Servo Motors

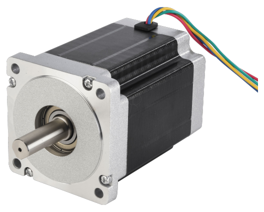
Yesterday we learned how to use basic hobby motors. Today we're going to practice using servo motors.

Motors

- Devices that convert electrical energy into rotational kinetic energy

Basic DC Motors

Servo Motors

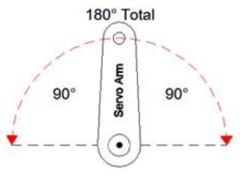


Stepper Motors

Yesterday we learned how to use basic hobby motors. Today we’re going to practice using servo motors.

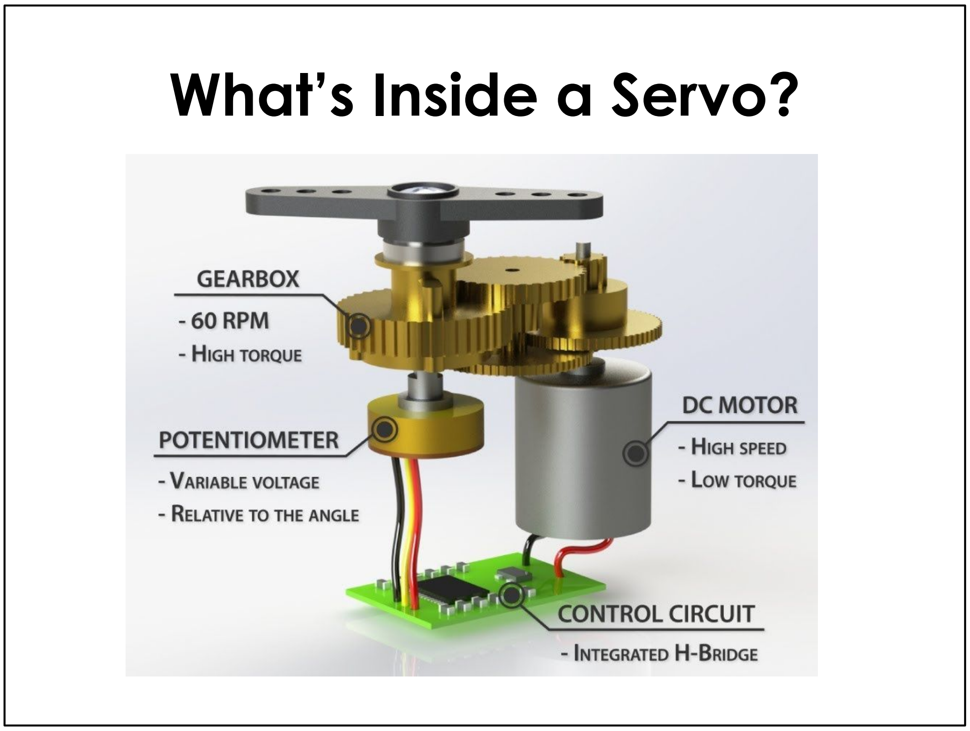
Standard Servos

- Precise: Can rotate to a specific angle
- Limited rotation
  - often 180°
- High torque at high speeds
- Good for back and forth, open and closed, dials
- What are some devices that could use servo motors?

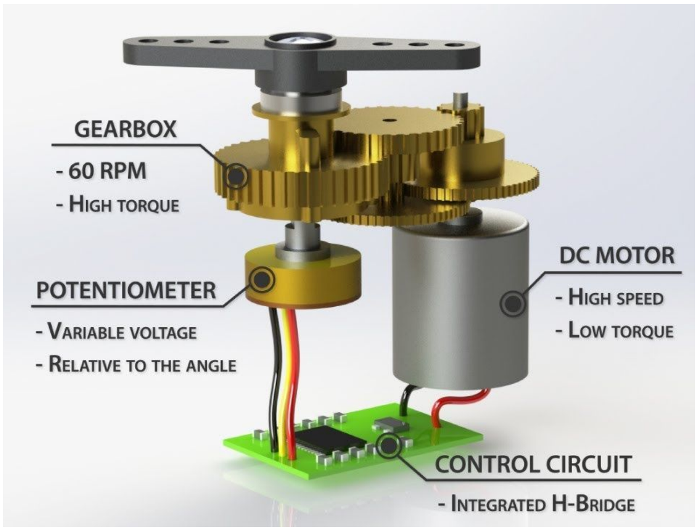


Robotic arms, factory equipment, camera lenses, DVD players (to extend/retract the disk tray)

