# Servo library

## Methods

### attach()

Attach the Servo variable to a pin. Note that in Arduino IDE 0016 and earlier, the Servo library supports servos on only two pins: 9 and 10.

#### Syntax

servo.attach(pin)

servo.attach(pin, min, max)

#### Parameters

* *servo*: a variable of type Servo
* *pin*: the number of the pin that the servo is attached to
* *min* (optional): the pulse width, in microseconds, corresponding to the minimum (0 degree) angle on the servo (defaults to 544)
* *max* (optional): the pulse width, in microseconds, corresponding to the maximum (180 degree) angle on the servo (defaults to 2400)

#### Example

#include <Servo.h>

Servo myservo;

void setup()

{

myservo.attach(9);

}

void loop() {}

#### See also

* [attached()](#attached)
* [detach()](#detach)

### write()

Writes a value to the servo, controlling the shaft accordingly. On a standard servo, this will set the angle of the shaft (in degrees), moving the shaft to that orientation. On a continuous rotation servo, this will set the speed of the servo (with 0 being full-speed in one direction, 180 being full speed in the other, and a value near 90 being no movement).

#### Syntax

servo.write(angle)

#### Parameters

* *servo*: a variable of type Servo
* *angle*: the value to write to the servo, from 0 to 180

#### Example

#include <Servo.h>

Servo myservo;

void setup()

{

myservo.attach(9);

myservo.write(90); // set servo to mid-point

}

void loop() {}

#### See also

* [attach()](#attach)
* [read()](#read)

### writeMicroseconds()

Writes a value in microseconds (us) to the servo, controlling the shaft accordingly. On a standard servo, this will set the angle of the shaft. On standard servos a parameter value of 1000 is fully counter-clockwise, 2000 is fully clockwise, and 1500 is in the middle.

Note that some manufactures do not follow this standard very closely so that servos often respond to values between 700 and 2300. Feel free to increase these endpoints until the servo no longer continues to increase its range. Note however that attempting to drive a servo past its endpoints (often indicated by a growling sound) is a high-current state, and should be avoided.

Continuous-rotation servos will respond to the writeMicrosecond function in an manner analogous to the write function.

#### Syntax

servo.writeMicroseconds(us)

#### Parameters

* *servo*: a variable of type Servo
* *us*: the value of the parameter in microseconds (int)

#### Example

#include <Servo.h>

Servo myservo;

void setup()

{

myservo.attach(9);

myservo.writeMicroseconds(1500); // set servo to mid-point

}

void loop() {}

#### See also

* [attach()](#attach)
* [read()](#read)

### read()

Read the current angle of the servo (the value passed to the last call to [write()](#write)).

#### Syntax

servo.read()

#### Parameters

* *servo*: a variable of type Servo

#### Returns

The angle of the servo, from 0 to 180 degrees.

#### See also

* [write()](#write)

### attached()

Check whether the Servo variable is attached to a pin.

#### Syntax

servo.attached()

#### Parameters

* *servo*: a variable of type Servo

#### Returns

true if the servo is attached to pin; false otherwise.

#### See also

* [attach()](#attach)
* [detach()](#detach)

### detach()

Detach the Servo variable from its pin. If all Servo variables are detached, then pins 9 and 10 can be used for PWM output with [analogWrite()](https://www.arduino.cc/reference/en/language/functions/analog-io/analogwrite/).

#### Syntax

servo.detach()

#### Parameters

* *servo*: a variable of type Servo

#### See also

* [attach()](#attach)
* [attached()](#attached)