

# 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

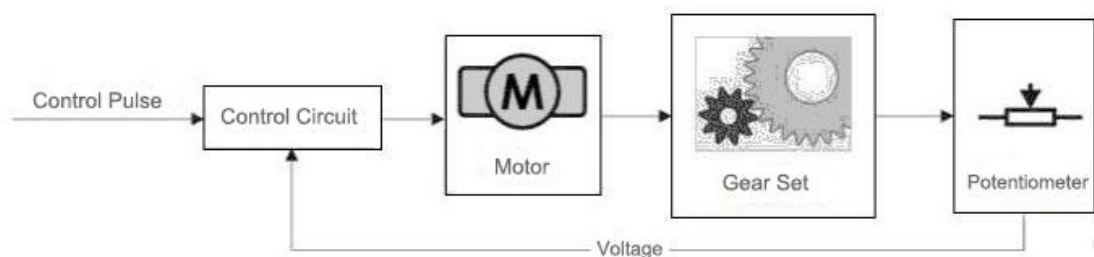
### 2.1 Servo Internal Structure

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.

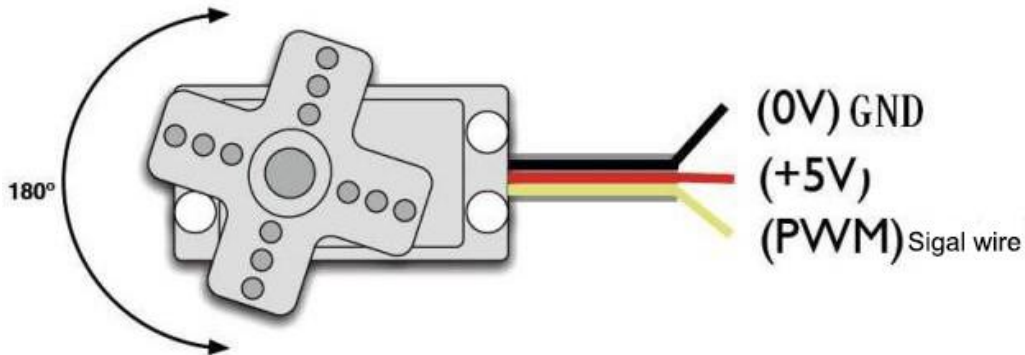


The reduction gear is driven by motors and its output terminal drives a linear potentiometer for positional detection. This potentiometer converts the angle into a proportional voltage feedback to the control circuit. Then the control circuit compares the proportional voltage with the angle corresponding 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







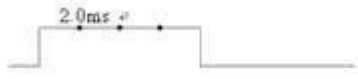



Servo motors have three wires: power, ground, and signal.



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.)

Input a periodic positive pulse signal. The high level of this periodic pulse signal is usually between 1ms-2ms, and the low level time should be between 5ms and 20ms.

Analog servo is required to maintain periodic signal all the time to keep the servo angle. When the signal is lost, servo will no longer output power. The digital servo is used in uHand2.0 As long as the correct high-level signal is sent once, the locked angle can be maintained, and no strict requirement for the low level time.

输入正脉冲宽度（周期为 20ms）	舵机输出臂位置
	
	
	
	
	

### 3. Program Analyst

- 1) The pulse width of the signal for controlling ranges 500us-2500us. 2500us–20000us is low level.
- 2) There is a way for you: take 500us high level as an example, that is, it is 500us high level + 2000 low level + 17500us low level; 1000us high level is 1000us high level + 1500us low level + 17500us low level. Therefore, we actually only need to control the high and low level division of 2500us and keep the low level at other times.
- 3) We know that a whole signal period is 20ms, that is, 20000us, which means eight 2500us. Therefore, cyclic control of the 2500us high and low level division of eight servos in a 20ms period can realize the operation of controlling eight servos with a timer.
- 4) If want to control the servo, the pin for controlling the servo needs to be configured to Push-Pull output mode.

```

119 void InitPWM(void)
120 {
121     GPIO_InitTypeDef GPIO_InitStructure;
122
123     InitTimer3();
124
125     RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
126     GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5 | GPIO_Pin_8;
127     GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP; //push-pull output
128     GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
129     GPIO_Init(GPIOB, &GPIO_InitStructure);
130
131     RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC, ENABLE);
132     GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10 | GPIO_Pin_11 | GPIO_Pin_12;
133     GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP; //push-pull output
134     GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
135     GPIO_Init(GPIOC, &GPIO_InitStructure);
136
137     RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOD, ENABLE);
138     GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2;
139     GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP; //push-pull output
140     GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
141     GPIO_Init(GPIOD, &GPIO_InitStructure);
142 }

