```ino
#include <Servo.h>

// Arduino pin assignment
#define PIN_SERVO 10

// configurable parameters for Servo
#define _DUTY_MIN 550 // servo full clock-wise position (0 degree)
#define _DUTY_NEU 1475 // servo neutral position (90 degree)
#define _DUTY_MAX 2400 // servo full counter-clockwise position (180 degree)

#define _SERVO_SPEED 0.3 // servo angular speed (unit: degree/sec)

// Loop Interval
#define _INTERVAL_SERVO   20 // servo interval (unit: msec)
#define _INTERVAL_SERIAL  100 // serial interval (unit: msec)
```

상수 선언부

서보 핀 번호

서보 duty 범위

서보 각속도

서보 작동 주기
시리얼 작동 주기

```ino
// global variables
unsigned long last_sampling_time_servo;  // unit: msec
unsigned long last_sampling_time_serial; // unit: msec
bool event_servo, event_serial;

Servo myservo;

float duty_change_per_interval; // maximum duty difference per interval
int duty_target;     // Target duty time
float duty_curr;      // Current duty time

int toggle_interval, toggle_interval_cnt;
```

변수 선언부

서보 마지막 작동 시각
시리얼 마지막 작동 시각
이번 실행에 서보, 시리얼을 작동할건지 여부

주기당 duty 변화량
목표 duty
현재 duty

서보 저속 구동을 할 때 정수형에 저장하면 소수점 아래가 손실돼서 0이 되어버린다(ex. 각속도 0.3°/s -> 주기당 duty 변화량 = (duty 범위) * (서보 각속도°/s / 180.0°/s) * (서보 작동 주기ms / 1000.0ms) = 0.0617). 때문에 "주기당 duty 변화량"과 "현재 duty"는 float를 쓴다.

토글까지 실행할 횟수, 현재 실행 횟수

```ino
void setup() {
  // initialize GPIO pins
  myservo.attach(PIN_SERVO);

  duty_target = duty_curr = _DUTY_MIN;
  myservo.writeMicroseconds(duty_curr);

  // initialize serial port
  Serial.begin(57600);  // <----- baud rate

  // convert angular velocity into duty change per interval.
  duty_change_per_interval =
    (float)(_DUTY_MAX - _DUTY_MIN) * (_SERVO_SPEED / 180.0) * (_INTERVAL_SERVO / 1000.0);

  // initialize variables for servo update.
  toggle_interval = (180.0 / _SERVO_SPEED) * 1000 / _INTERVAL_SERVO;
  toggle_interval_cnt = toggle_interval;

  // initialize last sampling time
  last_sampling_time_servo = 0;
  last_sampling_time_serial = 0;
}
```

void setup()

서보, 시리얼 작동 시작

변수 초기값 지정

```ino
void loop() {
  // wait until next event time.
  unsigned long time_curr = millis();
  if (time_curr >= (last_sampling_time_servo + _INTERVAL_SERVO)) {
    last_sampling_time_servo += _INTERVAL_SERVO;
    event_servo = true;
  }
  if (time_curr >= (last_sampling_time_serial + _INTERVAL_SERIAL)) {
    last_sampling_time_serial += _INTERVAL_SERIAL;
    event_serial = true;
  }
```

void loop()

서보와 시리얼의 작동 여부 결정

```ino
  if (event_servo) {
    event_servo = false;
    // adjust duty_curr toward duty_target by duty_change_per_interval
    if (duty_target > duty_curr) {
      duty_curr += duty_change_per_interval;
      if (duty_curr > duty_target)
          duty_curr = duty_target;
    } else {
      duty_curr -= duty_change_per_interval;
      if (duty_curr < duty_target)
        duty_curr = duty_target;
    }
    // update servo position
    myservo.writeMicroseconds((int)duty_curr);
    // toggle duty_target between _DUTY_MIN and _DUTY_MAX.
    if (toggle_interval_cnt >= toggle_interval) {
      toggle_interval_cnt = 0;
      if (duty_target == _DUTY_MIN)
        duty_target = _DUTY_MAX;
      else
        duty_target = _DUTY_MIN;
    } else {
      toggle_interval_cnt++;
    }
  }
```

서보 작동시

duty 목표값 까지만 "주기당 duty 변화량"만큼 duty를 변경

서보 작동

toggle_interval 만큼 작동 됐으면 작동 횟수를 0으로 초기화하고
서보 작동 방향을 반전시킴

서보 작동 횟수(카운트) 증가

```ino
  if (event_serial) {
    event_serial = false;
    Serial.print("Min:550");
    Serial.print(",duty_target:"); Serial.print(duty_target);
    Serial.print(",duty_curr:");   Serial.print(duty_curr);
    //Serial.print(",duty_change_per_interval:");   Serial.print(duty_change_per_interval);
    Serial.println(",Max:2400");
  }
  delay(5);
}
```

시리얼 작동시
시리얼 출력