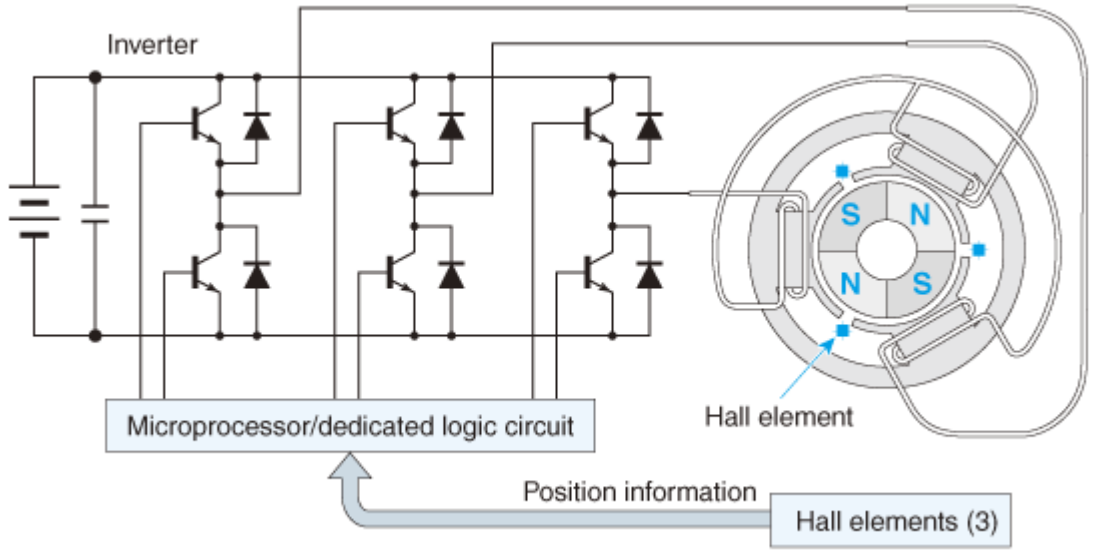
Xilinx Zynq FPGA, TI DSP, MCU 기반의 회로 설계 및 임베디드 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com

BLDC Motor Operation Principle

BLDC 전동기의 구성과 동작 원리를 살펴보자!



BLDC Motor 도 크게 다른 것은 없다.
우리가 전기/전자 회로 시간에 학습한 트랜지스터와
역기전력을 방어하기 위한 다이오드 정도의 이론을 알고 있으면
좌측의 회로 정도야 너무나도 쉬운 회로가 된다.

위 아래 트랜지스터들의 연결을 보면 하나가 살면 하나는 반드시 죽는다.
그리고 한쪽은 Emitter 쪽에서 신호가 들어가는 반면 다른 하나는 Drain 을 하고 있다.
결국 위로 들어갈때는 정방향으로 돌고 아래로 들어갈때는 반대 방향으로 돌게 된다.

모터 내부는 코일로 되어 있으므로 사실상 3 상 회로에서
학습했던 Y-Delta Circuit 의 코일 버전이라고 보면 되겠다.
단, 임피던스로 변환하였을 경우에만 가능하며
그렇지 않다면 미분 방정식이 복잡해지기 때문에 라플라스 변환이 역시나 필요하다.
사실 트랜지스터들도 걸려 있고 다양하게 걸려 있지만
소자 내부 저항과 도선 저항들을 고려하여 단순하게 제어 방정식을 표현해보자.

$$V = iR + \frac{d\lambda}{dt}$$

쇄교 자속의 변화량이 전압을 만들어낼 수 있음을 잊지말자!
맥스웰 방정식에 의해서도 시변 자기장과 전기장은 전기장과 자기장을 만들어낼 수 있다.
전기장의 구배가 전위에 해당하므로 사실상 쇄 자속의 변화량은 전압이 된다고 해석해도 무방하다.

우리가 이 쇄교 자속을 다루면서 잘 생각해봐야할 부분이 있다.

