밸런싱 로봇
(임베디드 스쿨 LV1프로젝트)

임베디드스쿨1기
Lv1과정
2021. 01. 14
손표훈

목차

- 개요
- 시스템 구성
- MPU6050
- 상보 필터
- PID 제어기
- 구동영상
- M&S(작성 중)

● 개요

➢ 밸런싱 로봇의 원리

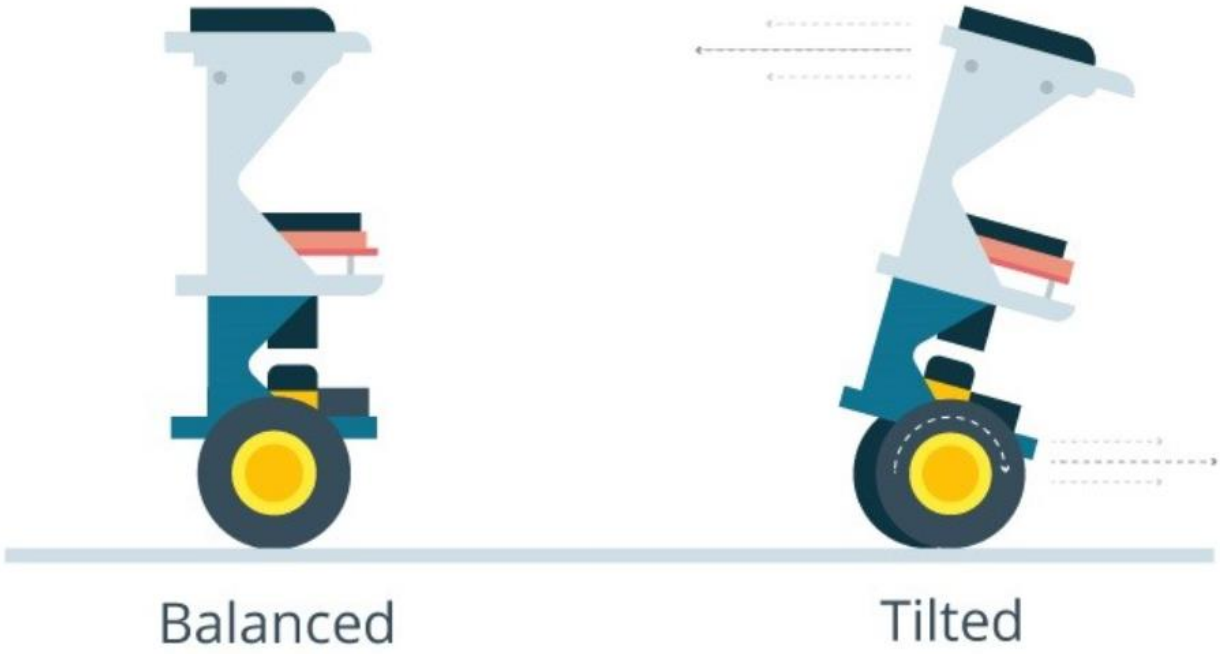
- 아래 그림과 같이 막대기를 손 위에 올려 놓고 기울어지는 방향으로 이동하면서 막대기가 손에서 떨어지지 않게 하는 것, 즉, 무게중심이 이동한 방향과 거리만큼 움직이면서 무게중심을 잡는 것이다.
- 2 자유도 운동(2-DOF) : 로봇 몸체의 기울기(PITCH)와 로봇의 병진운동

● 개요

➢ **목적**

- 동역학 모델링과 제어알고리즘, 센서신호처리에 관한 자료의 접근이 수월한 밸런싱 로봇 제어 경험을 통해 시스템 모델링의 기초에 대한 지식 습득

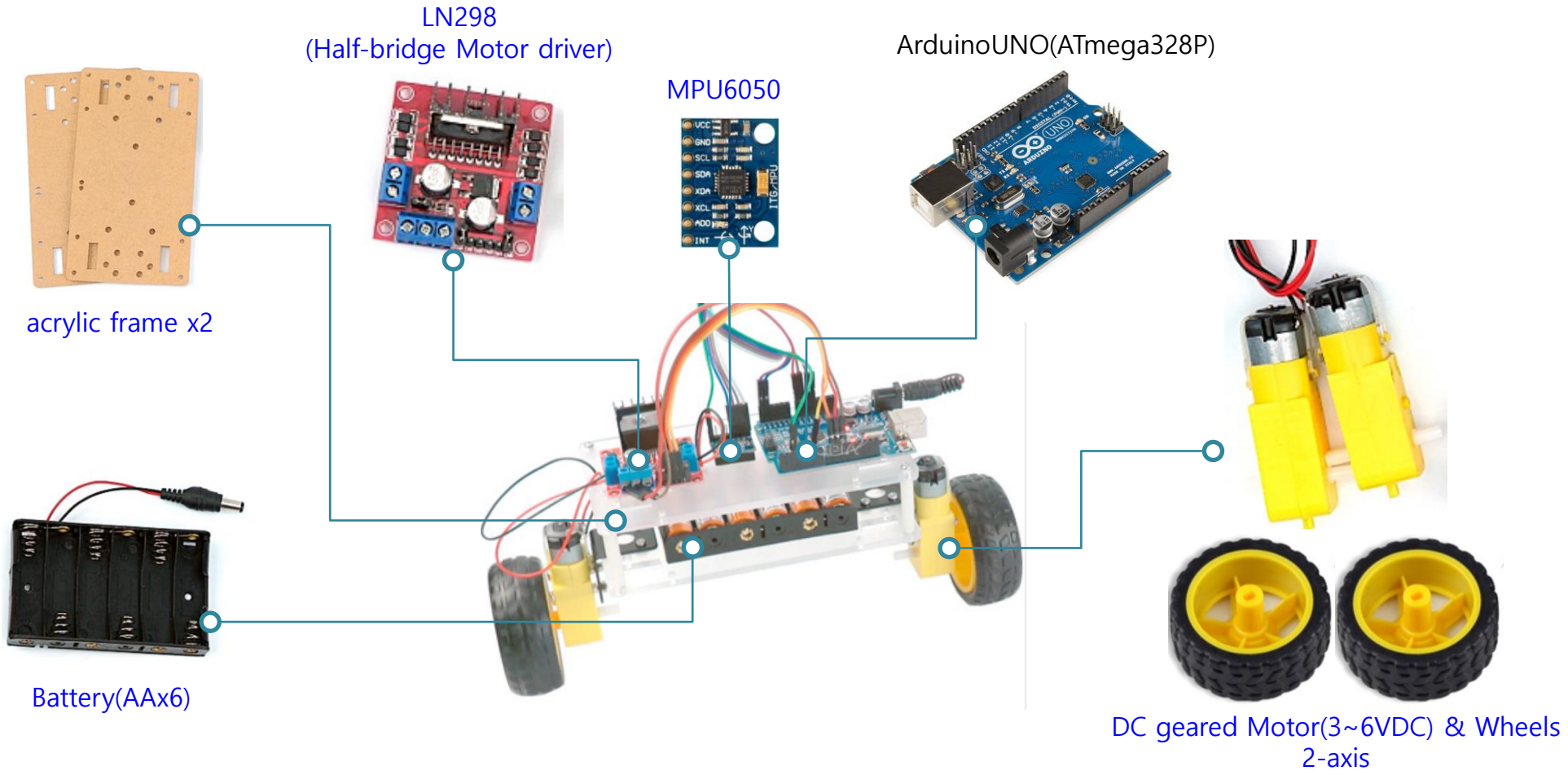
- 습득한 기초 지식을 통해 목표인 소형로켓 설계(추진연료 : 물)와 RC비행기 설계



● 개요

➢ 구성품

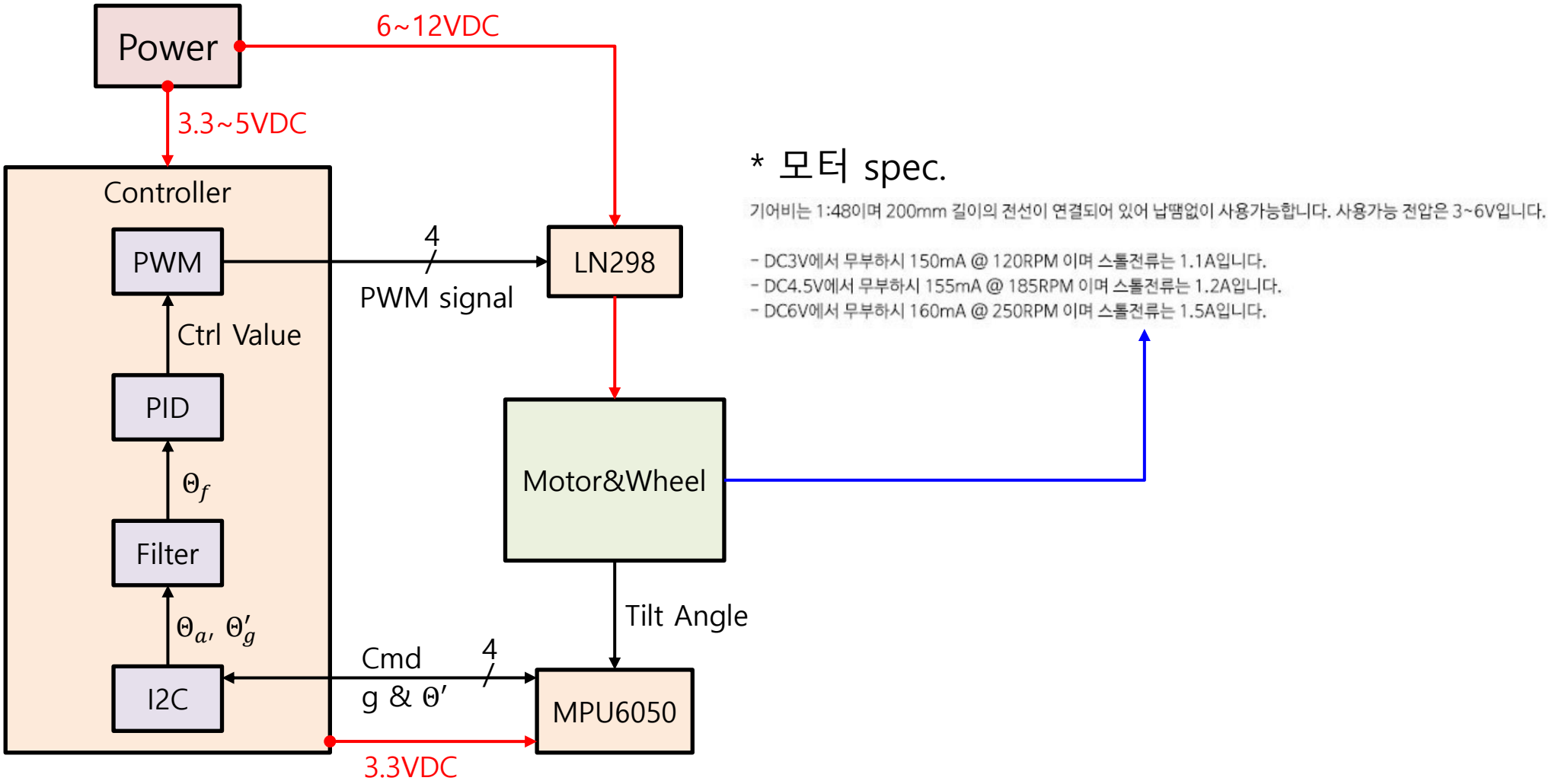
- 싸이피아의 "SBOT" 제품을 사용하여 로봇의 프레임, 센서, 모터, 모터 드라이버, 바퀴를 구성
- 컨트롤러 : ArduinoUNO(ATmega328P – 8bit MCU)



● 시스템 구성

➢ 시스템 구성 블록도

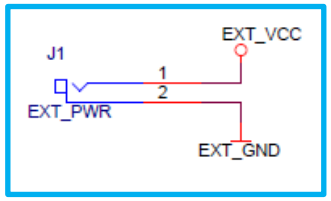
- 가속도+자이로센서가 감지한 로봇의 자세(각도)를 PID연산을 통해 제어한다.



