

Lesson 7 Control the Robot Arm by the wired Handle controller

Table

1. Wiring and assembly inspection	2
1.1 Wiring inspection	2
1.2 Structural inspection	3
2. Learn and use src files	3
2.1 Understand the src folder	3
2.2 How to use src files correctly	4
3. Compile and upload code	5
3.1 Open the code	5
3.2 Select development board and port	6
3.3 Install SCoop library	7
3.4 Compile and upload code	8
4. Safety and Precautions (important)	9
5. How to control the robotic arm by the Handle controller	9
5.1 Understand the buttons and joystick on the Handle controller	9
5.2 Analysis of the actions of each button and joystick controlling the servo	11
6. Trouble Shooting	24
6.1 Arduino IDE failed to recognize Robot Arm Hat	24
6.2 Code compilation failed	24
6.3 The robotic arm is not working	25
6.4 The working methods do not match the description	26
6.5 Robotic arm shaking	26
7. Any questions and suggestions are welcome	26

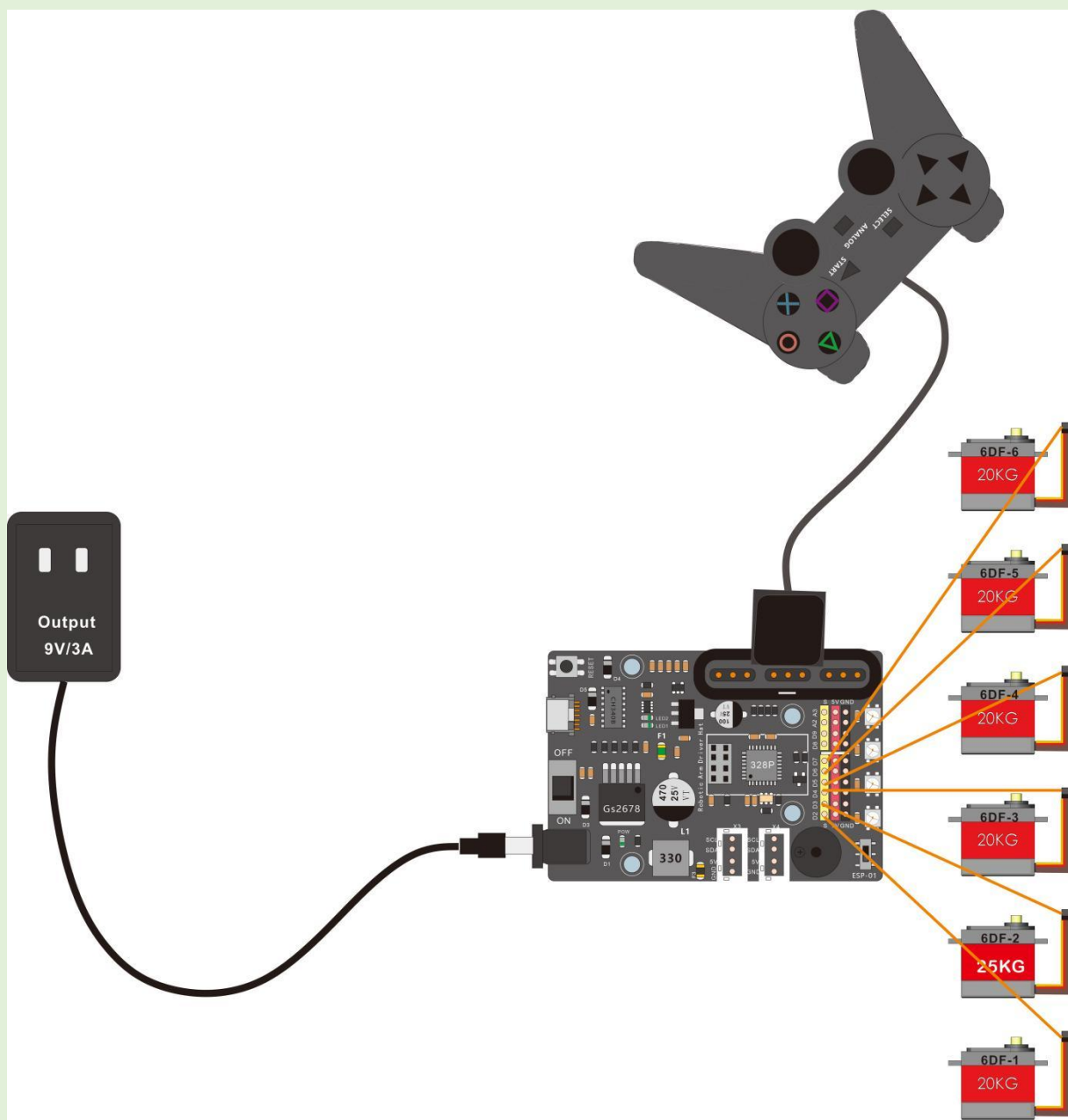
1. Wiring and assembly inspection

After assembling the robotic arm, do not immediately turn on the power or upload the code, but first check whether the wiring of the robotic arm and the assembly of each component are correct.

1.1 Wiring inspection

Please refer to the following wiring diagram to check if the 6 servos are connected to the correct positions.

Firstly, confirm that the signal pin of the servos is connected to the signal pin on the Robot Arm Hat instead of GND. Secondly, confirm if each servos is connected to the corresponding pin position on the Robot Arm Hat.



Six servos are labeled with 6DF-1, 6DF-2, 6DF-3, 6DF-4, 6DF-5, and 6DF-6 for easy differentiation. The signal pins of the 6 servos correspond to the signal pins on the Robot Arm Hat as shown in the table below.

Servo No.	Signal Pin of Robotic Arm Hat
6DF-1	D2
6DF-2	D3
6DF-3	D4
6DF-4	D5
6DF-5	D6
6DF-6	D7

1.2 Structural inspection

Manually rotate the various axis parts of the robotic arm without turning on the power, and confirm that each axis rotates smoothly without any jamming or mechanical obstruction. If there are any problems, please carefully check where the assembly is wrong according to the assembly steps.

2. Learn and use src files

For ease of operation, it is recommended not to connect the adapter to the robotic arm at this time, nor to connect the Handle wired handle to the robotic arm. Pick up the robotic arm and place it close to the computer on the desktop. Use a Type-C USB cable to connect the computer to the Robotic Arm Hat.

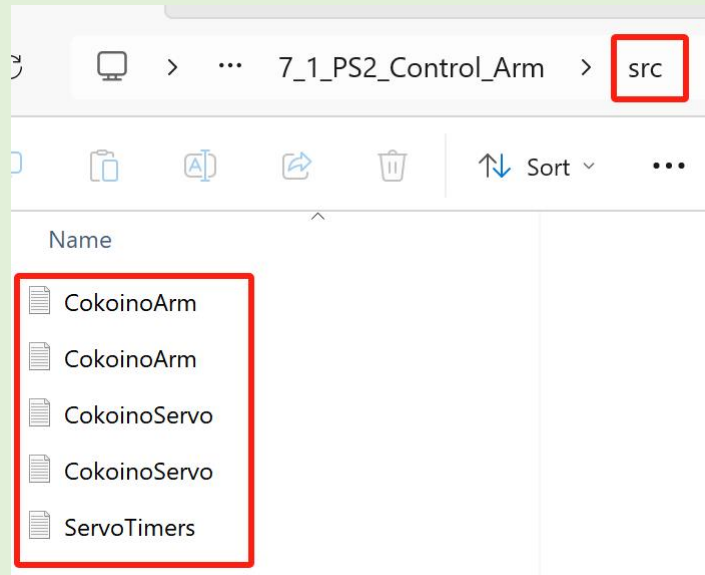
2.1 Understand the src folder

The src folder of Arduino is mainly used to store the source code files of the library. In the development process of Arduino library, the src folder is an essential part, which contains the main C++ source code files (.cpp files) of the library. These source code files define the classes and functions of the library and are the core part of implementing the library's functionality.

By organizing the code in the src folder, modularization of the code can be achieved, making it clearer and easier to manage. In addition, the code in the src folder can be shared and reused by multiple Arduino projects, improving the reusability and development efficiency of the code.

2.2 How to use src files correctly

Our robotic arm uses 6 servos, and using Arduino's existing Servo library to write code to control the actions of the 6 servos can be quite complex and cumbersome. To simplify the code, we wrote the CokoinoServo library and CokoinoArm library ourselves. Put them all in the src folder, as shown in the following figure:



How to use the library written by oneself?

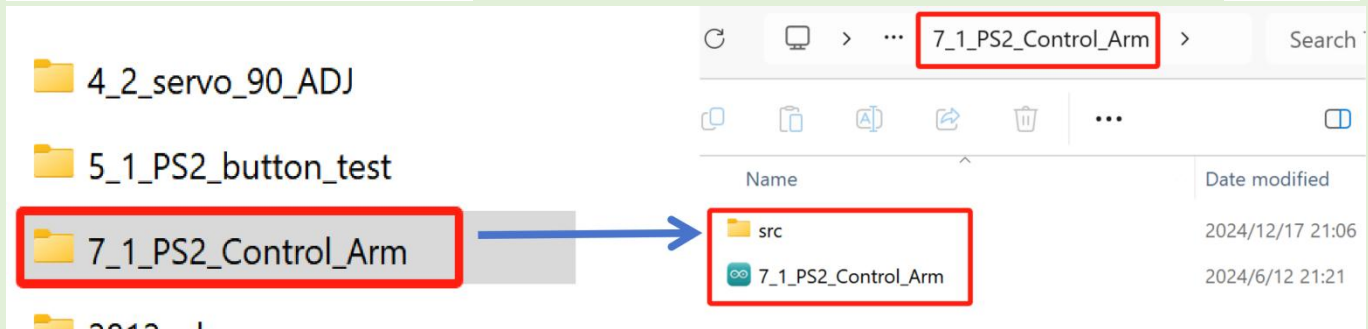
The first method is to install the library into the Arduino IDE and then call it in the code using `#include <xxx.h>`.

The method of installing libraries into Arduino IDE has been explained in detail in the previous Lesson, and will not be described here.

The second method is to directly use the functions in the library in the code, and then place the src folder containing the library in the folder where the code is located. This way, opening the code can directly call the library in the src folder.

