Exercise 5 – Control a Servo Motor

Introduction

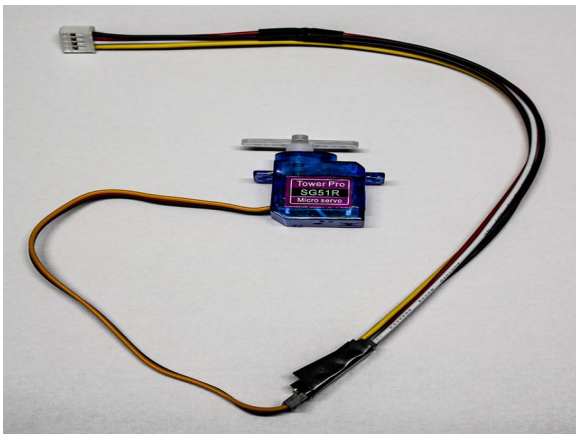
The document **`Packages.pdf`** in **`ada-remoteio-tutorial/`** contains annotated package specifications for the packages that you will be using during this tutorial session. For this exercise you will need to reference the packages **`PWM`**, **`PWM.RemoteIO`**, **`Servo`**, **`Servo.PWM_Template`**, and **`Servo.PWM`**.

Servo motors are controlled by a fixed frequency, variable width pulse train. The width of the pulses sets the servo motor position. The servo motor includes an internal feedback mechanism that will seek and hold the position that matches the control pulse width. As long as the servo motor is not overloaded mechanically, its position will always match the control pulse width and therefore position feedback to the control point is not necessary.

Conventional hobby RC servo motors require a 50 Hz pulse train with pulse width constrained to the range of 1.0 to 2.0 milliseconds. A pulse width of 1.0 milliseconds selects the “minimum” position and 2.0 milliseconds selects the “maximum” position. A pulse width of 1.5 milliseconds selects the center or “neutral” position.

We will be using the PWM (Pulse Width Modulated) output hardware of the LPC1114 I/O Processor to generate the pulse train for controlling the servo motor.

Hardware Setup



Plug the servo motor assembly from the tutorial hardware kit into Grove socket **`J2`** (Raspberry Pi Zero) or **`J4`** (BeagleBone Green). This attaches the servo motor control line (orange wire) to Remote I/O PWM channel 1.

The servos provided for this exercise are rated for a 4.8V power supply, but we will be running them at 3.3V. We will lose some torque because of the reduced power supply voltage.

Instructions

1. The following simple program contains all of the “boilerplate” code for controlling a servo motor using the Remote I/O Library:

```
with PWM.RemoteIO;
with RemoteIO.Client.hidapi;
with RemoteIO.LPC1114;
with Servo.PWM;

use type Servo.Position;

procedure Test_Servo is

   remdev : RemoteIO.Client.Device;
   PWM1   : PWM.Output;
   Servo1 : Servo.Output;

begin
   remdev := RemoteIO.Client.hidapi.Create;
   PWM1   := PWM.RemoteIO.Create(remdev, RemoteIO.LPC1114.PWM1, 50);
   Servo1 := Servo.PWM.Create(PWM1, Servo.NeutralPosition);
end Test_Servo;
```

2. Create a new program file **test_servo.adb** in **gps** and paste the above code into it. Build and run **test_servo** as is, first.You will observe that after running **test_servo**, the servo motor will move to the center or neutral position.

The servo motors provided for this tutorial all have 90 degree rotation. The minimum position will be 45 degrees from neutral in one direction and the maximum position will be 45 degrees from neutral in the other direction. (Whether minimum is clockwise or counterclockwise from the neutral position will depend on the particular servo motor. Different manufacturers have picked one direction or the other. A high end servo motor might even have a switch for selecting the direction of rotation.)

3. Add the following code fragment just before the final **end** statement, and then rebuild and run **test_servo**:

```
   delay 1.0;
   Servo1.put(Servo.MinimumPosition);
   delay 1.0;
   Servo1.put(Servo.MaximumPosition);
   delay 1.0;
   Servo1.put(Servo.NeutralPosition);
```

This will move the servo motor from neutral to minimum to maximum and back to neutral.

4. After **test_servo** terminates, the servo motor will be left in the neutral position. Try moving the white actuator with your fingers. Even though the servo motor is operating at reduced voltage and therefore reduced torque, it will hold the neutral position surprisingly strongly.

Going Further

1. Add a loop to **`test_servo`** that will move the servo motor back and forth.

2. Add code to control the position of the servo motor using the potentiometer. Plug the potentiometer assembly into **`J5`** (BeagleBone Green) or **`J3`** (Raspberry Pi Zero); it will be connected to **`RemoteIO.LPC1114.AIN3`**.

3. What would you need to change to reverse the direction of rotation?

4. Add code to implement a neutral override. Plug the button assembly into **`J6`** (BeagleBone Green) or **`J4`** (Raspberry Pi Zero); it will be connected to **`RemoteIO.LPC1114.GPIO4`**. Your code should move the servo motor to the neutral position while the button is pressed.