

Lesson 5 Test the Wired Handle controller

Table

1. What do you need to prepare 2

2. Introduction of the Wired Handle controller 2



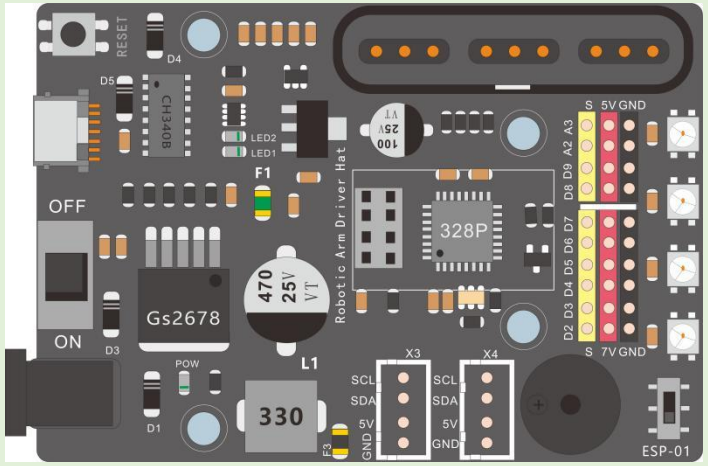
3. Hardware Connection Circuit 7

4. Upload the code and test 8

5. Code 13

6. Any questions and suggestions are welcome 15

1. What do you need to prepare

Components	Quantity	Picture
USB cable	1	
Wired Handle controller	1	
Robotic Arm Drive Hat	1	

2. Introduction of the Wired Handle controller

2.1 preface

The game controller of Playstation 2 has been very perfect since its development. It is well-designed and

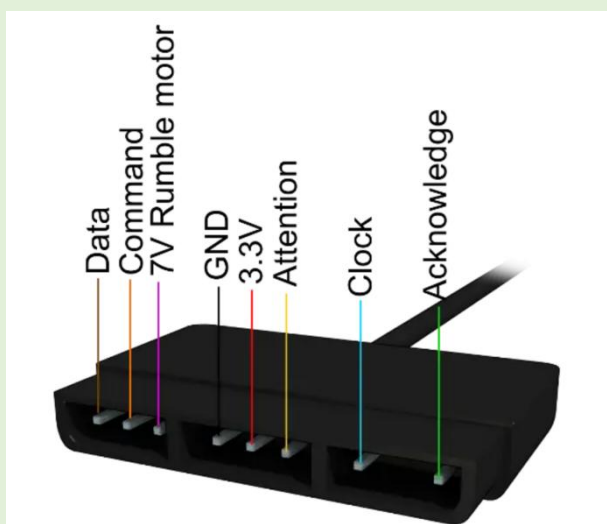
powerful, making it very suitable for use as the control end of Arduino projects, such as remote control cars, robotic arms, and multi legged robots. Although the PlayStation 2 game console is outdated, the Wired Handle controller is still being produced and sold in large quantities, indicating its great popularity. Not only that, the Wired Handle controller can be easily connected to Arduino for communication, it also includes two high-quality joysticks, and all its buttons are pressure-sensitive, which adds more functions to the control of the device.

2.2 Wired Handle controller interface definition

The Wired Handle controller interface has 8 pins on the male head, as defined in the table below

Wired Handle controller Pin	Definition
Pin1	Data
Pin2	Command
Pin3	7V Rumble motor
Pin4	GND
Pin5	3.3V
Pin6	Attention
Pin7	Clock
Pin8	Acknowledge