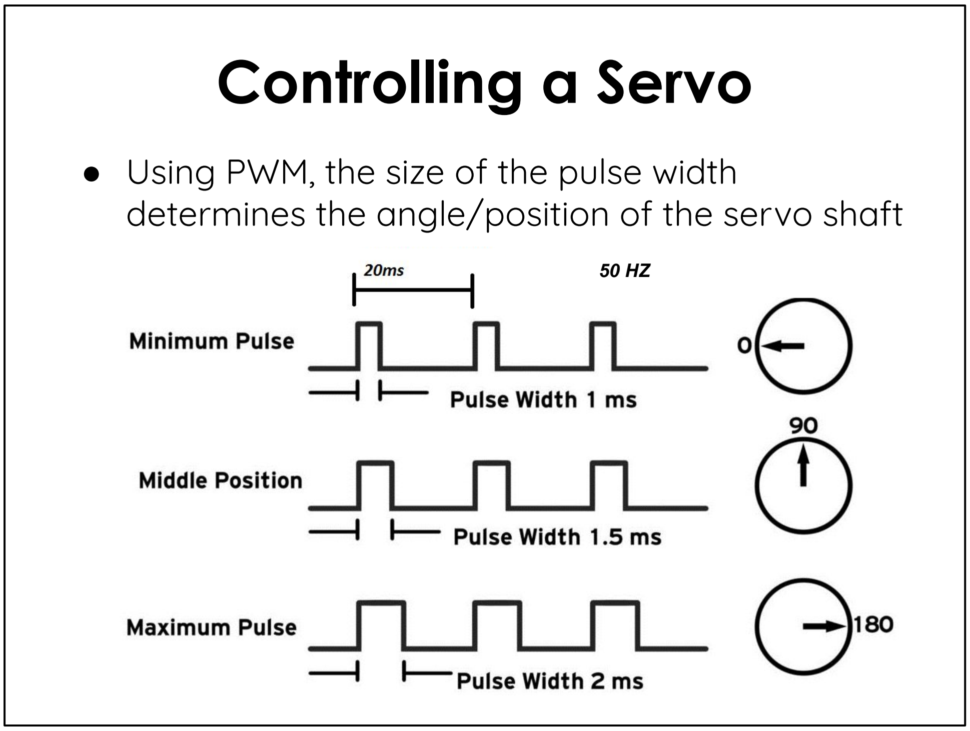
Other advantages: More power efficient than stepper motors
Other disadvantages: more expensive



What's Inside a Servo?



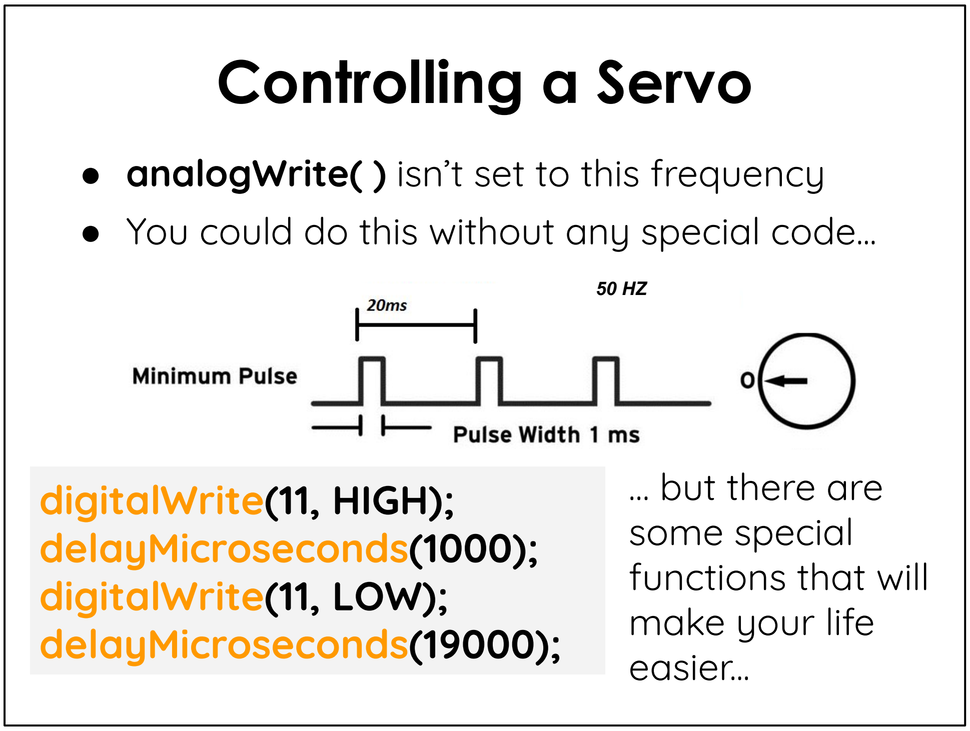
Recap from intro lesson: Chip controls the DC motor, turning it on until the motor has rotated the correct amount and then turning it off.



