

## 10.0.1 Lab 7 - Motion Control

### 10.0.1.1 - Purpose

To use a setpoint table to generate smooth motion profiles.

### 10.0.1.2 - Background

A simple (e.g., proportional) controller will move faster when further from the setpoint (desired position). This means at the start of motion there is normally a period of fast acceleration, resulting in a 'jerk' and high acceleration. At the end of motion the controller slows smoothly to a stop. In many cases we prefer motion that starts and stops smoothly. This is achieved by adding motion control.

A block diagram for a motion controller is shown in Figure 10.1. In this system a target position is used to generate a set of points and times along a smooth path in a setpoint schedule table. These points are then used as setpoints for the feedback control loop. Motion starts at the beginning of the table with the first setpoint. At the given times, the setpoint is updated to the new value in the table. In this system the controller is always shooting for the next point on a smooth path.

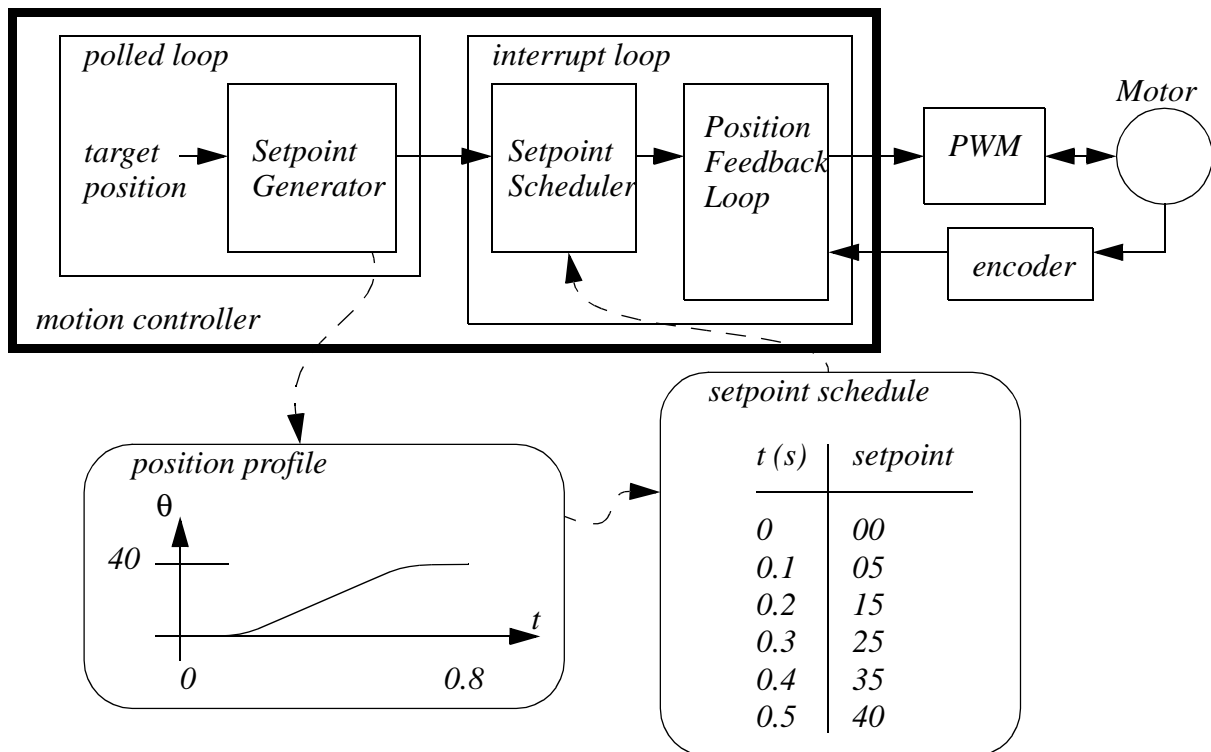


Figure 10.1 Motion generation

### 10.0.1.3 - Theory

There are multiple methods for generating a smooth motion path, this section outlines two of these. The first is shown in Figure 10.2, it controls motion using a parametric function 'p(u)'. The function value varies from 0 to 1 as the parameter 'u' varies from 0 to 1. However, the parameters of the function are selected so that the motion starts and stops with a velocity of zero. In this case the final polynomial equation, (3), is fairly simple. This equation can then be used in equation (1) to generate a smooth motion path between any arbitrary start and end point, with arbitrary start and end times.