The code required for this course is 7_1_Handle_Control_Arm.ino

This code uses functions from the CokoinoServo and CokoinoArm libraries that we have written ourselves. When using them, we need to place the src folder containing the CokoinoServo and CokoinoArm libraries in the folder where 7_1_SS2_Control. Arm. ino is located, as shown in the following figure:



3. Compile and upload code

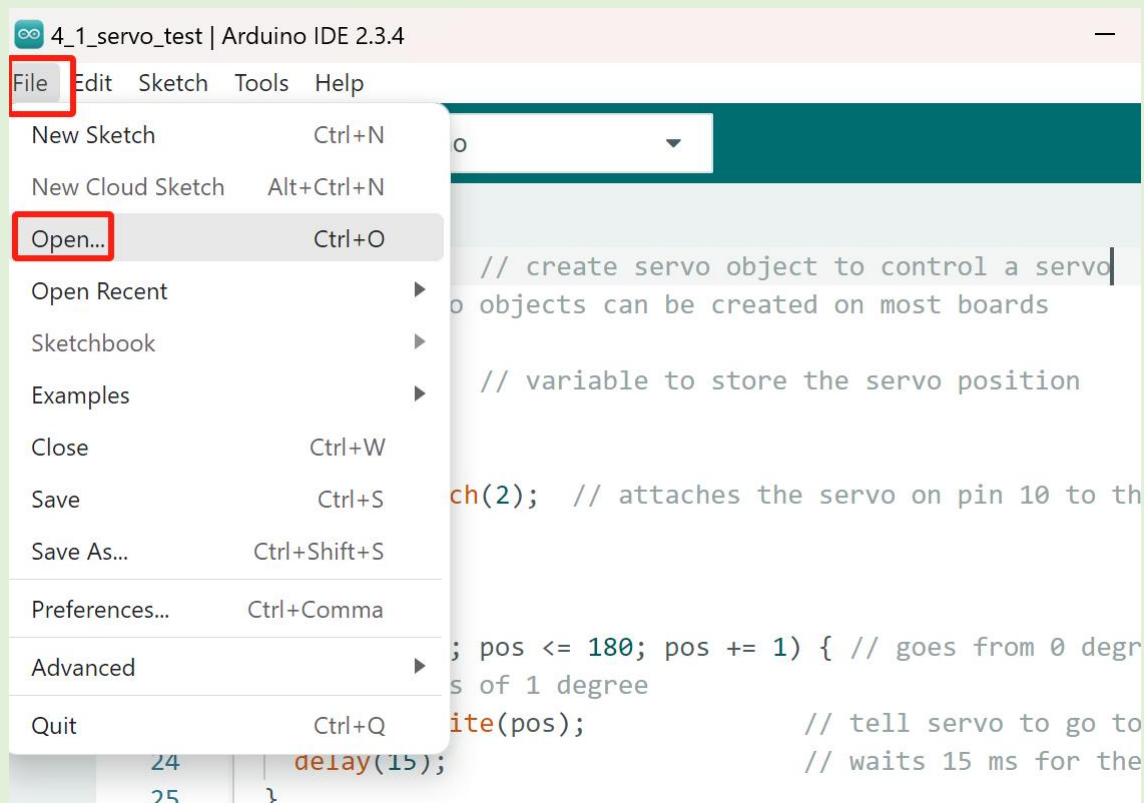
3.1 Open the code

The code used in this lesson is placed in this folder: “[E:\CKK0017-main\Tutorial\sketches](#)”

3.1.1 Double-click the Arduino IDE shortcut on the desktop to open it



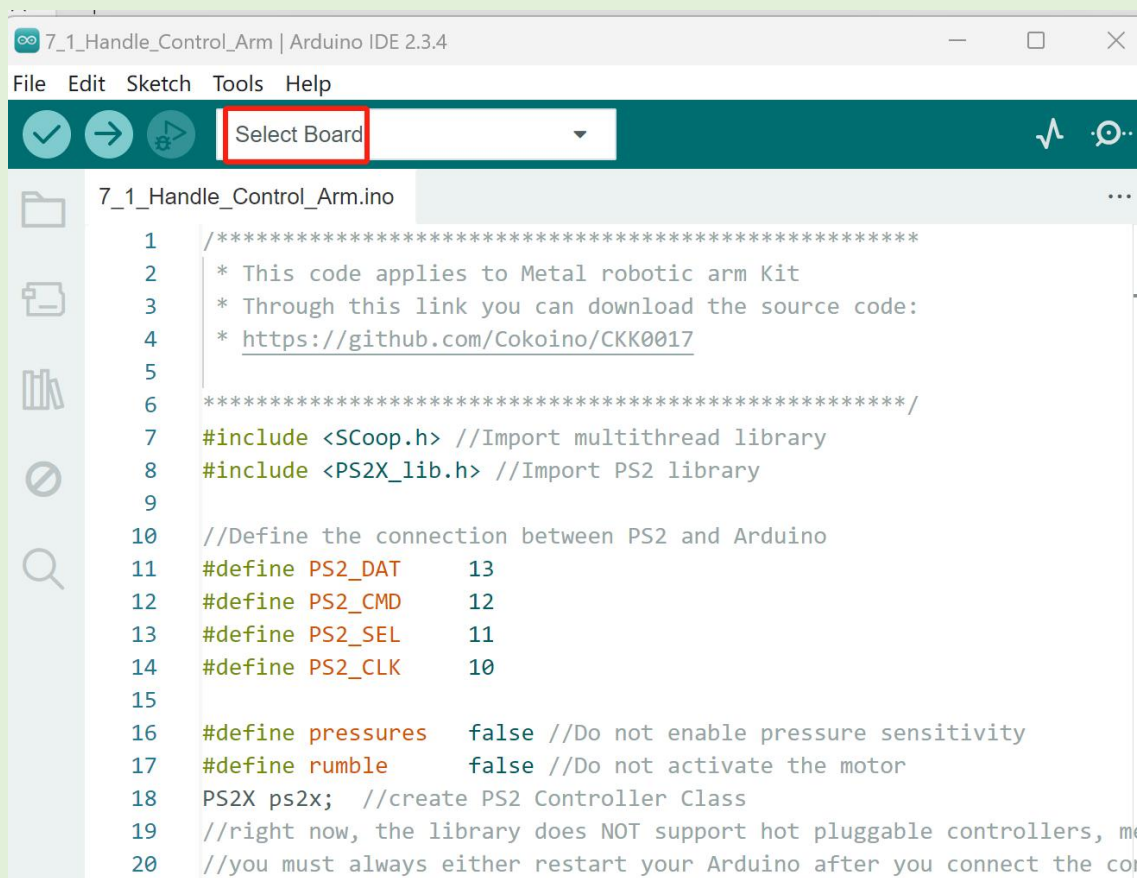
3.1.2 Click "File" --- "open"



3.1.3 Select the code in the folder named 7_1_Handle_Control_Arm:

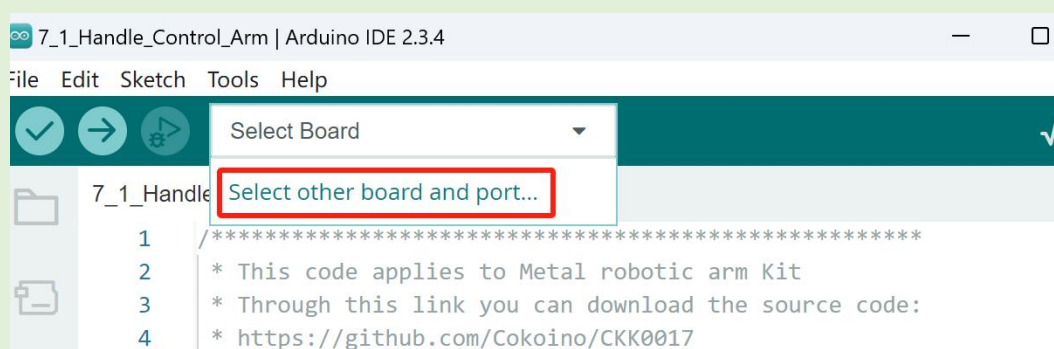
E:\CKK0017-main\Tutorial\sketches\7_1_Handle_Control_Arm.

Click "open", Open the code interface as follows:



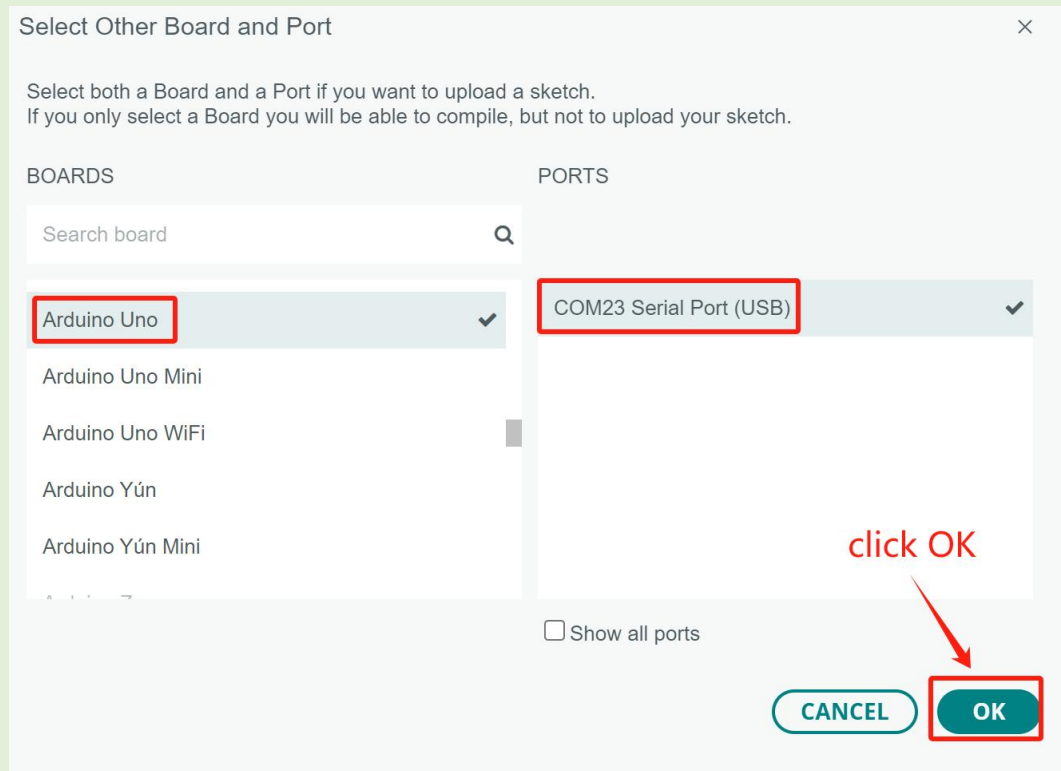
3.2 Select development board and port

3.2.1 Click the "Select Board", then click "Select other board and port..."



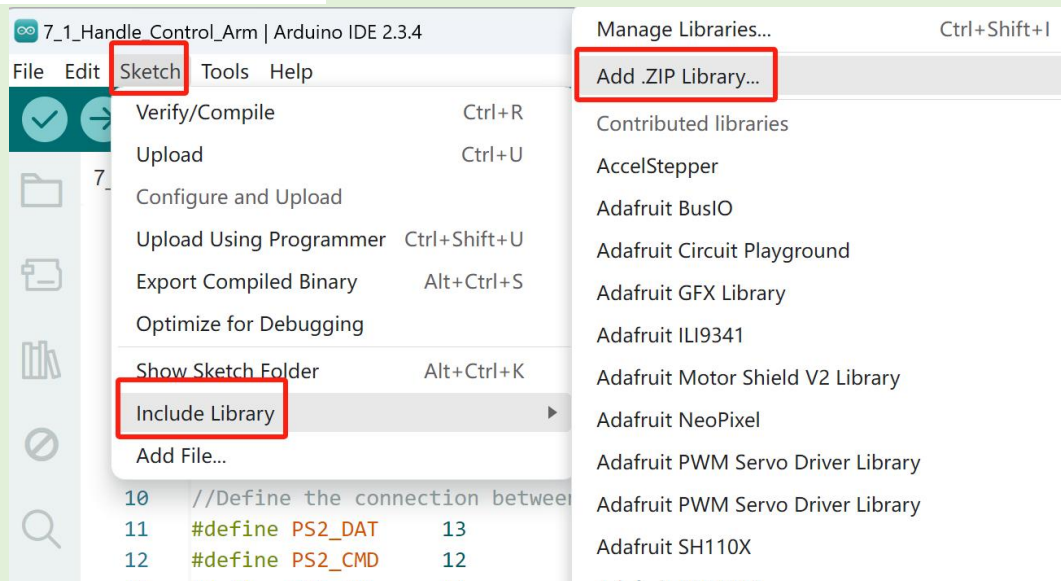
3.2.2 Click and select Arduino UNO in the "BOARDS" dropdown menu, then select the corresponding COM port of Robot Arm Hat on the computer in the "PORTS" menu. On my computer, it recognizes and

displays COM23 (COM port is commonly known as an input output port for a device normally PC which enables communication between Arduino and PC. You can check your arduino com number in device manager, the com port of our arduino board is recognized as COM23 in this tutorial), then click "OK".