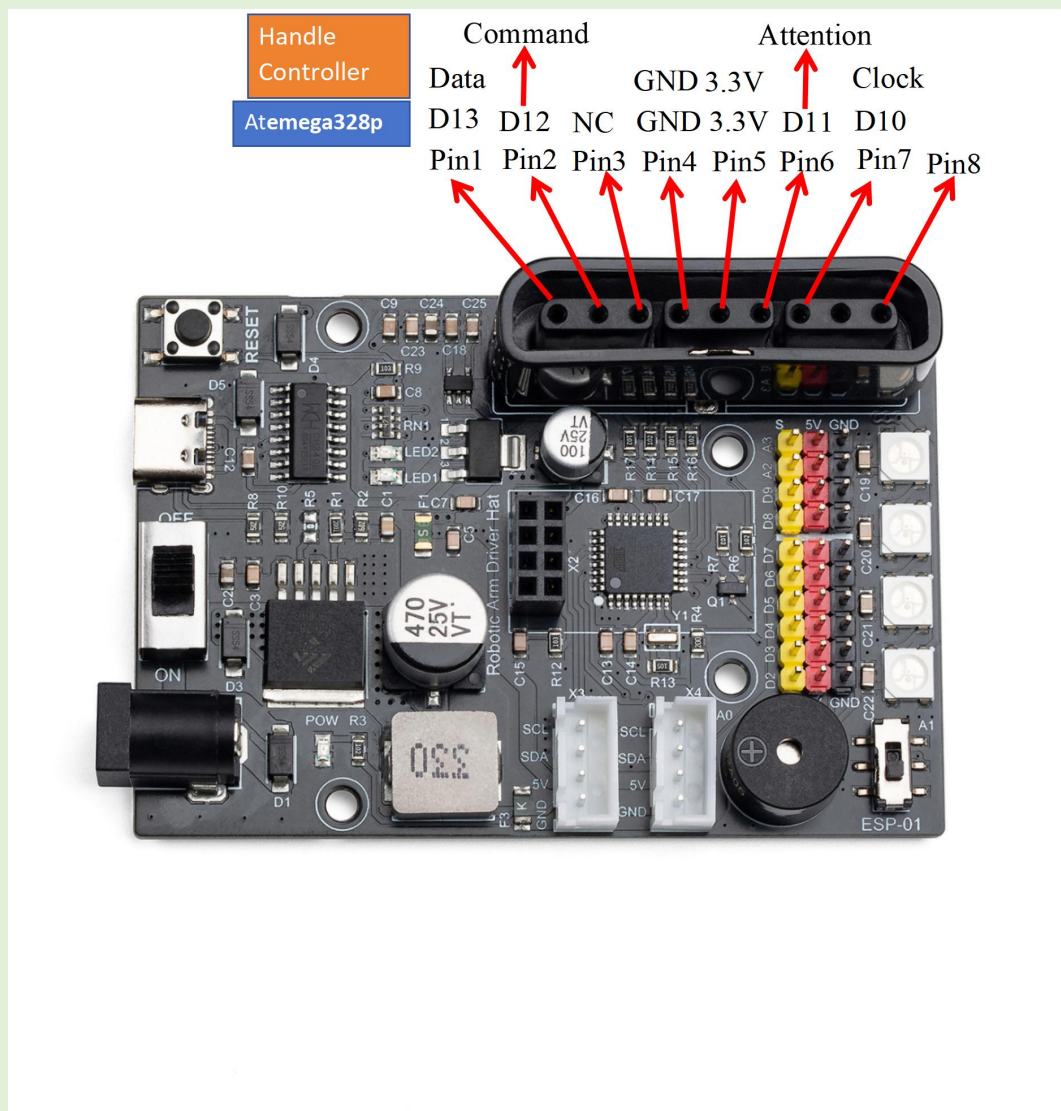
Please refer to the following figure for details



The Handle controller interface female head on the Robotic Arm Hat also has 8 pins, as defined in the table below

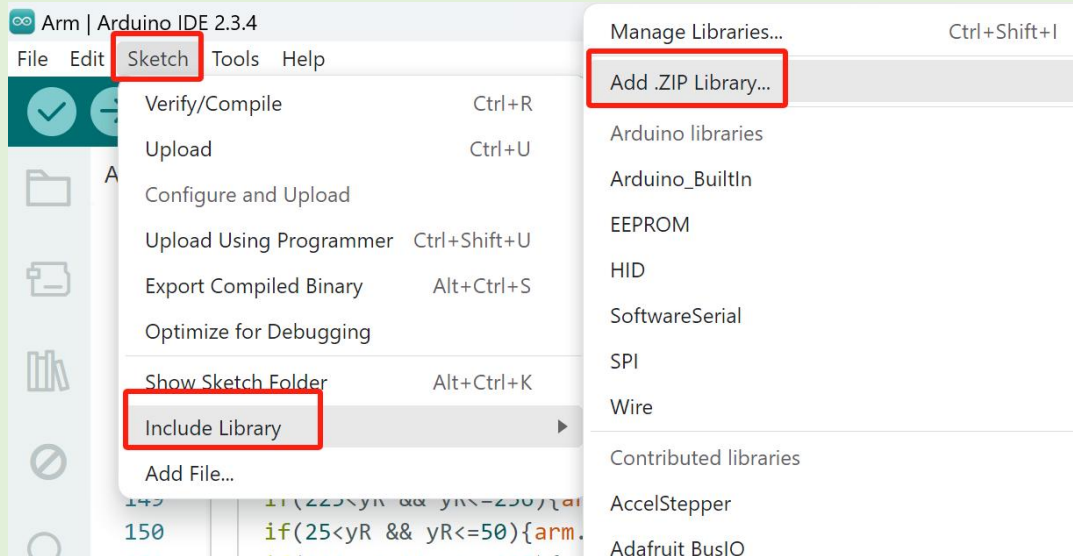
Handle controller interface female head pins	Robotic Arm Driver Hat(Atmega328p)
Pin1	D13
Pin2	D12
Pin3	NC
Pin4	GND
Pin5	3.3V
Pin6	D11
Pin7	D10
Pin8	NC

Please refer to the following figure for details



2.3 Wired Handle controller applied to Arduino

Firstly, we would like to thank Bill Porter for writing a library specifically for Wired Handle controllers, which makes it easy for users to use the Playstation 2 game controller through Arduino. This library is the PS2X library, and its source code can be obtained from Github. After downloading, you will receive a zip file, which can be directly imported into the PS2X library through the import ZIP library file function of Arduino IDE.



The example files included in the library can effectively demonstrate the functionality of the library and utilize all the features of the Playstation 2 controller, covering the basic content required to drive motors, servos, servers, and switches using the Playstation 2 controller.