BLDC Motor 는 마모되는 것을 없게 하기 위해서 안쪽과 바깥쪽에 모두 코일을 이용한 전자석 기법으로 모터를 회전 시킨다.
그러므로 고려해야하는 쇄교 자속이란 고정된 권선에 쇄교하는 자속과 영구 자석이 돌면서 만들어내는 쇄교 자속의 합으로 구성된다.

$$\boldsymbol{\lambda = \lambda_s + \lambda_f}$$

즉 앞서서 만들었던 식을 아래와 같이 적을 수 있다.

$$\boldsymbol{V = iR + \frac{d\lambda}{dt} = iR + \frac{d(\lambda_s + \lambda_f)}{dt} = iR + \frac{d\lambda_s}{dt} + \varepsilon}$$

영구자석의 쇄교 자속 변화량이 역기전력임을 상기하며 각각의 위상차가 120 도씩 난다는 것도 기억해두자!
또 여기서 고정된 코일간의 쇄교 자속도 고려해야 하는 상황이 도출되므로 결국 식을 행렬식으로 표현하는 것이 좋다.

$$\boldsymbol{\lambda = L_s i_{123} = \begin{bmatrix} L_{11} & L_{12} & L_{13} \\ L_{21} & L_{22} & L_{23} \\ L_{31} & L_{32} & L_{33} \end{bmatrix} \begin{bmatrix} i_1 \\ i_2 \\ i_3 \end{bmatrix}}$$

여기서 각 인덕턴스들은 뒷첨자 전류에 의해 발생한 자속 중 앞첨자 권선에 쇄교하는 자속에 대한 뒷첨자 권선 전류와의 비를 나타내는 인덕턴스에 해당한다.

$$\boldsymbol{L_{11} = L_{22} = L_{33} = L_s = L_m + L_{ls}}$$
$$\boldsymbol{L_{12} = L_{13} = L_{21} = L_{23} = L_{31} = L_{32} = -\frac{1}{2}\mathrm{L_m} = \mathrm{M}}$$

$\boldsymbol{L_m}$ 과 $\boldsymbol{L_{ls}}$ 은 각각 자화 인덕턴스와 누설 인덕턴스에 해당한다.

$$\boldsymbol{\frac{d}{dt}[L_s i_1 + M i_2 + M i_3] = \frac{d}{dt}[L_s i_1 - M i_1] \qquad (\because i_1 + i_2 + i_3 = 0)}$$
$$\boldsymbol{V_1 = i_1 R_s + (L_s - M)\frac{di_1}{dt} + \varepsilon_1}$$

이제 속도 제어를 알아보도록 하자!

우선 우리가 이론적으로 학습할 부분은 Sensor 가 있는 상태에서 BLDC 를 해석해보는 것이다.
가령 Encoder 같은 것을 외부에 달더라도 센서가 있는 형태라고 볼 수 있다.
다만 엔코더는 무겁기 때문에 기체 자체의 하중을 증가 시키고
하중이 증가되면 정지 마찰력과 운동 마찰력이 증가하게 되어
최대 속도는 떨어지고 최초 구동 토크가 더 증가하게 된다는 문제가 발생한다.

그러므로 Sensor 가 없는 고 토크 고 rpm 모터를 제어하기 위한 효율적인 방법이 필요하다.
바로 FoC(Field Oriented Control) 이라는 기법으로 물리학, 수학 지식을 기반으로 센서 없이도 피드백 제어를 수행할 수 있다.
당연히 Sensor 를 사용하는 System 에 비해 제어가 어렵지만 구성을 단순화 시킬 수 있고 하중 절감, 비용, 속도, 토크면에서 이득을 얻을 수 있다.
그러므로 우리는 FoC 제어를 활용해야함을 명심하자!

일반적으로 속도 제어에는 PI Controller 가 사용된다.

$$v = \left(K_p + \frac{K_i}{s}\right)(\overline{\omega_m} - \omega_m)$$

bar 가 붙어 있는 것은 목표치이며 bar 가 없는 것은 실제 각속도에 해당한다.
이제 Cortex-R5 보드에서 본격적으로 FoC 제어기를 설계하고 제어기 코드를 만들어보도록 하자!