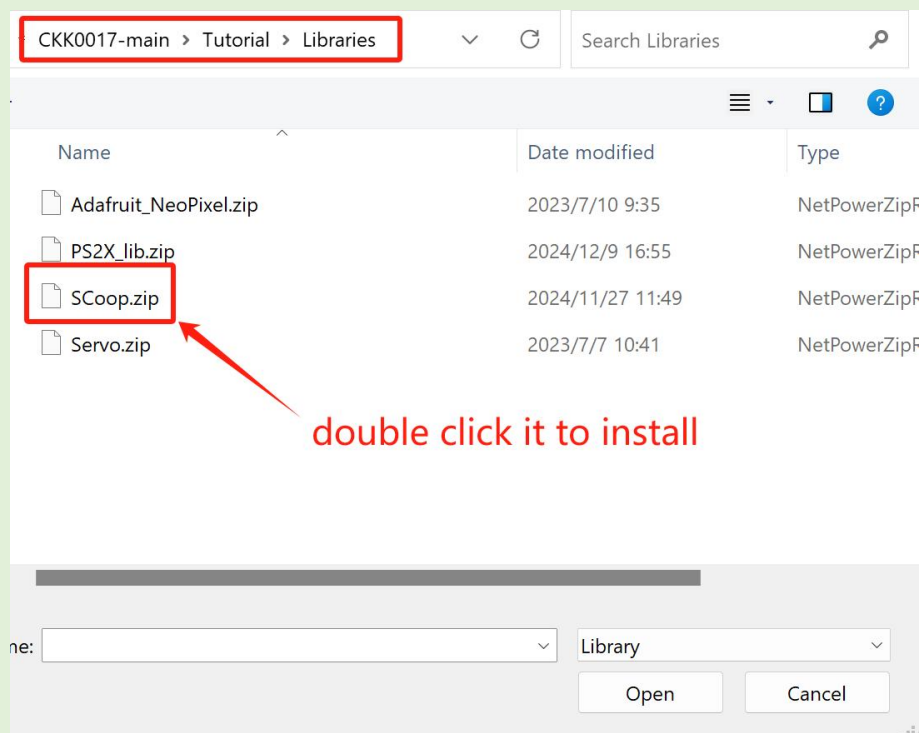
3.3 Install SCoop library

3.3.1 On the toolbar of Arduino IDE interface, click "Sketch" ---"Include Library"---"Add.ZIP Library..."




3.3.2 Find the SCoop.ZIP file in [E:CKK0017-main\Tutorial\Libraries\SCoop.zip](#)

Double click the SCoop.ZIP file, then it will be installed into Arduino IDE



3.4 Compile and upload code

3.4.1 Click compile button , successfully compiled the code will display “Done compiling”

3.4.2 Click upload button , successfully uploading the code will display “Done uploading”.

4. Safety and Precautions(important)

This metal robotic arm stands at a height of 45cm when fully standing, with a maximum swing arm length of 36cm. Adopting 5 maximum torques of 20KG CM and a maximum torque of 25KG CM's servo motor. Therefore, this metal robotic arm has a relatively large range of travel and rotational force during operation. When it is working, we need to be careful to keep our bodies away from its range of motion and avoid being touched by the moving robotic arm.

Before turning on the power to the control board, first hold the robotic arm upright by hand. This way, after turning on the power, the initialization action of the robotic arm will be relatively small.

When using the Handle controller to control the robotic arm, the body should maintain a distance of at least 0.5 meters from the robotic arm. When the robotic arm is in motion, hands or other parts of the body should not approach the robotic arm, and the body should not touch the joints of the robotic arm to prevent collision or self injury.

When there is a malfunction in the operation of the robotic arm, such as being entangled in joints by wires, tapes, or other similar objects, it is necessary to first turn off the power switch on the robotic arm control board before dealing with obstacles on the robotic arm.

This robotic arm is suitable for teenagers aged 13 and above to learn and use. Children aged 8-12 need to use this robotic arm under adult supervision.

Children under 8 years old are not allowed to use it and it needs to be kept out of their reach.

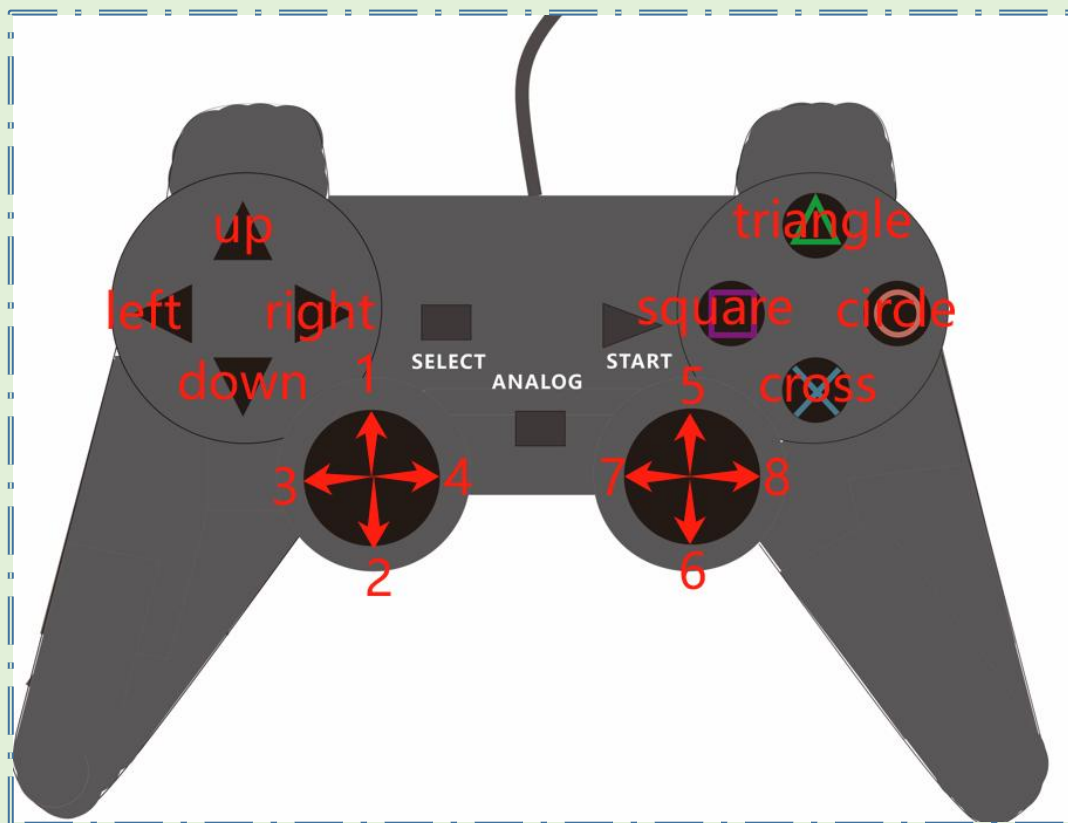
5. How to control the robotic arm by the Handle controller

There are buttons and joysticks on the Handle controller. We have defined the servos and actions controlled by each button/joystick in the code, so that the robotic arm can be controlled through the Handle controller.

5.1 Understand the buttons and joystick on the Handle controller

The Handle controller has left and right joysticks, each of which is divided into four control directions: front, back, left, and right. For the convenience of distinction, we use the number 1 to represent the front

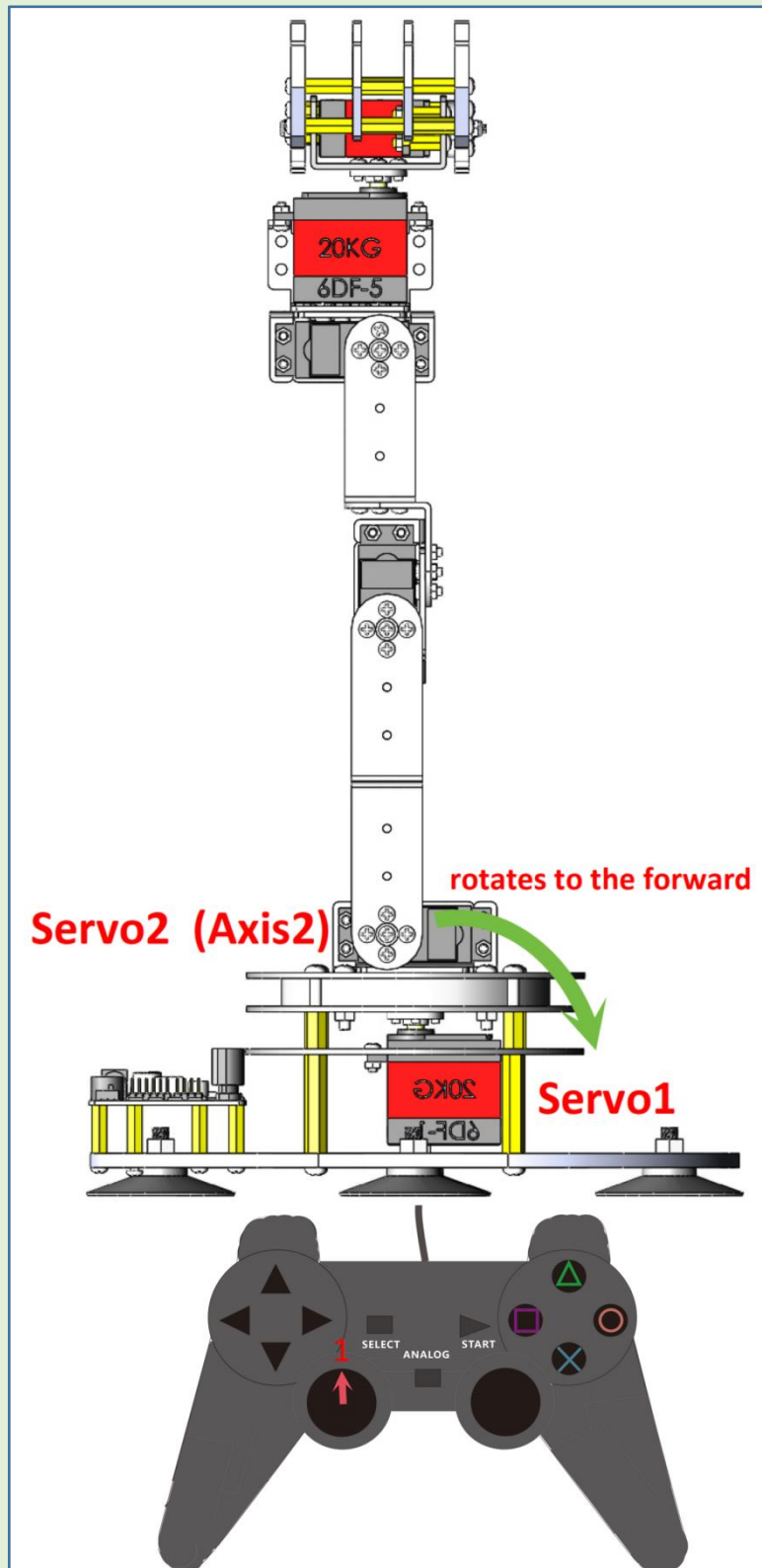
(control direction) of the left joystick, the number 2 to represent the back (control direction) of the left joystick, the number 3 to represent the left (control direction) of the left joystick, the number 4 to represent the right (control direction) of the left joystick, the number 5 to represent the front (control direction) of the right joystick, the number 6 to represent the back (control direction) of the right joystick, the number 7 to represent the left (control direction) of the right joystick, and the number 8 to represent the right (control direction) of the right joystick. The buttons include up, down, left, right, triangle, cross, square, circle, as shown in the following figure:



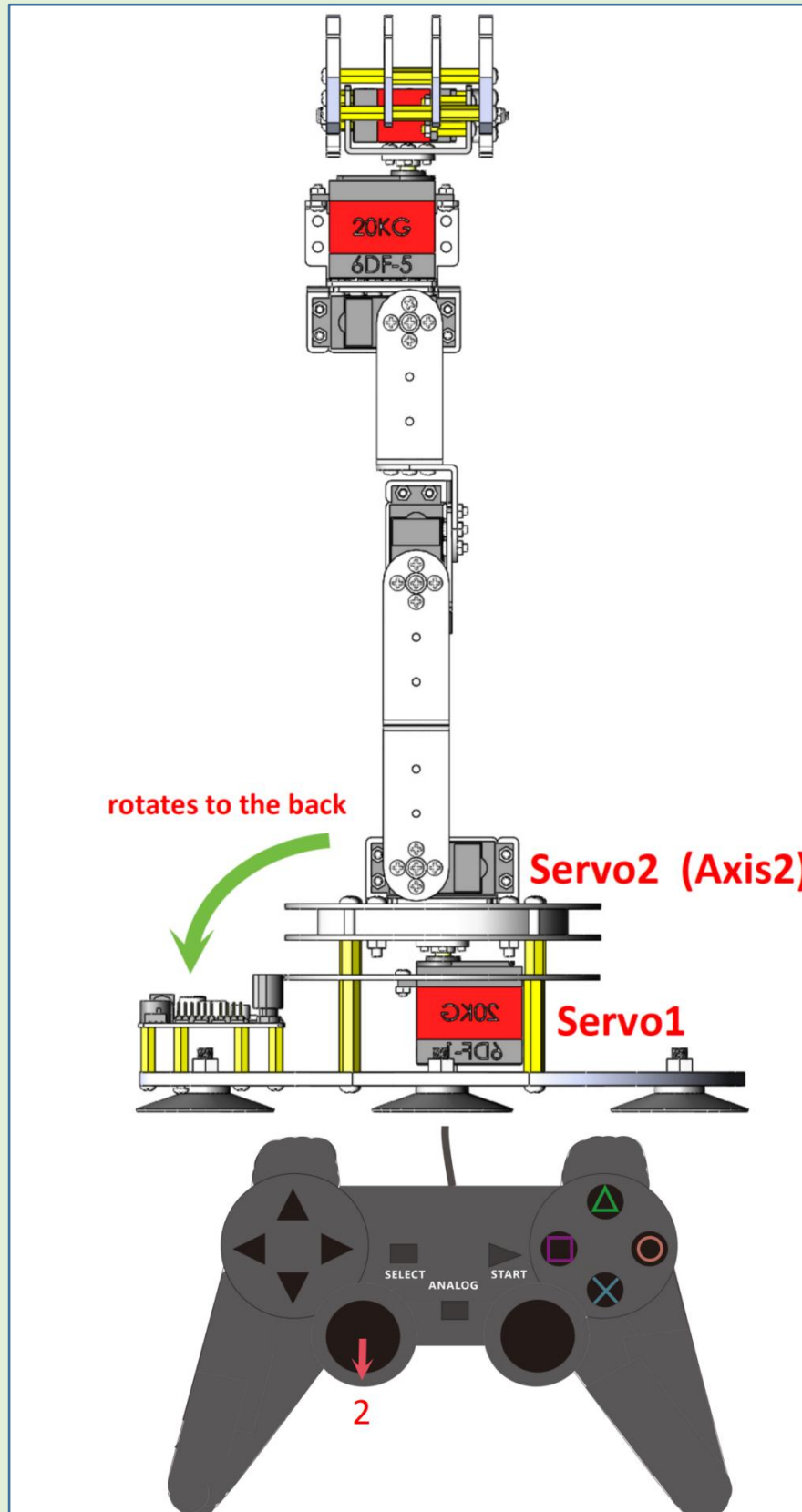
In the code, it is defined that the joystick controls the actions of servo2 (6DF-2) in the 1 and 2 directions, servo1 (6DF-1) in the 3 and 4 directions, servo3 (6DF-3) in the 5 and 6 directions, servo4 (6DF-4) in the up and down buttons, servo5 (6DF-5) in the left and right buttons, and servo6 (6DF-6) in the 7 and 8 directions.

5.2 Analysis of the actions of each button and joystick controlling the servo

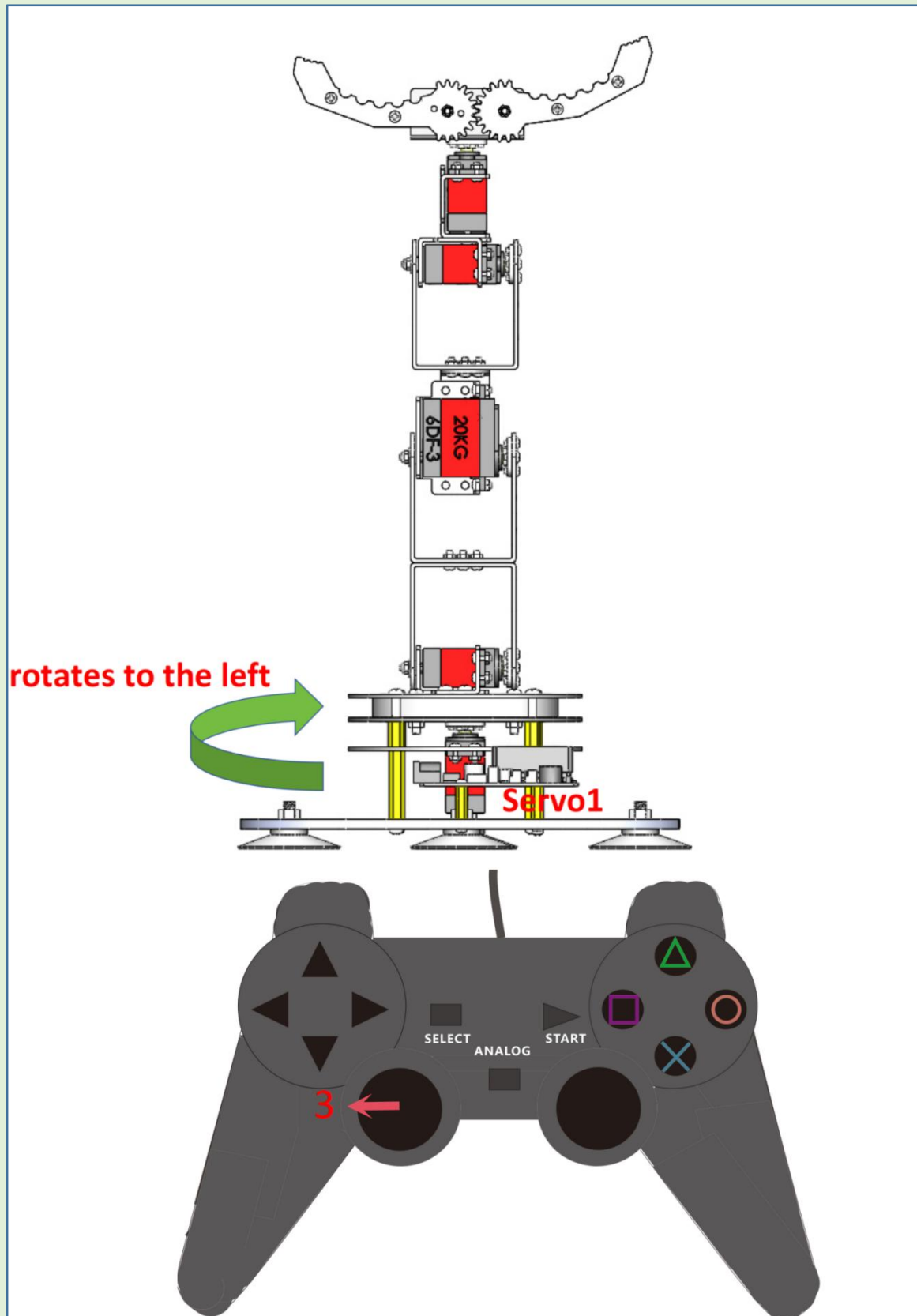
Number 1(Direction of joystick push): Control the servo2, move the Axis 2 forward while keeping the other axes stationary, so that the robotic arm moves forward along the Axis 2.