To use the library in Arduino, you need to call it using the following command:

```
#include <PS2X_lib.h>
```

```
PS2X ps2x; //create Wired Handle controller Class
```

In the setup section of the code, to let Arduino know how the Wired Handle controller is connected to it, the command is as follows:

```
ps2x.config_gamepad(PS2_CLK, PS2_CMD, PS2_SEL, PS2_DAT);
```

Based on the correspondence between the Wired Handle controller interface pins and the Atmega328p on the Robotic Arm Hat, this command can be specifically set as the digital pins of Atmega328p, as follows:

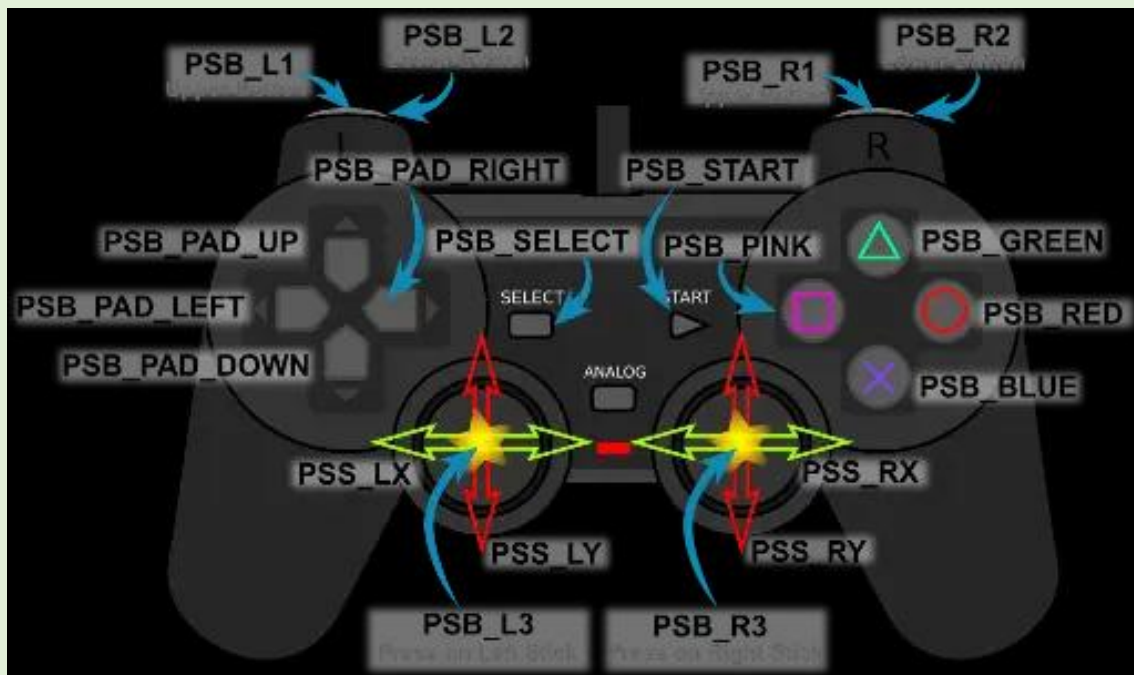
```
ps2x.config_gamepad(10, 12, 11, 13);
```

Connect the Wired Handle controller to the Robot Arm Hat, then upload the code to the Robot Arm Hat, and Arduino will continuously loop through the execution. After each loop is completed, Arduino communicates with the Wired Handle controller and collects all input data. This process is completed through the following command:

```
ps2x.read_gamepad();
```

2.4 The buttons of the Wired Handle controller correspond to the labels and usage methods in the PS2X library

The labels of Handle buttons and joysticks in the PS2X library are shown in the following figure



The colored buttons in the above picture can be referenced by their corresponding names, such as:

The green triangle button can be referenced using [PSB_TRIANGLE](#), the red circular button can be referenced using [PSB_CIRCLE](#), the blue CROSS button can be referenced using [PSB_CROSS](#), and the pink square button can be referenced using [PSB_SQUARE](#)

There are many ways to operate Handle buttons, such as pressing, holding, or pressing with different pressures, so there are also many methods that can be applied to buttons and joysticks. These methods in the PS2X library include:

Button Pressed, Button and Analog

Here are some examples of how to use these methods:

```
ps2x.ButtonPressed(PSB_RED)
```

Simply press the red circle button

```
ps2x.Button(PSB_PAD_DOWN)
```

Press and hold the down button continuously

`ps2x.Analog(PSAB_CROSS)`

Used to measure the pressure applied to the "X" button (if pressure sensitivity is enabled). The button prefix has already been in the form of 'PSAB'.

