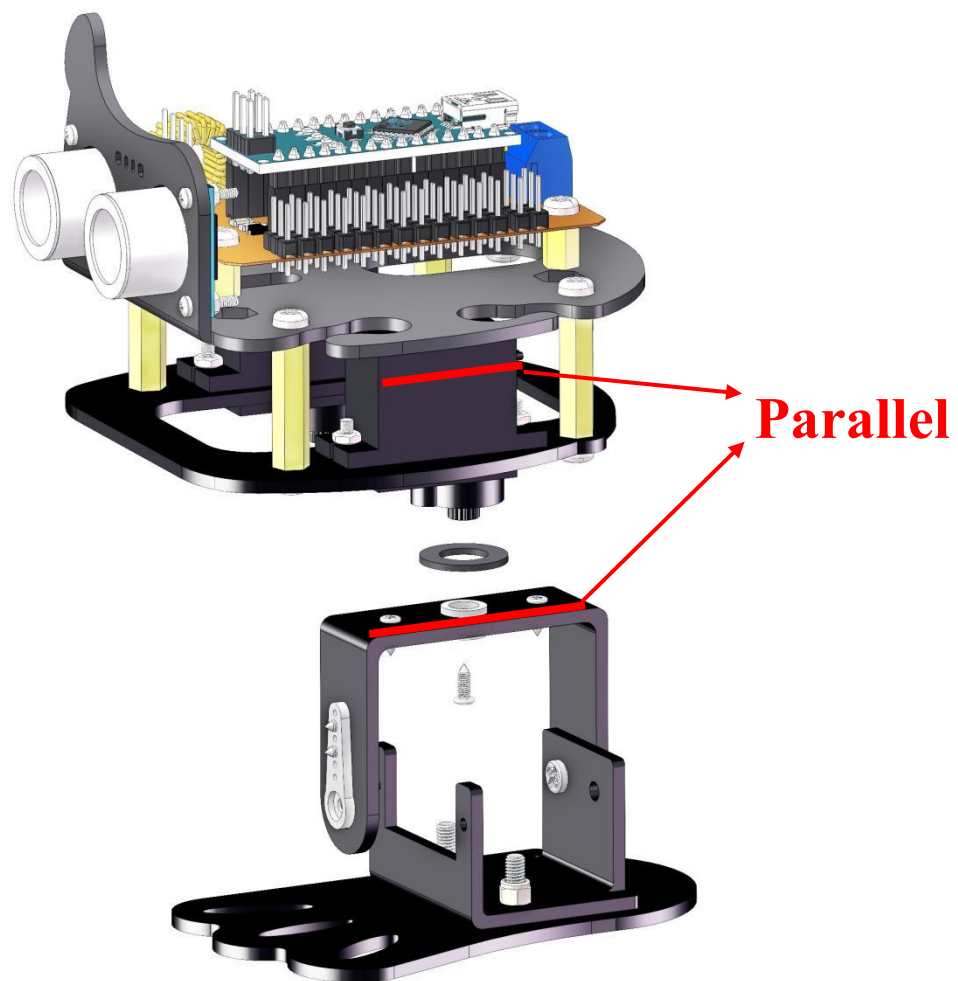
Note: Only one function can be used at the same time. Starting multiple functions might break down the robot.

In the current step, use INSTALL mode. Then select the corresponding **Board**, **Processor** and **Port**. The code is then uploaded into the **Nano** board. **Don't forget to toggle the power switch to ON**. When the servo control board is powered on, the servo will rotate to the position specified by the program.

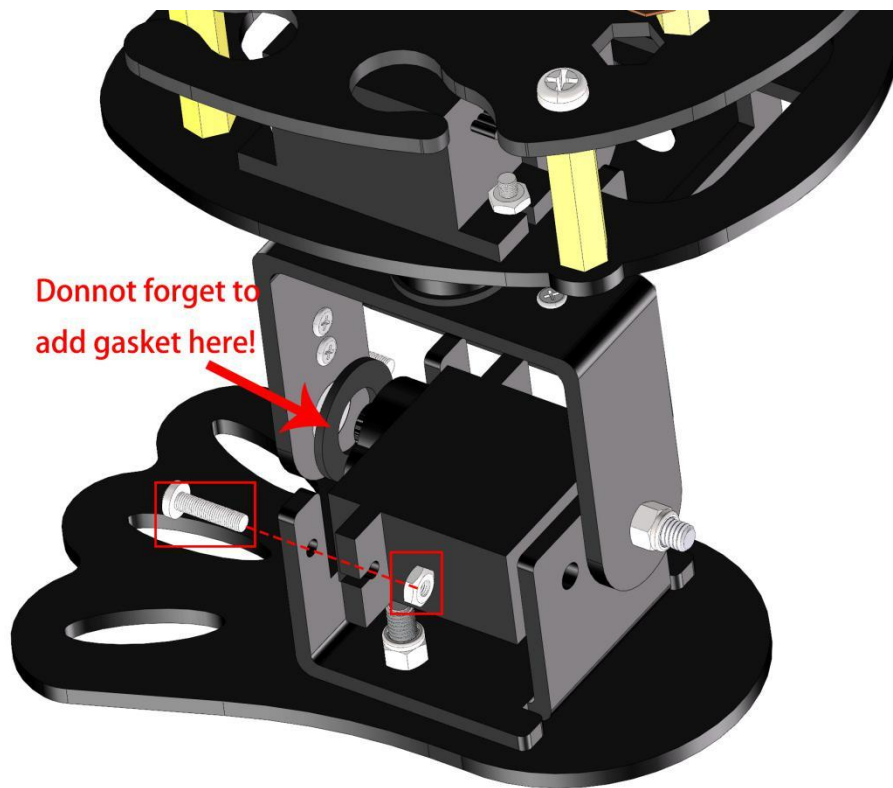
Foot Assembly

Note: Keep on the power until the whole step.