Controlling a Servo

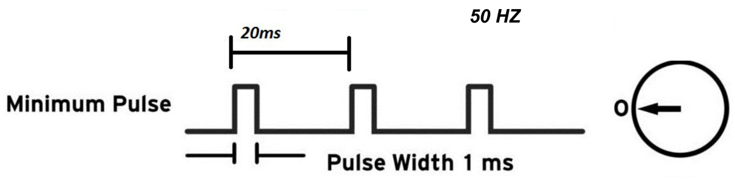
- Using PWM, the size of the pulse width determines the angle/position of the servo shaft

We communicate with the Servo by sending pulses of alternating high and low voltage. The chip inside the servo receives the pulses and translates them into a specific angle.



Controlling a Servo

- **analogWrite( )** isn't set to this frequency
- You could do this without any special code...



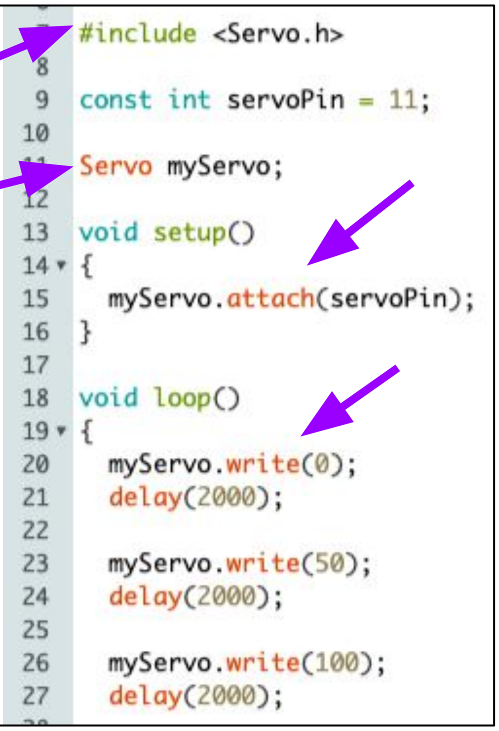
```
digitalWrite(11, HIGH);
delayMicroseconds(1000);
digitalWrite(11, LOW);
delayMicroseconds(19000);
```

... but there are some special functions that will make your life easier...

If we have the same thing repeating over and over many times, what should we do to make our code less repetitive? Make a function. Other people have already written a bunch of really useful functions, for example to set the servo to any angle. This prewritten code is stored in a "library" which we can access by typing #include <Servo.h>

Coding with Servos

- There are some useful structures and functions for controlling servos
- To access them, you need the **Servo Library**



```
#include <Servo.h>

const int servoPin = 11;

Servo myServo;

void setup()
{
  myServo.attach(servoPin);
}

void loop()
{
  myServo.write(0);
  delay(2000);

  myServo.write(50);
  delay(2000);

  myServo.write(100);
  delay(2000);
```

Libraries

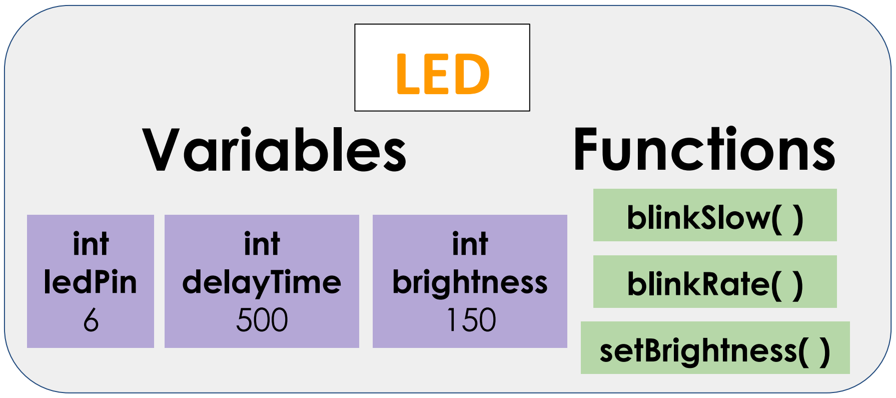
- A **library** is a collection of code that makes it easier to operate a sensor or actuator
  - This code is not normally included in the standard sketch (to save memory, keep sketches simple)