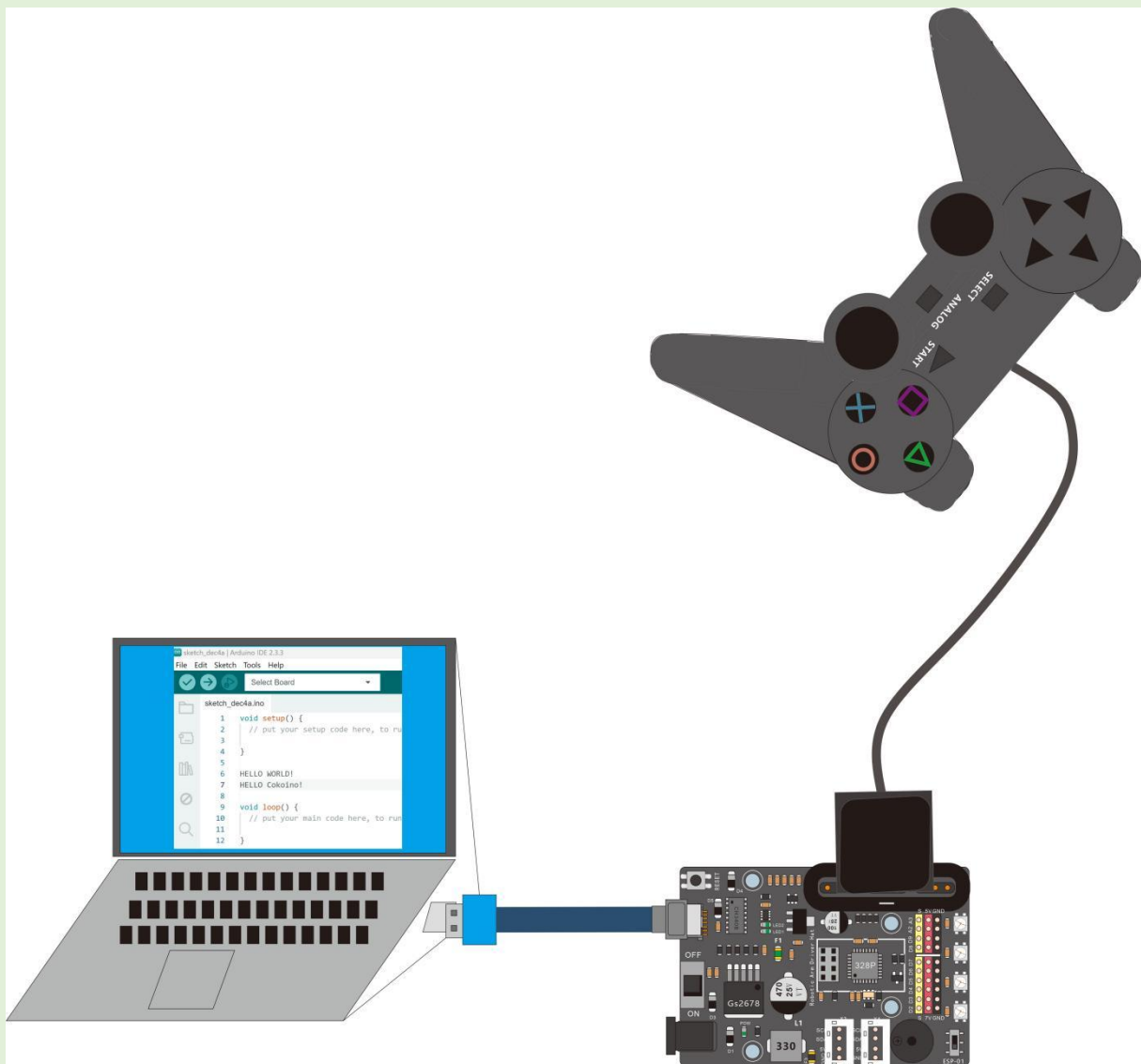
This feature cannot be used on Robot Arm Hat because pressure sensitivity is not enabled.

`ps2x.Analog(PSS_RY)`

Obtain simulated joystick readings in the vertical direction of the right joystick

3. Hardware Connection Circuit

Insert the male connector of the Wired Handle controller into the corresponding female connector on the Robot Arm Driver Hat, and then connect the Robot Arm Driver Hat to the computer using a Type-C USB cable. As shown in the following figure



