소프트웨어 프로젝트 2

서보모터, Part III

2024년 2학기

국민대학교
소프트웨어학부/인공지능학부
주용수, 최진우, 한재섭, 허대영
{ysjoo, jaeseob, jnwochoi, dyheo}@kookmin.ac.kr

서보 속도 제어

- 서보 인터페이스 자체에는 속도제어 기능이 포함되지 않음
  - 연속된 위치 입력값들의 변화량에 따라 속도가 간접적으로 제어됨

- 프로그램에서는 위치(각도) 입력이 급격하게 변할 수 있음
  - 예, 센서에서 잡히는 노이즈가 서보의 입력으로 연결되는 경우
    서보가 한쪽으로 최대 속도로 움직이는 중, 갑자기 반대 방향 입력이 발생

- 급격한 위치 변화의 부작용
  - 서보 기어 박스에 큰 충격이 가해짐 -> 서보 수명 저하 또는 물리적 고장
  - 제어 불안정 발생
    - 기어박스 진동 ==> 서보 몸체 진동 ==> 프레임 진동 ==>
      프레임에 장착된 센서 진동 ==> 센서 측정치에 스파이크 형태의 노이즈 유입==>
      급격한 위치변화 입력 ==> … 의 악순환 반복

서보 속도 제어

- 위치 입력값의 단위 시간당 최대 변화량을 제한한다면?
  - 서보의 최대 속도를 제한하는 것과 동일한 효과를 얻을 수 있음

- **`duty_change_per_interval`**
  - 서보 속도 제어 업데이트 1주기에 증감 가능한 duty의 최대값

- 예제 (서보 속도 제어 업데이트 주기: 20 ms로 가정)
  - **`duty_change_per_interval`**을 5 us로 설정하는 경우,
    - 서보의 현재 위치: **`duty_curr`** = 1000 us (0° 로 가정)
    - 서보의 목표 위치: **`duty_target`** = 2000 us (180° 로 가정)
    - 질문 1: **`duty_curr`** 값이 **`duty_target`** 값에 도달하는 시간?
      - 답: X seconds
    - 질문 2: **`duty_change_per_interval`** = 5 us를 각속도로 환산?
      - 답: Y° / sec

서보 속도 제어 코드 구현

- 서보 속도를 duty 변화량 대신 각속도로 설정
  - **_SERVO_SPEED**: 서보의 각속도(초당 각도 변화량, angular speed)
  - **_SERVO_SPEED**(°/sec)를 **duty_change_per_interval**, 즉 **INTERVAL** 시간 동안의 duty 변화량으로 환산해야 함
  - **duty_change_per_interval =**
    (_DUTY_MAX - _DUTY_MIN) * _SERVO_SPEED / 180 * INTERVAL / 1000;

```
14  #define _SERVO_SPEED 30 // servo speed limit (unit: degree/second)
15  #define INTERVAL 20     // servo update interval (unit: msec)
```

```
22  int duty_change_per_interval; // maximum duty difference per interval
23  int duty_target;    // Target duty time
24  int duty_curr;      // Current duty time
```

```
40    duty_change_per_interval =
41      (_DUTY_MAX - _DUTY_MIN) * (_SERVO_SPEED / 180) * (INTERVAL / 1000);
```

서보 속도 제어 코드 구현

- **duty_target**: 이동하고자 하는 목표 위치
- **duty_curr**: 서보에 실제로 입력할 위치
  - **duty_curr**는 매 주기마다 **duty_target** 값에 가까워지도록 **duty_change_per_interval** 만큼만 증감됨

```
61    // adjust duty_curr toward duty_target by duty_change_per_interval
62    if (duty_target > duty_curr) {
63      duty_curr += duty_change_per_interval;
64      if (duty_curr > duty_target)
65          duty_curr = duty_target;
66    } else {
67      duty_curr -= duty_change_per_interval;
68      if (duty_curr < duty_target)
69        duty_curr = duty_target;
70    }
```

서보 속도 제어 코드 구현

- **_POS_START**: 설정된 서보 시작 위치
- **_POS_END**: 설정된 서보 끝 위치

```
11  #define _POS_START (_DUTY_MIN + 100)
12  #define _POS_END   (_DUTY_MAX - 100)
```