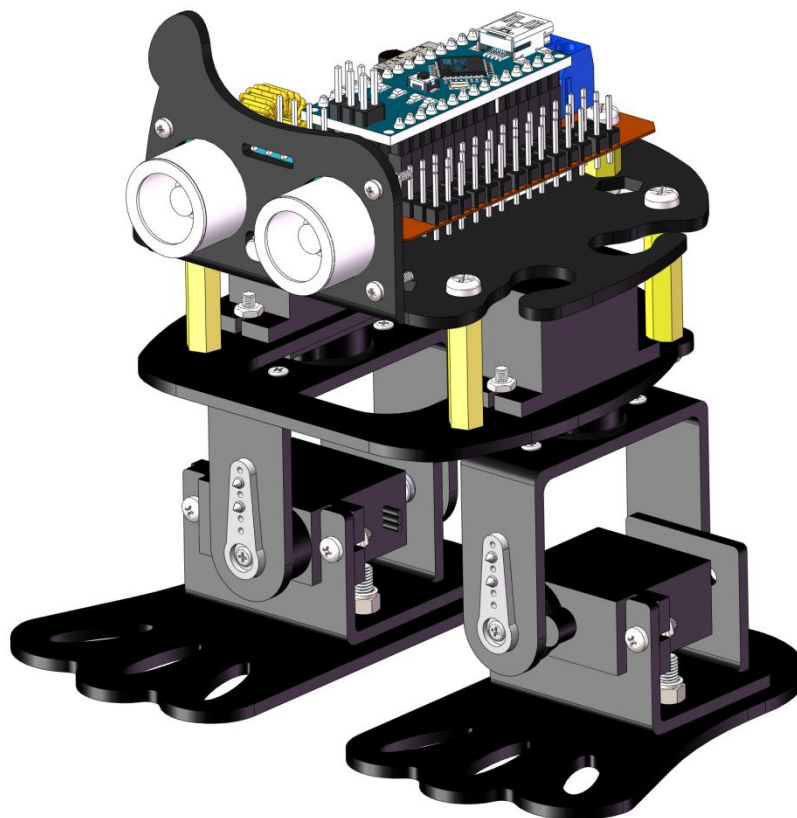
Assemble the left leg with the smallest screws in the packaged with servo, a gasket plate is needed between the servo and left leg. Try to keep the edges of the 4th board and the servo parallel to each other. If deviation are found at installation, it is normal and we will adjust them later when calibrating.



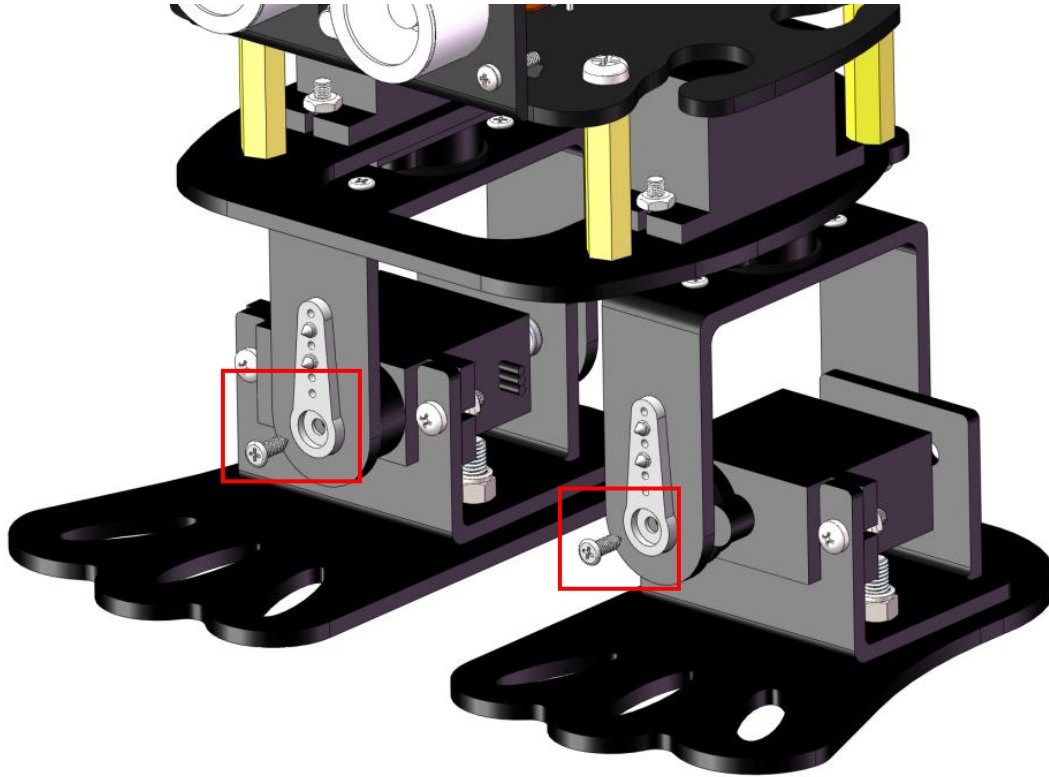
Insert a servo (in working condition) into the servo shaft of the left foot. Besides 2 M2*8 screws and 2 M2 nuts, a gasket plate is needed between the servo and left leg.