In Standard Arduino Sketches…

Always Included:	**Not Included Unless Instructed To:**
int, float	Servo
delay( )	attach( ), write( )
digitalWrite( )	

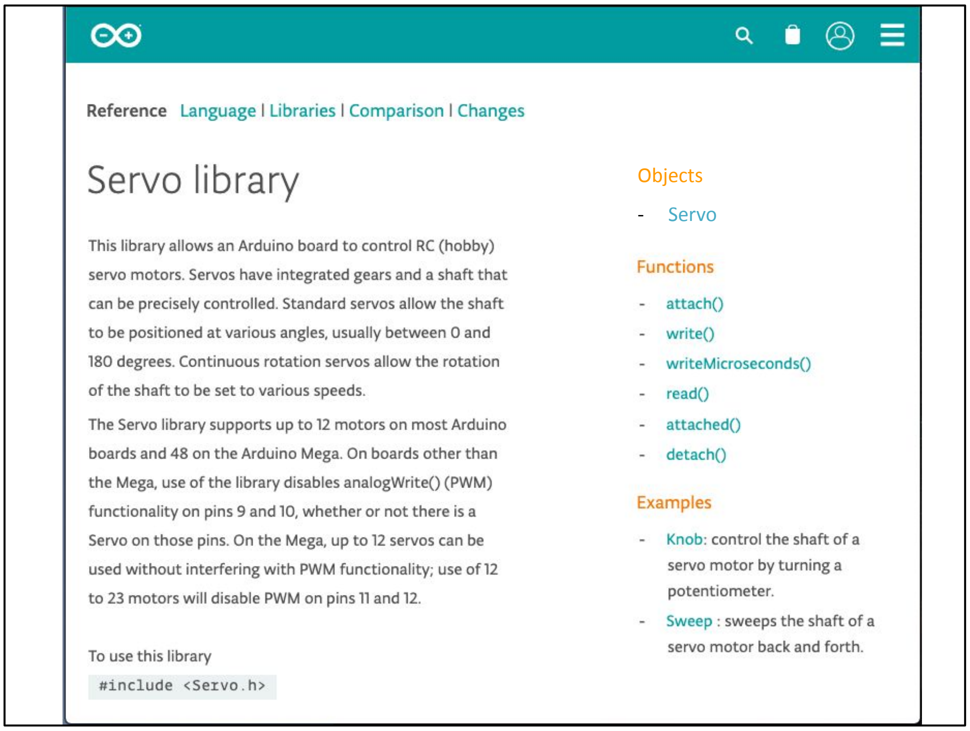
What’s in a Library?

- Constants and Variables
- Functions
- Objects
  - Variables and Functions



Objects

- A structure (“package”) of variables and functions grouped together
- A template for creating copies (“instances”)



Reference Language | Libraries | Comparison | Changes

Servo library

This library allows an Arduino board to control RC (hobby) servo motors. Servos have integrated gears and a shaft that can be precisely controlled. Standard servos allow the shaft to be positioned at various angles, usually between 0 and 180 degrees. Continuous rotation servos allow the rotation of the shaft to be set to various speeds.

The Servo library supports up to 12 motors on most Arduino boards and 48 on the Arduino Mega. On boards other than the Mega, use of the library disables analogWrite() (PWM) functionality on pins 9 and 10, whether or not there is a Servo on those pins. On the Mega, up to 12 servos can be used without interfering with PWM functionality; use of 12 to 23 motors will disable PWM on pins 11 and 12.