● 시스템 구성

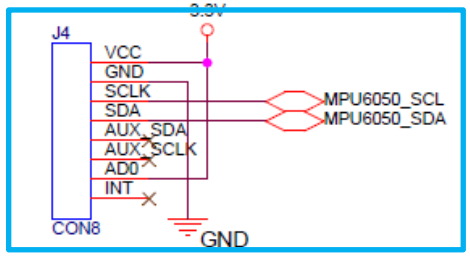
➢ 주요 회로 : Controller

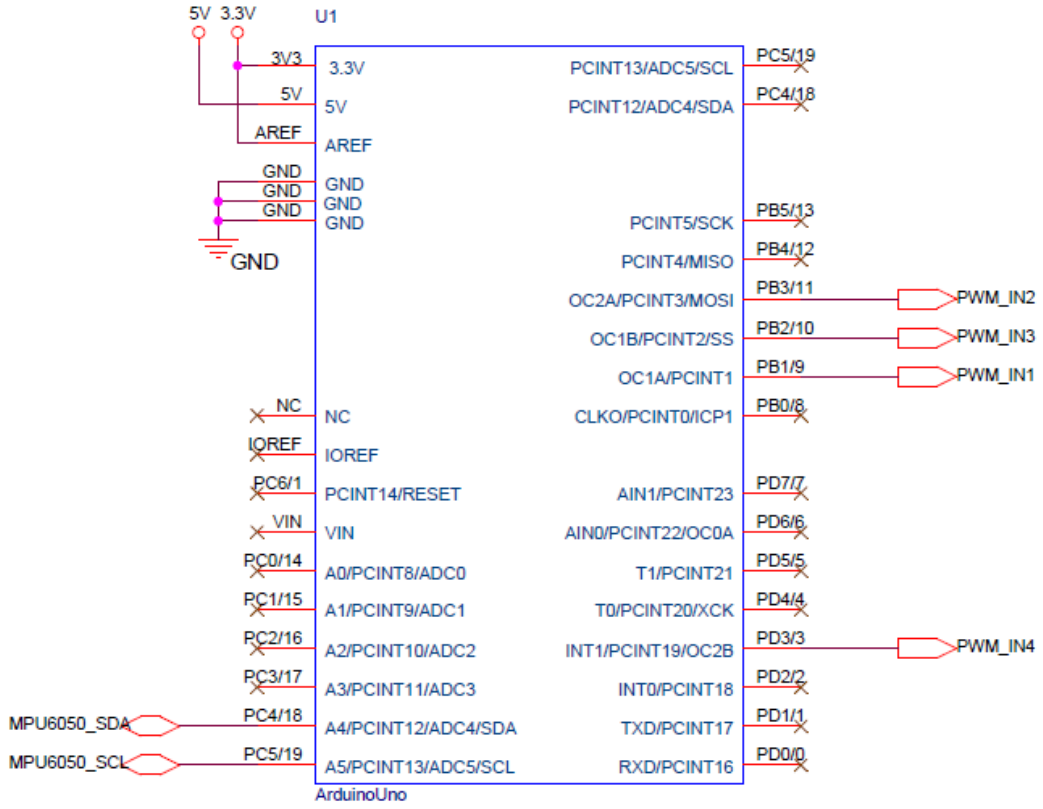
- MPU6050 인터페이스 : I2C 1CH
- 모터구동용 PWM : 8bit Timer 2CH(Timer0, Timer2), 16bit Timer 1CH(Timer1)
- 밸런싱 로봇 프로젝트에서 PWM출력은 2CH의 8bit Timer를 사용, 제어주기를 위해 Timer1을 사용



외부 전원(12V 어댑터 사용)

MPU6050 커넥터



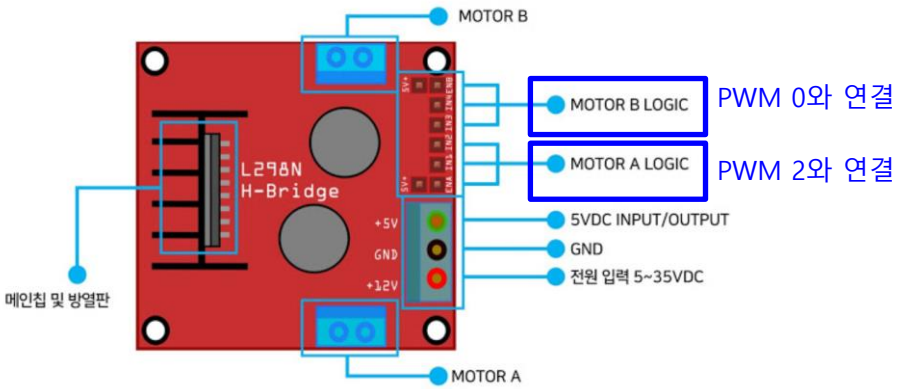


Arduino Uno(ATmega328P)

● 시스템 구성

➢ 주요 회로 – LN298

- LN298 H-bridge 모터 드라이버
- 2ch의 H-bridge 회로로 구성
- LN298의 EN핀은 Logic "High"로 항상 인가된 상태로 구성



ELECTRICAL CHARACTERISTICS (V_S = 42V; V_{SS} = 5V, T_j = 25°C; unless otherwise specified)

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
V_S	Supply Voltage (pin 4)	Operative Condition		V_{IH} +2.5		46	V
V_{SS}	Logic Supply Voltage (pin 9)			4.5	5	7	V
I_S	Quiescent Supply Current (pin 4)	V_{en} = H; I_L = 0	V_i = L		13	22	mA
			V_i = H		50	70	mA
		V_{en} = L	V_i = X			4	mA
I_{SS}	Quiescent Current from V_{SS} (pin 9)	V_{en} = H; I_L = 0	V_i = L		24	36	mA
			V_i = H		7	12	mA
V_{IL}	Input Low Voltage (pins 5, 7, 10, 12)			–0.3		1.5	V
V_{IH}	Input High Voltage (pins 5, 7, 10, 12)			2.3		VSS	V
I_{IL}	Low Voltage Input Current (pins 5, 7, 10, 12)	V_i = L				–10	µA
I_{IH}	High Voltage Input Current (pins 5, 7, 10, 12)	V_i = H ≤ V_{SS} –0.6V			30	100	µA
V_{en} = L	Enable Low Voltage (pins 6, 11)			–0.3		1.5	V
V_{en} = H	Enable High Voltage (pins 6, 11)			2.3		V_{SS}	V
I_{en} = L	Low Voltage Enable Current (pins 6, 11)	V_{en} = L				–10	µA
I_{en} = H	High Voltage Enable Current (pins 6, 11)	V_{en} = H ≤ V_{SS} –0.6V			30	100	µA
$V_{CEsat\ (H)}$	Source Saturation Voltage	I_L = 1A		0.95	1.35	1.7	V
		I_L = 2A			2	2.7	V
$V_{CEsat\ (L)}$	Sink Saturation Voltage	I_L = 1A (5)		0.85	1.2	1.6	V
		I_L = 2A (5)			1.7	2.3	V
V_{CEsat}	Total Drop	I_L = 1A (5)		1.80		3.2	V
		I_L = 2A (5)				4.9	V
V_{sens}	Sensing Voltage (pins 1, 15)			–1 (1)		2	V

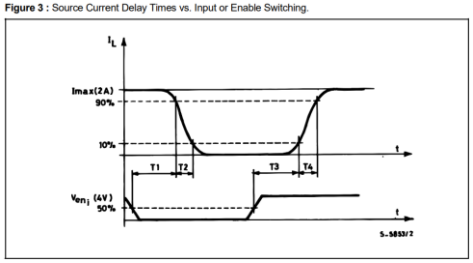
ELECTRICAL CHARACTERISTICS (continued)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
T_1 (V_i)	Source Current Turn-off Delay	0.5 V_i to 0.9 I_L (2); (4)		1.5		µs
T_2 (V_i)	Source Current Fall Time	0.9 I_L to 0.1 I_L (2); (4)		0.2		µs
T_3 (V_i)	Source Current Turn-on Delay	0.5 V_i to 0.1 I_L (2); (4)		2		µs
T_4 (V_i)	Source Current Rise Time	0.1 I_L to 0.9 I_L (2); (4)		0.7		µs
T_5 (V_i)	Sink Current Turn-off Delay	0.5 V_i to 0.9 I_L (3); (4)		0.7		µs
T_6 (V_i)	Sink Current Fall Time	0.9 I_L to 0.1 I_L (3); (4)		0.25		µs
T_7 (V_i)	Sink Current Turn-on Delay	0.5 V_i to 0.9 I_L (3); (4)		1.6		µs
T_8 (V_i)	Sink Current Rise Time	0.1 I_L to 0.9 I_L (3); (4)		0.2		µs
fc (V_i)	Commutation Frequency	I_L = 2A		25	40	KHz
T_1 (V_{en})	Source Current Turn-off Delay	0.5 V_{en} to 0.9 I_L (2); (4)		3		µs
T_2 (V_{en})	Source Current Fall Time	0.9 I_L to 0.1 I_L (2); (4)		1		µs
T_3 (V_{en})	Source Current Turn-on Delay	0.5 V_{en} to 0.1 I_L (2); (4)		0.3		µs
T_4 (V_{en})	Source Current Rise Time	0.1 I_L to 0.9 I_L (2); (4)		0.4		µs
T_5 (V_{en})	Sink Current Turn-off Delay	0.5 V_{en} to 0.9 I_L (3); (4)		2.2		µs
T_6 (V_{en})	Sink Current Fall Time	0.9 I_L to 0.1 I_L (3); (4)		0.35		µs
T_7 (V_{en})	Sink Current Turn-on Delay	0.5 V_{en} to 0.9 I_L (3); (4)		0.25		µs
T_8 (V_{en})	Sink Current Rise Time	0.1 I_L to 0.9 I_L (3); (4)		0.1		µs

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_S	Power Supply	50	V
V_{SS}	Logic Supply Voltage	7	V
V_i,V_{en}	Input and Enable Voltage	–0.3 to 7	V
I_O	Peak Output Current (each Channel)		
	– Non Repetitive (t = 100µs)	3	A
	–Repetitive (80% on –20% off; t_{on} = 10ms)	2.5	A
	–DC Operation	2	A
V_{sens}	Sensing Voltage	–1 to 2.3	V
P_{tot}	Total Power Dissipation (T_{case} = 75°C)	25	W
T_{op}	Junction Operating Temperature	–25 to 130	°C
T_{stg}, T_j	Storage and Junction Temperature	–40 to 150	°C

Figure 3 : Source Current Delay Times vs. Input or Enable Switching.