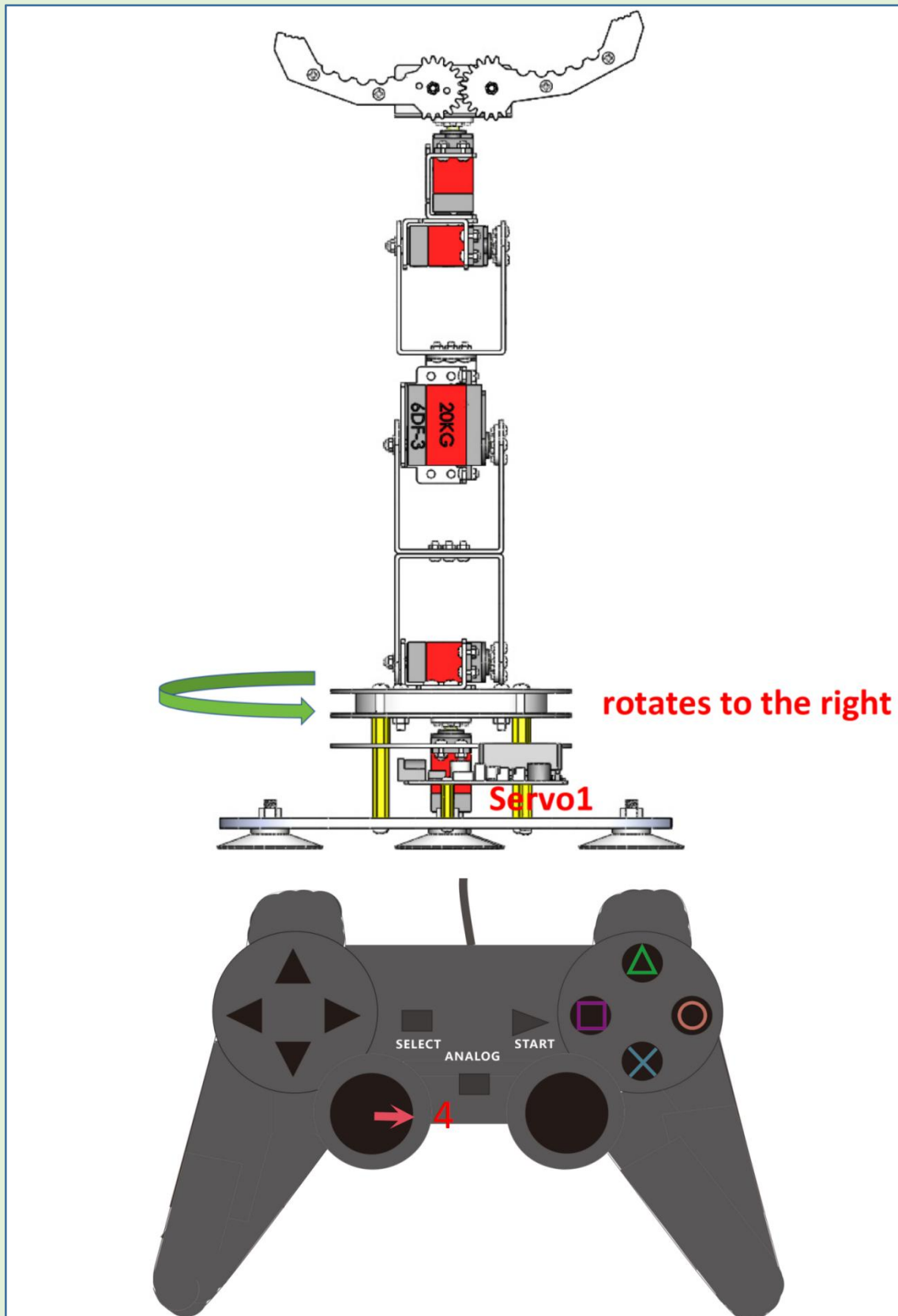
Number 2(Direction of joystick push): Control servo motor 2 to move the Axis backwards while keeping the other axes stationary, so that the robotic arm moves backwards along the Axis.



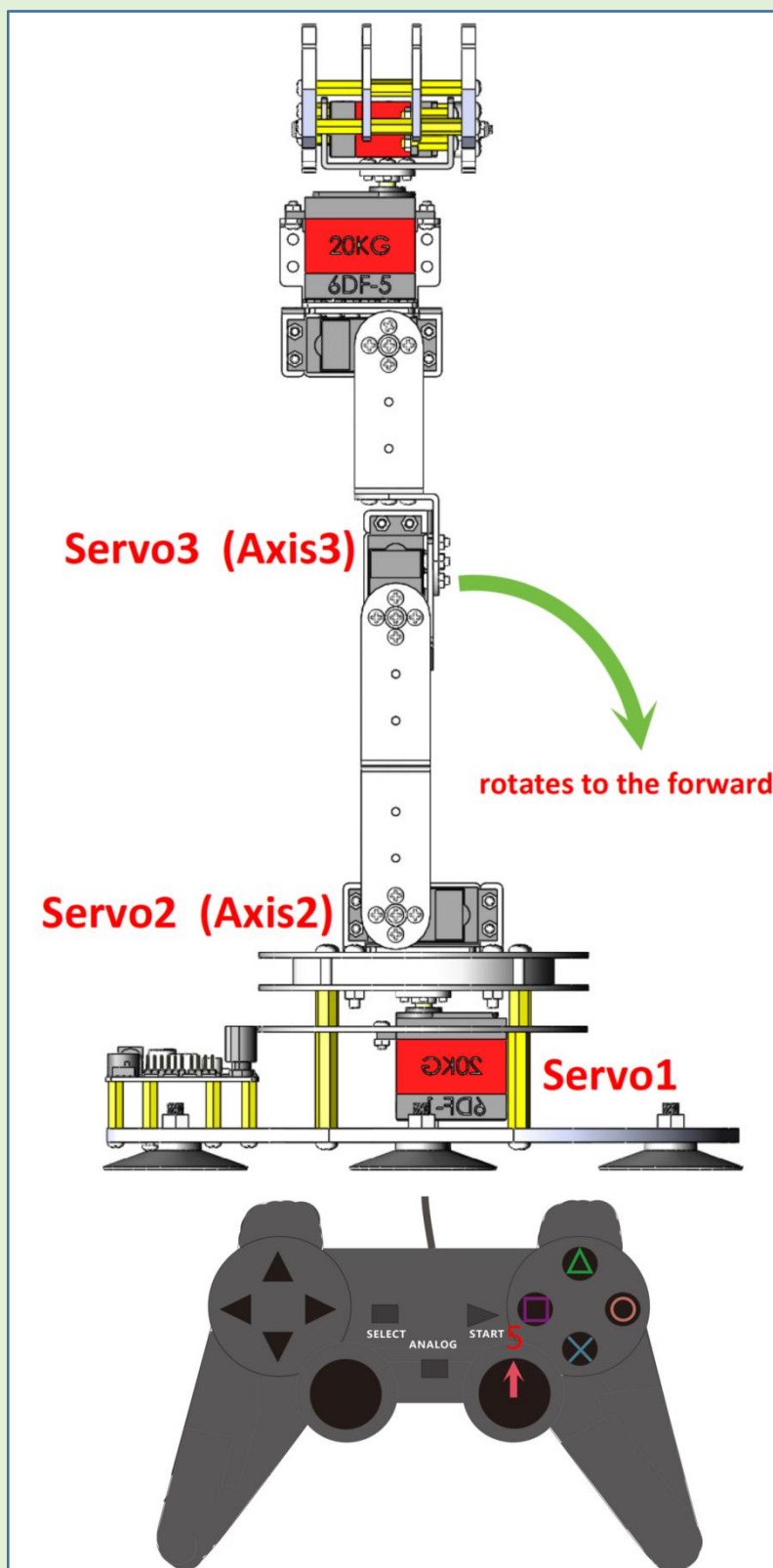
Number 3(Direction of joystick push): Control servo1 to rotate the base of the robotic arm to the left.



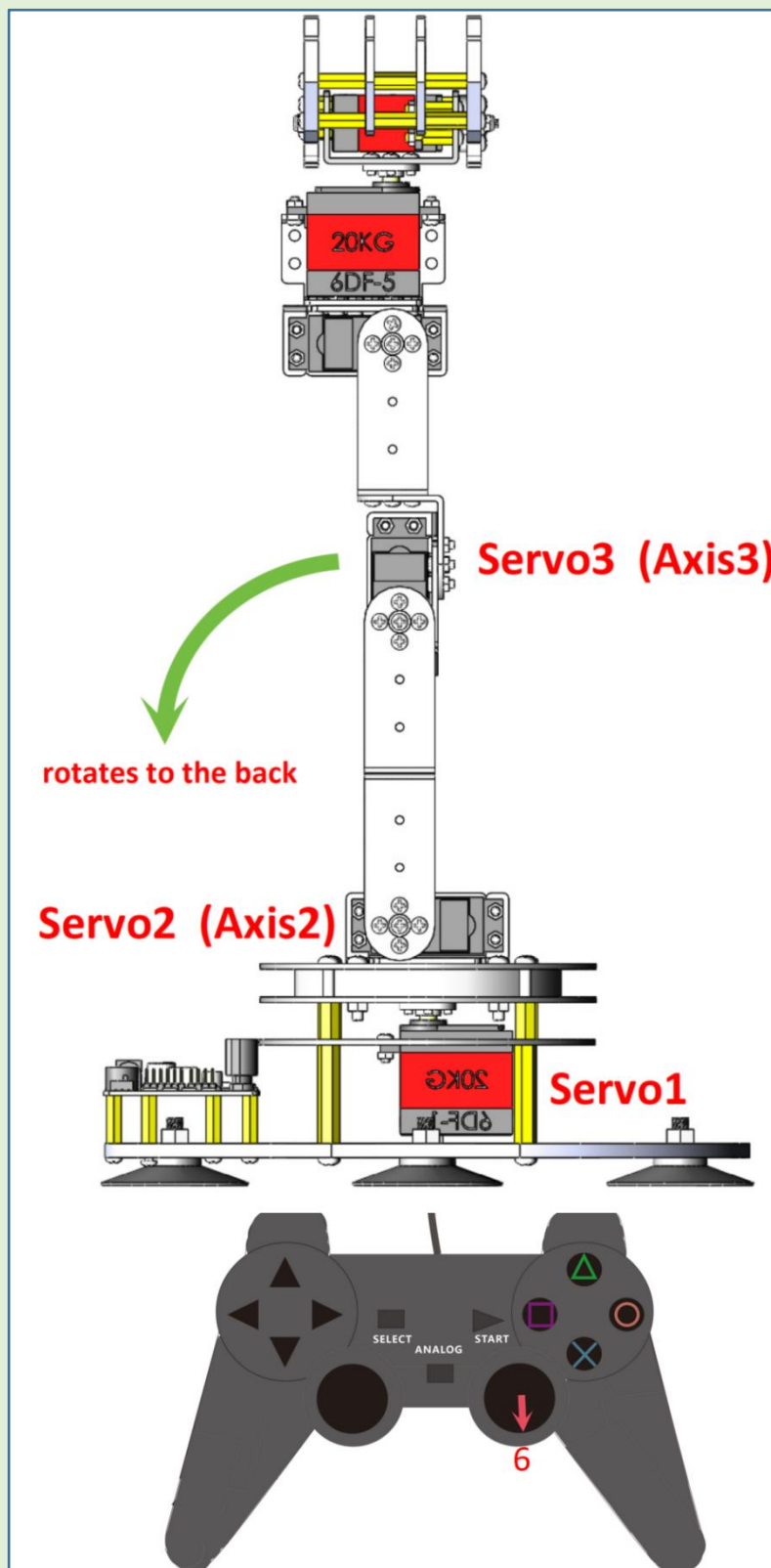
Number 4(Direction of joystick push): Control servo 1 to rotate the base of the robotic arm to the right.



Number 5(Direction of joystick push): Control the servo3, with Axis 3 moving forward and the other axes remaining stationary.

