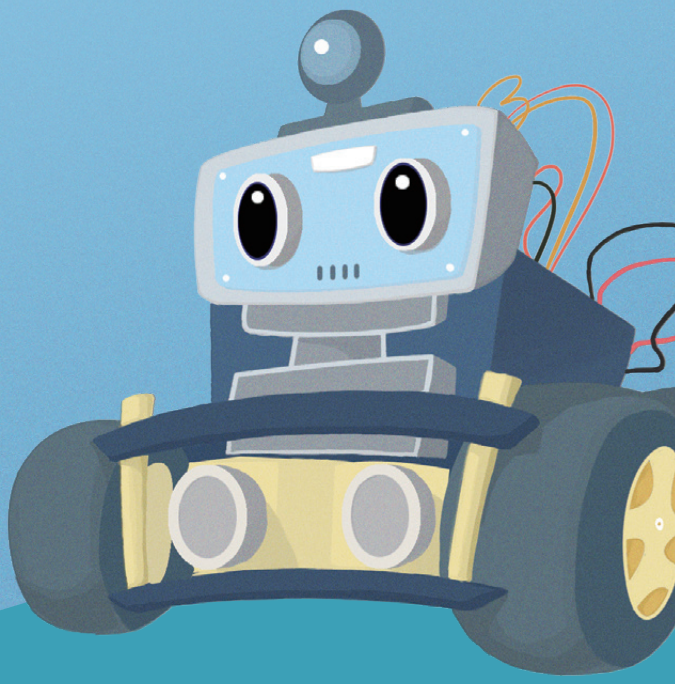
ELEGOO

# 4

# SMART ROBOT

## CAR V4.0 WITH CAMERA



# Servo Control

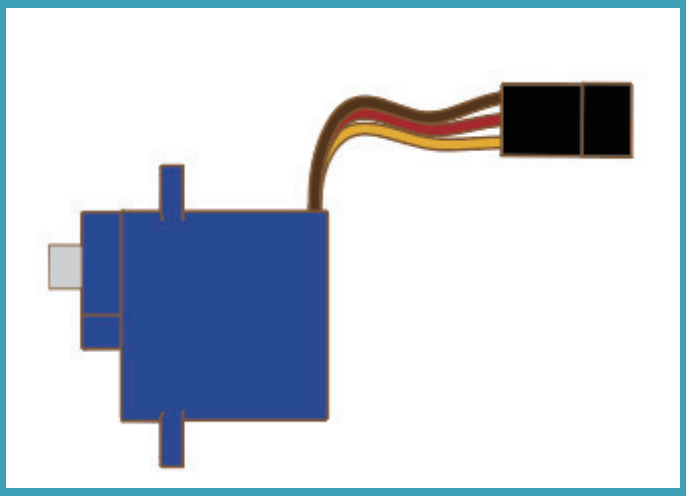
# Introduction:

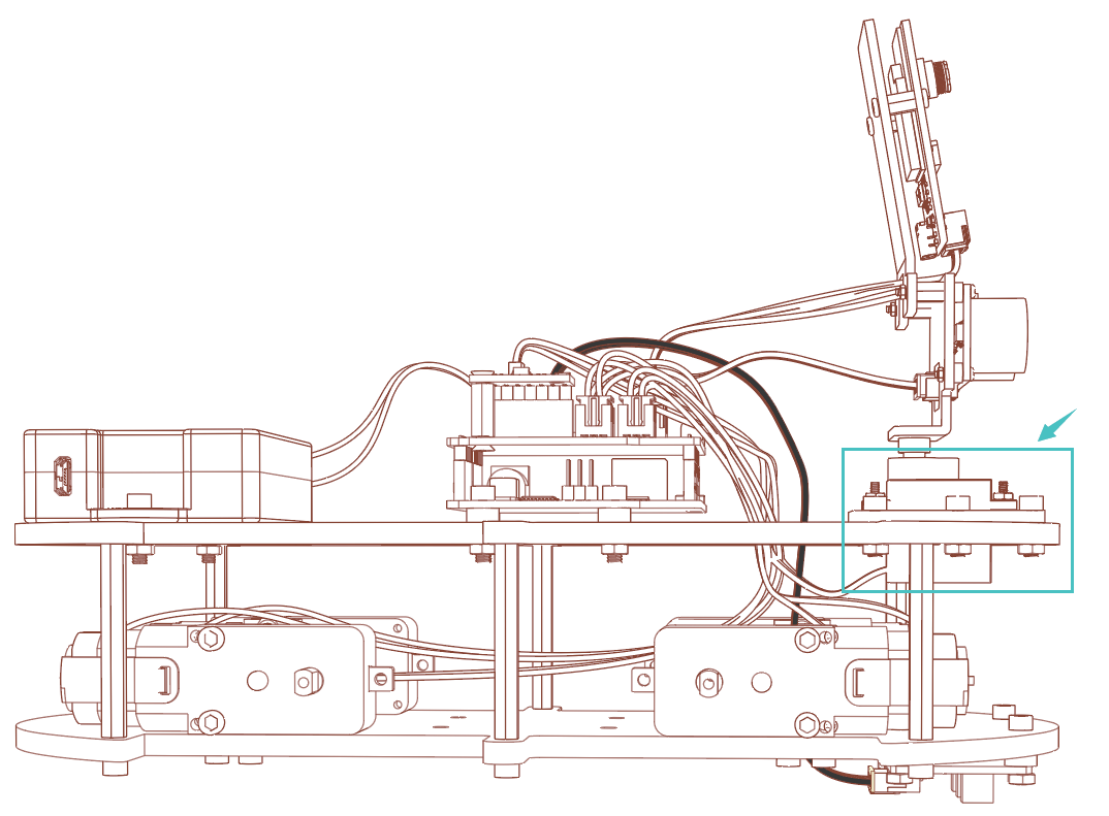
✚ In this lesson, we will teach you how to control the servo of the Smart Robot Car.