Assemble the right leg in the same way.

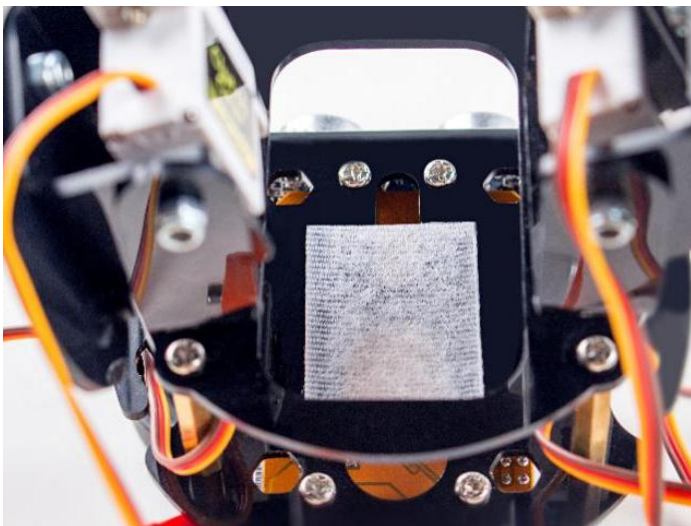


Secure the 2 legs with the smallest screws in the packaged with servo.

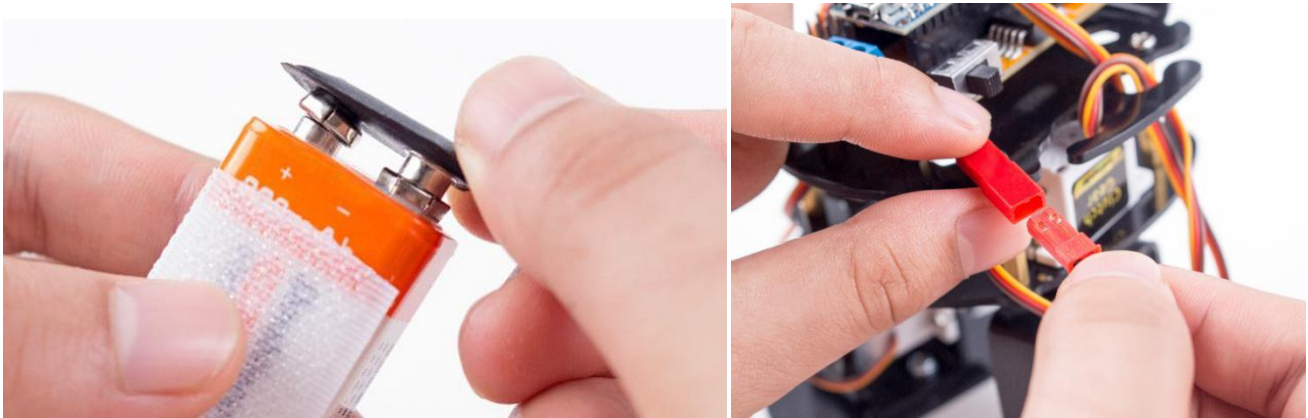


Battery Assembly

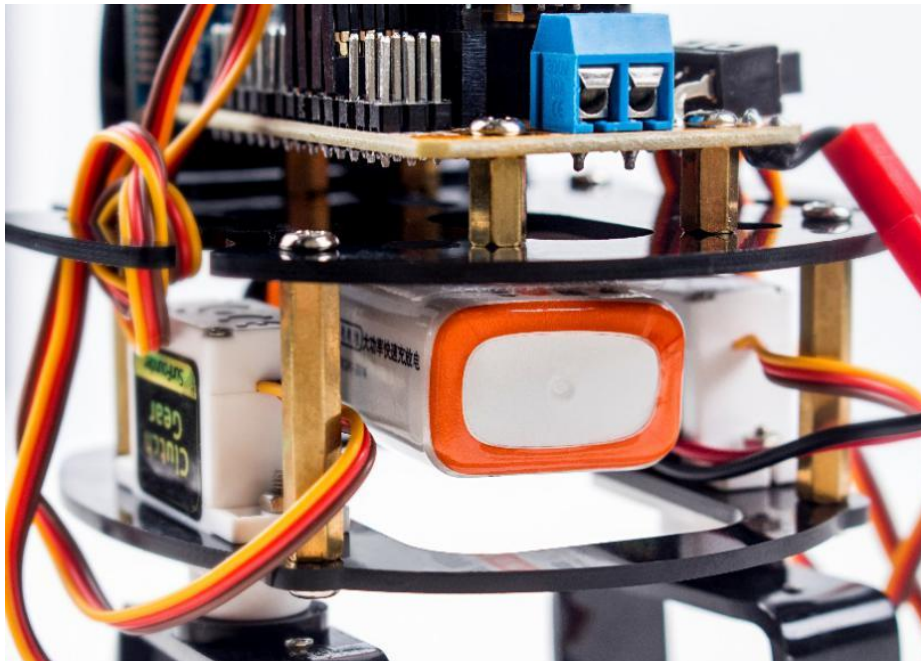
Attach one side of velcro tape to the bottom of the No. 1 board and the other side to the battery.