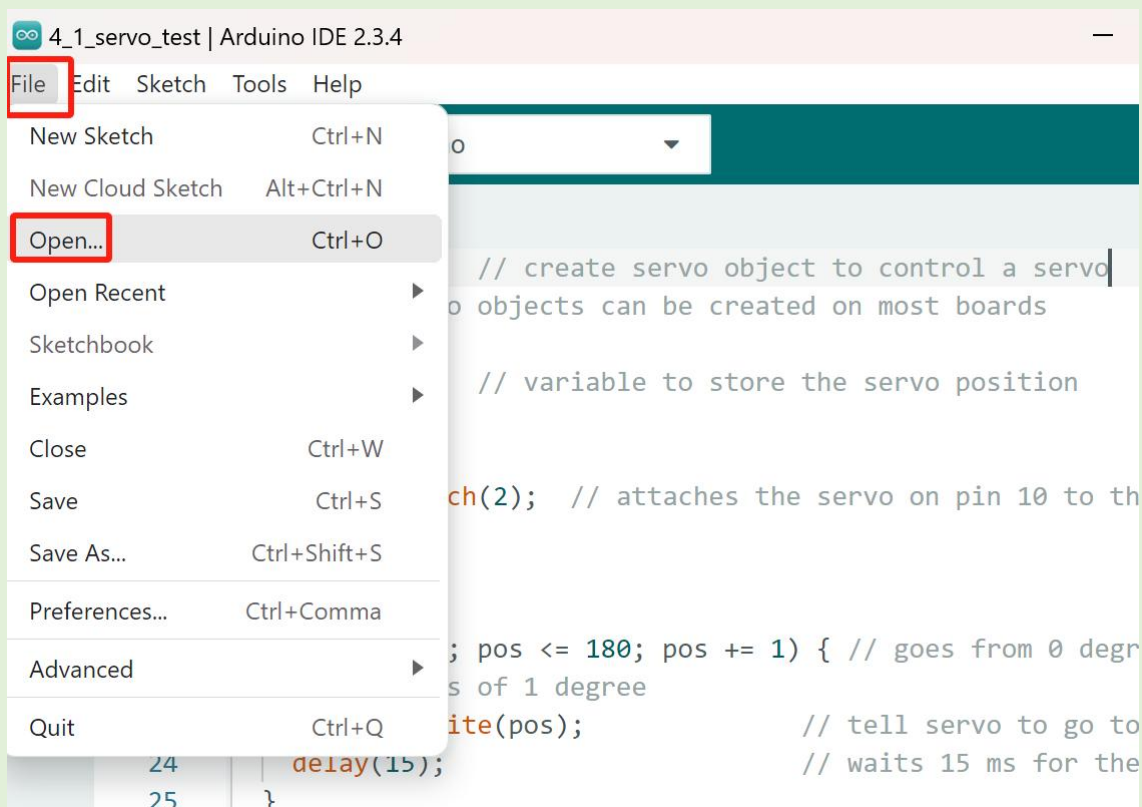
4. Upload the code and test

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

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



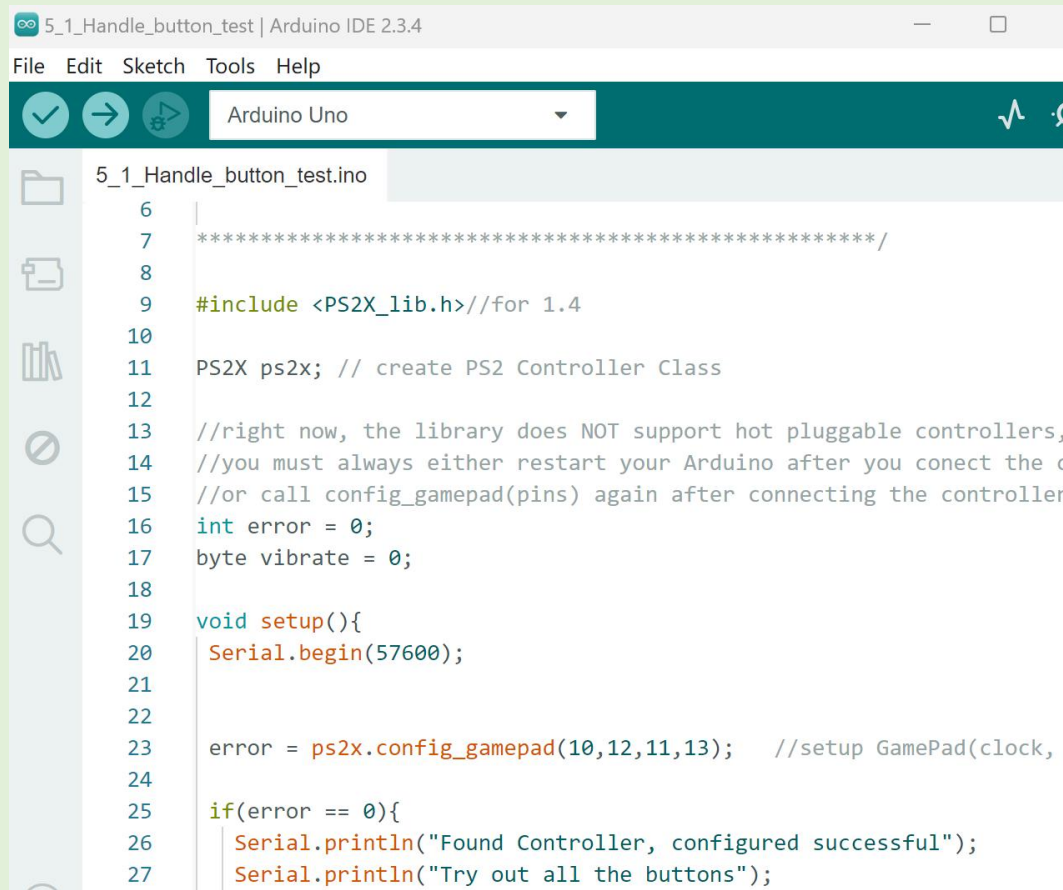
4.2 Click "File" --- "open"



4.3 Select the code in the folder named 5_1_Handle_button_test:

[E:\CKK0017-main\Tutorial\sketches\5_1_Handle_button_test](#).