# Preparation:

✚ A Smart Robot Car (with battery)
A USB Wire

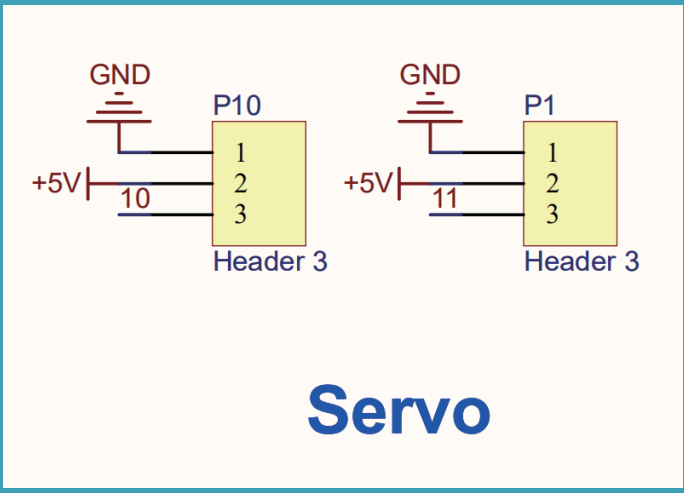
✚ Servo is a type of motor, also known as the servo motor. It is similar to stepper motor. The difference between servo and stepper motor is that the stepper motor can set how many angles to turn, while the servo can set the position to turn.





Please open the file of the last folder for more details: Related chip information -> SmartRobot-Shield

✚ As shown in the picture, servo can be connected to the pin D10 or pin D11 of the UNO.




And the control of the servo is also relatively simple. After adding the official library, you can modify the angle to control it at will.

✚ Please open the Demo1 in the current folder:

First of all, let’ s take a look at the definition of the ultrasonic relative pins and variables: (In this example, we connected servo with pin D10.)

C:/ // in DeviceDriverSet_xxx0.h

```
/*Servo*/
#include <Servo.h>
class DeviceDriverSet_Servo
{
public:
 void DeviceDriverSet_Servo_Init(unsigned int Position_angle);
#if _Test_DeviceDriverSet
 void DeviceDriverSet_Servo_Test(void);
#endif
 void DeviceDriverSet_Servo_control(unsigned int Position_angle);
private:
#define PIN_Servo_z 10
};
```

Let's first learn about the common functions

| | |
|---|---|
| uint8_t attach(int pin) | attach the given pin to the next free channel, sets pinMode, returns channel number or 0 if failure |
| uint8_t attach(int pin, int min, int max) | as above but also sets min and max values for writes |
| void write(int value); | if value is < 200 its treated as an angle, otherwise as pulse width in microseconds |
| void detach() | Detach the servo from its interface, which can continue to be used as the PWM interface |

And then, initial the servo which are connected to pin D10 and pin D11, and then set the servo position to 90 degrees.

```
// in DeviceDriverSet_xxx0.cpp

Servo myservo; // create servo object to control a servo
void DeviceDriverSet_Servo::DeviceDriverSet_Servo_Init
(unsigned int Position_angle)
{
 myservo.attach(PIN_Servo_z, 500, 2400);
 //500: 0 degree  2400: 180 degree
 myservo.attach(PIN_Servo_z);
 myservo.write(Position_angle);
 //sets the servo position according to the 90 (middle)
 delay(500);
}
```

And then choose the servo you need according to the rotation angles of the servo.

```
// in DeviceDriverSet_xxx0.cpp

/*0.17sec/60degree(4.8v)*/
void DeviceDriverSet_Servo::DeviceDriverSet_Servo_control
(unsigned int Position_angle)
{
 myservo.attach(PIN_Servo_z);
 myservo.write(Position_angle);
 delay(450);
 myservo.detach();
}
```

Finally, upload the program. (Please toggle the “Upload-Cam” button to “Upload” when uploading the program.) After is has been uploaded successfully, you will see the servo rotate half a round and than back to the middle position.