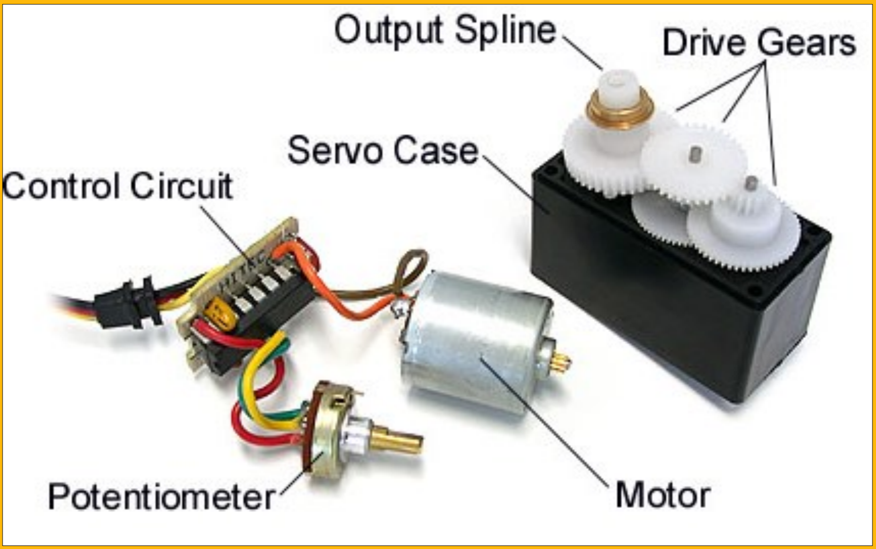


Servo Controller

<u>Electronics 123</u>

Located at :
102 East Park Ave
Columbiana, OH 4440

Contact us on :
Tel: 330-482-9944
Fax: 330-266-7307
E-mail: support@electronics123

Visit our website :
www.electronics123.co

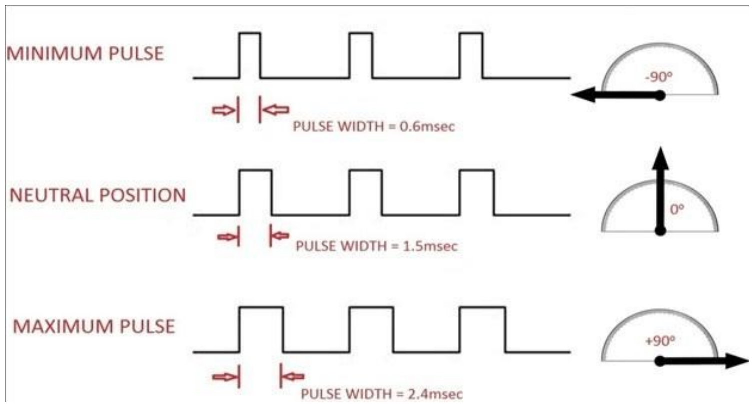
What is a Servomotor

To fully understand how the servo works, you need to take a look under the hood. Inside there is a pretty simple set-up, a small DC motor, potentiometer, and a control circuit. The motor is attached by gears to the control wheel. As the motor rotates, the potentiometer's resistance changes, so the control circuit can precisely regulate how much movement there is and in which direction.

When the shaft of the motor is at the desired position, power supplied to the motor is stopped. If not, the motor is turned in the appropriate direction. The desired position is sent via electrical pulses through the signal wire. The motor's speed is proportional to the difference between its actual position and desired position. So if the motor is near the desired position, it will turn slowly, otherwise it will turn fast. This is called **proportional control.** This means the motor will only run as hard as necessary to accomplish the task at hand, a very efficient little guy.

How is the servo controlled

Servos are controlled by sending an electrical pulse of variable width, or **pulse width modulation** (PWM), through the control wire. There is a minimum pulse, a maximum pulse, and a repetition rate. A servo motor can usually only turn 90 degrees in either direction for a total of 180 degree movement. The motor's neutral position is defined as the position where the servo has the same amount of potential rotation in the both the clockwise or counter-clockwise direction. The PWM sent to the motor determines position of the shaft, and based on the duration of the pulse sent via the control wire, the rotor will turn to the desired position. The servo motor expects to see a pulse every 20 milliseconds (ms) and the length of the pulse will determine how far the motor turns. For example, a 1.5ms pulse will make the motor turn to the 90-degree position. Shorter than 1.5ms moves it to 0 degrees, and any longer than 1.5ms will turn the servo to 180 degrees, as diagramed below



Servo Motor Applications

Servos are used in radio-controlled airplanes to position control surfaces like elevators, rudders, walking a robot, or operating grippers. Servo motors are small, have built-in control circuitry and have good power for their size.

In food services and pharmaceuticals, the tools are designed to be used in harsher environments, where the potential for corrosion is high due to being washed at high pressures and temperatures repeatedly to maintain strict hygiene standards. Servos are also used in **in-line manufacturing**, where high repetition is needed yet precise work is necessary.

Of course, you don't have to know how a servo works to use one, but as with most electronics, the more you understand, the more doors open for expanded projects and projects' capabilities. Whether you're a hobbyist building robots, an engineer designing industrial systems, or just constantly curious, where will servo motors take you?

In workshop

Last weekend we briefly covered the working of a servo motor, we explained the basic principals of the controller circuit, we manufactured our own PCB’s and we successfully assembled and tested each circuit. Now I will fill in the missing links and go in depth on the working of this circuit and the motor driver built into the servo casing.