$$\theta(t) = \theta_{start} + (\theta_{end} - \theta_{start})p\left(\frac{t - t_{start}}{t_{end} - t_{start}}\right) \quad (1)$$

where,

$\theta_{start}, \theta_{end}$  = start and end positions of motion

$t_{start}, t_{end}$  = start and end times for the motion

$$p(u) = Au^3 + Bu^2 + Cu + D \quad (2)$$

The constraints for the polynomial are,

$$p(0) = 0 \quad p(1) = 1$$

$$\frac{d}{dt}p(0) = 0 \quad \frac{d}{dt}p(1) = 0$$

These can be used to calculate the polynomial coefficients,

$$0 = A0^3 + B0^2 + C0 + D \quad \therefore D = 0$$

$$0 = 3A0^2 + 2B0 + C \quad \therefore C = 0$$

$$0 = 3A1^2 + 2B1 \quad \therefore B = \left(-\frac{3}{2}\right)A$$

$$1 = A1^3 + B1^2 + (0)0 + 0 \quad \therefore A = -2$$

$$\therefore B = 3$$

$$p(u) = -2u^3 + 3u^2 \quad (3)$$

Figure 10.2 Generating smooth motion paths

The example in Figure 10.3 shows the use of a trigonometric function, instead of a polynomial. This function was used to generate the points in the following sample program in Figure 10.4.

where,

$$p(u) = A \sin(Bt + C) + D$$

The coefficients can be calculated using the conditions used previously,

$$\frac{d}{dt}p(0) = AB \cos(B(0) + C) = 0$$

$$\therefore \cos(C) = 0$$

$$\therefore C = -\frac{\pi}{2}$$

$$\frac{d}{dt}p(1) = AB \cos(B(1) + C) = 0$$

$$\therefore \cos(B + C) = 0$$

$$\therefore B + C = \frac{\pi}{2}$$

$$\therefore B = \pi$$

$$p(0) = A \sin\left(B(0) - \frac{\pi}{2}\right) + D = 0$$

$$\therefore A = D$$

$$p(1) = A \sin\left(\pi(1) - \frac{\pi}{2}\right) + D = 1$$

$$\therefore A(1) + A = 1$$

$$\therefore A = \frac{1}{2}$$

The final relationship is,

$$p(u) = \frac{1}{2} \sin\left(\pi t - \frac{\pi}{2}\right) + \frac{1}{2}$$

Figure 10.3 Generating smooth motion paths

The program in Figure 10.4 generates a motion table that can then be used to update setpoints. The function 'table\_init()' must be called once when the program starts to set up global time and table values. When a new target position has been specified the 'table\_generate()' function is called to generate the setpoint table. The 'table\_update()' function is called once every interrupt scan to check the setpoint table, and update the global setpoint variable, 'point\_current' at scheduled times. This function also includes a simple clock to keep track of the system time.

```

#define          TABLE_SIZE      11
int    point_master[TABLE_SIZE] = {0, 24, 95, 206, 345, 500, 655, 794, 905, 976, 1000};
int    point_position[TABLE_SIZE];
int    point_time[TABLE_SIZE];
int    point_start_time;
int    point_index;

int    ticks; /* variables to keep a system clock count */
int    point_current; /* a global variable to track position */

int table_init(){ /* initialize the setpoint table */
    ticks = 0; /* set the clock to zero */
    point_current = 0; /* start the system at zero */
    point_index = TABLE_SIZE; /* mark the table as empty */
}

void table_generate(int start, int end, int duration_sec){
    unsigned i;

    point_time[0] = ticks + 10; /* delay the start slightly */
    point_position[0] = start;

    for(i = 1; i < TABLE_SIZE; i++){
        point_time[i] = point_time[0] +
            (unsigned long)i * duration_sec * 250 / (TABLE_SIZE - 1);
        point_position[i] = start + (long int)(end - start) * point_master[i] / 1000;
    }
    point_index = 0;
}

int table_update(){/* interrupt driven encoder update */
    ticks++; /* update the clock */

    if(point_index < TABLE_SIZE){
        if(point_time[point_index] == ticks){
            point_current = point_position[point_index++];
            outint16(point_current);
            putchar("\n");
        }
    }
    return point_current;
}

```

*Figure 10.4* Subroutines for motion profile generation and use

#### 10.0.1.4 - Prelab

1. Modify the subroutines in the prelab to use 15 motion points, instead of 11.
2. Write a program to use the setpoint generation routines with the position feedback control system developed in previous labs. The user interface should allow the user to increment or decrement a target value using 'i' or 'd'. The user should be able to set the duration using key '0' to '9'. The target value will not

update the motion table until 'g' is input. 's' will stop the motion.

#### **10.0.1.5 - Equipment**

computer with a WinAVR compiler  
ATMega32 board with an L293D Push-pull four channel driver  
1 motor  
1 encoder  
1 multimeter  
external power supply

#### **10.0.1.6 - Experimental**

1. Connect the control system using an L293D to drive the motor, and an encoder as done in previous labs.
2. Test the program developed for the prelab. Determine all necessary parameters, such as the deadband limits.
3. For the proportional and integral gain values use  $K_p=10$ ,  $K_i = 0$ . Supply different setpoints and measure the steady state position and record other observations.
4. Vary the time of the motion and observe the system behaviour.
5. Repeat step 3 for different values of proportional and integral gains.
5. Plot the results on a single graph, be sure to indicate the setpoints.