BLDC Motor Control 예비보고서

20101449 한가은

1. 실험 개요

1.1. 실험 목적

전기적 정류자를 SW로 구현하여 BLDC Motor를 원하는 position이 되도록, 원하는 velocity로 회전하도록 제어한다. 이 때 PID 제어 기법이 사용된다.

1.2. 실험 대상 및 배경

1.2.1. BLDC motor란

Brush가 없는 DC 모터로, 고정자로 코일을, 회전자로 영구자석을 갖는다. 전기회로로 구현된 3상 입력의 상전환을 인가받아 회전하며, 상전환 시점을 결정하기 위해 홀센서가 내장된다.

DC 모터와 비교하여 기계적 마모가 없다는 장점을 가지며, 스테핑 모터와 비교하여 전기회로로 구현된 입력 상의 변경을 인가받아 회전한다는 유사점을 갖는다. 또한, 스테핑 모터와 마찬가지로 open loop 구동이 가능하며, 이 때 홀센서를 통해 탈조를 방지할 수 있다는 특징을 갖는다.

구동을 위해 인가받는 3상 전압은 각 High, Low, Hi-Z 상태 중 하나를 가지며, 각 입력 상에 대한 자기장의 방향에 회전자가 정렬되는 방식으로 구동된다. 구체적인 구동 매커니즘은 다음 순서를 따른다.

1. 현재 회전자 위치(홀센서 상태) 확인
2. 회전하려는 방향의 3상 입력 인가
3. 회전자 위치 변경에 따른 3상 입력 변경

이 때, 회전자와 자기장 방향이 90도 차이가 날 때 최대 토크를 가지며, open-loop 구동의 경우 탈조 현상에 유의해야한다.

1.2.2. BLDC motor에 사용되는 제어 기법

BLDC 모터의 경우 구동 이후 DC 모터와 차이가 발생하지 않는다. 즉, BLDC 모터의 제어에는 DC 모터와 마찬가지로 PID 제어 기법이 사용된다.

실험에 적용될 PID 제어 기법을 다음과 같이 코드로 나타낼 수 있다.

```
// compute PID
    sumErr += err;
    u = (Kp * err) + (Ki * sumErr) + (Kd * (err-prevErr));
    prevErr = err;
```

2. 실험 내용 및 예상 구현방법

2.1. 실험 절차

실험은 네 단계로 진행되며, 각 단계는 다음과 같다.

2.1.1. PWM을 이용한 3상 신호 생성 및 BLDC의 open-loop 구동

BLDC 모터 코일 3개 단자에 각각 독립적인 PWM 파형의 듀티비를 설정하여 모터를 구동한다.

- Hi-z는 0x800, High와 Low는 0x800을 기준으로 서로 대칭이 되도록 설정하며, 다음과 같은 코드로 작성할 수 있다.

  ```
  #define PWMD 0x7FF
  #define PWMZ 0x800
  #define PWML (PWMZ - PWMD)
  #define PWMH (PWMZ + PWMD)
  ```

  이와 같은 구현 시, PWMD값을 줄일수록 모터에 인가되는 전압 값이 작아져 탈조가 발생하는 delay가 커진다.

- delay 조절을 통해 속도를 조절하고, 탈조가 발생하는 delay를 확인한다.

2.1.2. BLDC 입력 3상에 따른 Hall 센서 신호 발생

코일에 인가한 상에 따라 홀센서 값을 측정하여 표를 작성한다.

- BLDCHALL 레지스터를 통해 홀센서 상을 알아낸다. 이 때, BLDCHALL의 하위 3bit만 필요하므로 `*BLDCHALL & 0x7;` 의 masking 작업을 한다.
- L,Z,H의 조합은 상 경계의 회전자 위치를 만드므로, 홀센서 값을 알아내는 용도에 부적합하다. 따라서 Hi-Z가 없는 상태에서의 홀센서 상태를 관찰한다. 이 때, 홀센서 값의 측정은 상전환 후 delay가 적용된 뒤 측정해야한다. 다음과 같은 코드로 작성할 수 있다.

  ```
  while(1){
      // 120 degree
      *BLDC2 = PWMH;
      *BLDC1 = PWML;
      *BLDC0 = PWMH;
      delay_ms(1000);
      hall = *BLDCHALL & 0x7;
  ```

```
    MACRO_PRINT((tmp_string, "HLH: %d\n", hall));

    // 180 degree
    *BLDC2 = PWMH;
    *BLDC1 = PWML;
    *BLDC0 = PWML;
    delay_ms(1000);
    hall = *BLDCHALL & 0x7;
    MACRO_PRINT((tmp_string, "HLL: %d\n", hall));
    ...
}
```

2.1.3. Hall 센서 신호를 이용한 BLDC 구동

실험 2.1.2.의 결과를 통해 홀센서를 이용한 전기적 정류자를 구현한다.

- `void BLDCDrive(float duty)`
  - 현재 hall 센서의 상태를 읽고, 일정한 방향으로 회전하기 위한 코일의 상을 출력한다.
  - duty값에 따라 회전 속도를 조절한다. 탈조를 주의한다.
  - 상전환 순서는 고정하고, High와 Low의 PWM 값을 반전시켜 회전 방향을 반전시킨다.
  - main()의 while(1) loop에서 해당 함수를 호출하며, 이 때 while(1) 내부에는 MACRO_PRINT, delay_us와 같은 함수의 사용을 금한다.

2.1.4. Hall 센서 신호를 카운트하여 피드백 제어 적용

Hall 센서 신호를 카운트하여 회전 각도를 알아내고, 위치 피드백 PID 제어를 수행한다. 실험에 사용될 BLDC 모터에서는 한바퀴에 12번의 Hall 센서 변화가 발생한다. 단, 기어비를 고려해야 하며, 실험을 통해 360도 회전에 몇 개의 Hall 센서 count가 발생하는지 알아내야 한다.

- `void BLDCDrive(float duty)` 내부에 Hall 센서 신호 카운터를 구현한다.
- 카운터 값 또는 환산된 회전 각도를 전역변수 `hallPos`라 하면, 타이머 ISR에서의 PID 제어는 다음과 같은 코드로 작성할 수 있다.

```
interrupt void ISRtimer0()
{
    float y, err, u;

    // set reference angle
    y = 360.0;

    // compute err
    err = hallPos - y;

    // compute PID
    sumErr += err;
```

```
    u = (Kp * err) + (Ki * sumErr) + (Kd * (err-prevErr));
    prevErr = err;

    // print usb monotor
    UMAddData(y, hallPos, err, u);
}
```

- Timer ISR에서 계산한 제어 입력 u를 전역변수로 선언하고, main 함수의 while(1) loop 안에서 다음과 같이 구동에 사용할 수 있다.

```
    main(){
    ..
        while(1){
            ..
            // drive the motor by controll input u
            BLDCDrive(u);
            ..
        }
    }
```

- PID 제어는 다음과 같은 순서로 한다.
  1. 응답 속도를 만족시키기 위하여 P gain을 증가시킨다.
  2. Overshoot을 줄이기 위해 D gain을 증가시킨다. 과도한 D gain에 따른 노이즈 민감성을 주의한다.
  3. 정상상태 오차를 줄이기 위해 I gain을 증가시킨다. 과도한 I gain에 따른 overshoot 증가를 초래하는 상황을 주의한다.