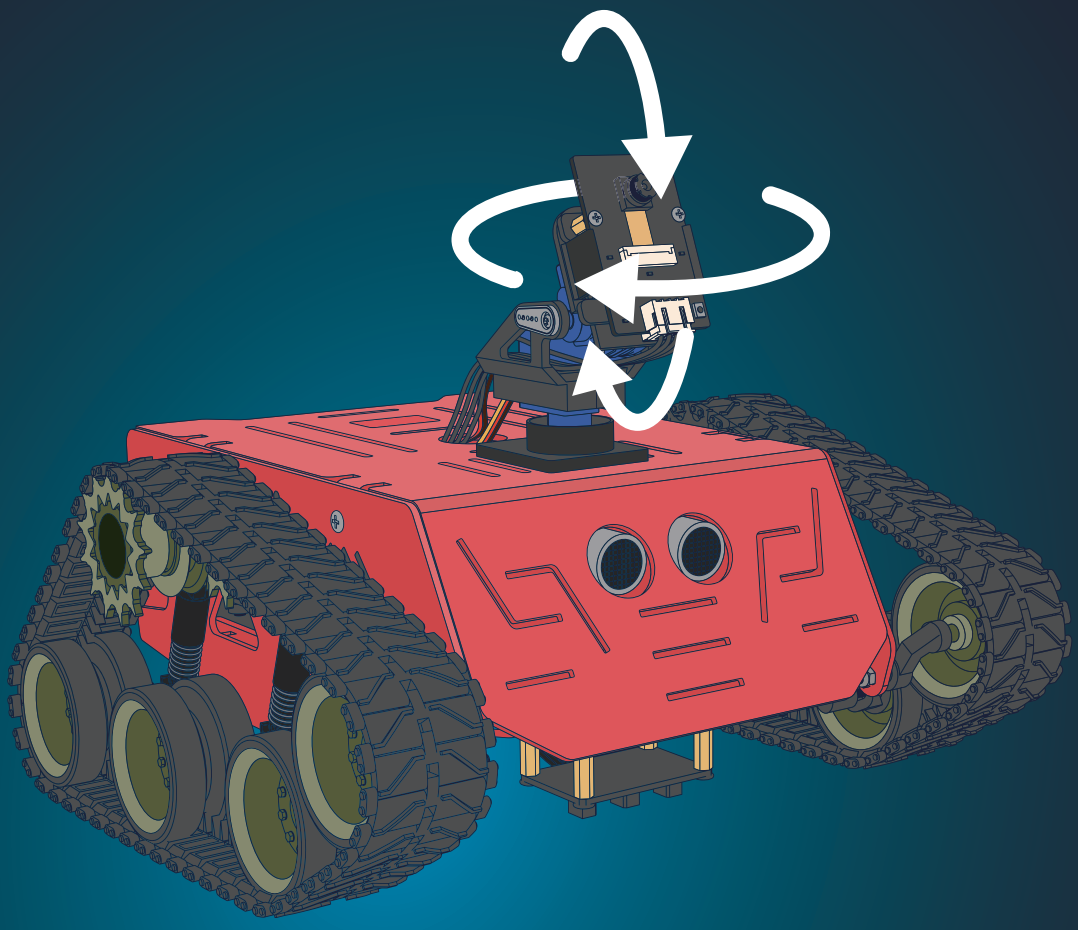
ELEGOO

4 Lesson

ServoControl



ELEGOO.COM

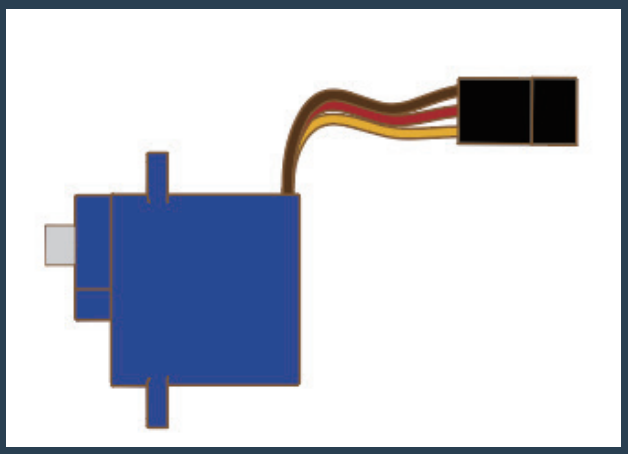
Introduction:

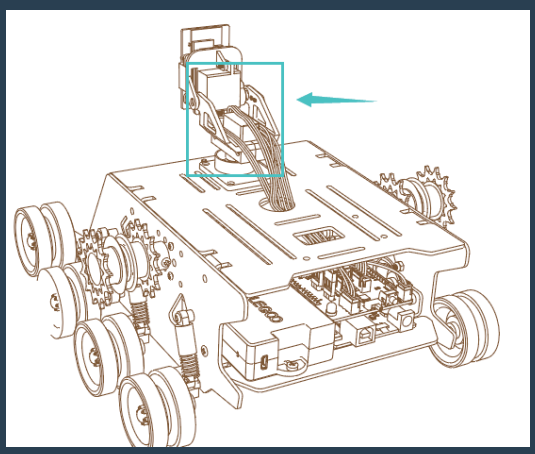
In this lesson, we will teach you how to control the servo of the Conqueror Robot Tank Car.