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

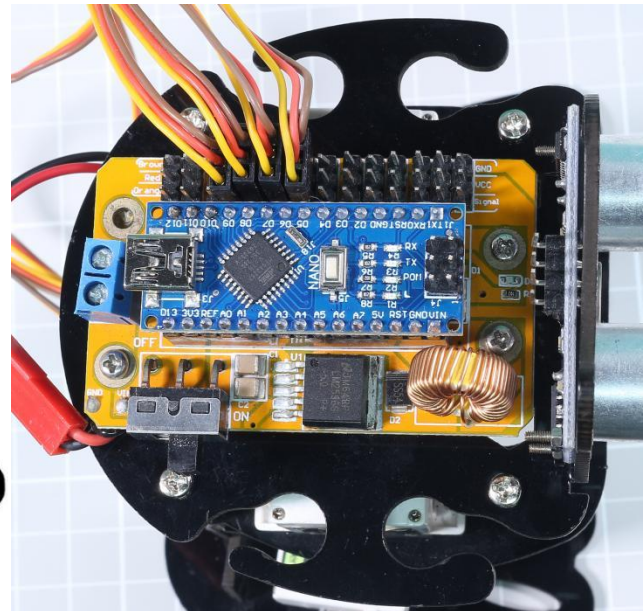
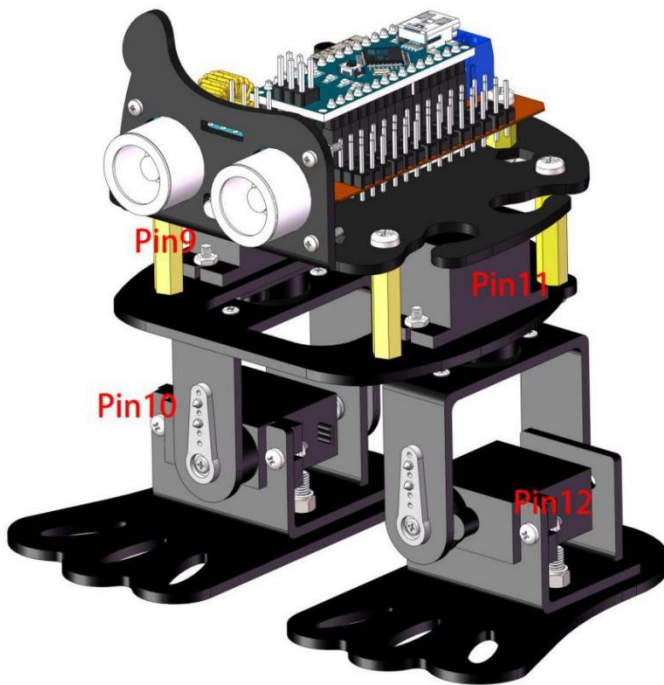


Lastly, paste the battery on the No. 1 board.

