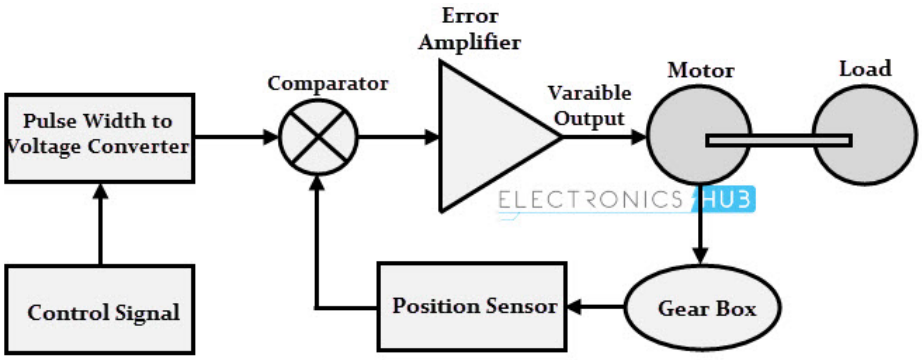
Servo Motors and how to use them

Servo motor is a special type of motor which is automatically operated up to a certain limit for a given command with the help of error-sensing feedback to correct the performance.

The safest bet for positioning tasks in a controller driven circuit is a stepper motor, but in many cases Servo motors, with its easy to use package, work just fine.

How they work:

This is a basic Schematic of the various components involved in the working of a Servo:



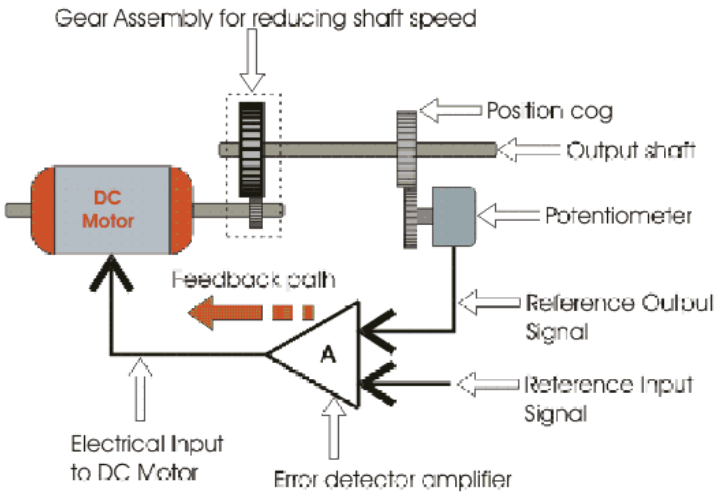
A servo system mainly consists of three basic components - a controlled device, an output sensor, a feedback system. This is an automatic closed loop control system. Here instead of controlling a device by applying the variable input signal, the device is controlled by a feedback signal generated by comparing output signal and reference input signal.

When reference input signal or command signal is applied to the system, it is compared with output reference signal of the system produced by output sensor, and a third signal is produced by the feedback system. This third signal acts as an input signal of controlled device. This input signal to the device presents as long as there is a logical difference between reference input signal and the output signal of the system. After the device achieves its desired output, there will be no longer the logical difference between reference input signal and reference output signal of the system. Then, the third signal produced by comparing theses above said signals will not remain enough to operate the device further and to produce a further output of the system until the next reference input

signal or command signal is applied to the system. Hence, the primary task of a servomechanism is to maintain the output of a system at the desired value in the presence of disturbances.

Servo Motor Control

let us consider an example of servomotor that we have given a signal to rotate by an angle of 45 degrees and then stop and wait for further instruction. The shaft of the DC motor is coupled with another shaft called output shaft, with the help of gear assembly. This gear assembly is used to step down the high rpm of the motor's shaft to low rpm at the output shaft of the servo system.



As the angle of rotation of the shaft increases from 0deg to 45deg the voltage from potentiometer increases. At 45^{o} this voltage reaches to a value which is equal to the given input command voltage to the system. As at this position of the shaft, there is no difference between the signal voltage coming from the potentiometer and reference input voltage (command signal) to the system, the output voltage of the amplifier becomes zero.

As per the picture given above the output electrical voltage signal of the amplifier, acts as input voltage of the DC motor. Hence, the motor will stop rotating after the shaft rotates by 45^{o}. The motor will be at this rest position until another command is given to the system for further movement of the shaft in the desired direction. From this example we can understand the most basic **servo motor theory** and how **servo motor control** is achieved.

Although in practical servo motor control system, instead of using simple potentiometer we use digital or analog position sensor encoder.