Material Preparation:

A Conqueror Robot Tank Car (with battery)
A USB Cable

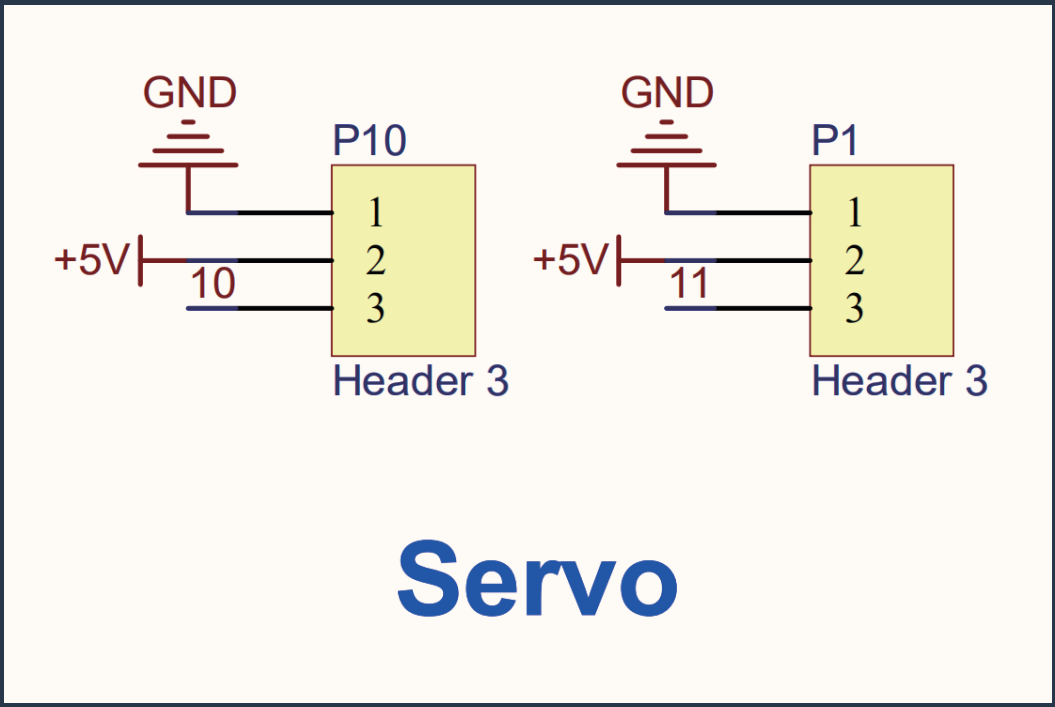
Servo is a type of motor, also known as a servo motor. It is similar to stepper motor. The difference between servo and stepper motor is that the stepper motor can set how many angles to turn, while the servo can set the position to turn.





Please open the last folder: Related chip information -> Conqueror-Shield

As shown in the picture, the servo can be connected to the D10 and D11 pins of UNO.



And the control of 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 to the definition of related pins and variables of ultrasonic.

```
// 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);
 void DeviceDriverSet_Servo_controls
 (uint8_t Servo, unsigned int Position_angle);

private:
#define PIN_Servo_z 10
#define PIN_Servo_y 11
};
```

Let’s learn the common functions as follows first:

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.

Then, initial the servo which are connected to the D10 and D11 pins, 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);

  myservo.attach(PIN_Servo_y, 500, 2400); //500: 0 degree  2400: 180 degree
  myservo.attach(PIN_Servo_y);
  myservo.write(Position_angle);
//sets the servo position according to the 90 (middle)
  delay(500);
  myservo.detach();
}
```

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

```
// in DeviceDriverSet_xxx0.cpp

void DeviceDriverSet_Servo::DeviceDriverSet_Servo_controls
(uint8_t Servo, unsigned int Position_angle)
{
  if (Servo == 1) //Servo_z
  {
    myservo.attach(PIN_Servo_z);
    myservo.write(10 * Position_angle);
    //delay(30 * 3);
    delay(500);
  }
  else if (Servo == 2) //Servo_y
  {
    myservo.attach(PIN_Servo_y);
    myservo.write(10 * Position_angle);
    //delay(30 * 3);
    delay(500);
  }
  myservo.detach();
}
```

We program the application function of the servo as follows:

```
// in ApplicationFunctionSet_xxx0.cpp

void
ApplicationFunctionSet::ApplicationFunctionSet_Servo(uint8_t Set_Servo)
```

```
// in ApplicationFunctionSet_xxx0.cpp

void setup()
{
  // put your setup code here, to run once:
  Application_FunctionSet.ApplicationFunctionSet_Init();
  for (int i = 1; i <= 5; i++)
  {
    for (int j = 0; j < 5; j++)
    {
      Application_FunctionSet.ApplicationFunctionSet_Servo(i);
    }
  }
}
```

Finally, upload the program. (Please toggle the “Upload-Cam” button to “Upload” when uploading the program.) You will find that the servo moving up and down, turning left and right in sequence after uploading the program successfully.