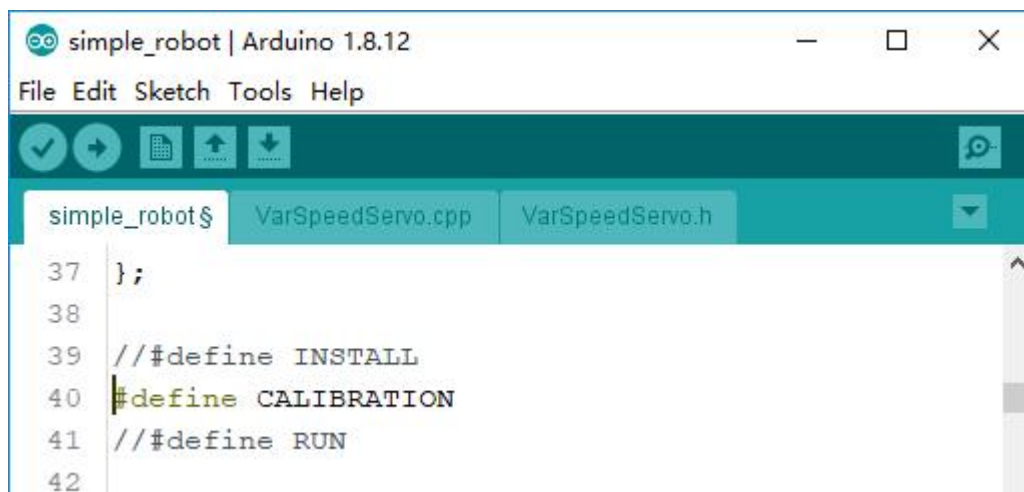


Servo CALIBRATION Test

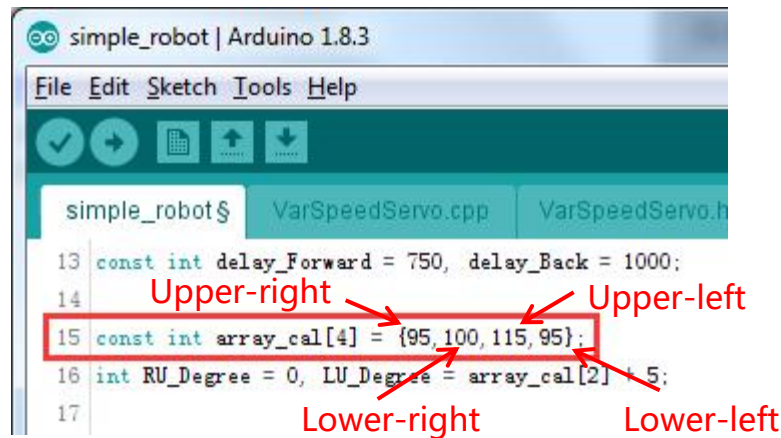
Check the assembly of the 4 servos according to the picture as shown.



Open the program *simple_robot.ino* and go to **Line 39**. Set *#define CALIBRATION* as *able* and disable the other two. Then select the correct board and port, and upload the sketch.



If the robot is not fully upright, the angle can be manually calibrated. Go to **Line 15** to rectify it.



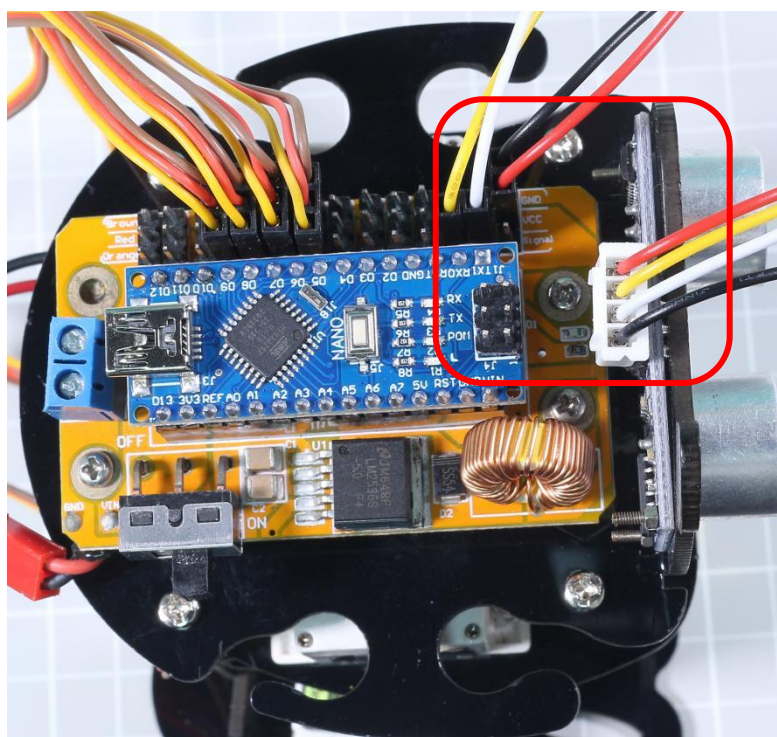
The basic principle of calibration: increased values can rotate the servo shaft clockwise and vice versa. For example, if the right leg is toe out, you need to decrease the upper-right servo' s angle; if it is toe in, you need to increase the angle.

Tips for calibration:

- ① The calibration method for the left leg works the opposite way for right leg.
- ② 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.
- ③ The calibration method for the left foot works the opposite way for right foot.