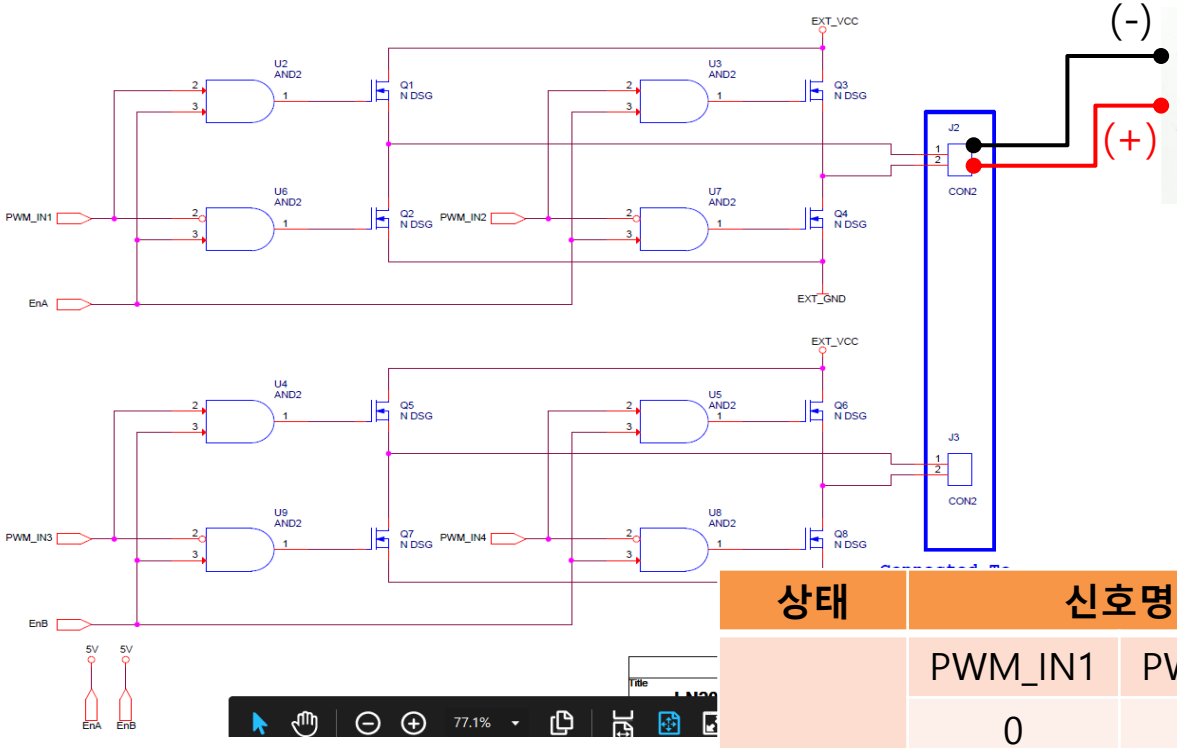


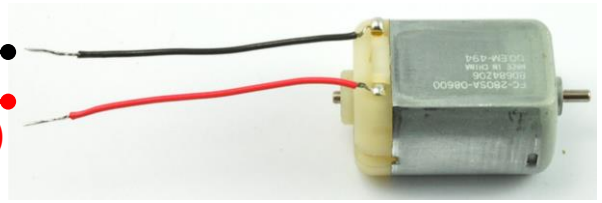
*모터의 효율과
내부 FET/BJT의 열과 관련

● 시스템 구성

➢ LN298N의 동작원리

- 모터의 + -> -핀으로 전류가 흐르면 CW, 반대로 흐르면 CCW라 가정
- Enable핀은 항상 “High”상태





상태	신호명		FET상태(ON/OFF)				방향
Logic “0” or “1”	PWM_IN1	PWM_IN2	Q1	Q2	Q3	Q4	CW, CCW, STOP
	0	0	OFF	ON	OFF	ON	STOP
	0	1	OFF	ON	ON	OFF	CW
	1	0	ON	OFF	OFF	ON	CCW
	1	1	ON	OFF	OFF	ON	STOP

MPU6050

➢ MPU6050이란?

- 자이로센서와 가속도센서가 결합된 6축 센서 모듈
- I2C, SPI 인터페이스 지원

7.6 Overview

The MPU-60X0 is comprised of the following key blocks and functions:

- Three-axis MEMS rate gyroscope sensor with 16-bit ADCs and signal conditioning
- Three-axis MEMS accelerometer sensor with 16-bit ADCs and signal conditioning
- Digital Motion Processor (DMP) engine
- Primary I^2C and SPI (MPU-6000 only) serial communications interfaces
- Auxiliary I^2C serial interface for 3rd party magnetometer & other sensors
- Clocking
- Sensor Data Registers
- FIFO
- Interrupts
- Digital-Output Temperature Sensor
- Gyroscope & Accelerometer Self-test
- Bias and LDO
- Charge Pump

각 센서는 16bit의 분해능을 가진다.

I2C와 SPI인터페이스 둘 다 사용가능

MPU6050의 기능을 설정하기 위한 Data Register
ADC 결과를 저장하는 Data Register FIFO
ADC 완료 여부를 확인 할 수 있는 인터럽트 기능

● MPU6050

➢ MPU6050 특징

- 자이로 센서 : X,Y,Z 3축의 각속도를 센싱 ±250deg/s ~ ±2000deg/s
- 가속도 센서 : ±2g ~ ±16g
- 16bit의 분해능을 가짐

5.1 Gyroscope Features

The triple-axis MEMS gyroscope in the MPU-60X0 includes a wide range of features:

- Digital-output X-, Y-, and Z-Axis angular rate sensors (gyroscopes) with a user-programmable full-scale range of ±250, ±500, ±1000, and ±2000°/sec
- External sync signal connected to the FSYNC pin supports image, video and GPS synchronization
- Integrated 16-bit ADCs enable simultaneous sampling of gyros
- Enhanced bias and sensitivity temperature stability reduces the need for user calibration
- Improved low-frequency noise performance
- Digitally-programmable low-pass filter
- Gyroscope operating current: 3.6mA
- Standby current: 5μA
- Factory calibrated sensitivity scale factor
- User self-test

5.2 Accelerometer Features

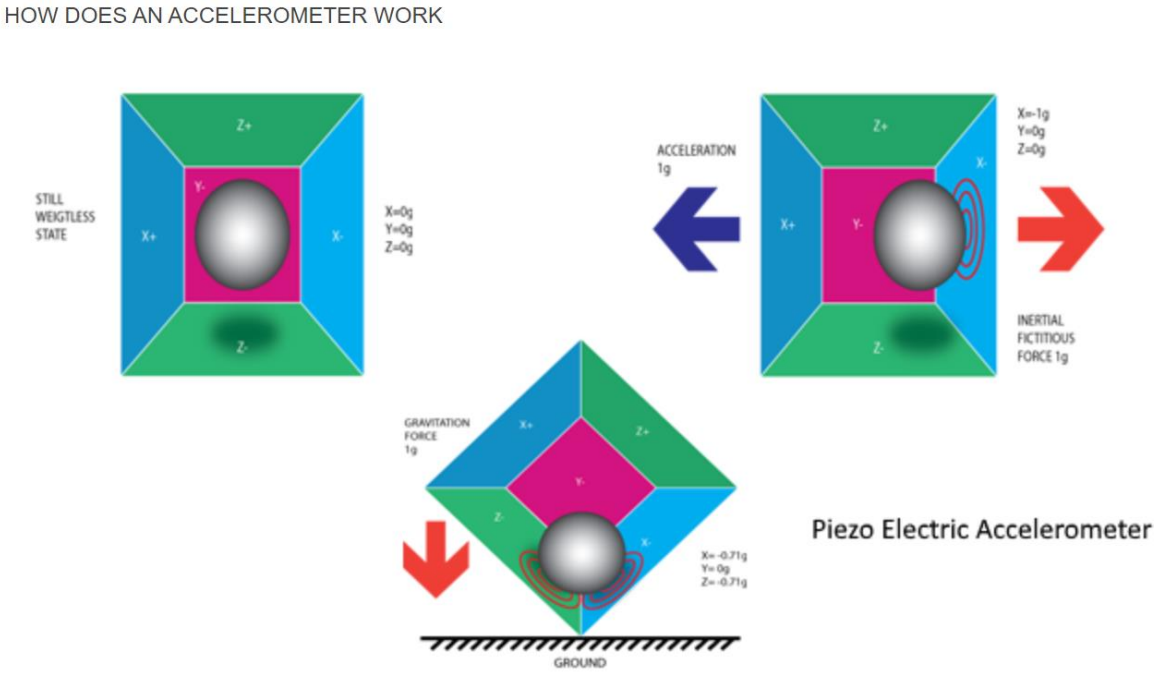
The triple-axis MEMS accelerometer in MPU-60X0 includes a wide range of features:

- Digital-output triple-axis accelerometer with a programmable full scale range of ±2*g*, ±4*g*, ±8*g* and ±16*g*
- Integrated 16-bit ADCs enable simultaneous sampling of accelerometers while requiring no external multiplexer
- Accelerometer normal operating current: 500μA
- Low power accelerometer mode current: 10μA at 1.25Hz, 20μA at 5Hz, 60μA at 20Hz, 110μA at 40Hz
- Orientation detection and signaling
- Tap detection
- User-programmable interrupts
- High-G interrupt
- User self-test

● MPU6050

➢ 가속도센서란?

- 속도의 변화량을 측정하는 센서
- 가속도 센서의 출력은 " 가속도"
- 단위 : g(중력가속도이며, 1g = 9.8$^{m}/_{s^2}$)
- 가속도 센서의 가속도를 측정하는 원리는 아래와 같다(예시)



*기본 원리 : 뉴턴의 제 2법칙(F=ma)
- 가운데 공이 들어있는 박스의 벽면은 Piezo 크리스탈로 구성
- 각 벽면은 x,y,z 3축을 의미
- 센서의 움직임에 따라 공이 벽면을 부딪혀 전류를 발생시킴
- 전류의 극성에 따라 방향이 결정되고, 크기가 힘의 크기가 됨

*그림 출처 : https://acoptex.com/project/118/basics-project-020a-mpu-6050-gy-521-gy-521-module-3-axis-gyroscope-and-accelerometer-at-acoptexcom/

➢ 가속도센서를 이용한 각도 측정

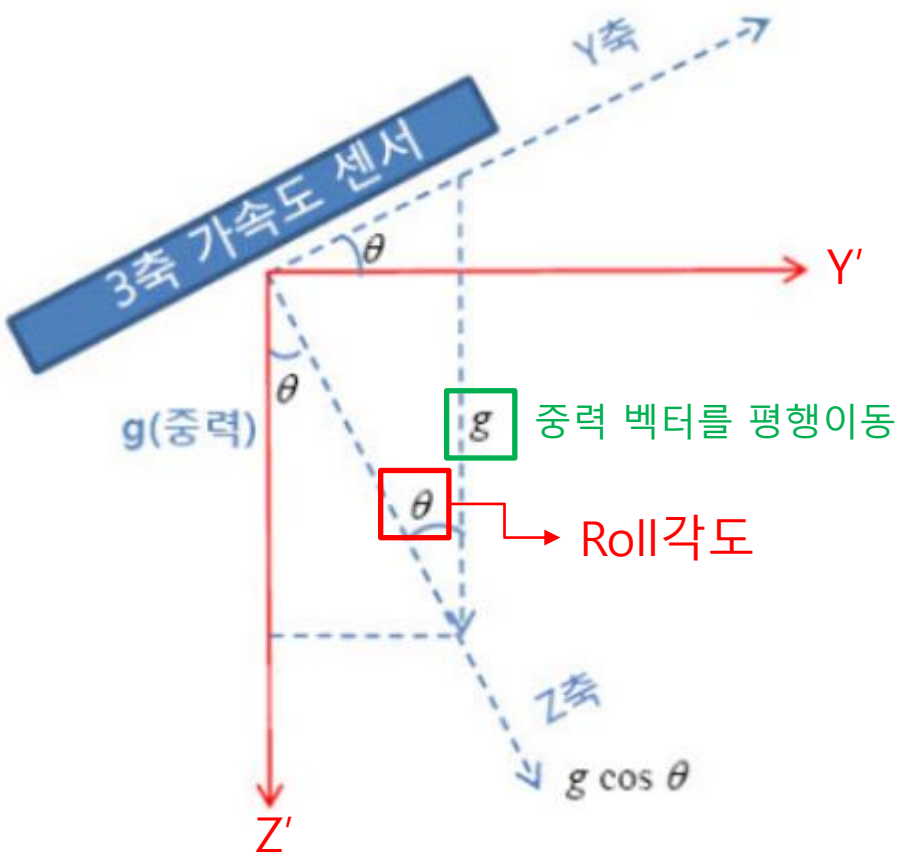


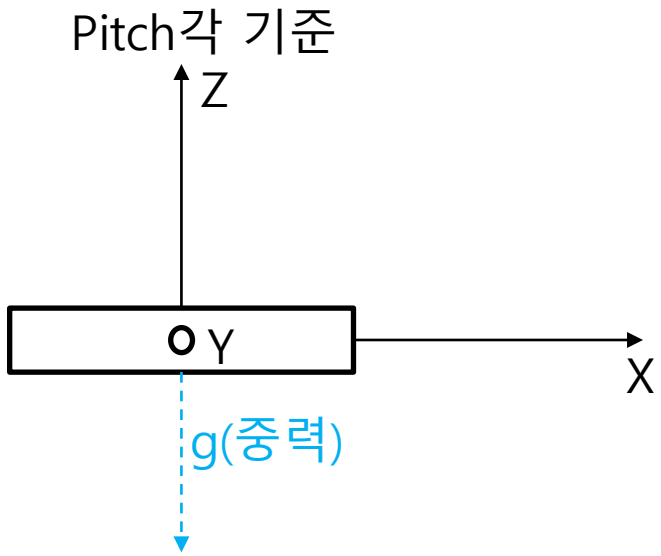
*그림 출처 : https://pinkwink.kr/73

* 이때 Roll각도는 다음과 같다
- $\Theta = \tan^{-1}\frac{Y}{Z}$ -> Euler각
- Y : Y축 가속도센서 출력
- Z : Z축 가속도센서 출력
- Y': 초기 Y축의 위치
- Z': 초기 Z축의 위치