Using a Servo Motor with Arduino Uno Microcontroller

Servo.h Library:

This library allows an Arduino board to control RC (hobby) servo motors. 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.**

Circuit

Servo motors have three wires: power, ground, and signal. The power wire is typically red, and should be connected to the 5V pin on the Arduino board. The ground wire is typically black or brown and should be connected to a ground pin on the Arduino board. The signal pin is typically yellow, orange or white and should be connected to a digital pin on the Arduino board. Note that servos draw considerable power, so if you need to drive more than one or two, you'll probably need to power them from a separate supply (i.e. not the +5V pin on your Arduino). Be sure to connect the grounds of the Arduino and external power supply together.

Examples

- Knob: Control the position of a servo with a potentiometer.
- Sweep: Sweep the shaft of a servo motor back and forth.

Functions

1) attach()

Description

Attach the Servo variable to a pin. Note that in Arduino 0016 and earlier, the Servo library supports only 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)

2) write()

Description

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

3) writeMicroseconds()

Description

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 analogous manner to the write function.

Syntax

servo.writeMicroseconds(uS)

Parameters

servo: a variable of type Servo

uS: the value of the parameter in microseconds (*int*)

4) read()

Description

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

Syntax

servo.read()

Parameters

servo: a variable of type Servo

Returns

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

5) attached()

Description

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.

6) detach()

Description

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().

Syntax

servo.detach()