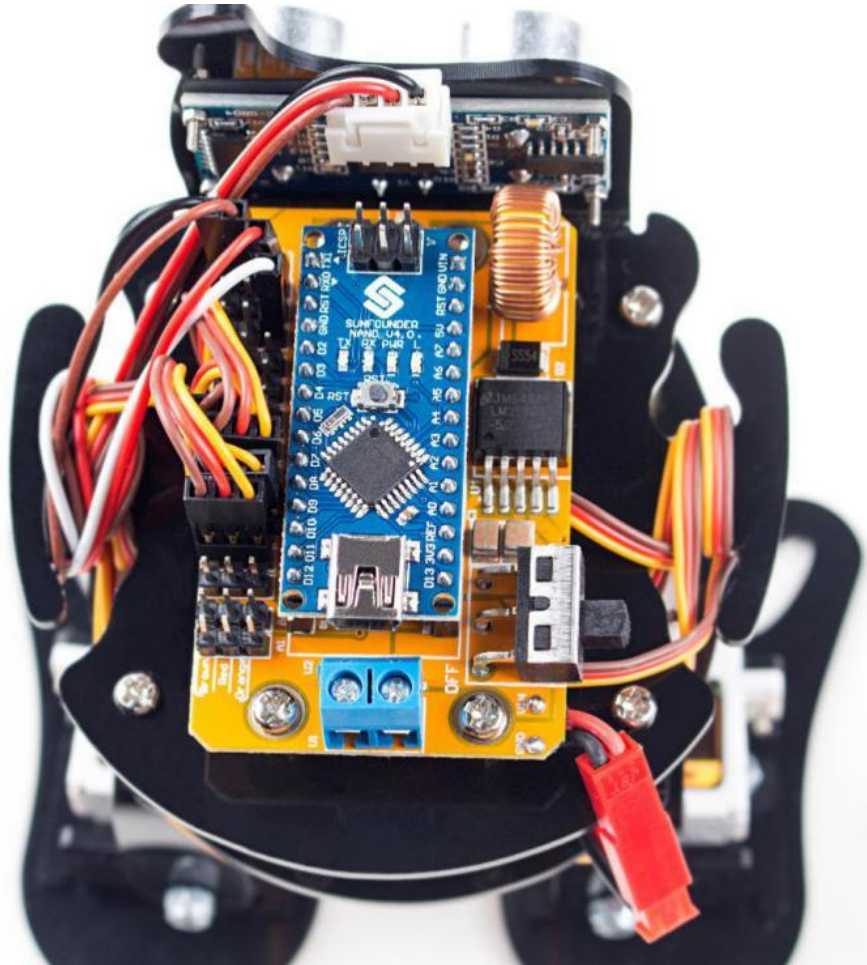
Ultrasonic Connecting

Connect **pin Trig** of the ultrasonic to **pin 4** of the board, **Echo** to **pin 3**, **Vcc** to **VCC** and **Gnd** to **GND**.



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.

Example

Here, we provide you with two sample programs to play Sloth:

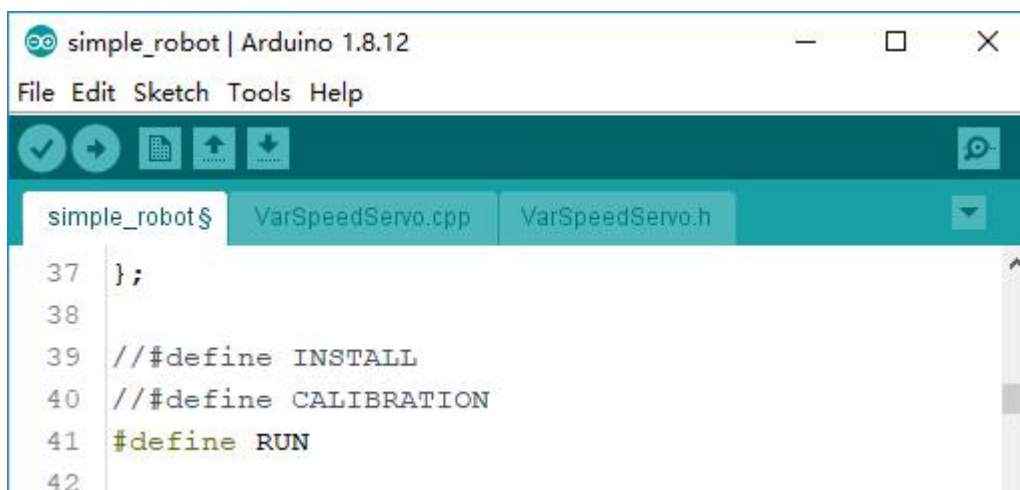
Simple Robot

In this code we write mobile obstacle avoidance for the robot. After the program is burned, sloth will go straight ahead. If it senses an obstacle ahead, it will step back and turn to find a new direction.

Open the program *simple_robot.ino* under the path of *DIY_4-DOF_Robot_Kit_-_Sloth\Code\simple_robot*.

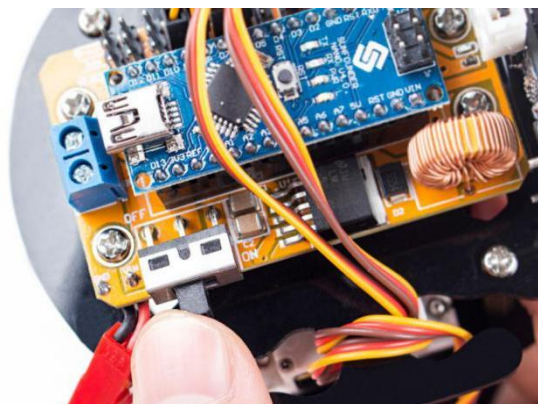
(This is also the program what we use to install and calibrate the servo.)

Go to **Line 39** again, set *#define RUN* as *able* and disable the other two, then upload the code to the Nano board.



```
simple_robot | Arduino 1.8.12
File Edit Sketch Tools Help
simple_robot$ VarSpeedServo.cpp VarSpeedServo.h
37 };
38
39 // #define INSTALL
40 // #define CALIBRATION
41 #define RUN
42
```

After burning successfully, unplug the USB cable and slide the power switch to ON.



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

Dancing

In this code, we write the basic actions of sloth and compose them into a dance.

