Interfacing a Servo to the OOPic

Purpose

- To introduce the servo as a mechatronic actuator
- To discover how to control the movement of a servo using the OOPic
- To practice interfacing power electronics and switches

Components

Qty.	Item
1	Oricom Technologies OOBOT 40-II microcontroller and serial port cable
4	470 Ω resistors
2	tact switches
1	green LED
1	red LED
1	jumper wire with female crimp-on socket
1	jumper block
1	Futaba FP-S148 servo

Introduction

In this lab you will explore the workings of a radio control (R/C) servo and how the OOPic can be used to control one.

An R/C servo consists of a dc motor, gear train, potentiometer, and some control circuitry all mounted compactly in a case. R/C servos are commonly used in radio-controlled cars, airplanes, and boats to provide limited rotational motion to steer, move control surfaces, etc. R/C servos are attractive for educational use in mechatronics, because they are relatively inexpensive (about $12-$20), they can put out about 42 oz/in of torque, they can easily be modified to produce continuous shaft rotation at relatively slow speeds, and they can easily be controlled by a microcontroller. There are three wires, white, red, and black on the servo leading from a 3-pin female connector to the case. These carry the control signal, power, and ground return respectively.

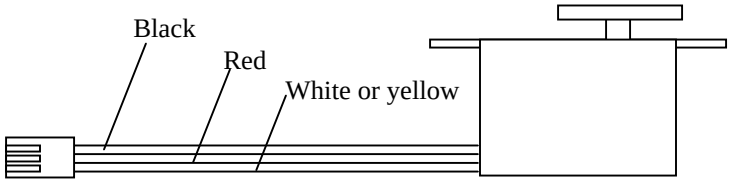


Figure 1 R/C Servo. The R/C servo uses three wires: white carries the control signal, red carries power (usually 4.8 V to 6 V), black is ground.

Figure 2 shows how an R/C servo is made to rotate. The control circuitry inside the servo must receive a stream of pulses whose widths may vary between about 1 ms and 2 ms, and these pulses must occur at intervals of about 10 to 20 ms. A potentiometer coupled to the rotation of the output shaft produces a voltage corresponding to the angle of the shaft. The control circuitry compares the “average” (i.e., low pass filtered) voltage of the control signal

with the voltage from the potentiometer, and the shaft rotates until the two voltages are the same.

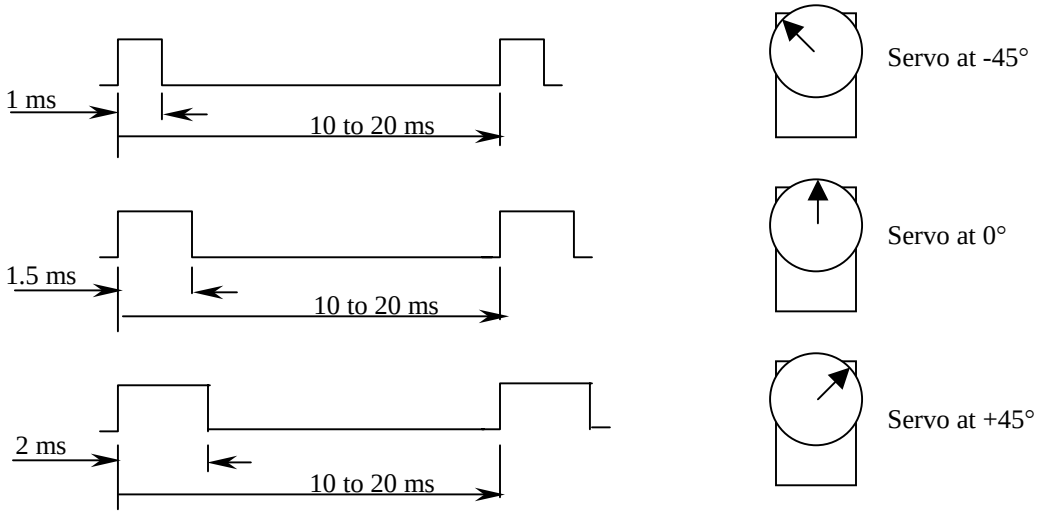


Figure 2 R/C Servo Operation. The R/C servo needs to see a pulse-width modulated control signal in order to position the output shaft. Pulse-widths vary between approximately 1ms - 2 ms, and have a period of 10 ms - 20 ms

Procedure

Servo Control Circuit

Figure 3 shows the circuit you will use in this lab.