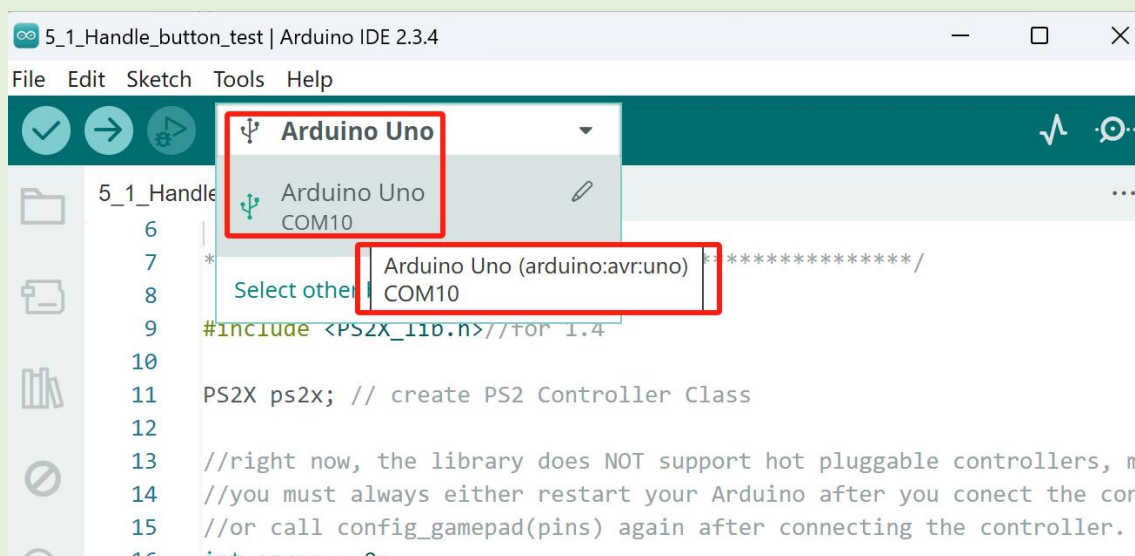
Click "open"



```

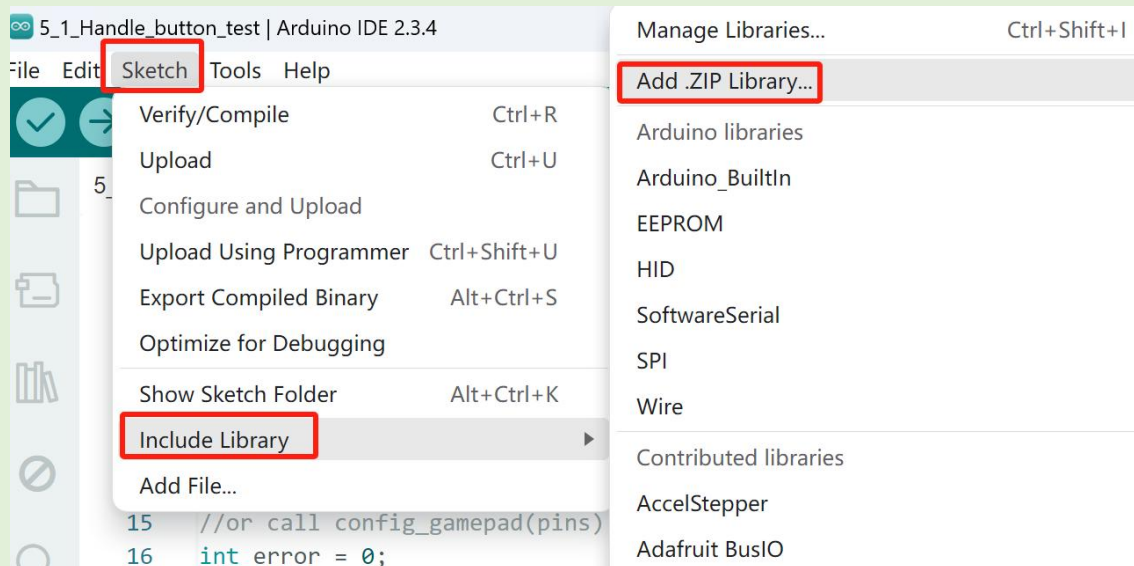
5_1_Handle_button_test | Arduino IDE 2.3.4
File Edit Sketch Tools Help
Arduino Uno
5_1_Handle_button_test.ino
6
7 *****
8
9 #include <PS2X_lib.h> //for 1.4
10
11 PS2X ps2x; // create PS2 Controller Class
12
13 //right now, the library does NOT support hot pluggable controllers,
14 //you must always either restart your Arduino after you conect the c
15 //or call config_gamepad(pins) again after connecting the controller
16 int error = 0;
17 byte vibrate = 0;
18
19 void setup(){
20   Serial.begin(57600);
21
22
23   error = ps2x.config_gamepad(10,12,11,13); //setup GamePad(clock,
24
25   if(error == 0){
26     Serial.println("Found Controller, configured successful");
27     Serial.println("Try out all the buttons");
  
```

4.4 Select the board "Arduino UNO" and Port "COM10" (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 COM10 in this tutorial)