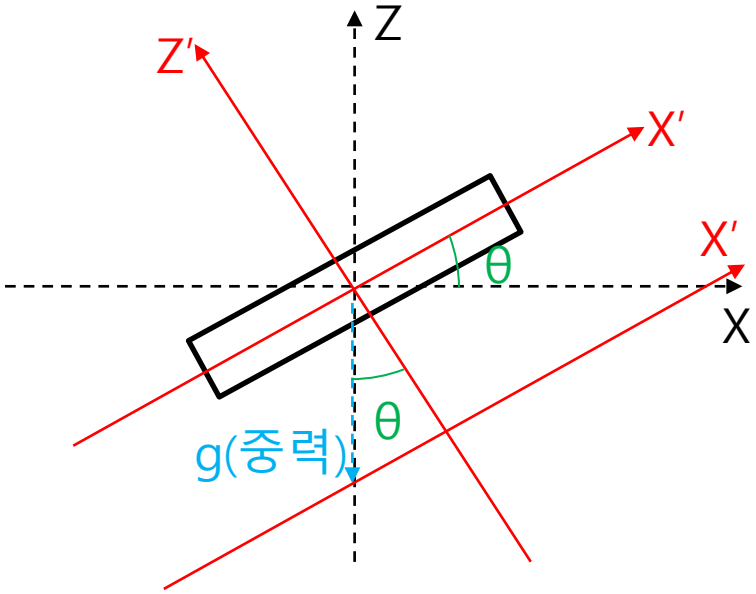
* 가속도 센서로는 yaw각도를 측정 할 수 없음..
- 기준 축이 바뀌므로....

(다음 장에서 계속)

MPU6050



- 가속도 센서가 Z축을 기준으로 가만히 있을 때
- 센서에 작용하는 가속도는 중력가속도뿐
- 이 때 Z축 값으로 1G가 측정된다.
- Z축을 제외한 X,Y축의 가속도 값은 0이 된다.



- 옆의 그림과 같이 센서가 기울었다면,
- Z축의 가속도 크기 = |g|cos(θ)
- X축의 가속도 크기 = |g|sin (θ)
- 두 축의 합은 1g(9.8m/s^2)이 된다.

➢ 자이로 센서란?

- 회전하는 물체의 각속도를 측정
- 자이로 센서의 출력은 "각속도"
- 자이로 센서의 원리

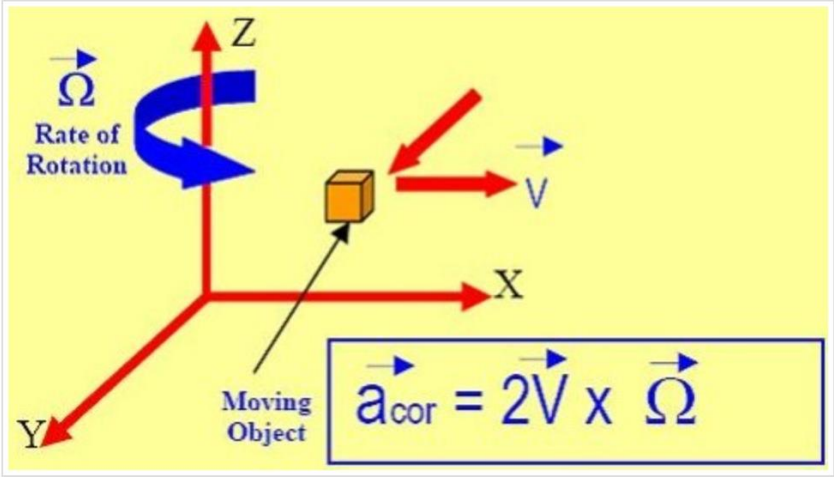


*그림 출처 : https://ibmhdd.tistory.com/3

*기본 원리 : 뉴턴의 제 2법칙(F=ma), 코리올리 효과
- a_{cor} : 코리올리 가속도
- Z축으로 회전시 물체가 V의 속도로 X축으로 이동하면, 코리올리 힘에 의해 코리올리 가속도는 Y축으로 발생한다.
- Ω : 회전계의 각속도
- F=ma에 의해 $F_{cor} = m2V\Omega$에서 2, m, V가 일정하므로 코리올리 힘은 각속도와 비례

* 코리올리 효과란?
- 회전하는 계에서 느껴지는 "관성력 "

● MPU6050

➢ 자이로 센서란?

- 단위 : °/s(DPS : Degree Per Second)이며, x, y, z축에 대한 각속도
- 각속도 $\omega = \frac{d\theta}{dt}$
- 자이로센서로 부터 각도를 얻기 위해서 센서출력을 "적분"하여 각도 값을 얻어야 한다.

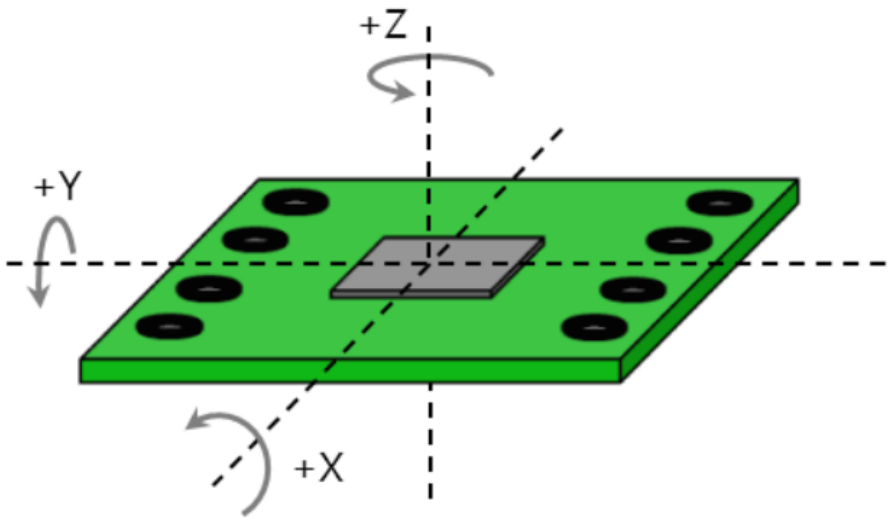
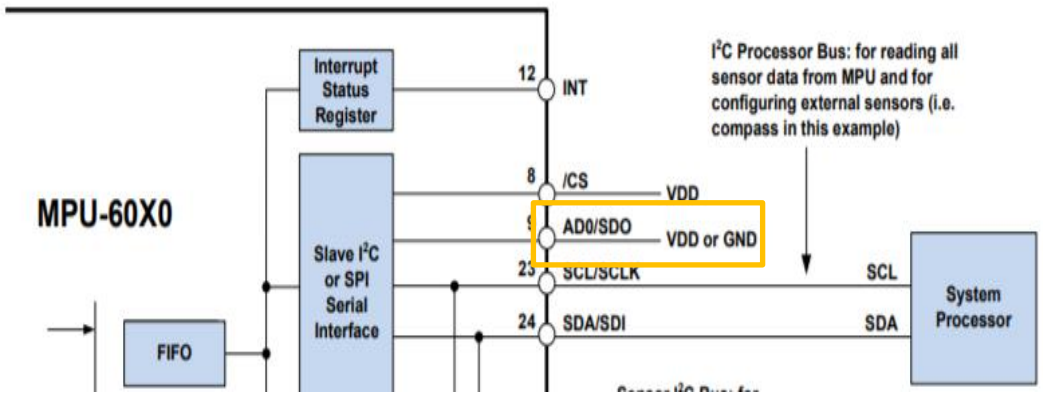


*그림 출처 : https://m.blog.naver.com/PostView.nhn?blogId=lagrange0115&logNo=220767476955&proxyReferer=https:%2F%2Fwww.google.com%2F

● MPU6050

➢ MPU6050 I2C 인터페이스

- I2C인터페이스와 프로토콜은 아래와 같다.



* AD0핀에 VDD or GND에 따라 MPU6050의 I2C Address가 다르다.

I²C ADDRESS	AD0 = 0		1101000			
	AD0 = 1		1101001			
DIGITAL INPUTS (SDI/SDA, AD0,						

MPU6050에 데이터 쓰기

Single-Byte Write Sequence

Master	S	AD+W		RA		DATA		P
Slave			ACK		ACK		ACK	

Burst Write Sequence

Master	S	AD+W		RA		DATA		DATA		P
Slave			ACK		ACK		ACK		ACK	

MPU6050으로 부터 데이터 읽기

Single-Byte Read Sequence

Master	S	AD+W		RA		S	AD+R			NACK	P
Slave			ACK		ACK			ACK	DATA		

Burst Read Sequence

Master	S	AD+W		RA		S	AD+R			ACK		NACK	P
Slave			ACK		ACK			ACK	DATA		DATA		

➢ MPU6050 Register Map

- Power Management1 Register

4.28 Register 107 – Power Management 1
PWR_MGMT_1

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
6B	107	DEVICE _RESET	SLEEP	CYCLE	-	TEMP_DIS	CLKSEL[2:0]		

Note: The device will come up in sleep mode upon power-up.

- 우선 센서를 사용하기 위해...
- 데이터시트 9페이지에 센서에 전원이 인가되면 센서는 "슬립모드"로 들어간다.
- 이 슬립모드를 해제하기 위해 Power Management1의 bit6을 0으로 설정해야한다.
- 센서의 샘플링을 위한 클럭소스를 선택
- Bit0~2를 설정, 아래와 같이 클럭소스를 선택할 수 있다.

CLKSEL	Clock Source
0	Internal 8MHz oscillator
1	PLL with X axis gyroscope reference
2	PLL with Y axis gyroscope reference
3	PLL with Z axis gyroscope reference
4	PLL with external 32.768kHz reference
5	PLL with external 19.2MHz reference
6	Reserved
7	Stops the clock and keeps the timing generator in reset

➢ MPU6050 Register Map

- Configuration Register

4.3 Register 26 – Configuration
CONFIG

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1A	26	-	-	EXT_SYNC_SET[2:0]			DLPF_CFG[2:0]		

- 내부 디지털 저역통과 필터 설정을 해줄 수 있다.

The DLPF is configured by *DLPF_CFG*. The accelerometer and gyroscope are filtered according to the value of *DLPF_CFG* as shown in the table below.

DLPF_CFG	Accelerometer (F_s = 1kHz)		Gyroscope		
	Bandwidth (Hz)	Delay (ms)	Bandwidth (Hz)	Delay (ms)	Fs (kHz)
0	260	0	256	0.98	8
1	184	2.0	188	1.9	1
2	94	3.0	98	2.8	1
3	44	4.9	42	4.8	1
4	21	8.5	20	8.3	1
5	10	13.8	10	13.4	1
6	5	19.0	5	18.6	1
7	RESERVED		RESERVED		8

- 8가지의 설정을 할 수 있고, 0을 선택하면 DLPF를 Disable하는 것

➢ MPU6050 Register Map

- Sample Rate Divider Register

4.2 Register 25 – Sample Rate Divider
SMPRT_DIV

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
19	25	SMPLRT_DIV[7:0]							

- 센서 샘플링 속도를 설정해 줄 수 있다.
- 8bit인 0~255의 값으로 아래와 같이 설정할 수 있다.