- **toggle_interval_cnt** 는 몇 번의 **loop()** 만에 180°를 움직일지를 결정

```
48    // initialize variables for servo update.
49    toggle_interval = (180.0 / _SERVO_SPEED) * 1000 / INTERVAL;
50    toggle_interval_cnt = toggle_interval;
```

```
81    // toggle duty_target between _DUTY_MIN and _DUTY_MAX.
82    if (toggle_interval_cnt >= toggle_interval) {
83      toggle_interval_cnt = 0;
84      if (duty_target == _POS_START)
85        duty_target = _POS_END;
86      else
87        duty_target = _POS_START;
88    } else {
89      toggle_interval_cnt++;
90    }
```

실습 1: 코드 디버깅

- 예제 코드: 12_example_1.ino
  - https://www.dropbox.com/scl/fi/fidclsqhrjogvlofffte t/12_example_1.ino?rlkey=wnrp6aa6fyreru24ywc41gta2&dl=0
  - ???로 되어 있는 _DUTY_MIN/NEU/MAX 값을 이전 실험에서 구한 값으로 설정
- line 40, 41의 식은 정상 동작하지 않음 ==> 수정하시고..
  - 원인 분석 및 디버깅이 끝난 뒤 line 46: **`while (1) {}`** 제거

```
38    // convert angular speed into duty change per interval.
39    // Next two lines are WRONG. you have to modify.
40    duty_change_per_interval =
41      (_DUTY_MAX - _DUTY_MIN) * (_SERVO_SPEED / 180) * (INTERVAL / 1000);
42
43    // remove 'while(1) { }' lines after modify
44    Serial.print("duty_change_per_interval:");
45    Serial.println(duty_change_per_interval);
46    while (1) { }
```

실습 1: 코드 디버깅

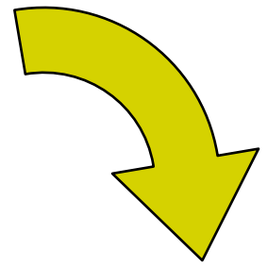
- 변수형 : **`float`** VS. **`int`**
  - 정수형끼리 곱셈을 할 때, 범위를 초과하는 overflow를 주의
  - 정수형끼리 나눗셈을 할 때, 소수점 이하의 정보가 사라지는 문제에 유의
- 해결책
  - 중간 계산과정에서 필요할 경우 명시적으로 **`float`** 속성을 부여하여 부동소수점 연산이 수행될 수 있도록 할 것

- **`float`** 속성 부여 방법
  - **`(float) var`** , **`(float) 1000`** , **`1000.0`**

- **`float`** 속성의 적용 범위도 고려
  - **`100.0 / (3 / 4)`** VS. **`100.0 / 3 / 4`**

실습 1: 코드 디버깅

```c
#include <stdio.h>

int main()
{
  int a;

  a = 95.0 * (3 / 4);
  printf("95.0 * (3 / 4) = %d\n", a);

  a = 95 * (3.0 / 4);
  printf("95 * (3.0 / 4) = %d\n", a);

  a = 95 * (3 / 4.0);
  printf("95 * (3 / 4.0) = %d\n", a);

  a = 95.0 * 3 / 4;
  printf("95.0 * 3 / 4 = %d\n", a);

  return 0;
}
```



```
95.0 * (3 / 4) = 0
95 * (3.0 / 4) = 71
95 * (3 / 4.0) = 71
95.0 * 3 / 4 = 71
```

실습 2: 서보 속도 변경

- 서보 속도가 실제로 변화하는 **_SERVO_SPEED**의 범위를 구할 것
  - **_SERVO_SPEED** 값을 변경해가면서 서보 모터 움직임을 관찰
  - 최저속도: 더 낮은 값에 대해서는 서보가 전혀 움직이지 않음
  - 최고속도: 더 높은 값에 대해서는 서보 속도가 더이상 증가하지 않음
    - 더 이상 증가하지 않는지 어떻게 확인할 수 있는가?

- 주어진 interval 20ms에서 구현 가능한 최고 각속도는?
  - 힌트: 서보의 위치 입력 범위는 0°~180°로 제한되어 있음
- 서보가 반응하는 최저 각속도를 a°/sec (a>1)라고 할 경우 (a - 1)°/sec에 대해서는 서보가 반응하지 않는 이유를 찾아볼 것
  - 시리얼 모니터에서 관련 변수의 실제 값 및 변화 추이를 관찰