

## Lesson 5 Multi-channel Servo Control

### 1. Project Purpose

Learn the principle of the servo control and control several servos to rotate.

### 2. Project Principle

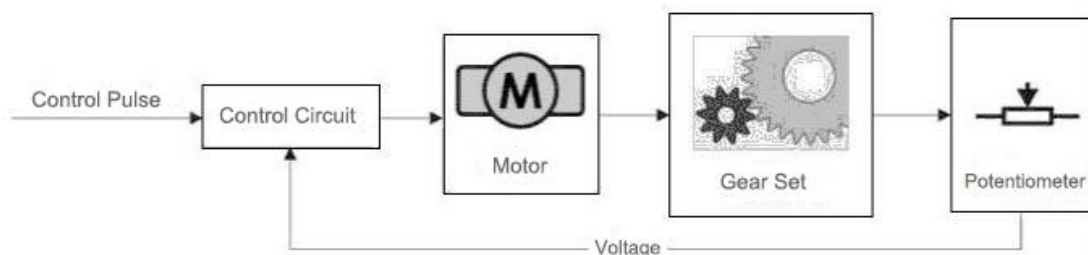
#### 2.1 Servo Internal Structure

Servo consists of several parts namely small DC motor, a set of change gears, a linear feedback potentiometer, and a control circuit.

Of these, the high-speed DC motor provides the raw power for the servo and drives the reduction gear set to produce the high torque output. The greater the gear ratio, the greater the output torque of the servo, which means that it can drive a heavier load (limited by the gear strength), but the lower the output speed (response speed).

#### 2.2 Servo Working Principle

Servo is a typical closed loop feedback system. Its principle can refer to the figure below.

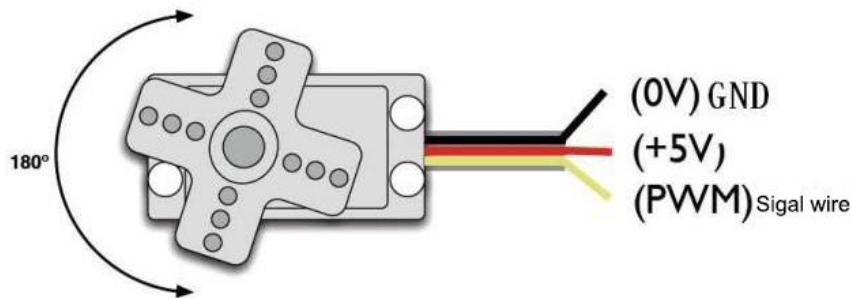


The reduction gear is driven by motors and its output terminal drives a linear potentiometer for positional detection. This potentiometer converts the angle into a proportional voltage feedback to the control circuit. Then the control circuit compares the proportional voltage with the angle corresponding to the input control signal and drives the motor to rotate clockwise or counterclockwise so as to make the potentiometer feedback angle to approach

to the anticipated angle of the control signal, which achieves the accurate the purpose of accurate positioning of the servo motor.

## 2.3 How to control servo

Servo motors have three wires: power, ground, and signal.




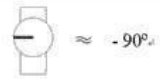

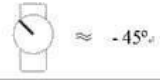




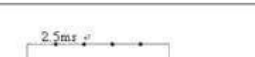

The power and ground wires are used to provide energy required for the internal DC motor and the control circuit. The voltage usually ranges between 5V and 8V, and the power supply should be isolated from the power supply of the processing system as much as possible. (because it will generate noise.)

Since the servo will also pull down the amplifier voltage during heavy load, the ratio of power supply for the whole system must be reasonable.

Input a periodic positive pulse signal. The high level time of this periodic pulse signal is usually between 1ms-2ms, and the low level time should be between 5ms and 20ms. The analog servo needs to maintain a periodic signal all the time to keep the angle of the servo. When the signal is lost, servo will no longer output power.

In this case we are using a digital servo that can maintain the locked angle by sending the correct high level signal once, so the low level time requirement is not strict.

In this section, we are using a digital servo, as long as the correct high-level signal is sent once to maintain the locked angle, so the requirement for low-level time is not strict.