The Sample Rate is generated by dividing the gyroscope output rate by *SMPLRT_DIV*:

Sample Rate = Gyroscope Output Rate / (1 + *SMPLRT_DIV*)

where Gyroscope Output Rate = 8kHz when the DLPF is disabled (*DLPF_CFG* = 0 or 7), and 1kHz when the DLPF is enabled (see Register 26).

Note: The accelerometer output rate is 1kHz. This means that for a Sample Rate greater than 1kHz, the same accelerometer sample may be output to the FIFO, DMP, and sensor registers more than once.

➢ MPU6050 Register Map

- Gyroscope Configuration Register

4.4 Register 27 – Gyroscope Configuration
GYRO_CONFIG

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1B	27	XG_ST	YG_ST	ZG_ST	FS_SEL[1:0]		-	-	-

- 센서의 스케일 범위를 아래와 같이 설정한다.

FS_SEL selects the full scale range of the gyroscope outputs according to the following table.

FS_SEL	Full Scale Range
0	± 250 °/s
1	± 500 °/s
2	± 1000 °/s
3	± 2000 °/s

➢ MPU6050 Register Map

- Accelerometer Configuration Register

4.5 Register 28 – Accelerometer Configuration
ACCEL_CONFIG

Type: Read/Write

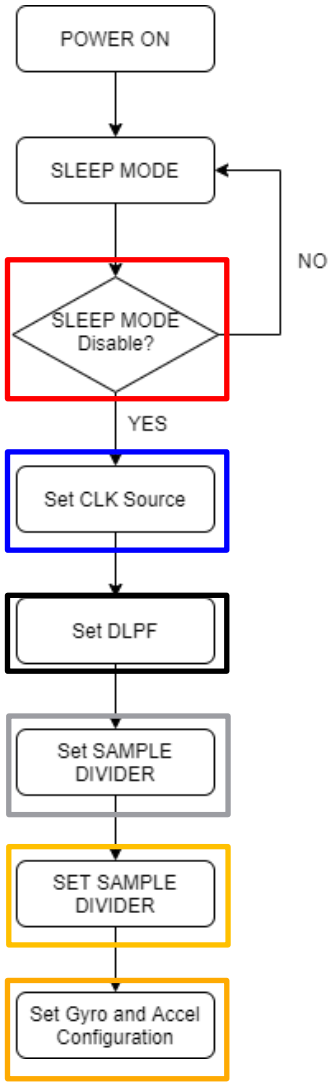
Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1C	28	XA_ST	YA_ST	ZA_ST	AFS_SEL[1:0]		-		

- 센서의 스케일 범위를 아래와 같이 설정한다.

AFS_SEL selects the full scale range of the accelerometer outputs according to the following table.

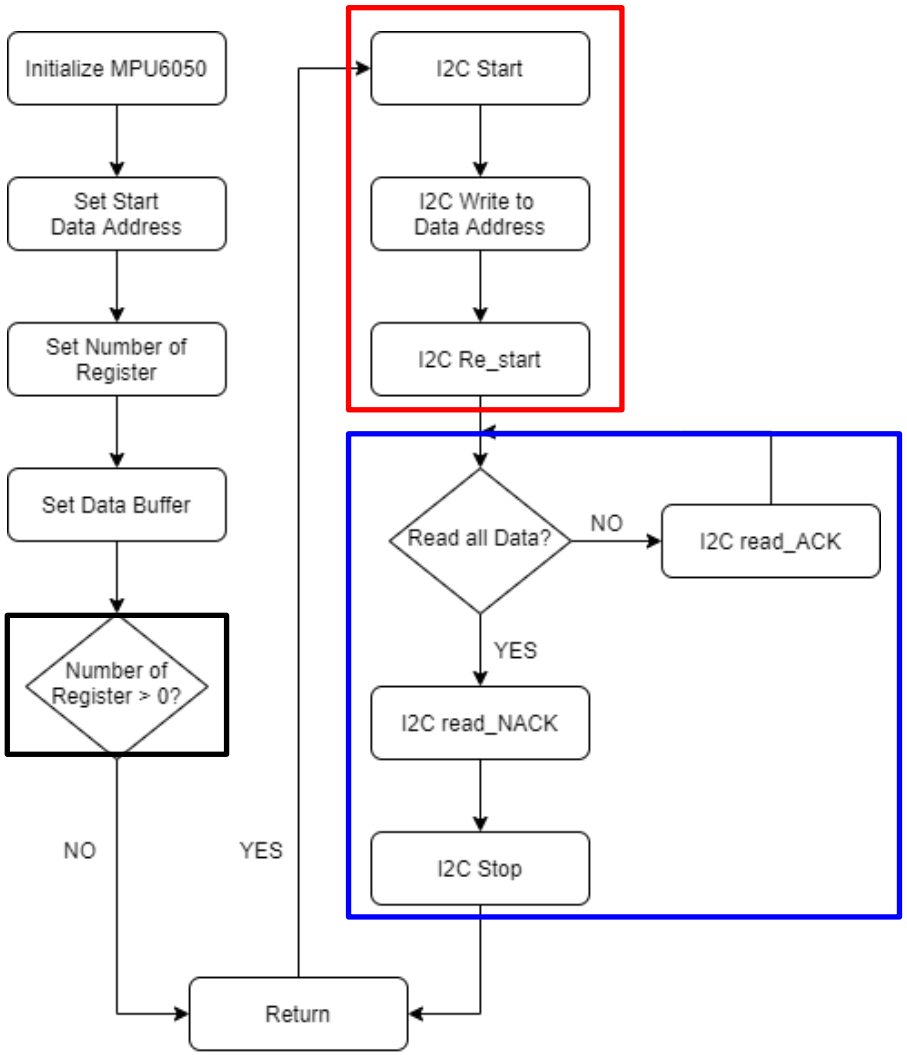
AFS_SEL	Full Scale Range
0	± 2*g*
1	± 4*g*
2	± 8*g*
3	± 16*g*

➢ Initialize MPU6050




```
//Register : PWR management1, Set Bit : SLEEP MODE, Set SLEEP MODE Bit length : 1, Disable SLEEP MODE
MPU6050_WriteBits(PWR_MGMT_1, SLEEP_MODE_BIT, SLEEP_MODE_LEN, DISABLE);

//Register : PWR management1, Set Bit : CLK Select, Set CLK Select length : 3, Set CLK source : PLL with X axis gyroscope reference
MPU6050_WriteBits(PWR_MGMT_1, CLKSEL_BIT, CLKSEL_LEN, CLKSEL1);

//Register : Interrupt Enable, Set Bit : DATA_RDY_INT_EN, Set DATA_RDY_INT_EN length : 1, Enable DATA_RDY_INT
//MPU6050_WriteBits(INT_ENABLE, DATA_RDY_INT_EN, DATA_RDY_INT_EN_LEN, ENABLE);

//Register : Configuration, Set Bit : Digital LowPassFilter Config, Set DLPF Bit length : 3,
//Set DLPF config3 : Accele = 44Hz(Bandwidth), 4.9ms(Delay), Fs = 1KHz / Gyro = 42Hz(Bandwidth), 4.8ms(Delay), Fs = 1KHz -> 2020.12.07
//Set DLPF config0 : Accele = 260Hz(Bandwidth), 0ms(Delay), Fs = 1KHz / Gyro = 256Hz(Bandwidth), 0.98ms(Delay), Fs = 8KHz -> 2020.12.28
MPU6050_WriteBits(CONFIG, DLPF_CFG_BIT, DLPF_CFG_LEN, DLPF_CFG0);

//Register : Sample Rate, Writes 0x04 : Sample Rate = Gyroscope Output Rate / (1 + SMPLRT_DIV(=4))
//where Gyroscope Output Rate = 8kHz when the DLPF is disabled (DLPF_CFG = 0 or 7), and 1kHz
//when the DLPF is enabled
MPU6050_WriteByte(SMPLRT_DIV, 4);

//Register : Gyro Configuration, Set Bit : Gyro Full Scale, Set Bit length : 2, Set Gyro Full Scale : +,-2000 degree/s
MPU6050_WriteBits(GYRO_CONFIG, GYRO_FS_SEL_BIT, GYRO_FS_SEL_LEN, GYRO_FS_SEL2000);

//Register : Accelerometer Configuration, Set Bit : Accelerometer Full Scale, Set Bit length : 2, Set Accel Full Scale : +,-16g
MPU6050_WriteBits(ACCEL_CONFIG, ACCEL_AFS_SEL_BIT, ACCEL_AFS_SEL_LEN, ACCEL_AFS_SEL16G);
```

➢ Get MPU6050 Raw Data




```
uint8_t MPU6050_ReadBytes(uint8_t reg_addr, uint8_t len, uint8_t* pData)
{
    uint8_t cnt = 0;
    if(len > 0)
    {
        i2c_start((MPU6050_ADD0 << 1) | I2C_WRITE);
        i2c_write(reg_addr);
        i2c_rep_start((MPU6050_ADD0 << 1) | I2C_READ);

        for(int i = 0; i < len; i++)
        {
            cnt++;
            if(i == len-1)
                pData[i] = i2c_readNak();
            else
                pData[i] = i2c_readAck();
        }
    }
    i2c_stop();
    return cnt;
}
```

* reg_addr : Accel_Xout_H = 0x3B
* len : 14 = 0x3B ~ 0x48(Accel_Xout_H ~ Gyro_Zout_L)

● MPU6050

➢ Scaling Raw MPU6050 Data

```
void Calc_Accel_RollPitch(pAxis_Data AccelData, pAngle_3Dim Accel_Angle)
{
    Angle_3Dim Angle_Tmp;

    Angle_Tmp.Roll = (float)AccelData -> Xaxis / (float)ACCEL_LSB_FS16G;
    Angle_Tmp.Pitch = (float)AccelData -> Yaxis / (float)ACCEL_LSB_FS16G;
    Angle_Tmp.Yaw = (float)AccelData -> Zaxis / (float)ACCEL_LSB_FS16G;

    Accel_Angle->Roll = atan2(Angle_Tmp.Pitch, Angle_Tmp.Yaw) * 180.0F / (float)PI;
    Accel_Angle->Pitch = atan2(Angle_Tmp.Roll, Angle_Tmp.Yaw) * 180.0F / (float)PI;

    #ifdef ForTestMPU6050
    printf("Roll = %.2f\tPitch = %.2f\n", Accel_Angle->Roll, Accel_Angle->Pitch);
    #endif
}

void Calc_GyroData(pAxis_Data GyroData, pAngle_3Dim Gyro_Angle)
{
    Gyro_Angle->Roll = (float)GyroData->Xaxis / (float)GYRO_LSB_FS2000;
    Gyro_Angle->Pitch = (float)GyroData->Yaxis / (float)GYRO_LSB_FS2000;
    Gyro_Angle->Yaw = (float)GyroData->Zaxis / (float)GYRO_LSB_FS2000;
}
```