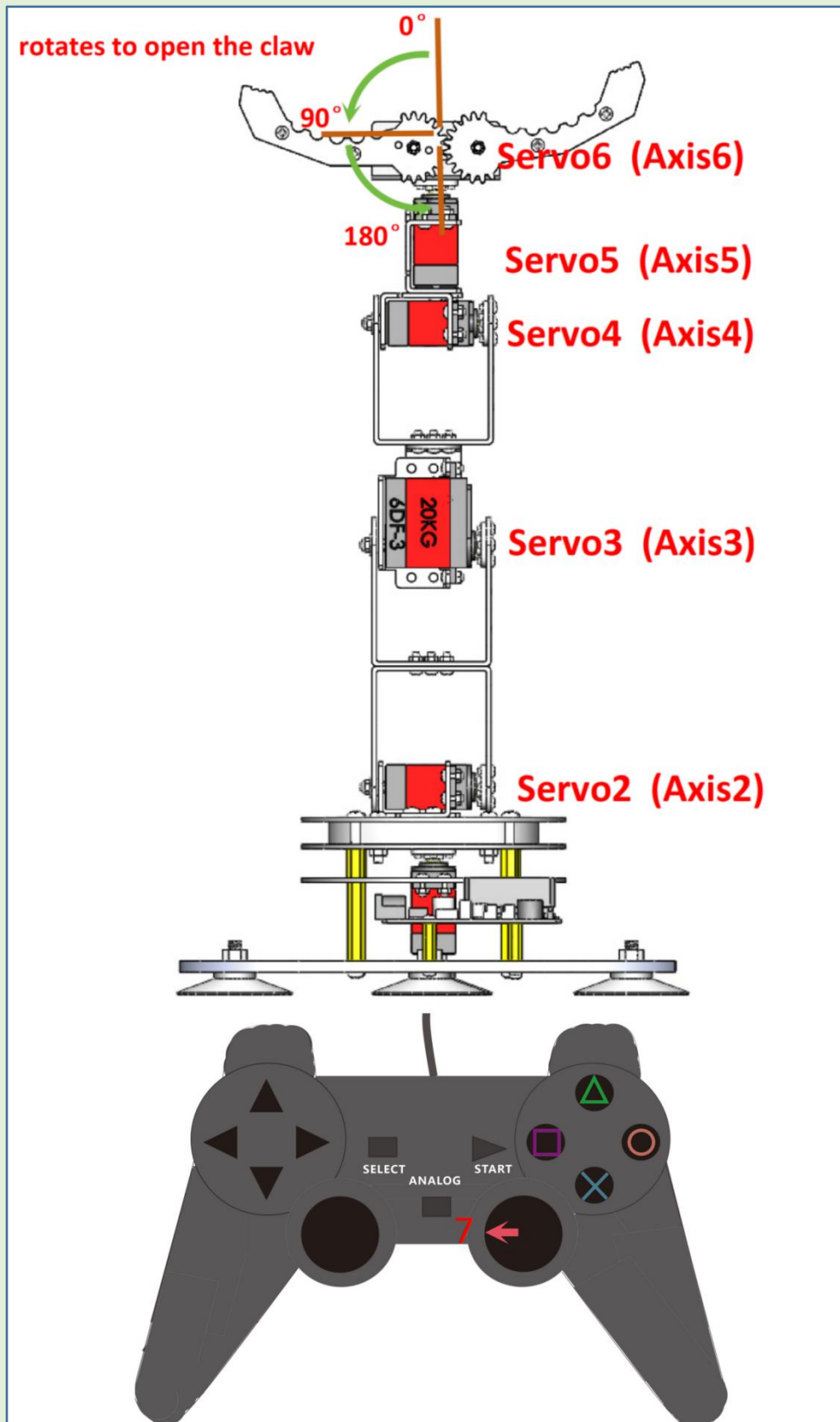


Number 6(Direction of joystick push): Control the servo3, with Axis 3 moving backwards and the other axes remaining stationary.

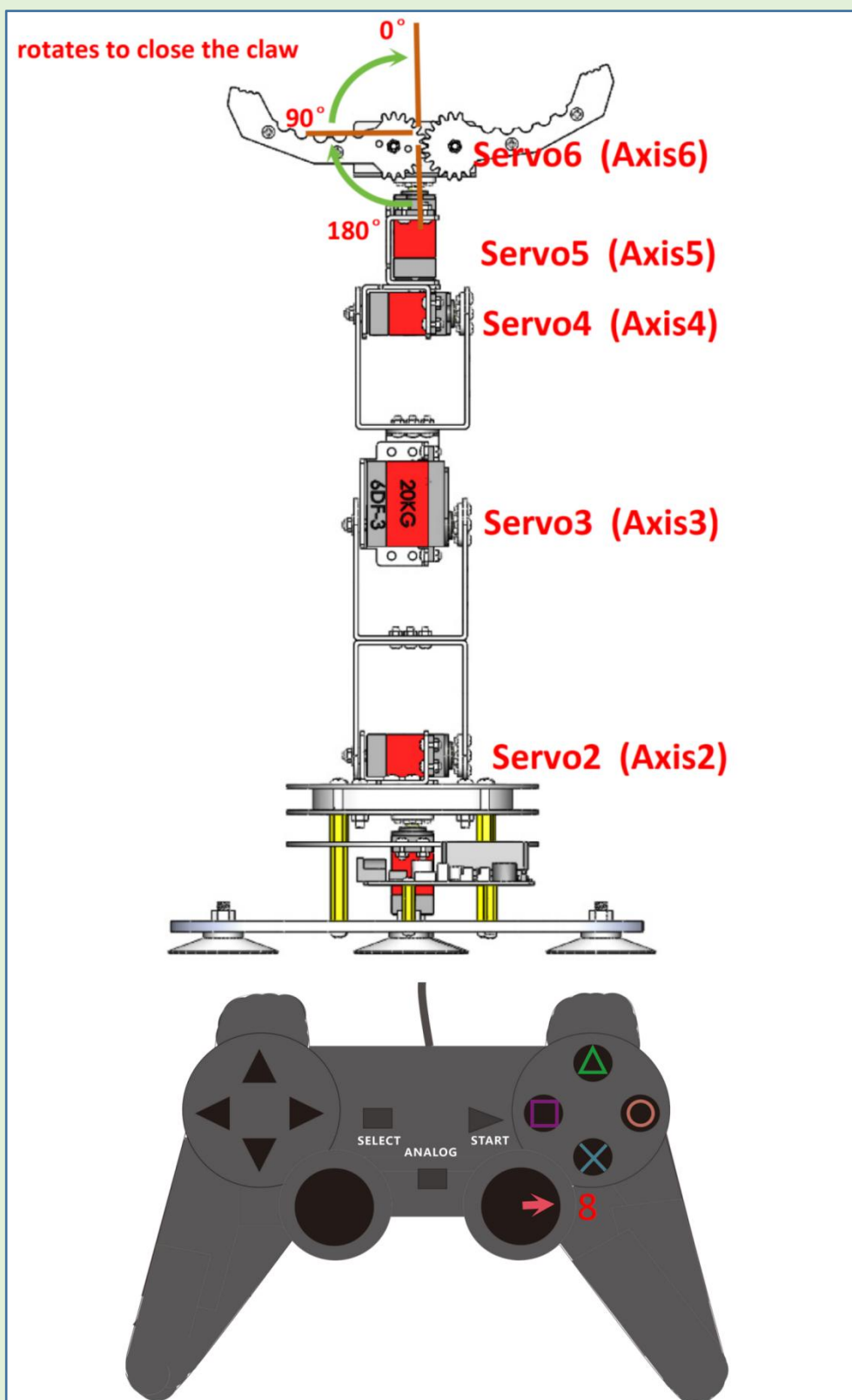


Number 7(Direction of joystick push): Control servo 6, open the claws, and keep the other axes stationary.

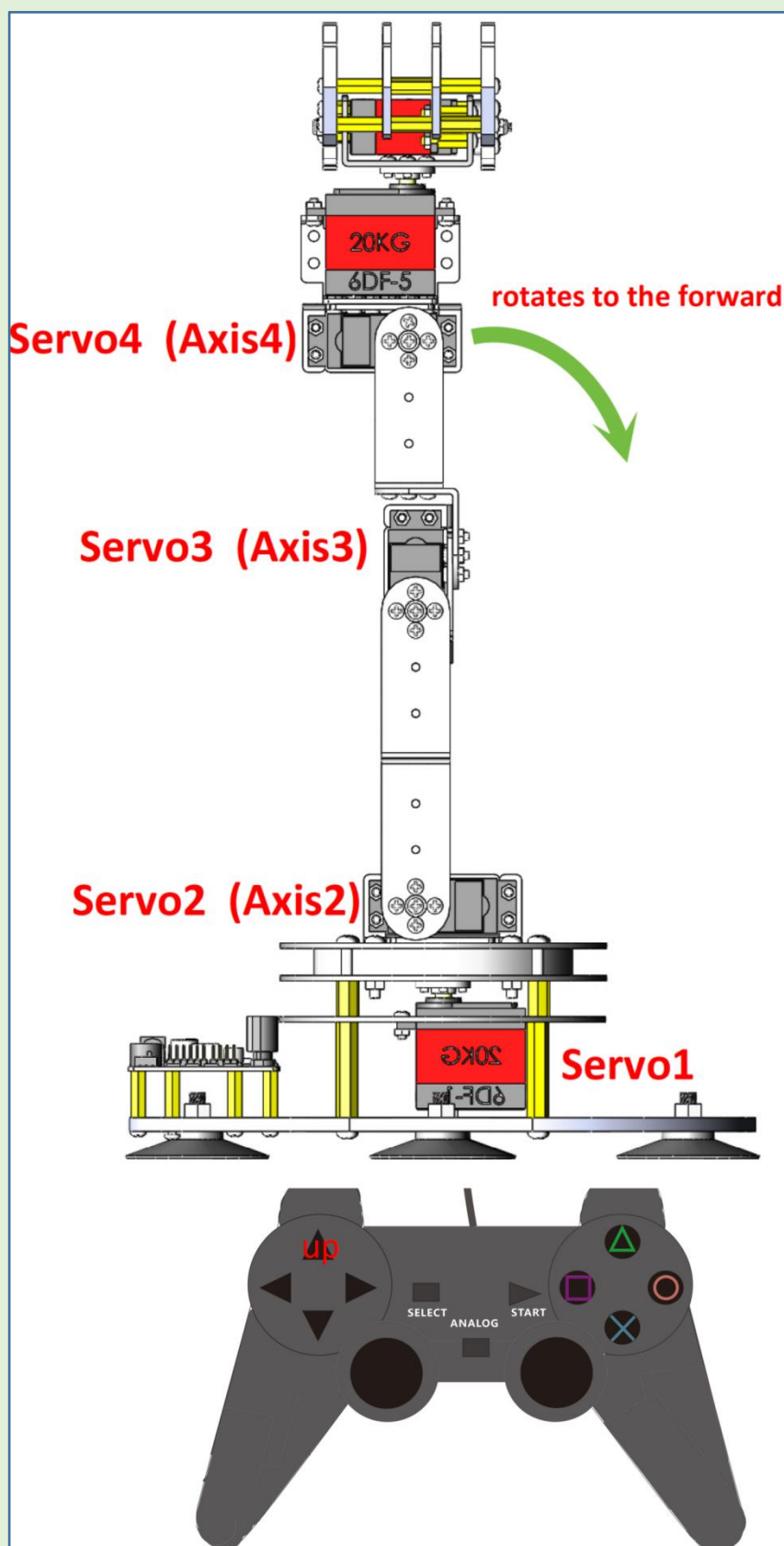
Due to structural interference, the claws can be opened up to a maximum of 120 ° position.