The servo Brain

This is the circuit inside the servo casing, this circuit is tasked with decoding the pulse width and comparing the on period to a preset value read from the internal potentiometer. This can be done with a microcontroller or a rather complex digital and analog network of IC’s.

The working principle behind this concept is by comparison, the complexity and size of the circuit is determined by the method used to do the decoding.

If you do the decoding with a microcontroller you setup the microcontroller in your hardware as a pulse counter, internally a cheep microcontroller typically has 2 or 3 timers. Usually two 8 bit and one 16bit timer. You can use any of these timers as long as your timer has a counter function internally that is latchable to your external setup. You will typically setup your code to have a reference timer running in the back ground. Lets say for arguments sake timer 1 is latched to your signal, and timer 0 is your reference timer running internally, timer 0 will have to have an overflow period longer than your signal feed rate, or you will end up reading no pulse hence making your servo change position continuously. Timer 0 will trigger an interrupt on overflow (reaching max count value) and then you can tell the microcontroller to read the value stored in timer1's register.

Timer 1 can be setup to start counting if it gets a positive going edge, (from low to high), and to stop counting if the edge goes negative again. This value read by timer 1 will be the on period of the signal. By mathematically manipulating the value in your code you can get a value ranging from 0 to what ever you want. But since this timer is 16 bit in most situations and to get the best accuracy you will work with the largest number possible.

So lets say for arguments sake an on period of 0.6ms gives you a value of 600, and an on period of 2.4ms gives you a value of 2400. this is a rather nice range to work with.

Next you will read your potentiometer value with the microcontrollers internal ADC. This will give you a value of say 0 for –90 deg, and a value of 1023 for +90deg. By comparing the values of the timer and the ADC with the correct mathematical expression you can determine the direction of rotation as well as the position of the servo with pin point accuracy.

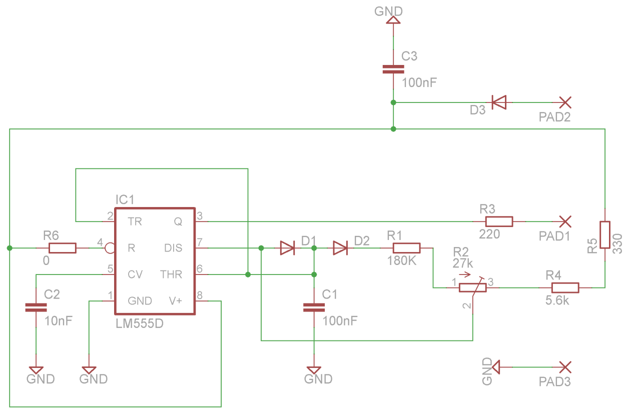
Interfacing to the real world

"The real world", your application, here is some real hidden factors to consider. Microcontrollers can not do more than one thing at a time. They can not multitask. Yes, sure that might not be a problem at first then you play with the servo alone, but lets say you add a led to your circuit and you want to make it change its flashing speed according to the position of the servo. The refresh rate of the servo is too fast for you to register any change on the led's flashing pattern if you setup your code to run dependent on the timer setup of the servo. So you will need to write an extra piece of code for your led. And now for the problem, if you delay the microcontroller by say 1 sec, or for any period longer than your servo refresh rate your interrupts will fire and your code will cycle again and you will end up with one of two things happening, having not changed the state of the led, or the servo will only change position after 1 sec has passed.

To get rid of these unnecessary complex coding methods and timer dependencies we can make use of a bit of analog circuitry that will perform the fixed time dependent functions and this will free up the microcontroller to have a less time based code setup.

This was my aim with the circuit we discussed in the workshop. By replacing the potentiometer with a digital potentiometer that has volatile memory you only need to write a value to the digital pot and you are free to do what ever you want with the microcontroller until you wish to change the position of the servo again, the analog circuitry takes care of keeping the servo in position, not your time dependent PWM signal from the microcontroller.

The circuit



Note that here the duty cycle is less than 50% so normal A-stable operation will not work.

The design equation is:

T = R2 * C1

C1 = 100nF

R2 = ?

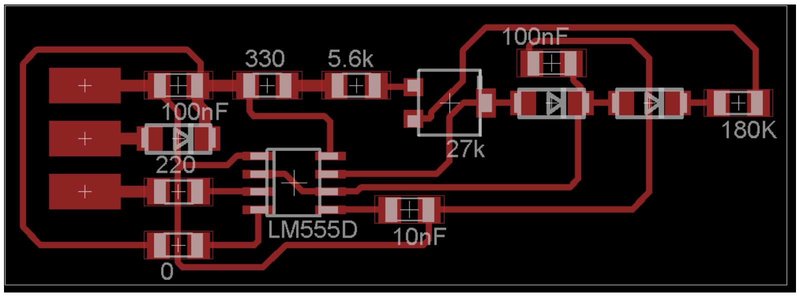
T1 = 0.6ms

T2 = 2.4ms

Rmin = 6K

Rmax = 24K

Hence I have a fixed resistance value in series with the pot, these values will give us the time ranges we are looking for. The diode configuration sets the voltage difference between pin 7 and 6 causing a duty cycle near 10%.



Parts List

LM555D	AE125
10nF	AD962
100nF	AD966
180K	AD790
5.6K	AD772
330E	AD756
220E	AD754
ASK FOR DIODES AND TRIM POT FOR THIS CIRCUIT.	
DIODE 1A 1000V STANDARD RECT	
POT 50K SMD SINGLE TURN	