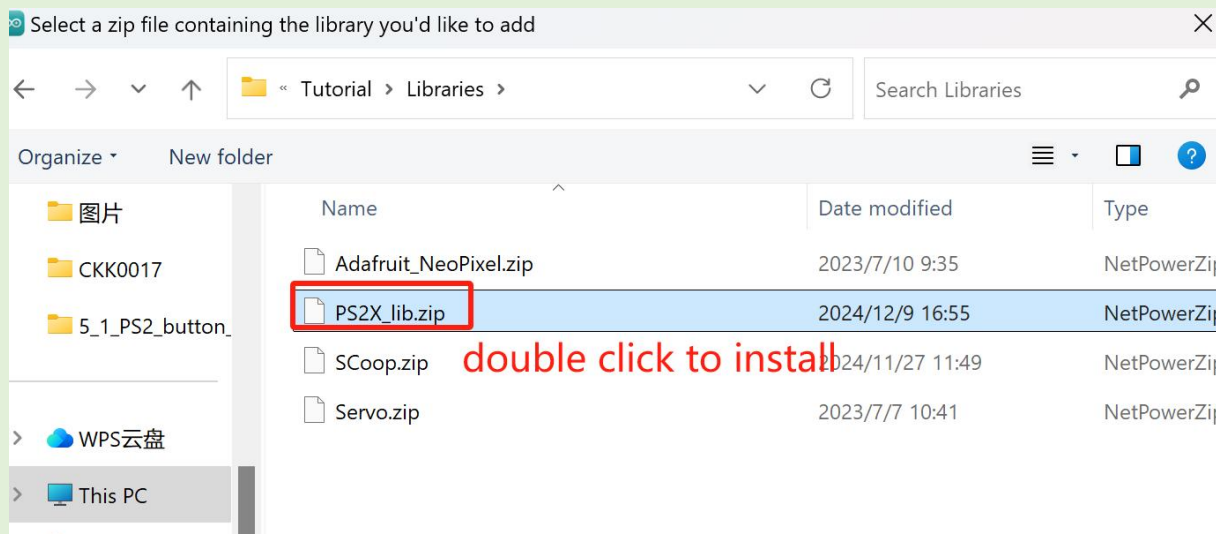
4.5 Install library [PS2X_lib.h](#)


After opening the 5_1_Handle_button_test code, click "Sketch" ---"Include Library"---"Add.ZIP Library..."




Find the PS2X_lib.ZIP file in E:\CKK0017-main\Tutorial\Libraries\PS2X_lib.zip

Double click the PS2X_lib.ZIP file, then it will be installed into Arduino IDE



4.6 Click compile button , successfully compiled the code will display "Done compiling"

4.7 Click upload button , successfully uploading the code will display "Done uploading".

4.8 After the code was successfully uploaded, we began testing the functionality of the buttons and joystick on the Wired Handle controller.

Firstly, click to open the Serial Monitor of Arduino IDE.



Secondly, adjust the baud rate to 57600, monitor will output three lines of information:

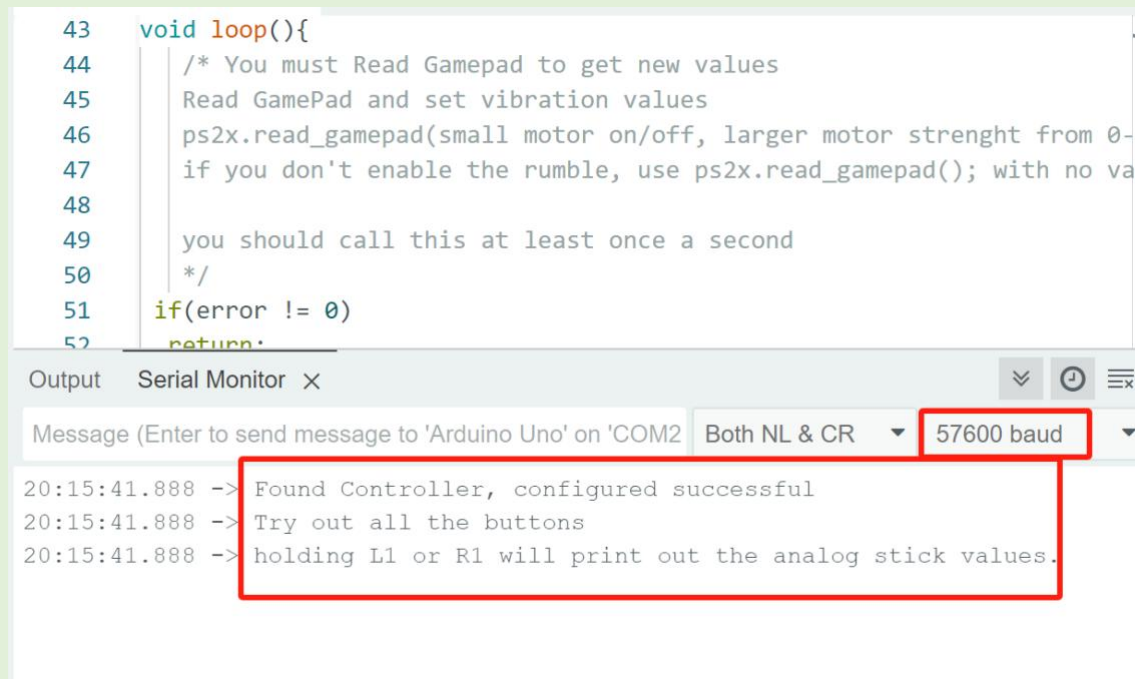
Found Controller, configured successful

Try out all the buttons

holding L1 or R1 will print out the analog stick values.

Thirdly, press different buttons to observe whether the corresponding status information is displayed on the monitor.

For example, holding down the UP button will display "UP is being held" on the monitor



Press and hold the square button, and the monitor will display "SQUARE is being held"

```
35 Serial.println("Controller found but not accepting commands");
36
37 //Serial.print(ps2x.Analog(1), HEX);
38
39 //ps2x.enableRumble(); //enable rumble vibration motors
40 //ps2x.enablePressures(); //enable reading the pressure values fr
```

Output Serial Monitor X

Message (Enter to send message to 'Arduino Uno' on 'COM23')

```
20:23:50.341 -> UP is being held
20:23:50.388 -> UP is being held
20:23:50.422 -> UP is being held
20:30:56.567 -> SQUARE is being held
20:30:56.653 -> SQUARE is being held
20:30:56.700 -> SQUARE is being held
```

When the square button is being pressed

Press and hold the L1 button while pushing the left joystick, and the monitor will display the status value of the left joystick.