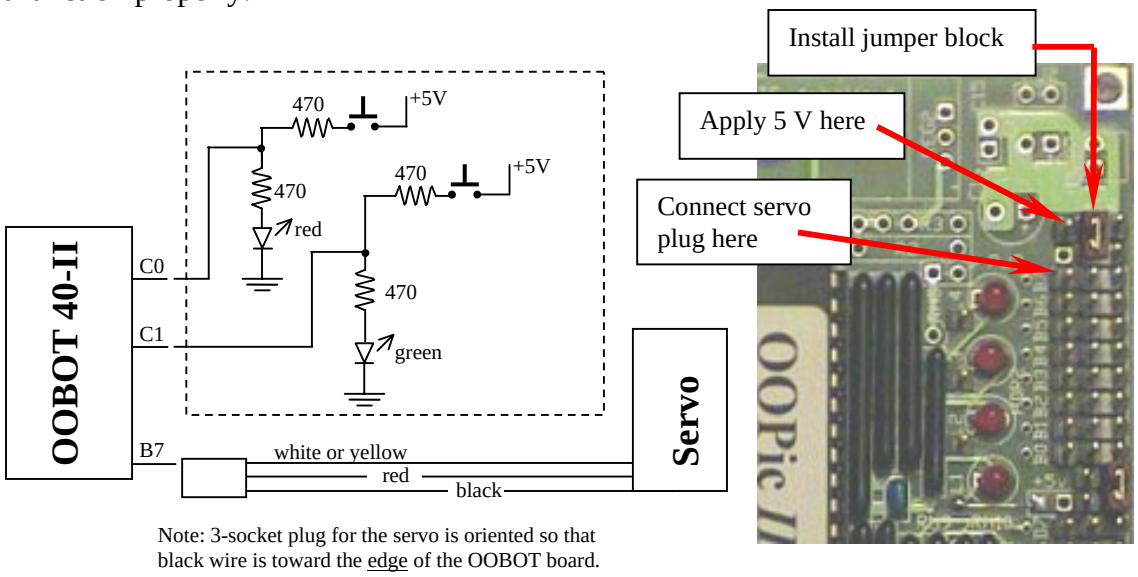
1. Build and test the section of the circuit shown in the dashed rectangle first. **Do not** connect the servo to the OOPic yet! Don't forget to supply power and ground to the microcontroller board. Be careful when you insert the tact switches into the solderless breadboard that you get the leads oriented properly. Make sure that each pair of legs that are tied together internally plug into the ***same*** row of 5 holes on the breadboard. Look for the dot on the underside of the switch. The legs on the side closest to the dot are tied together internally. Test this sub-circuit by entering and running the following program:

```
// Switch Control Test Program
// Put your name here
// Put the date here
oButton SW1 = new oButton;              // Define switch 1 (SW1) as an oButton object
oButton SW2 = new oButton;              // Define switch 2 (SW2) as an oButton object
sub void main (void)
{
  // Set up SW1 parameters
  SW1.Mode = 0;         // Mode=0 and Invert=0 causes SW1 to act like a push button
  SW1.InvertIn = 0;     // i.e., pressed=on, released=off
  SW1.Style = 0;        // Style=0 means LED will be steady when on, not blinking
  SW1.IOLine = 16;      // SW1 connected to IO line 16 (pin C0)
  SW1.Option = 0;       // Causes state of LED on line 16 to be controlled by Value property
  // Set up SW2 parameters
```

```
  SW2.Mode = 0;       // Mode=0 and Invert=0 causes SW2 to act like a push button
  SW2.InvertIn = 0;   // i.e., pressed=on, released=off
  SW2.Style = 0;      // Style=0 means LED will be steady when on, not blinking
  SW2.IOLine = 17;    // SW2 connected to IO line 17 (pin C1)
  SW2.Option = 0;     // Causes state of LED on line 17 to be controlled by Value property
}
```

Your program should allow you turn on the red or green LED's when the corresponding switch is pressed. Save this program to your floppy disk.

Explore the effects of changing the Mode, InvertIn, and Style parameters. Note: the OOBOT board has a 330-ohm resistor in series with the C0 and C1 pins, and also each of the pins has a 10 kohm resistor to ground. This arrangement of resistors will cause the InvertIn=1 option to not function properly.



Note: 3-socket plug for the servo is oriented so that black wire is toward the edge of the OOBOT board.

Figure 3 Servo Control Circuit. +5 Volts from the fixed power supply is applied to the board to power the servo as shown above. A jumper block needs to be installed to bring power from the +5 V supply to the center pins of HDR 2.

2. After you have proven that the switches and LED's have been wired correctly, add the servo. Connect the servo to the OOBOT board at pin B7 with the servo's 3-socket plug. Orient the plug so that the black wire (servo ground) is toward the edge of the board. Use a jumper from the 5 V post on the fixed power supply to the servo power pin shown in Figure 3. Make sure that you install a jumper block as shown above. This will supply 5 V to the center pins in HDR 2.

3. Enter and run the following program:

```
// Servo test program
// Put your name here
// Put the date here
oServo Servo = new oServo;    // Declare Servo as a new oServo object
sub void main (void)
{
  Servo.IOLine = 15;          // Use IO line 15 to drive Servo on pin B7 on OOBOB board
  Servo.Center = 26;          // Set servo center pulse rate
  Servo.Operate = 1;          // Activate Servo object
```

```
    while(1)
    {
        Servo = 10;                 // Rotate to the left
        OOPic.Delay = 100;          // Wait for 1 second
        Servo = 50;                 // Rotate to the right
        OOPic.Delay = 100;          // Wait for 1 second
    }
}
```

What does the servo do? Look at the OOPic manual for the description on the oServo object. Explore changing the Value property of your oServo object between the range of about 2 to about 60. **Caution**, going beyond these values could result in the servo gears getting stripped. We don't want you to have to buy another servo for the lab!

Add some code to the previous program to turn on one of the LED's when the servo rotates in one direction and to turn on the other when the servo rotates in the opposite direction.

4. Write a program that will cause the servo to rotate to one extreme when one of the switches is pressed and will rotate to the other extreme when the other button is pressed. If both switches are pressed, then the servo should rotate to its central position. Save your program to your floppy, and include it in your lab report.

End Notes

It is relatively easy to modify a servo so that it can rotate continuously. An excellent step-by-step guide for doing so can be found at:

http://www.seattlerobotics.org/guide/servohack.html

If you use the Futaba FP-S148, it is especially easy. You don't even need to replace the potentiometer as noted in the link above. You can manually position the pot shaft to its center position.

Once you've hacked the servo, by using the oServo object, you then have a low-cost, relatively powerful variable speed dc motor drive system.

Using a servo can be a low-cost and efficient approach for giving mobility to your robot. Tower Hobbies (http://www.towerhobbies.com/) has the best prices around for Futaba servos (3 for $39.47).