```

- 5) The minimum resolution of the servo control signal is 1us so the time base of the timer can be configured to 1us. Turn on the timer interrupt, and then modify the output status of I/O port and the time of next interrupt in timer interrupt.

```

99 void InitTimer3(void)
100 {
101     NVIC_InitTypeDef NVIC_InitStructure;
102
103     RCC->APB1ENR|=1<<1; //TIM3 clock enable
104     TIM3->ARR=10000 - 1; //Set time auto reload value//1ms
105     TIM3->PSC=72 - 1; //Prescaler 72, get 1Mhz counting clock
106     //Both must be set at the same time to use interrupt
107     TIM3->DIER|=1<<0; //allow update interruption
108     // TIM3->DIER|=1<<6; //allow interrupts to be triggered
109     TIM3->CR1|=0x01; //enable Timer 3
110
111     NVIC_InitStructure.NVIC_IRQChannel = TIM3_IRQn; //TIM3 interrupt
112     NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0; //Preemption priority 3
113     NVIC_InitStructure.NVIC_IRQChannelSubPriority = 3; //SubPriority 3
114     NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE; //Enable IRQ channel
115     NVIC_Init(&NVIC_InitStructure); //Initial peripherals NVIC register according to the parameters specified in NVIC_InitStruct

```

- 6) The following is the code for timer interrupt.

```

144 //Convert the PWM pulse width to the value of the auto-load register
145 void Timer3ARRValue(uint16 pwm)
146 {
147     TIM3->ARR = pwm + 1;
148 }
149
150
151 //Timer 3 interrupt service program
152 void TIM3_IRQHandler(void)
153 {
154     static uint16 i = 1;
155
156     if(TIM3->SR4 & 0x0001) //Overflow interrupt
157     {
158         switch(i)
159         {
160             case 1:
161                 //SERVO0 = 1; //PWM control pin high level
162                 //Assign timer0. After counting Pwm0Duty pulses, interruption is generated. The next interruption will enter case statement.
163                 Timer3ARRValue(ServoPwmDuty[0]);
164                 break;
165             case 2:
166                 //SERVO0 = 0; //PWM Control pin low level
167                 //The interruption generated by this counter assignment indicates the start of the task to be performed by the next unit
168                 Timer3ARRValue(2500-ServoPwmDuty[0]);
169                 break;
170             case 3:
171                 SERVO1 = 1;
172                 Timer3ARRValue(ServoPwmDuty[1]);
173                 break;
174             case 4:
175                 SERVO1 = 0; //PPWM Control pin low level
176                 Timer3ARRValue(2500-ServoPwmDuty[1]);
177                 break;
178             case 5:
179                 SERVO2 = 1;
180                 Timer3ARRValue(ServoPwmDuty[2]);
181                 break;

```

- 7) Let's briefly analyze the switch statement. Suppose the interrupt is entered for the first time now. After entering the interrupt, execute case 1 ,and then set the I/O port of SERVO0 to high level, and set the time of timer interrupt to the high level time of the servo signal.
- 8) When entering the next interrupt, execute case 2, and set the I/O port of SERVO0 to low level, and then set the interrupt time of the timer to the remaining time in 2500us.By repeating this process 8 times, a period of 20ms can be realized.

```

182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

```

case 6:
    SERVO2 = 0; //PWM controls pin low level
    Timer3ARRValue(2500-ServoPwmDuty[2]);
    break;
case 7:
    SERVO3 = 1;
    Timer3ARRValue(ServoPwmDuty[3]);
    break;
case 8:
    SERVO3 = 0; //PWM controls pin low level
    Timer3ARRValue(2500-ServoPwmDuty[3]);
    break;
case 9:
    SERVO4 = 1;
    Timer3ARRValue(ServoPwmDuty[4]);
    break;
case 10:
    SERVO4 = 0; //PWM controls pin low level
    Timer3ARRValue(2500-ServoPwmDuty[4]);
    break;
case 11:
    SERVO5 = 1;
    Timer3ARRValue(ServoPwmDuty[5]);
    break;
case 12:
    SERVO5 = 0; //PWM controls pin low level
    Timer3ARRValue(2500-ServoPwmDuty[5]);
    break;
case 13:
    SERVO6 = 1;
    Timer3ARRValue(ServoPwmDuty[6]);
    break;
case 14:
    SERVO6 = 0; //PWM controls pin low level
    Timer3ARRValue(2500-ServoPwmDuty[6]);
    break;
case 15:
    SERVO7 = 1;
    Timer3ARRValue(ServoPwmDuty[7]);
    break;
case 16:
    SERVO7 = 0; //PWM controls pin low level
    Timer3ARRValue(2500-ServoPwmDuty[7]);
    i = 0;
    break;
}
i++;
}
TIM3->SR&=~(1<<0); //Clear interruption mark

```

- 9) Because there are only 6-ch servo ports on the palm controller, comment out SERVO0 and SERVO7. In this way, when operating the array, the subscript of the array is the servo corresponding to the servo number.