Output Serial Monitor X

Message (Enter to send message to 'Arduino Uno' on 'COM23')

```
20:34:53.306 -> L1 is being held
20:34:53.306 -> Stick Values:91,100
20:34:53.351 -> L1 is being held
20:34:53.351 -> Stick Values:91,100
20:34:53.383 -> L1 is being held
20:34:53.383 -> Stick Values:91,100
20:34:53.474 -> L1 is being held
20:34:53.474 -> Stick Values:91,107
20:34:53.506 -> L1 is being held
20:34:53.506 -> Stick Values:91,107
```

The status values of LX and LY range from 0 to 255

Press the other buttons one by one and observe if the monitor displays any status.

5. Code

5_1_Handle_button_test code:

```
1.  #include <PS2X_lib.h> //for 1.4
2.
3.  PS2X ps2x; // create PS2 Controller Class
4.
5.  //right now, the library does NOT support hot pluggable controllers, meaning
6.  //you must always either restart your Arduino after you conect the controller,
7.  //or call config_gamepad(pins) again after connecting the controller.
8.  int error = 0;
9.  byte vibrate = 0;
10. void setup(){
11.    Serial.begin(57600);
12.
13.    error = ps2x.config_gamepad(10,12,11,13); //setup GamePad(clock, command, attention, data) pins, check for error
14.
15.    if(error == 0){
16.        Serial.println("Found Controller, configured successful");
17.        Serial.println("Try out all the buttons");
18.        Serial.println("holding L1 or R1 will print out the analog stick values.");
19.    }
20.
21.    else if(error == 1)
22.        Serial.println("No controller found, check wiring");
23.
24.    else if(error == 2)
25.        Serial.println("Controller found but not accepting commands");
26.
27.    //Serial.print(ps2x.Analog(1), HEX);
28.
29.    //ps2x.enableRumble(); //enable rumble vibration motors
30.    //ps2x.enablePressures(); //enable reading the pressure values from the buttons.
31. }
32.
33. void loop(){
34.    /* You must Read Gamepad to get new values
35.    Read GamePad and set vibration values
36.    ps2x.read_gamepad(small motor on/off, larger motor strenght from 0-255)
37.    if you don't enable the rumble, use ps2x.read_gamepad(); with no values
38.
39.    you should call this at least once a second
```

```
40.     */
41.     if(error != 0)
42.         return;
43.
44.     ps2x.read_gamepad(false, vibrate);    //read controller and set large motor to spin at 'vibrate' speed
45.
46.     if(ps2x.Button(PSB_START)) {          //will be TRUE as long as button is pressed
47.         Serial.println("Start is being held");
48.     }
49.     if(ps2x.Button(PSB_SELECT)){
50.         Serial.println("Select is being held");
51.     }
52.     if(ps2x.Button(PSB_PAD_UP)) {          //will be TRUE as long as button is pressed
53.         Serial.println("UP is being held");
54.     }
55.     if(ps2x.Button(PSB_PAD_RIGHT)){
56.         Serial.println("RIGHT is being held");
57.     }
58.     if(ps2x.Button(PSB_PAD_LEFT)){
59.         Serial.println("LEFT is being held");
60.     }
61.     if(ps2x.Button(PSB_PAD_DOWN)){
62.         Serial.println("DOWN is being held");
63.     }
64.     if(ps2x.Button(PSB_TRIANGLE)){
65.         Serial.println("TRIANGLE is being held");
66.     }
67.
68.     if(ps2x.Button(PSB_CROSS)){
69.         Serial.println("CROSS is being held");
70.     }
71.     if(ps2x.Button(PSB_SQUARE)){
72.         Serial.println("SQUARE is being held");
73.     }
74.     if(ps2x.Button(PSB_CIRCLE)){
75.         Serial.println("CIRCLE is being held");
76.     }
77.
78.     if(ps2x.Button(PSB_L2)){
79.         Serial.println("L2 is being held");
80.     }
81.     if(ps2x.Button(PSB_R2)){
82.         Serial.println("R2 is being held");
83.     }
84.     if(ps2x.Button(PSB_L1)){
```



```
85.      Serial.println("L1 is being held");
86.      Serial.print("Stick Values:");
87.      Serial.print(ps2x.Analog(PSS_LX), DEC); //Left stick, X axis.
88.      Serial.print(",");
89.      Serial.println(ps2x.Analog(PSS_LY), DEC); //Left stick, Y axis.
90.      }
91.      if(ps2x.Button(PSB_R1)){
92.          Serial.println("R1 is being held");
93.          Serial.print("Stick Values:");
94.          Serial.print(ps2x.Analog(PSS_RX), DEC); //Right stick, X axis.
95.          Serial.print(",");
96.          Serial.println(ps2x.Analog(PSS_RY), DEC); //Right stick, Y axis.
97.      }
98.
99.      delay(50);
100.
101.      }
```

6. 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