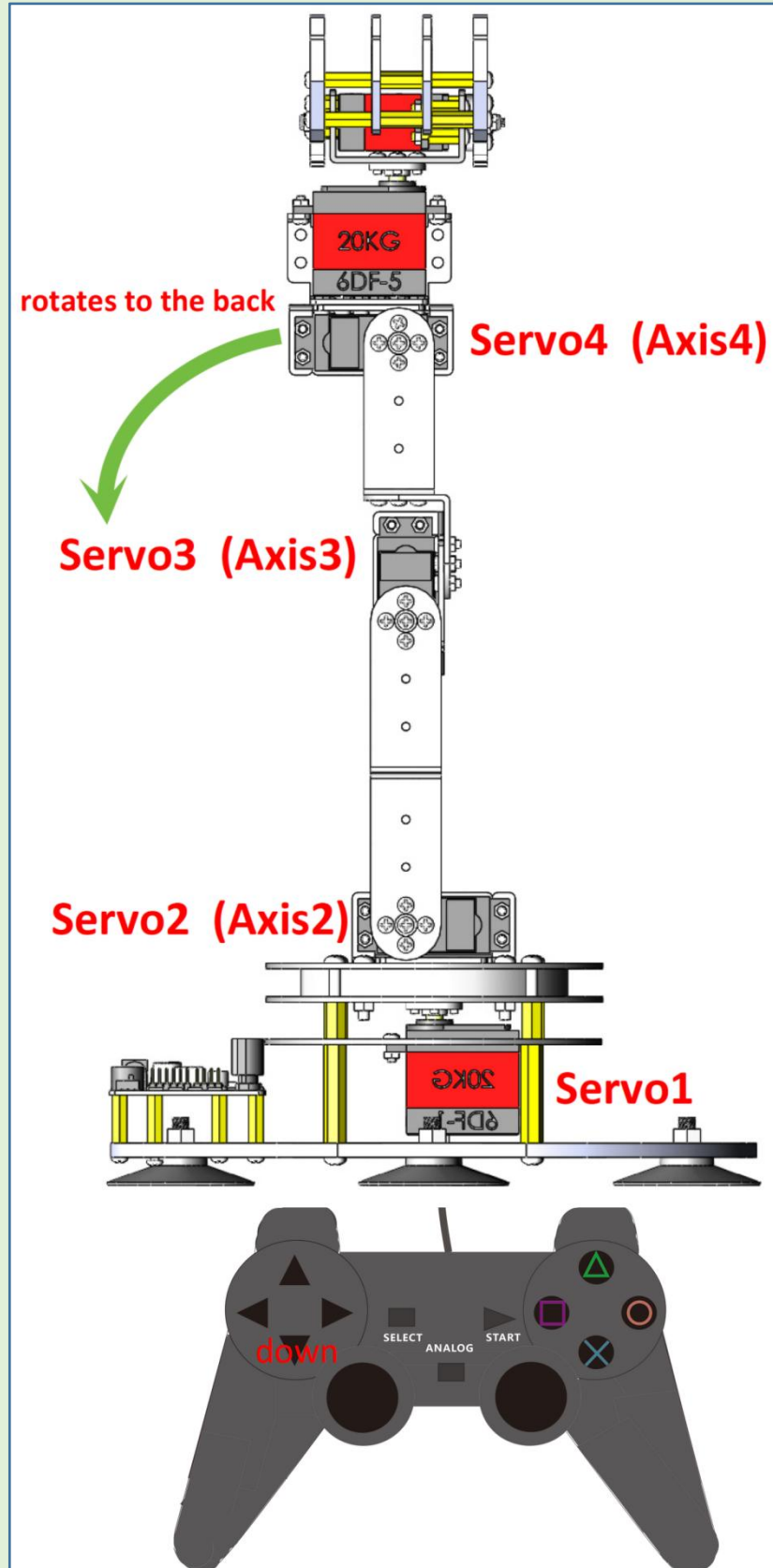
Number 8(Direction of joystick push): Control servo 6, turn off the claws, and keep the other axes stationary.



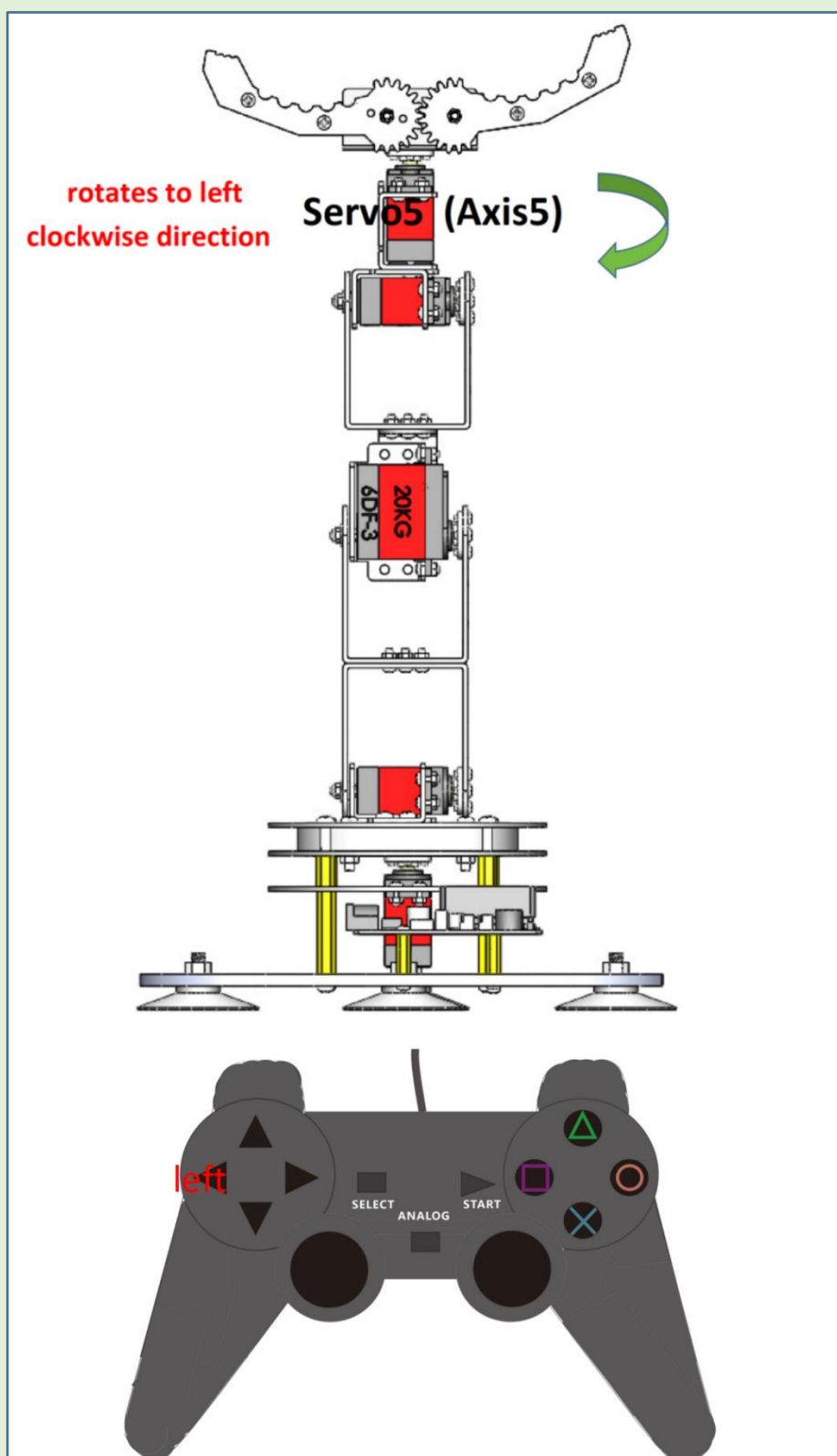
“up” button: Press and hold the up button to control servo4, Axis4 moves forward while the other axes remain stationary. After releasing the button, servo4 stops moving.。



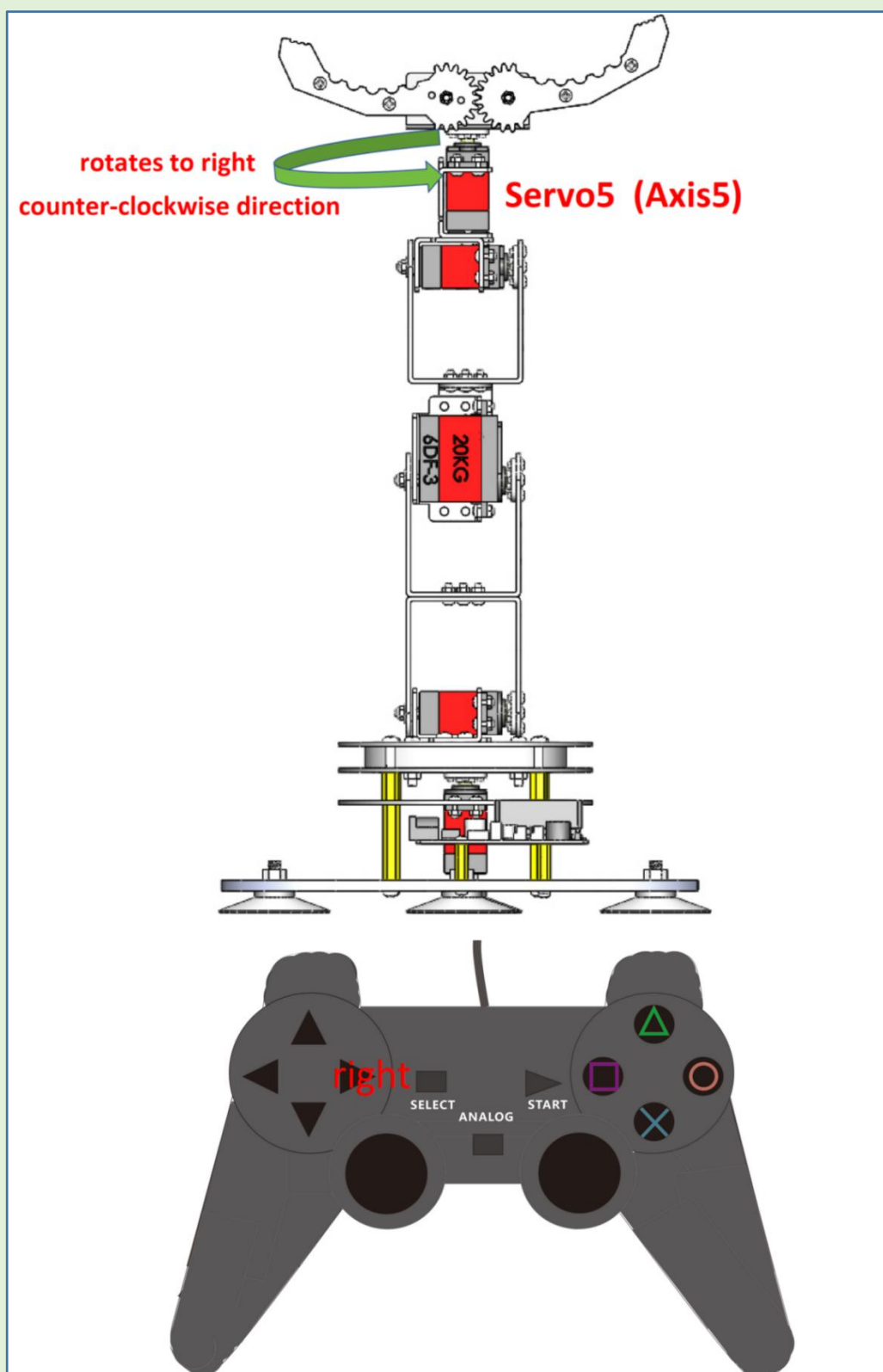
“down” button: Press and hold the down button to control servo 4. Axis 4 moves backwards while the other axes remain stationary. After releasing the button, servo 4 stops moving.