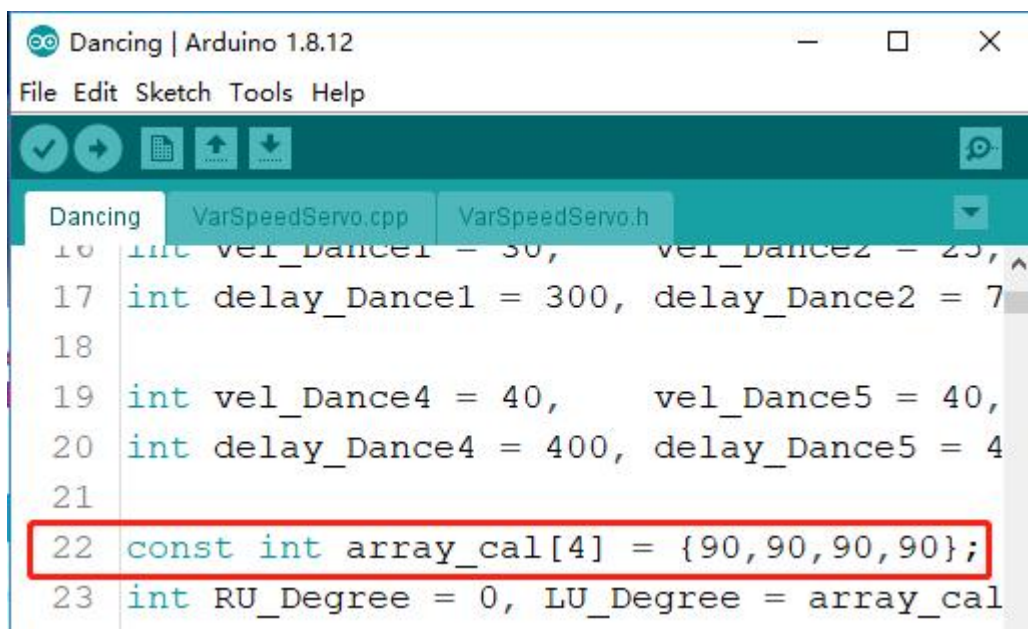
Open the program *Dancing.ino* under the path of *DIY_4-DOF_Robot_Kit - _Sloth\Code\Dancing*. Go to **Line 196**, select the RUN function by rectifying *#define*.



```
Dancing | Arduino 1.8.12
File Edit Sketch Tools Help
[Icons]
Dancing VarSpeedServo.cpp VarSpeedServo.h
193
194 // #define INSTALL
195 // #define CALIBRATION
196 #define RUN
```

After burning successfully, unplug the USB cable and press the power button on the servo control board. You will see the robot dancing.

Note: The program also needs to be calibrated in the same way as **Servo CALIBRATION Test** in **Assembly**. If there has been a precise calibration, you can modify the parameters in line 22 directly.



```
Dancing | Arduino 1.8.12
File Edit Sketch Tools Help
[Icons]
Dancing VarSpeedServo.cpp VarSpeedServo.h
16 int vel_Dance1 = 30,    vel_Dance2 = 25,
17 int delay_Dance1 = 300, delay_Dance2 = 7
18
19 int vel_Dance4 = 40,    vel_Dance5 = 40,
20 int delay_Dance4 = 400, delay_Dance5 = 4
21
22 const int array_cal[4] = {90,90,90,90};
23 int RU_Degree = 0, LU_Degree = array_cal
```

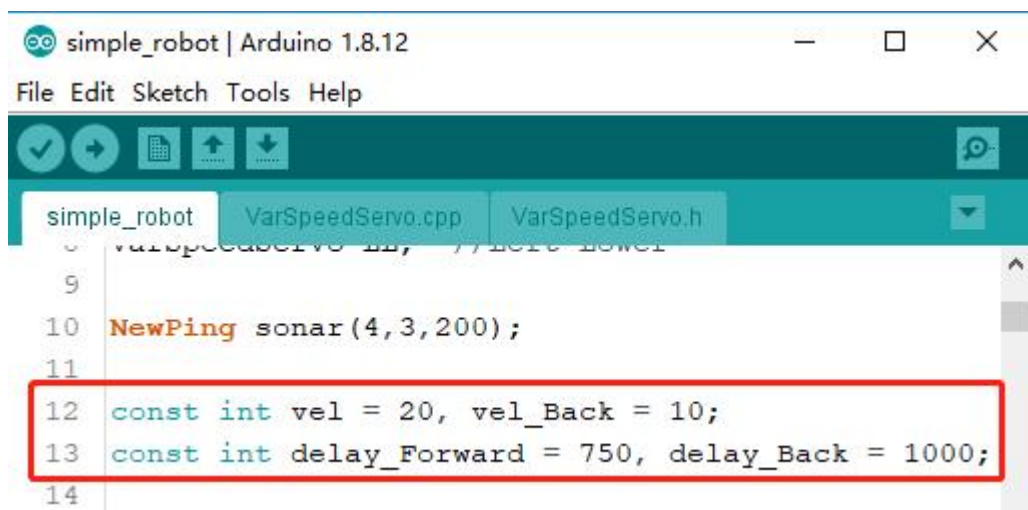
Q&A

Q1: How can we know the servo is damaged?

In **Test the Servo** 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.