Parameters

servo: a variable of type Servo

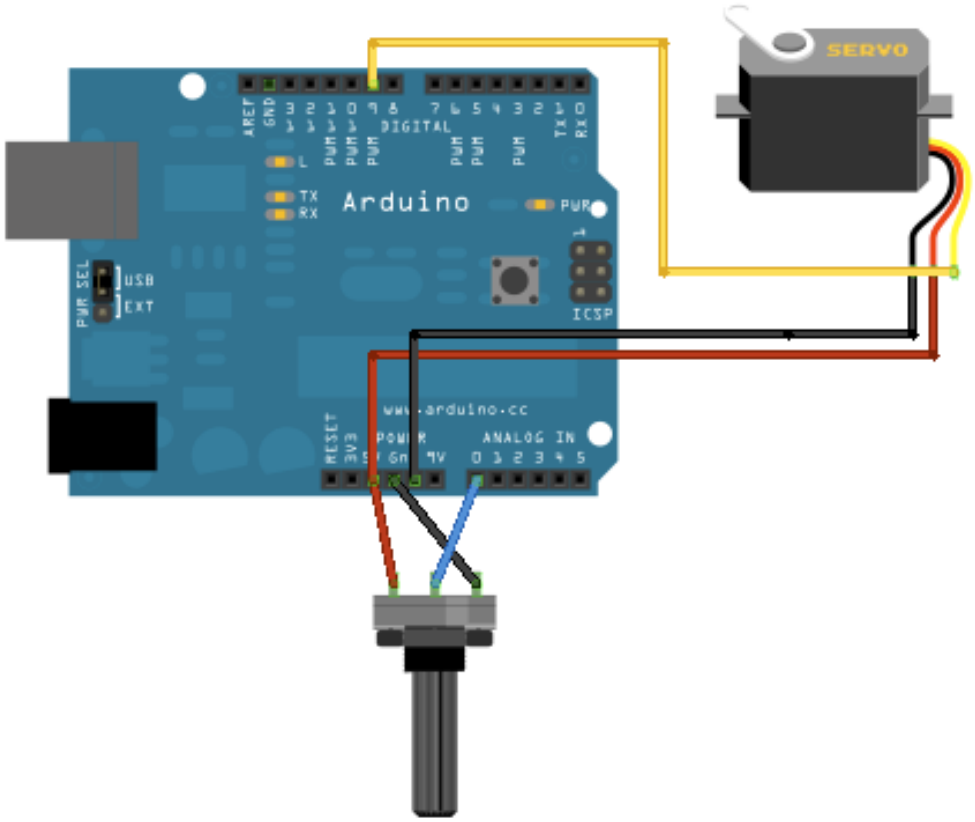
Hardware Required

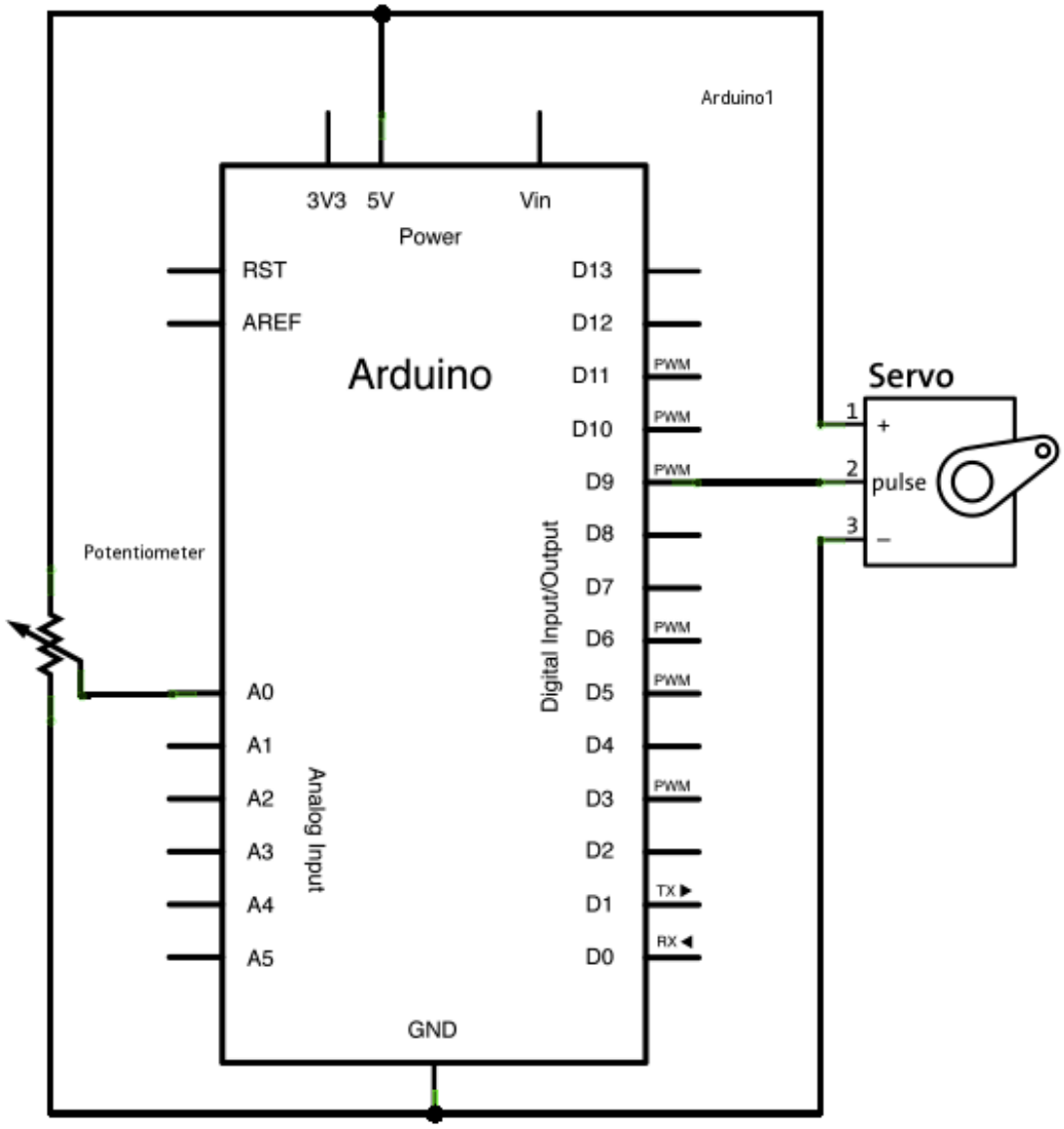
- Arduino or Genuino Board
- Servo Motor
- 10k ohm potentiometer
- hook-up wires

Circuit

Servo motors have three wires: power, ground, and signal. The power wire is typically red, and should be connected to the 5V pin on the Arduino or Genuino board. The ground wire is typically black or brown and should be connected to a ground pin on the board. The signal pin is typically yellow or orange and should be connected to pin 9 on the board.

The potentiometer should be wired so that its two outer pins are connected to power (+5V) and ground, and its middle pin is connected to analog input 0 on the board.





CODE

```
#include <Servo.h>

Servo myservo;  // create servo object to control a servo

int potpin = 0;  // analog pin used to connect the potentiometer
int val;    // variable to read the value from the analog pin
int pos = 0;    // variable to store the servo position
void setup() {
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
}
```

```
void loop() {
  val = analogRead(potpin);            // reads the value of the potentiometer (value between 0 and
1023)
  val = map(val, 0, 1023, 0, 180);     // scale it to use it with the servo (value between 0 and 180)
  myservo.write(val);                  // sets the servo position according to the scaled value
  delay(15);                           // waits for the servo to get there

//performing a sweep

for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
  for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
}
```