● MPU6050

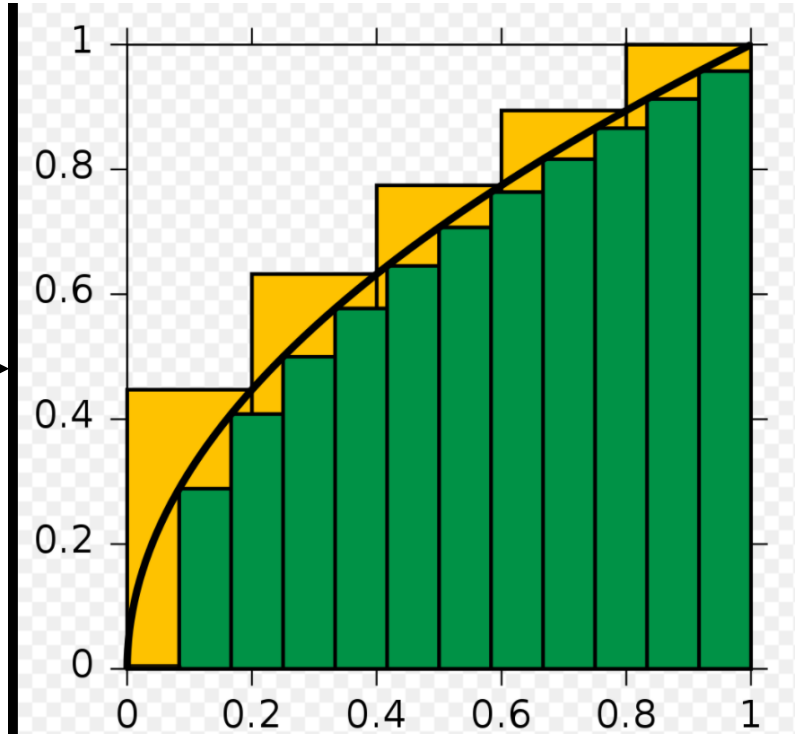
➢ Scaling Raw MPU6050 Data

```
void Calc_Accel_RollPitch(pAxis_Data AccelData, pAngle_3Dim Accel_Angle)
{
    Angle_3Dim Angle_Tmp;

    Angle_Tmp.Roll = (float)AccelData -> Xaxis / (float)ACCEL_LSB_FS16G;
    Angle_Tmp.Pitch = (float)AccelData -> Yaxis / (float)ACCEL_LSB_FS16G;
    Angle_Tmp.Yaw = (float)AccelData -> Zaxis / (float)ACCEL_LSB_FS16G;

    Accel_Angle->Roll = atan2(Angle_Tmp.Pitch, Angle_Tmp.Yaw) * 180.0F / (float)PI;
    Accel_Angle->Pitch = atan2(Angle_Tmp.Roll, Angle_Tmp.Yaw) * 180.0F / (float)PI;

    #ifdef ForTestMPU6050
    printf("Roll = %.2f\tPitch = %.2f\n", Accel_Angle->Roll, Accel_Angle->Pitch);
    #endif
}

void Calc_GyroData(pAxis_Data GyroData, pAngle_3Dim Gyro_Angle)
{
    Gyro_Angle->Roll = (float)GyroData->Xaxis / (float)GYRO_LSB_FS2000;
    Gyro_Angle->Pitch = (float)GyroData->Yaxis / (float)GYRO_LSB_FS2000;
    Gyro_Angle->Yaw = (float)GyroData->Zaxis / (float)GYRO_LSB_FS2000;
}
```

➢ Integral Gyro Data

```
void Integral_GyroData(pAxis_Data GyroData, pAngle_3Dim Gyro_Angle)
{
    Angle_3Dim Angle_Tmp;

    Angle_Tmp.Roll = (float)GyroData->Xaxis / (float)GYRO_LSB_FS2000;
    Angle_Tmp.Pitch = (float)GyroData->Yaxis / (float)GYRO_LSB_FS2000;
    Angle_Tmp.Yaw = (float)GyroData->Zaxis / (float)GYRO_LSB_FS2000;

    Gyro_Angle->Roll = Angle_Tmp.Roll*(float)dt + Gyro_Angle->Roll;
    Gyro_Angle->Pitch = Angle_Tmp.Pitch*(float)dt + Gyro_Angle->Pitch;
    Gyro_Angle->Yaw = Angle_Tmp.Yaw*(float)dt + Gyro_Angle->Yaw;
```

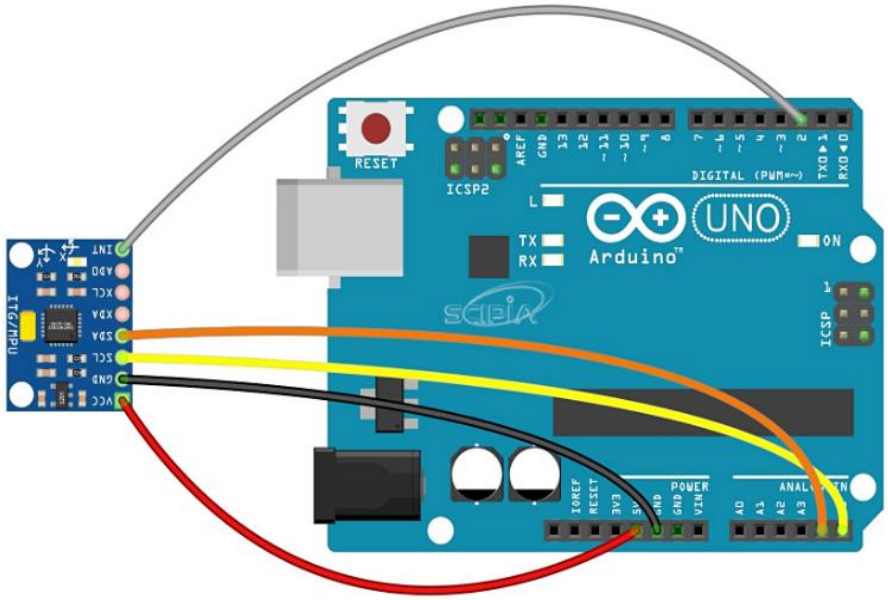
*그림 출처 : By I, KSmrq, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=2359630

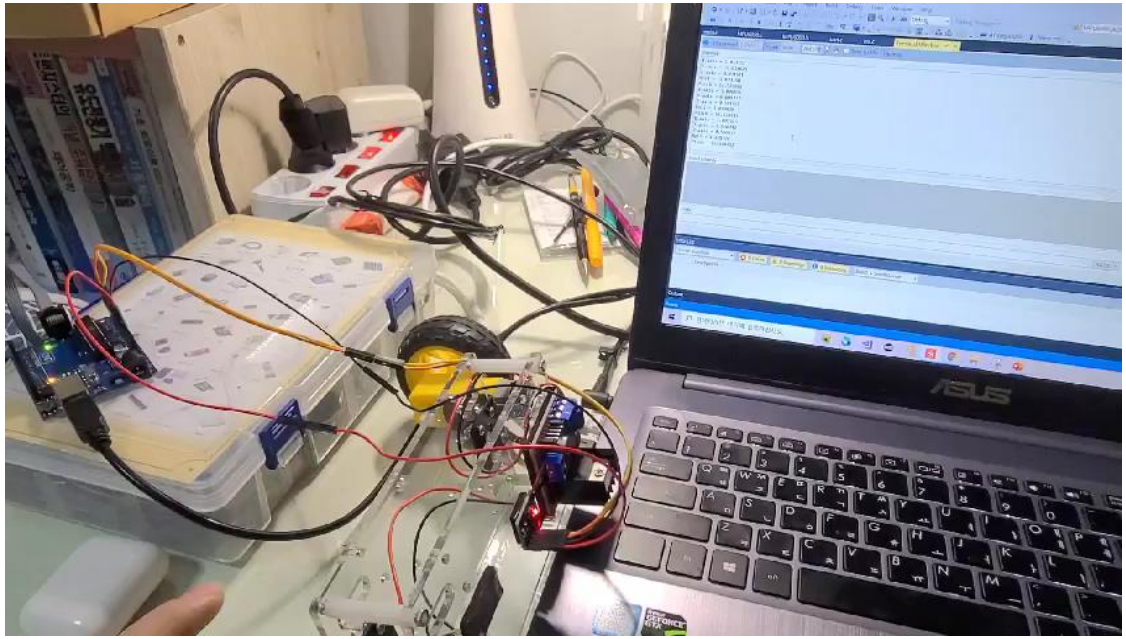


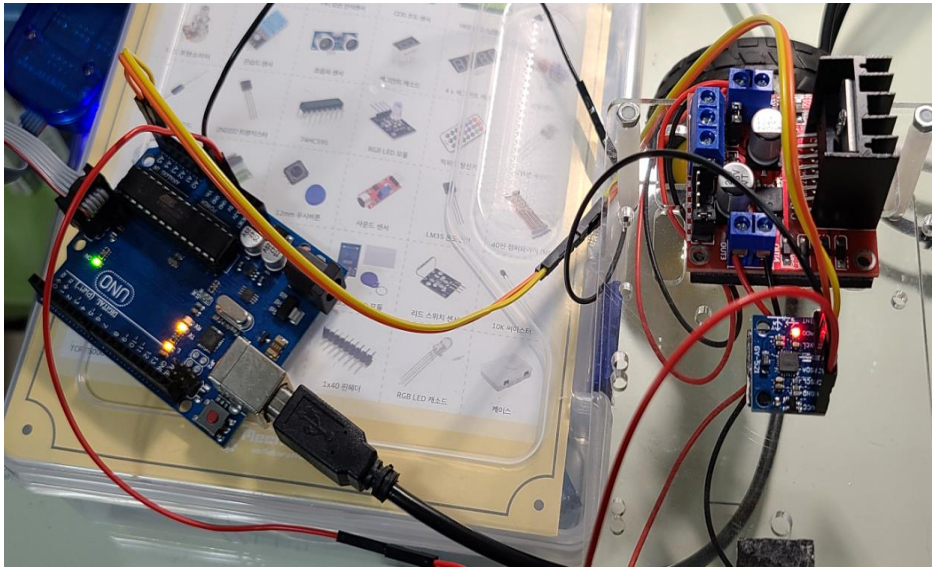
* 적분구간 = 1ms(제어주기=센서 데이터 획득주기)

● MPU6050

➢ 센서 테스트





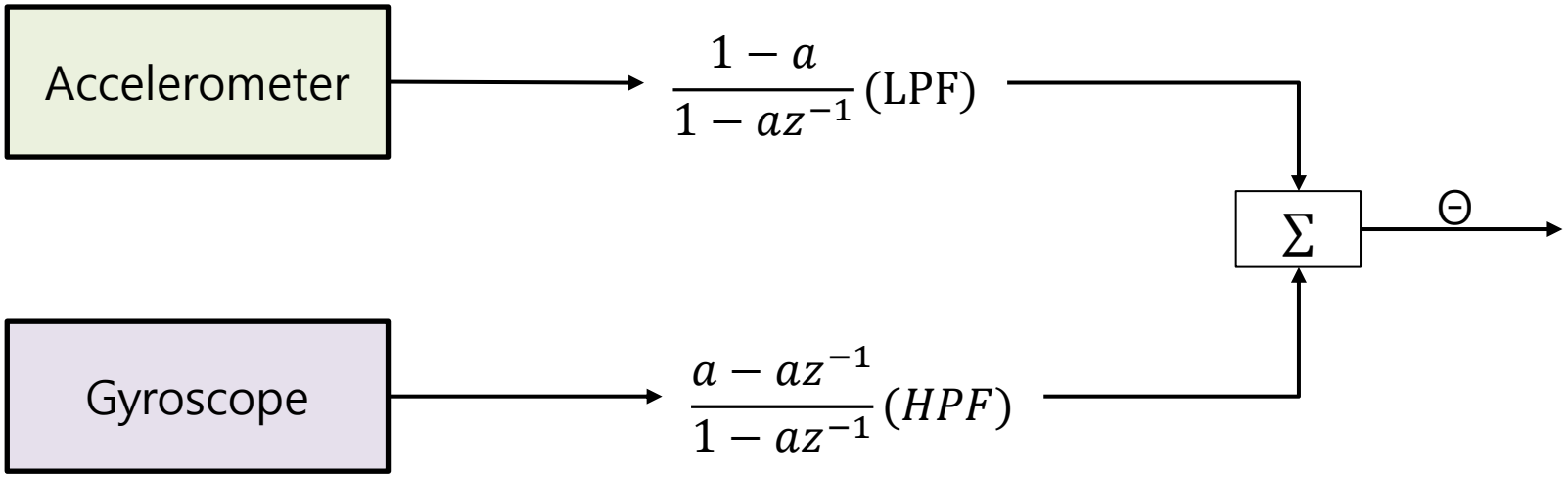


```
X-axis = 0.006836
Y-axis = 0.026855
Z-axis = 1.304199
Roll = 1.179641
Pitch = 0.300312
X-axis = 0.004883
Y-axis = 0.021484
Z-axis = 1.307617
Roll = 0.941295
Pitch = 0.213949
X-axis = 0.006348
Y-axis = 0.024414
Z-axis = 1.307617
Roll = 1.069625
Pitch = 0.278133
```

● 상보필터

➢ **상보필터란?**

- 가속도센서 단점 : 진동과 이동시에 발생하는 고주파 노이즈에 취약
- 자이로 센서 단점 : 자이로센서를 이용한 각도추정시 적분으로 인한 누적오차가 발생,
  각도 추정의 저주파성분의 드리프트 현상이 발생
- 가속도 센서는 고주파 노이즈를 차단하는 "Low Pass Filter"를 자이로센서는 저주파 노이즈를 차단하는 "High Pass Filter"를 적용하여 필터링된 결과 값을 더하여 서로 단점을 보완해주는 필터가 "상보 필터 " 이다.



