ME 461
RC Servo Motor Control Using Arduino

Objective:
The objective of this lab is to get something moving by using the Arduino boards for the first time.

Pre-Lab:
Signals in the form of square waves are commonly used in mechatronic systems for several purposes. One application of such periodic signals is in motor actuation. In this lab we will focus on RC Servo motors (will be referred to as *RC Servo* in the remainder of this document). A RC Servo is not simply a DC motor. Indeed, it is a combination of a DC motor, a gear train, a position sensor (commonly a potentiometer is used) and control circuitry as illustrated in figure 1.

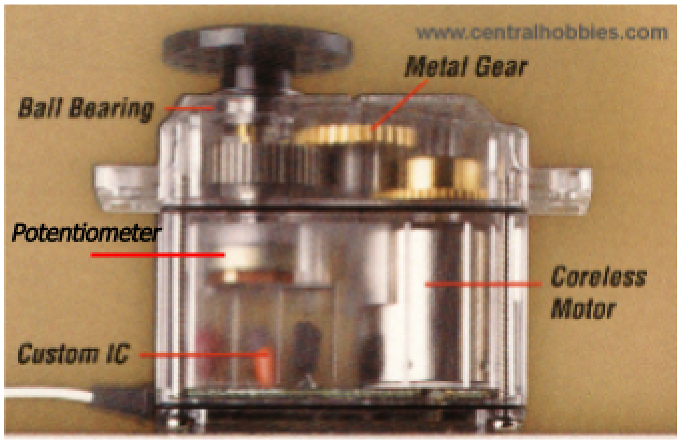

Figure 1. Components of a RC Servo [www.centralhobbies.com]

RC servos are small actuators widely used in remotely controlled (RC) toys (i.e. cars, planes, helicopters and boats). These are compact actuation mechanisms that provide an easy way of achieving position control in small scale applications where precision is not essential. Each RC servo is composed of a small DC motor attached to a gear train, a potentiometer and a simple controller. The gear reduction provides a reasonable amount of torque compared to their size. As the name implies, the RC servo has a sensor that tells the current position of the shaft to the controller that is built into this system, namely the potentiometer. The built-in controller compares the current position (measured from the pot) with the target (sent from Arduino) and creates a control signal that will move the output shaft to a position that decreases this difference. By now you should have started wondering how we can set the reference position (in other words, how to tell the motor where to go). The reference position is provided by using a PWM (pulse width modulated) signal. All of the RC servos that you will find in our lab will have a range of 0-180º. The built-in controller of RC servos controls the rotor position within this range. Generally, one end of this range is referred to as –90º and the other as +90º (even though there is no harm if you prefer 0-180º). Figure 2 illustrates the ideal working conditions of RC servos. As shown, RC servos can be rotated to specific positions by sending PWM signals, the ON (high) time of which determines the target position of the motor. In order to reach and maintain a position this PWM signal has to be **continuously sent** to the motor via its signal pin. Ideally the high time of these signals will change between 1 to 2 ms. Low time of these signals can be anywhere between 10 to 25 ms. Use of 10 ms for low time is recommended. Even though ideal high times should be within [1-2]ms range, in practice this range will slightly change and in this lab you will determine these limits empirically.

Figure 2 illustrates the type of PWM that you have to generate in order to control the position of the RC Servo. Even though only three discrete positions are illustrated; the servos can be controlled continuously between –90 (P_{min}) and 90(P_{max}) degrees.

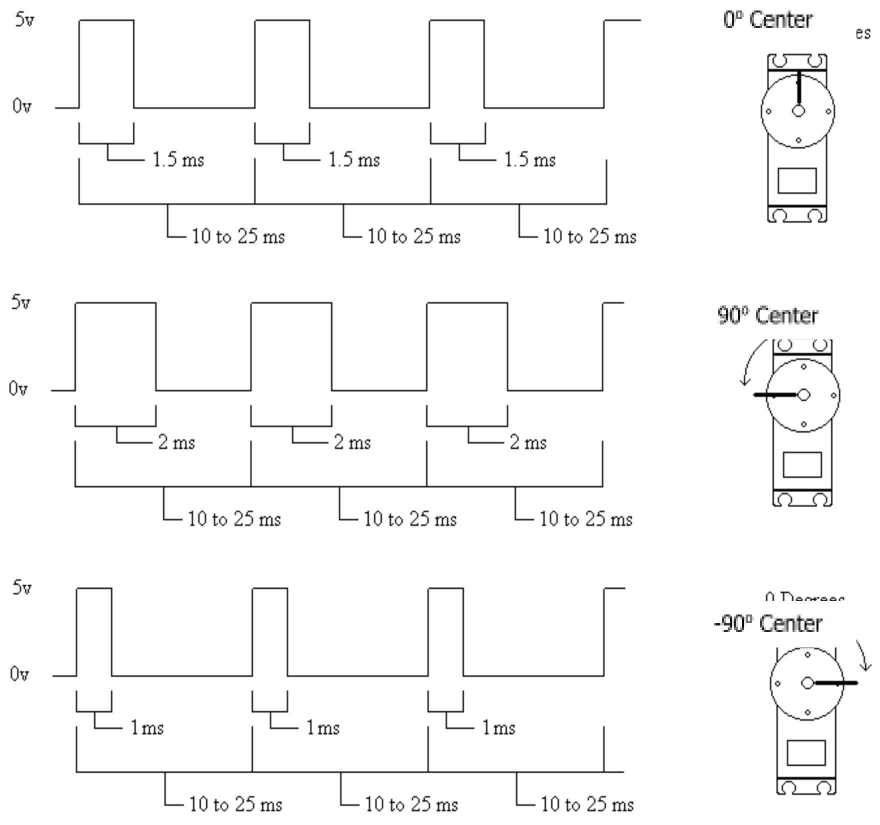

Figure 2. PWM Signal used in controlling a RC servo

Lab:
PART I:
The lab will be completed in a couple of steps.