To use this library

```
#include <Servo.h>
```

Objects

- Servo

Functions

- attach()
- write()
- writeMicroseconds()
- read()
- attached()
- detach()

Examples

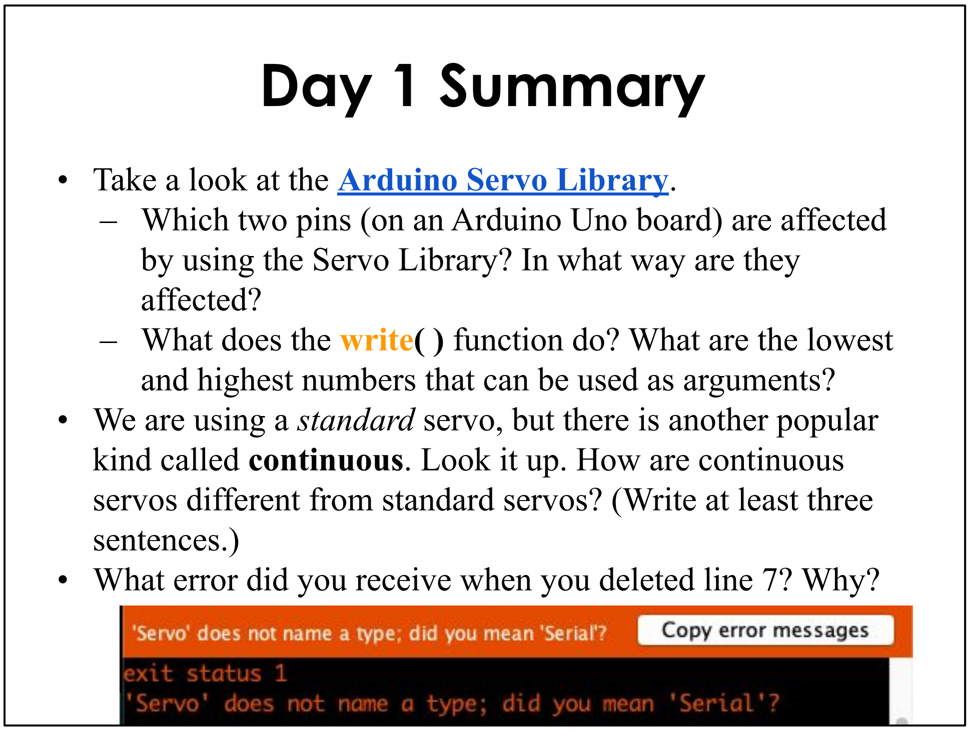
- Knob: control the shaft of a servo motor by turning a potentiometer.
- Sweep : sweeps the shaft of a servo motor back and forth.

You can learn more by checking out the documentation

Work Time!

- Work with your partner to begin the Servo Motors assignment (posted on Google Classroom)

me after working for 15 minutes
I've had a very long, hard day...



Day 1 Summary

- Take a look at the **Arduino Servo Library**.
  - Which two pins (on an Arduino Uno board) are affected by using the Servo Library? In what way are they affected?
  - What does the **write( )** function do? What are the lowest and highest numbers that can be used as arguments?
- We are using a *standard* servo, but there is another popular kind called **continuous**. Look it up. How are continuous servos different from standard servos? (Write at least three sentences.)
- What error did you receive when you deleted line 7? Why?

```
'Servo' does not name a type; did you mean 'Serial'?
exit status 1
'Servo' does not name a type; did you mean 'Serial'?
```

1. Take a look at the **Arduino Servo Library**.
   a. Which two pins (on an Arduino Uno board) are affected by using the Servo Library? In what way are they affected?

      Answer: pins 9 and 10; when using the Servo library you can't use pulse-width modulation for those pins (i.e. you can't use analogWrite() to have a voltage other than 0V/5V.

   b. What does the **write( )** function do? What are the lowest and highest numbers that can be used as arguments?

      Answer: For a normal servo this sets the angle. For a continuous servo this sets the speed. It takes numbers from 0 to 180.

2. We are using a *standard* servo, but there is another popular kind called **continuous**. Look it up. How are continuous servos different from standard servos? (Write at least three sentences.)
   Answer: a standard servo only spins 180 degrees while a continuous servo can spin indefinitely.

3. What error did you receive when you deleted line 7? Why?

Answer: ‘Servo does not name a type; did you mean ‘Serial’?’

This happens because without incorporating the servo library, Servo isn’t defined. The computer doesn’t know what it means.

Servo Motors Day

Work Time!

- Work with your partner to complete the Servo Motors assignment (posted on Google Classroom)

Summary

- One thing you accomplished

OR

- One bug you overcame

OR

- One idea you have for how to use stepper motors in the future