● 상보필터

➢ 상보필터 구현

1. 1차 상보 필터

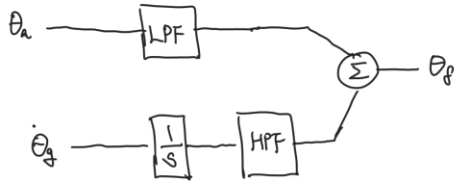
(1) 상보필터란?

① 가속도 센서와 자이로 센서의 센서값을 보정하기 위한 필터

② 가속도 센서는 긴시간(낮은 주파수 대역)에 걸친 데이터가 정확하다.

③ 자이로 센서는 짧은시간(높은 주파수 대역)에 걸친 데이터가 정확하다.

(2) 상보필터의 구성



- θ_f : 필터링된 각도 데이터
- θ_a = 가속도 센서의 각도 데이터
- $\dot{\theta}_g$ = 자이로 센서의 각가속도

① LPF의 전달함수

- $G_{LPF}(s) = \frac{1}{s\tau+1}$

② HPF의 전달함수

- $G_{HPF}(s) = \frac{s\tau}{s\tau+1}$

③ $\theta_f = \frac{1}{s\tau+1}\theta_a + \frac{1}{s}\frac{s\tau}{s\tau+1}\dot{\theta}_g$

④ $\theta_f(s\tau+1) = \theta_a + \frac{1}{s}s\tau\dot{\theta}_g$

$\Rightarrow s\tau\theta_f + \theta_f = \theta_a + \frac{1}{s}s\tau\dot{\theta}_g$

$\Rightarrow \theta_f = \theta_a + \tau\dot{\theta}_g - s\tau\theta_f$

⑤ ④식을 역라플라스 변환

$\theta_f(t) = \theta_a(t) + \tau\dot{\theta}_g(t) - \tau\frac{d\theta_f(t)}{dt}$

⑥ ⑤식을 Discrete Time으로 표현하면

- $t \to nT \quad dt = T$

$\to \theta_f[n] = \theta_a[n] + \tau\dot{\theta}_g[n] - \frac{\tau(\theta_f[n]-\theta_f[n-1])}{T}$

$\to \theta_f[n] + \frac{\tau}{T}\theta_f[n] = \theta_a[n] + \tau\dot{\theta}_g[n] + \frac{\tau}{T}\theta_f[n-1]$

$\to \frac{\tau+T}{T}\theta_f[n] = \theta_a[n] + \tau\dot{\theta}_g[n] + \frac{\tau}{T}\theta_f[n-1]$

$\to \theta_f[n] = \frac{T}{\tau+T}\theta_a[n] + \frac{\tau T}{\tau+T}\dot{\theta}_g[n] + \frac{\tau}{\tau+T}\theta_f[n-1]$

여기서 $\frac{\tau}{\tau+T} = \alpha$, $1-\alpha = \frac{T}{\tau+T}$ 이므로

$\theta_f[n] = (1-\alpha)\theta_a[n] + \alpha T\dot{\theta}_g[n] + \alpha\theta_f[n-1] = (1-\alpha)\theta_a[n] + \alpha(\theta_f[n-1] + T\dot{\theta}_g[n])$

($\alpha T \to \frac{1}{s}$)

(2) τ 값 설정

① LPF의 전달 함수 $G_{LPF}(s) = \frac{1}{s\tau+1}$

② $|G_{LPF}(j\omega_c)| = \frac{1}{\sqrt{2}}$ 이 될 때의 ω_c 값이 차단각주파수.

③ $G_{LPF}(j\omega_c) = \frac{1}{j\omega_c\tau+1} \Rightarrow \frac{1-j\omega_c\tau}{(1+j\omega_c\tau)(1-j\omega_c\tau)} = \frac{1-j\omega_c\tau}{1-j^2\omega_c^2\tau^2} = \frac{1-j\omega_c\tau}{1+\omega_c^2\tau^2}$

④ $G_{LPF}(j\omega_c)$를 실수+허수로 나타내 보면

$G_{LPF}(j\omega_c) = \frac{1}{\omega_c^2\tau^2+1} - j\frac{\omega_c\tau}{\omega_c^2\tau^2+1}$

⑤ $|G_{LPF}(j\omega_c)| = \sqrt{Re^2+Im^2} = \sqrt{\frac{1}{(\omega_c^2\tau^2+1)^2} + \frac{\omega_c^2\tau^2}{(\omega_c^2\tau^2+1)^2}} = \sqrt{\frac{\omega_c^2\tau^2+1}{(\omega_c^2\tau^2+1)^2}} = \frac{1}{\sqrt{2}}$

⑥ 양변을 제곱 하면,

$\frac{1}{(\omega_c^2\tau^2+1)^2} + \frac{\omega_c^2\tau^2}{(\omega_c^2\tau^2+1)^2} = \frac{1}{2} \Rightarrow 2(1+\omega_c^2\tau^2) = (\omega_c^2\tau^2+1)^2$

$\Rightarrow 2+2\omega_c^2\tau^2 = \omega_c^4\tau^4 + 2\omega_c^2\tau^2 + 1 \Rightarrow \omega_c^2\tau^2 = 1 \quad \therefore \omega_c\tau = 1$

⑦ $\omega_c \to 2\pi f_c$ 라면

$2\pi f_c\tau = 1$ 이고 $\therefore f_c = \frac{1}{2\pi\tau}$ 이 된다.

(3) 상보필터 계수 $\alpha = \frac{\tau}{\tau+T_s}$ 여기서 f_c = 100헤르 라면,

① $\tau = \frac{1}{2\pi \times 100헤르}$ 이므로 $\tau \fallingdotseq 0.001591549$가 된다.

② T_s가 8ms 라면 $\alpha = \frac{0.001591549}{0.001591549+0.008} \fallingdotseq 0.1659324$

③ $\theta_f[n] = (1-0.1659324)\theta_a[n] + 0.1659324(\theta_f[n-1] + 0.008\cdot\dot{\theta}_g[n])$

● 상보필터

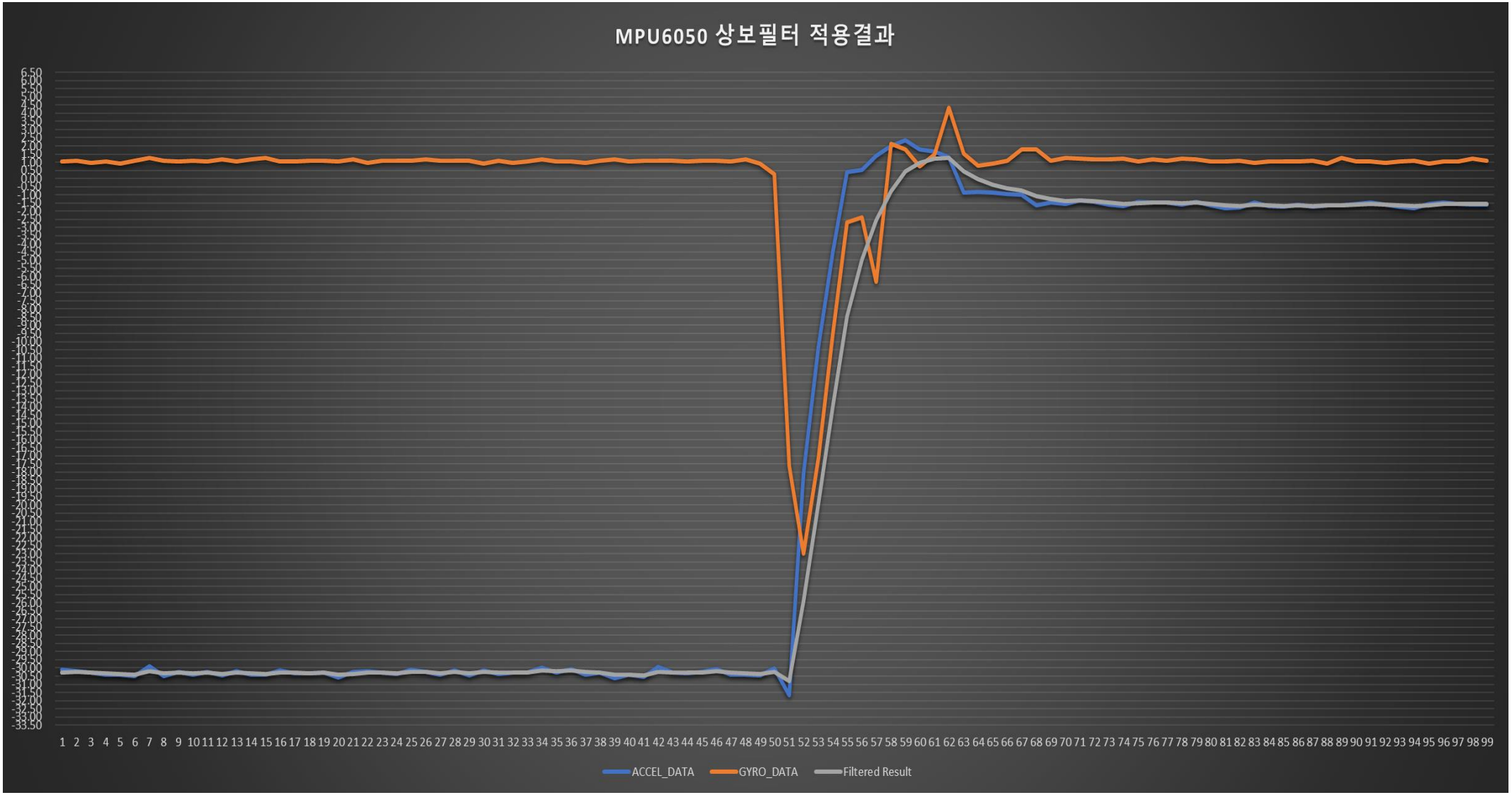
➢ 상보필터 구현 - 코드

```
Angle_3Dim CompleFilter(pAngle_3Dim AccelAngle, pAngle_3Dim GyroAngle)
{
    Filter_Angle.Roll = (1.0F-(float)alpha)*AccelAngle->Roll + (float)alpha*(preFilter_Angle.Roll + (float)dt * GyroAngle->Roll);
    Filter_Angle.Pitch = (1.0F-(float)alpha)*AccelAngle->Pitch + (float)alpha*(preFilter_Angle.Pitch + (float)dt * GyroAngle->Pitch);
    Filter_Angle.Yaw = (1.0F-(float)alpha)*AccelAngle->Yaw + (float)alpha*(preFilter_Angle.Yaw + (float)dt * GyroAngle->Yaw);

    preFilter_Angle = Filter_Angle;

    return Filter_Angle;
}
```

```
//#define dt 0.001F //Ts
//#define alpha 0.1659324 //alpha = tau/(tau+Ts) @fc(cut-off Frequency) = 100Hz

//2021.01.20
//Set Sampling Time = 1ms
//#define alpha 0.6141304F//tau = 1/(2*pi*fc), Ts = 1ms, alpha = tau/(tau+Ts)

//2021.03.06
//Set Sampling Time = 500us
//#define alpha 0.760942776

//2021.03.16
//fc = 2KHz
#define alpha 0.073711F
```

