WEEK 2
Experiment # 5

Maneuvers – Making Turns

If the same value is added to the center pulse width of one servo and subtracted from the center pulse width of the other, the Boe-Bot will travel in a straight line, either forward or backward. When the right servo gets a *pulsout* period of 500 (1.0 ms) and the left servo gets a *pulsout* period of 1000 (2.0 ms), the Boe-Bot goes forward. When the pulse periods for each servo are swapped, the Boe-Bot goes backward.

If both servos receive 1.0 ms pulses, they turn in the same direction and cause the Boe-Bot to rotate counterclockwise. If many pulses are applied, the Boe-Bot will keep rotating. If 35 or so pulses are applied, the net effect is in the neighborhood of a 90° left turn. The same principles apply if both servos receive 2.0 ms pulses, except that the Boe-Bot will rotate clockwise instead of counterclockwise.

Activity:

Type in the following codes onto the Stamp Editor.

```
‘ Program Listing 2.3, Turning in place.
‘ {$Stamp bs2}                           ‘ Stamp Directive

‘ ------Declarations -----
pulse_count var word                     ‘ Declare a var. for counting

‘ ----- Initialization -----
output 2                                 ‘ Set P2 as output pin
freqout 2, 2000, 3000                    ‘ Signal prog is (re)starting
low 12                                   ‘ Set P12 and P13 to output-low
low 13

‘ ---- Main routine -----
main:                                    ‘ Main routine
   left_turn:                            ‘ Left turn routine
      for pulse_count = 1 to 35          ‘ Sends 35 left rotate pulses
         pulsout 12, 500                 ‘ 1.0 ms pulse to right servo
         pulsout 13, 500                 ‘ 1.0 ms pulse to left servo
         pause 20                        ‘ pause for 20 ms
      next

      pause 500                          ‘ Pause for 0.5 s

   right_turn:                           ‘ Right turn routine
      for pulse_count = 1 to 35          ‘ Sends 35 right rotate pulses
         pulsout 12, 1000                ‘ 1.0 ms pulse to right servo
         pulsout 13, 1000                ‘ 1.0 ms pulse to left servo
         pause 20                        ‘ pause for 20 ms
      next

      pause 500                          ‘ Pause for 0.5 s

   stop                                  ‘ Stop until reset is pressed
```

Program Explanation:

The *left-turn* routine is used to rotate the Boe-Bot about 90° to the left, while the *right-turn* routine is used to rotate the Boe-Bot about 90° to the right. The turning of about 90° can be accomplished by setting the amount of the loop's count into 35. So, if you want to make about 45° of rotation, you can set the amount of the loop's count into 17 or so.
Save the program as 'Prog2_3.bs2' and run it.
Your Boe-Bot should remain still and plays the tone for 2 seconds. Then it rotates a quarter turn to the left and then rotates a quarter turn to the right, to its initial position.

Notes:
The value of 35 might not make the Boe-Bot to rotate exactly 90°. Try to to use other values to get one which will do the most precise quarter turn.

Task:

1. Add the forward and backward routines from Prog2_2.bs2 to Prog2_3.bs2. Make the Boe-Bot to go forward and backward before making the left and right maneuvers. Use the best value on the loop in order to make the Boe-Bot turns 90° to the left and right. Save the program as the same file (Prog2_3.bs2). Run it.
2. Add more capability to the Boe-Bot so that it can turn 180° to the left and to the right. Modify the program so that the Boe-Bot moves with the following directions: forward, 90° right turn, forward, 90° right turn, forward, 180° left turn, forward, 90° left turn, forward. Run it.

Maneuvers – Ramping

Ramping is a way to gradually increase the speed of the servos instead of suddenly making them go to the opposite direction. This technique can increase the life expectancies both of your Boe-Bot’s batteries and servos.

Activity:

Type in the following codes onto the Stamp Editor.

```
’ Program Listing 2.4, Ramping for Start and Stop.
’ {$Stamp bs2}                      ’ Stamp Directive

’ ------Declarations -----
pulse_count var word                ’ Declare a variable for counting
right_width var word                ’ Variable stores right pulse width
left_width var word                 ’ Variable stores left pulse width

’ ----- Initialization -----
output 2                            ’ Set P2 as output pin
freqout 2, 2000, 3000               ’ Signal program is (re)starting
low 12                              ’ Set P12 and P13 to output-low
low 13

’ ---- Main routine -----
main:                               ’ Main routine
   ramp_up_forward:                 ’ Routine ramps into forward motion
       for pulse_count=0 to 250 step 2 ’ For loop counts up, steps of 2
          pulsout 12, 750–pulse_count  ’ Pulse sent = 1.5ms–pulse_count
          pulsout 13, 750+pulse_count  ’ Pulse sent = 1.5ms+pulse_count
          pause 20                  ’ Pause for 20 ms
       next

   forward:                         ’ Forward routine
       for pulse_count = 1 to 100 ’ Loop that sends 100 forward pulses
          pulsout 12, 500           ’ 1.0 ms pulse to right servo
          pulsout 13,1000           ’ 2.0 ms pulse to left servo
          pause 20                  ’ Pause for 20 ms
       next

   ramp_down_forward:             ’ Routine ramps out of forward motion
      for pulse_count=250 to 0 step 2 ’ For loop count down, steps of 2
          pulsout 12, 750–pulse_count ’ Pulse sent is 1.5ms–pulse_count
          pulsout 13, 750+pulse_count ’ Pulse sent is 1.5ms+pulse_count
          pause 20                  ’ Pause for 20 ms
       next

   stop                             ’ Stop until reset is pressed
```

Program Explanation:

If there is no *step* argument used on the *for … next* command, it means that the value of *pulse_count* will be incremented OR decremented by 1. Note that on the *for … next* command on the *ramp_up_forward* and *ramp_down_forward* routine, there is *step 2* argument, which means that value of *pulse_count* variable will incremented OR decremented by 2.

The key to ramping is to modify the pulse period a little each time a pulse is sent to the servo until it reaches the desired value.

Take a look at *ramp_up_forward* routine. The first time through the *for … next* loop, the value of *pulse_count* is 0, so a 750 pulse width is sent to P12 and P13. The second time through, *pulse_count* is incremented to 2, so a 748 pulse width is sent to P12 and a 752 pulse width is sent to P13; and so on. Finally, the value of *pulse_count* would be 250, so a 500 pulse width is sent to P12 and a 1000 pulse width is sent to P13.

Save this program as Prog2_4.bs2. Run it.

Your Boe-Bot should remain still and plays the tone for 2 seconds. Then it gradually speed up moving forward. Then it moves for some time on its maximum speed and then gradually slow down to a complete stop. If your Boe-Bot doesn't stop completely, you may need to adjust the center *pulsout* period other than 750.

Task:

Develop routines that ramp into and back out of the backward routine on the Prog2_2.bs2 and add them to Prog2_4.bs2. Save it as the same file and run it.

Your Boe-Bot should remain still and plays the tone for 2 seconds. Then it gradually speed up moving forward. Then it moves for some time on its maximum speed and then gradually slow down to a complete stop. After that, it should gradually speed up moving backward, moving on its maximum speed for some time and then gradually slow down to a complete stop.