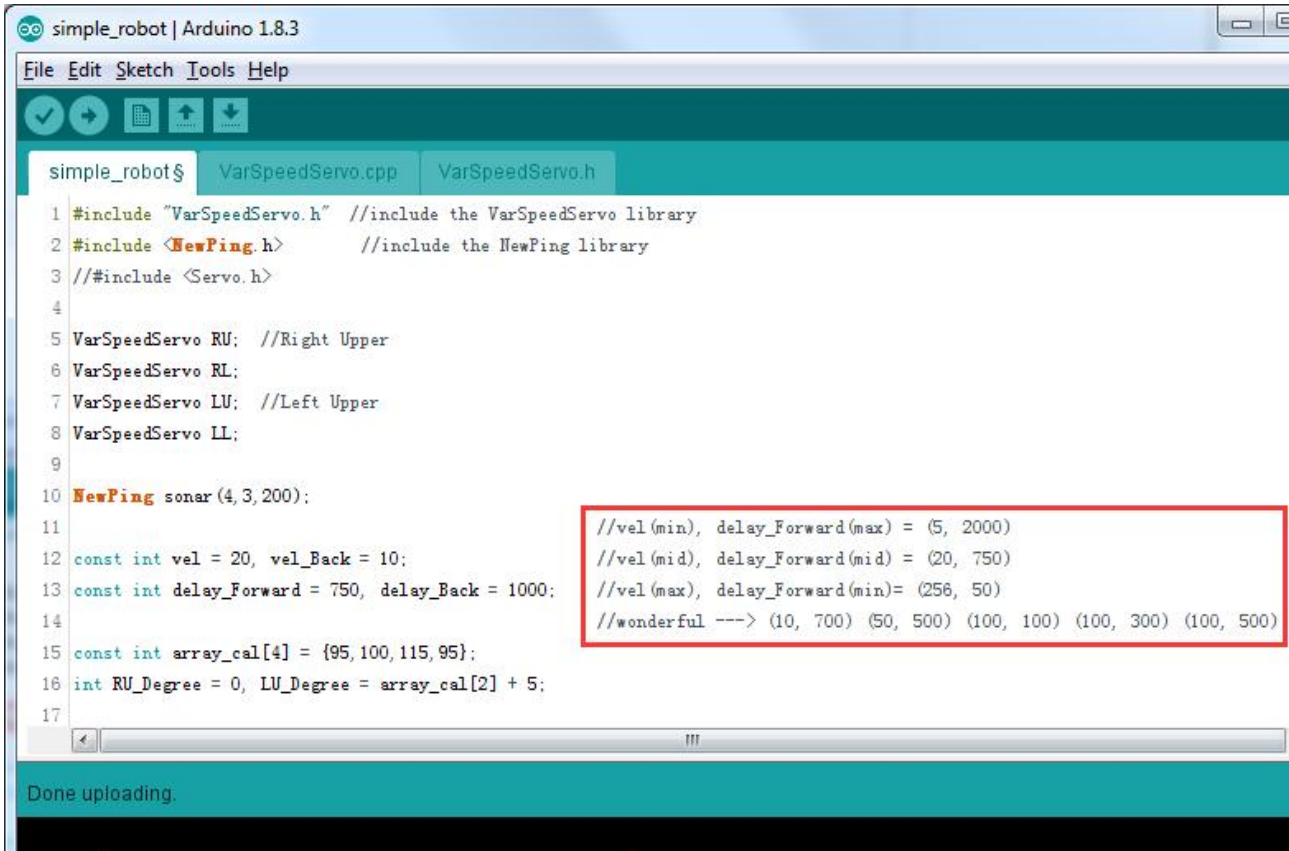
Q2: Why the Sloth reboots in running?

- 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. **"ve"** 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**.



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

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.



```
simple_robot | Arduino 1.8.3
File Edit Sketch Tools Help
simple_robot$ VarSpeedServo.cpp VarSpeedServo.h
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
12 const int vel = 20, vel_Back = 10;
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
//vel(min), delay_Forward(max) = (5, 2000)
//vel(mid), delay_Forward(mid) = (20, 750)
//vel(max), delay_Forward(min)= (256, 50)
//wonderful ---> (10, 700) (50, 500) (100, 100) (100, 300) (100, 500)
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?

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?

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.

Q6: What is macro definition (`#define`)?

The `#define` creates a macro, which is the association of an identifier or parameterized identifier with a token string. After the macro is defined, the compiler can substitute the token string for each occurrence of the identifier in the source file.

You can use the `#ifdef` directives anywhere `#if` can be used. The `#ifdef` identifier statement is equivalent to `#if 1` when identifier has been defined. It's equivalent to `#if 0` when identifier hasn't been defined, or has been undefined by the `#undef` directive. These directives check only for the presence or absence of identifiers defined with `#define`, not for identifiers declared in the C or C++ source code.

In sloth code, we use `#define` and `#ifdef` to start corresponding functions.

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 future journey of exploring 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

All contents including but not limited to texts, images, and code in this manual are owned by the SunFounder Company. You should only use it for personal study, investigation, enjoyment, or other non-commercial or nonprofit purposes, under the related regulations and copyrights laws, without infringing the legal rights of the author and relevant right holders. For any individual or organization that uses these for commercial profit without permission, the Company reserves the right to take legal action.