1. Properly interface the RC Servo to the Arduino.
2. Write a RC servo control program on Arduino.
3. Interface a potentiometer (POT) to Arduino.
4. Implement a simple communication protocol with Arduino
5. You are not allowed to use Arduino Servo library or PWM commands, just use microsecond delays and digital pin outs.

1. Figure 2 illustrates the theoretical operation of RC servos. However, in practice, the actual values for RC servo limits might deviate from these theoretical ones. Connect the servo pins as follows: middle pin is 5V (mostly supply wire is red), generally the darker colors such as black or brown are used to indicate ground, and the remaining pin is for signaling desired position. By the way, use a separate 5V source to feed the servo and make sure that both sources have the same ground. You can use a separate 7805 for this purpose. The proper wiring is shown in Figure 3. You can provide the signal from any digital output pin. (If you are using a small size servo, Arduino 5V output can be used instead of 7805)

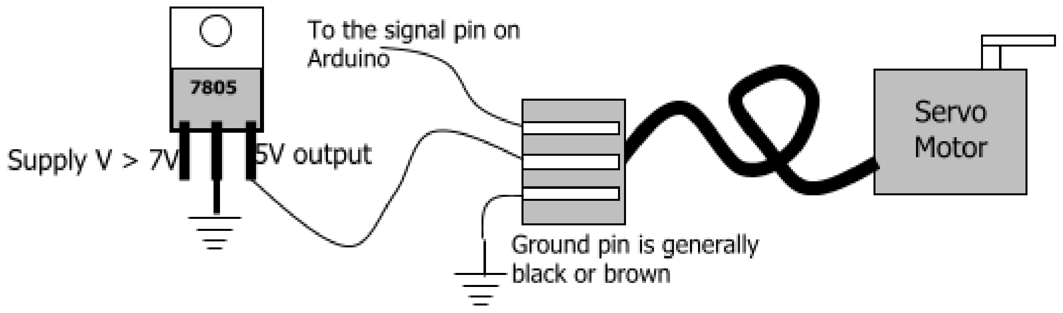
7805
To the signal pin on Arduino
Supply V > 7V
5V output
Servo Motor
Ground pin is generally black or brown

Figure 3. Interfacing the RC servo to Arduino

2. Now, write a program in Arduino which receives a control position from the serial port and converts this into a properly timed signal and **continuously** applies this signal to the RC Servo. The details of this messaging protocol are up to you, but if you send *an invalid value*, Arduino should stop sending PWM signals to the motor and hence the motor should be released. You are **not to use any servo library**, but write every step of this timed signal yourself.

3. At this step, you will try to figure out the actual timing limits for the specific servo you have. This will be an iterative process. You need to determine two different high time values P_{min} and P_{max} respectively. These times correspond to the two extreme positions of the RC servo you have. In order to find these values use the simple protocol you have implemented in the second stage. Systematically send position commands through the serial port, observe the motor response and determine both limit values. If you send a position command which is beyond the physical limits of the motor, it will sound weird (i.e. humming or buzzing) and potentially will start shaking and shivering in pain. In such a case rapidly send 0 (or an invalid value) to release the motor, and try another position command that is closer to the center.

Repeat these steps, until you find the limiting values for the servo you have, and record these values in milliseconds below. Also check the average of these values to see if it takes the motor shaft to the center position.

Pmin = ________________________ Pmax = ________________________

You are expected to present these values during your lab presentation.

4. Next step is to control the RC servo using your POT. Now that you have determined P_{min} and P_{max}, write a program that maps the POT limits to that of the RC servo. In other words, your program should maps POT output to the RC servo position, and you turn the POT from one end to the other, RC servo should turn in sync with the POT.

5. Implement a Python Program that acts like the screenshot illustrated in figure 4. You can use one scrollbar or a slider (the range of which has to be [0-180], to do this, play with the min,max properties of this control), one label, 5 radio buttons and arrange the form as shown. You will need to catch any change in scroll bar / slider state by monitoring proper events and: 1) update the label on top of the scroll bar 2) turn the servo motor to indicated position. The radio buttons on the other hand, provide preset positions as indicated. By either using the slider or the radio buttons the user should be able to control the RC servo position. If you press the 'RELEASE MOTOR' button, the motor should become free, i.e. you should be able to turn it by hand.

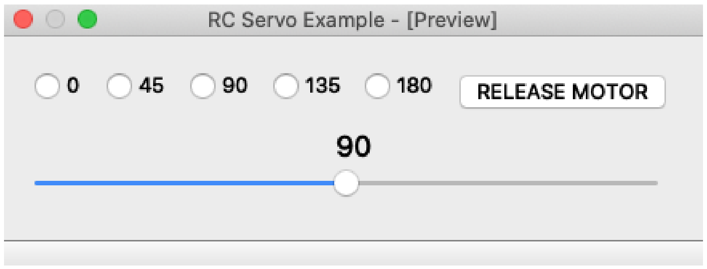

Figure 4. RC servo control computer user interface at runtime

PART II:
This part is very similar to what you did in Part I, but this time rather than Arduino you are expected to control the servo via your ESP8266 board. This part is open to creative improvisation and you are completely on your own! You are the sole designer of everything now and at the end I expect you to show me that you can control your RC Servo motor, similar to what you did in Part I. You are to explain briefly what you did and how you did it during lab presentations.

and don't forget to have fun…