输入正脉冲宽度 (周期为 20ms)	舵机输出臂位置
	 $\approx -90^\circ$
	 $\approx -45^\circ$
	 $\approx 0^\circ$
	 $\approx 45^\circ$
	 $\approx 90^\circ$

### 3. Program Analyst

- 1) Initialize servo connection. Then connect the 6 servos to different pins and set the angle limit range at the same time.

```

3 void setup()
4 {
5     pinMode(LED, OUTPUT);
6     pinMode(BUZZER, OUTPUT);
7     pinMode(KEY, INPUT_PULLUP);
8
9     InitPWM();
10    InitTimer2();
11
12    InitUart1();
13    InitPS2();
14    InitFlash();
15    InitMemory();
16
17 }
18

```

```

void InitPWM(void)
{
    myservo[0].attach(2, 500, 2500); // attaches the servo on pin 2 to the servo object
    myservo[1].attach(3, 500, 2500);
    myservo[2].attach(4, 500, 2500);
    myservo[3].attach(5, 500, 2500);
    myservo[4].attach(6, 500, 2500);
    myservo[5].attach(7, 500, 2500);
}

```

- 2) TaskRun is called in loop. TaskTimeHandle function is called in TaskRun and this function will be timed to call ServoPwmDutyCompare function.

```

138 void TaskTimeHandle(void)
139 {
140     static uint32 time = 10;
141     static uint32 times = 0;
142     if(gSystemTickCount > time)
143     {
144         time += 10;
145         times++;
146         if(times % 2 == 0) //20ms
147         {
148             ServoPwmDutyCompare();
149         }
150     }
151 }
152 }
153 }

46 void ServoPwmDutyCompare(void) //Pulse width change comparison and speed control
47 {
48     uint8 i;
49
50     static uint16 ServoPwmDutyIncTimes; //increments times
51     static bool ServoRunning = FALSE; //The servo is moving at the specified speed to the position corresponding to the specified pulse width
52     if(ServoPwmDutyHaveChange) //Calculate when the running stops and the pulse width changes ServoRunning == FALSE &&
53     {
54         ServoPwmDutyHaveChange = FALSE;
55         ServoPwmDutyIncTimes = ServoTime/20; //Use this sentence when the ServoPwmDutyCompare() function is called every 20ms
56         for(i=0;i<8;i++)
57         {
58             //if(ServoPwmDuty[i] != ServoPwmDutySet[i])
59             {
60                 if(ServoPwmDutySet[i] > ServoPwmDuty[i])
61                 {
62                     ServoPwmDutyInc[i] = ServoPwmDutySet[i] - ServoPwmDuty[i];
63                     ServoPwmDutyInc[i] = -ServoPwmDutyInc[i];
64                 }
65                 else
66                 {
67                     ServoPwmDutyInc[i] = ServoPwmDuty[i] - ServoPwmDutySet[i];
68                 }
69                 ServoPwmDutyInc[i] /= ServoPwmDutyIncTimes; //Pulse width for each increment
70             }
71         }
72     }
73     ServoRunning = TRUE; //The servo starts running
74 }
75 if(ServoRunning)
76 {
77     ServoPwmDutyIncTimes--;
78     for(i=0;i<8;i++)
79     {
80         if(ServoPwmDutyIncTimes == 0)
81         {
82             //The last increment will directly assign the set value to the current value
83             ServoPwmDuty[i] = ServoPwmDutySet[i];
84             ServoRunning = FALSE; //Reach the set position, the servo stops moving
85         }
86         else
87         {
88             ServoPwmDuty[i] = ServoPwmDutySet[i] +
89                 (signed short int)(ServoPwmDutyInc[i] * ServoPwmDutyIncTimes);
90         }
91     }
92     if((i >= 0) && (i <= 6))
93     {
94         myservo[i - 1].writeMicroseconds(ServoPwmDuty[i]);
95     }
96 }

```

- 3) When set the function parameters as the figure shown above, that is, the command of the servo rotation, the ServoPwmDuty Compare function will decompose the time according to the set value.
- 4) Then loop to judge the angle of all the servos that need to be changed and divide the rotation of servos into multiple times by decomposing the time so as to achieve the control to the time, that is, the control of the servos speed. The shorter the time, the faster the speed.