“left” button: Press and hold the left button to control servo5, and rotate Axis5 to the left (clockwise direction). After releasing the button, servo5 stops moving.



“right” button: Press and hold the right button to control servo5, and rotate Axis5 to the right (counterclockwise). After releasing the button, servo5 stops moving.



“triangle” button:Record an action.



After pushing the joystick or pressing the button to make the robotic arm perform an action, press the "triangle" button once, and this action will be memorized. After pushing the joystick or pressing the button again to make the robotic arm perform another action, press the "triangle" button once, and the second action will be memorized. In the code we provide, the default setting is to record 10 actions. After recording 10 actions, the buzzer will ring once to indicate that the action memory is complete.

If you want to remember more actions, you need to change the setting of the number of actions in the code. In line 31 of the code, you can modify the number, and the maximum should not exceed 100.

```
29 CokoinoArm arm;  
30 int xL,yL,xR,yR;  
31 const int act_max=10; //The default setting records 10 actions,  
32 int act[act_max][6]; //Support action memory for 6 servos  
33 int num=0,num_do=0;  
34  
35 #include <Adafruit_NeoPixel.h>  
36 #ifdef __AVR__
```

Change to the quantity you want to record, up to a maximum of 100

“cross” button:Execute recorded actions



After completing the memory action, press the "cross" button, the buzzer rings twice, and the robotic arm executes the memory action. After all actions are completed, the buzzer will sound for 3 seconds and stop.

6. Trouble Shooting

6.1 Arduino IDE failed to recognize Robot Arm Hat

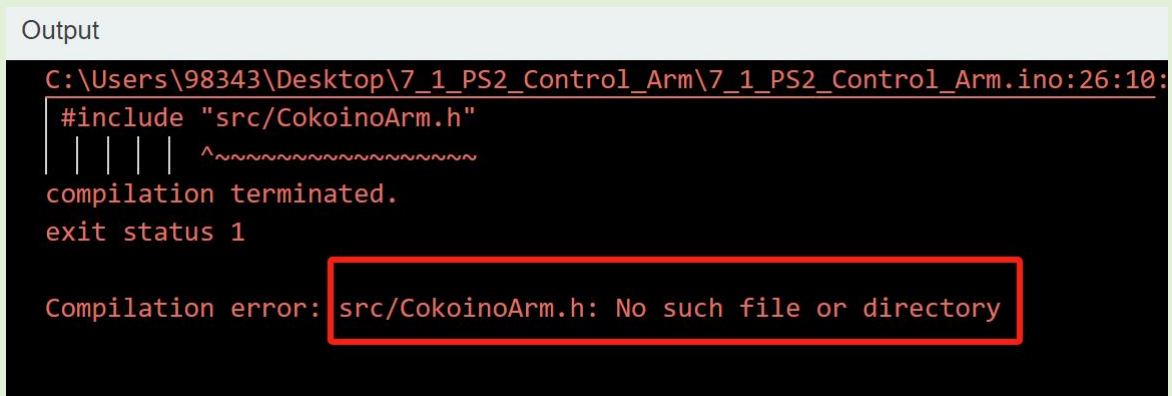
---Check if the USB driver is installed. To install the CH340 driver, please refer to Lesson 1.

---Check if the USB port of the computer is working properly. You can insert a USB flash drive for testing. If the computer can read the USB flash drive, the USB port is working properly.

---Check if there is a problem with the USB cable. If the USB port of the computer is functioning properly, connect the computer and phone with a USB cable. If the computer can read files from the phone, then the USB cable is functioning properly.

6.2 Code compilation failed

Compilation error : src/CokoinoArm.h: No such file or directory

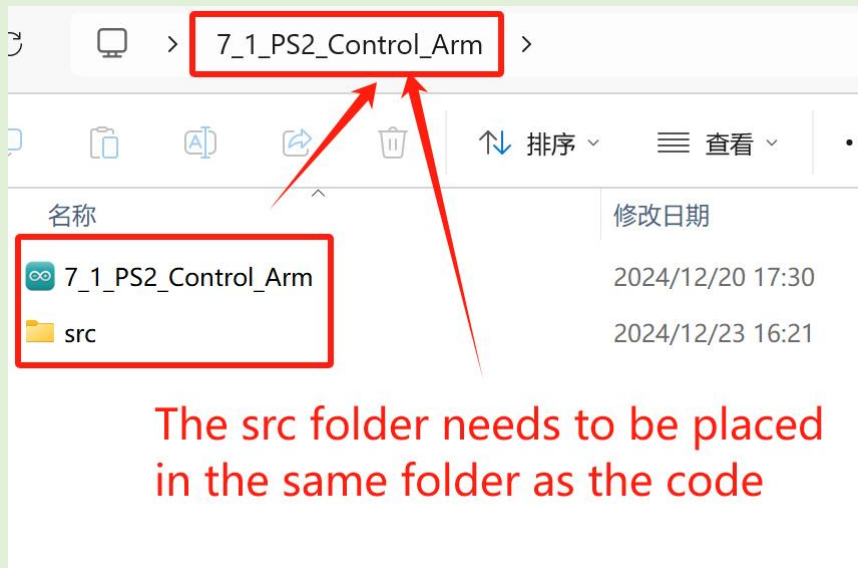


```
Output
C:\Users\98343\Desktop\7_1_PS2_Control_Arm\7_1_PS2_Control_Arm.ino:26:10:
#include "src/CokoinoArm.h"
| | | | ^~~~~~
compilation terminated.
exit status 1

Compilation error: src/CokoinoArm.h: No such file or directory
```

This error message appears during compilation because the src file was not placed in the folder where the code is located, and the code cannot call the libraries in the src file.

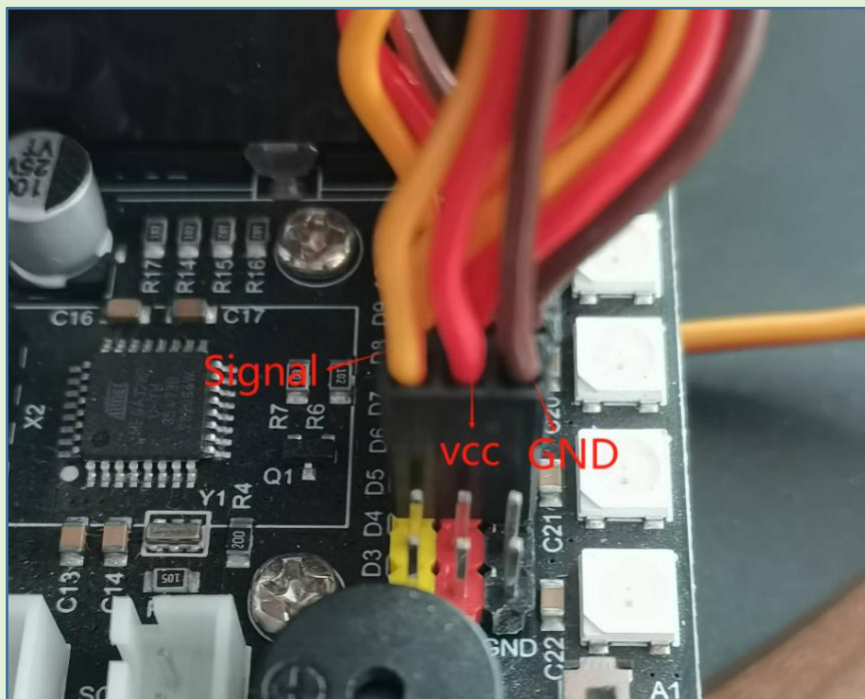
Solution: As shown in the following figure, be sure to put the src file and code in the same folder, because the src file contains some libraries that the code needs. Only by placing them in the same folder can the code call these libraries and compile successfully.



6.3 The robotic arm is not working

---Check if the power light on the Robot Arm Drive Hat is on. If it is not on, it means that there is no power on. Check if the switch is turned to the "ON" state, and if the power adapter is plugged into the Robot Arm Drive Hat and powered on.

---Check if the female terminal of the servo motor is plugged into the correct position on the Robot Arm Drive Hat. Signal corresponds to the yellow pin, VCC corresponds to the red pin, and GND corresponds to the black pin.



6.4 The working methods do not match the description

- Confirm that all 6 servos have been tested and adjusted to 90 degrees before assembly.
- The servos 1, 2, 3, 4, 5, and 6 correspond one-to-one to D2, D3,D4,D5,D6,D7 on the Robot Arm Drive Hat, Confirm that the servo sequence is not connected incorrectly.
- Confirm that the servo cable is long enough and not wrapped around the robotic arm.

6.5 Robotic arm shaking

---The robotic arm shakes without operating the Handle controller. This is because there is a gap in the servo gear, which is around 3° . When the robotic arm is in a standing position (when the center of gravity is not vertically downward), all the weight of the upper part of the Axis2 is loaded onto the Servo2, and under the action of gravity, it drives the servo panel on the Servo2 to rotate downwards towards the clearance angle. When the servo panel rotates to the limit position of the gear clearance, due to the inertia of gravity, it drives the Servo2 gear to rotate downwards by some angle. However, the program in the Robot Arm Drive Hat needs to maintain the angle before Servo2, and then drive Servo2 to rotate upwards back, and then Servo2 will rotate towards the gear clearance under the action of gravity. This cycle creates shaking.

Solution: Use your hands to straighten the body of the robotic arm so that it stands completely vertically with its center of gravity on Servo2 and vertically downward. At this point, the robotic arm will stop shaking.

7. Any questions and suggestions are welcome

Thank you for reading this document!

If you find any errors and omissions in the tutorial, or if you have any suggestions and questions, please feel free to contact us:

cokoino@outlook.com

We will do our best to make changes and publish revisions as soon as possible.

If you want to learn more about Arduino, Raspberry Pi, Smart Cars, Robotics and other interesting products in science and technology, please continue to visit our Amazon Store by search for "**LK COKOINO**" on Amazon. We will continue to launch fun, cost-effective, innovative and exciting products.

LK COKOINO