● 상보필터

➢ 상보필터 적용결과



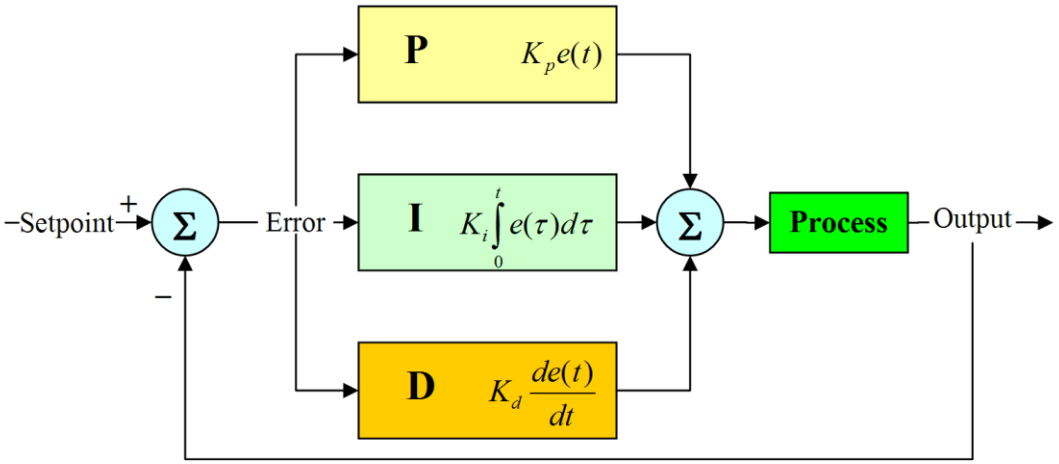
● 제어기 설계

➢ **PID 제어란?**

- Feedback 제어의 형태
- 비례(Proportional), 적분(Integral), 미분(Differential)를 의미
- 목표 값(또는 설정 값)과 제어하고자 하는 대상의 출력 값을 비교하여 오차를 계산
- 오차를 비례, 적분, 미분 연산을 통해 제어량을 조절한다.

$$u(t) = K_P e(t) + K_I \int e(t)dt + K_D \frac{de(t)}{dt}$$

*u(t) : 제어량, e(t) : 목표 값 – 출력 값



* 미분제어 : $K_D \frac{de(t)}{dt}$ 이지만, e(t)가 목표 값 – 출력 값으로
목표 값이 변하지 않는 상황이면 미분제어의 수식은 다음과 같다

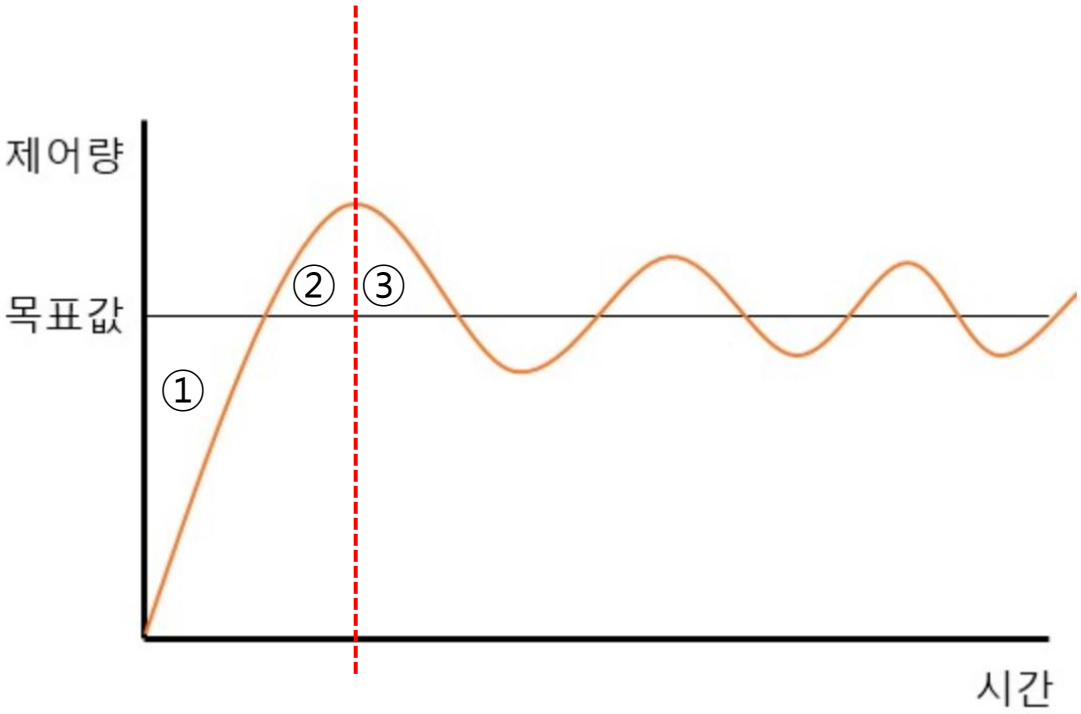
* 미분제어 = $-K_D \frac{dout(t)}{dt}$

*그림 출처 : By Arturo Urquizo - http://commons.wikimedia.org/wiki/File:PID.svg, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=17633925

● 제어기 설계

➢ PID 제어 동작원리

- 비례제어(P제어) : 오차에 비례하여 제어량이 변함, 응답속도를 높일 수 있지만 오버슈트, 정상상태 오차가 발생
- 적분제어(I제어) : 정상상태 오차를 없애고, 비례제어와 같이 응답속도를 높일 수 있지만,
  오차를 계속 누적하여 정상상태에 도달하여도 제어량이 줄지 않는 와인드업 현상 발생(Wind-up).
- 미분제어(D제어) : 오차의 급격한 변화량에 비례하여 제어량을 조절하며, 오버슛을 줄여준다.



*구간별 적분/미분제어 동작

- ①구간
  → 적분제어는 오차가 (+)가 되어 제어량을 늘려준다
  → 미분제어는 오차의 변화량이 증가하는 구간으로 제어량을 줄여준다

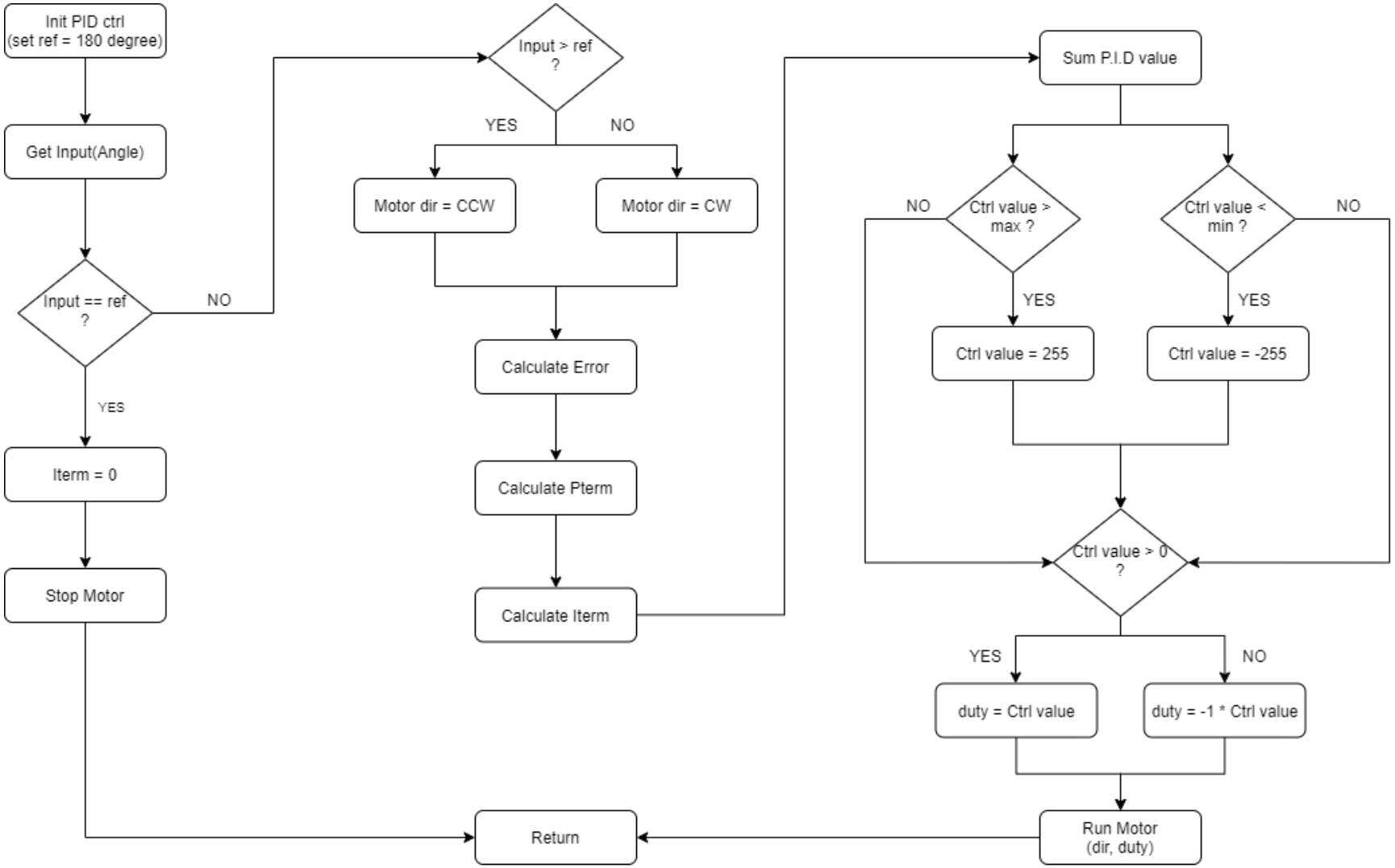
- ②구간
  → 적분제어는 오차가 (-)가 되어 제어량을 줄여준다
  → 미분제어는 오차의 변화량이 증가하는 구간으로 제어량을 줄여준다

- ③구간
  → 적분제어는 오차가 (-)가 되어 제어량을 줄여준다
  → 미분제어는 오차의 변화량이 줄어드는 구간으로 제어량을 늘려준다

$$u(t) = K_P e(t) + K_I \int e(t)dt - K_D \frac{dout(t)}{dt}$$

● 제어기 설계

➢ PID 제어 코드1



● 제어기 설계

➢ PID 제어 코드2

```
void RunPID_MotorCtrl(Angle_3Dim* AngleData)
{
    //0도를 180으로 변환하여 연산
    Input = AngleData->Pitch+180.0;
    if(Input == (float)ref)
    {
        Iterm = 0;
        MotorStop();
    }
    else
    {
        dir = (Input > (float)ref) ? (uint8_t)CCW : (uint8_t)CW;

        Cal_PID();

        if(u >= (float)Limit_Max) u = (float)Limit_Max;
        if(u <= (float)Limit_Min) u = (float)Limit_Min;

        //입력>기준값일 때 제어량은 음수이므로 양수로 변환하여 OCR값에 대입
        if(u < 0) u = -1 * u;

        //DeadBand 추가
        //if(Input >= (float)DeadBand_Min && Input <= (float)DeadBand_Max) u = 0;

        duty = (uint8_t)u;

        RunMotor(duty, dir);
    }

}
```

```
void Cal_PID(void)
{
    float dInput = 0;

    cur_err = (float)ref - Input;
    dInput = Input - preInput;

    Pterm = (float)Kp*cur_err;
    Iterm += cur_err*(float)dt;
    Dterm = dInput/(float)dt;

    //if(cur_err == 0) Iterm = 0.0F;

    preInput = Input;

    u = Pterm+(float)Ki*Iterm-(float)Kd*Dterm;
}
```

* 미분 = 차분방정식(후방 차분법)

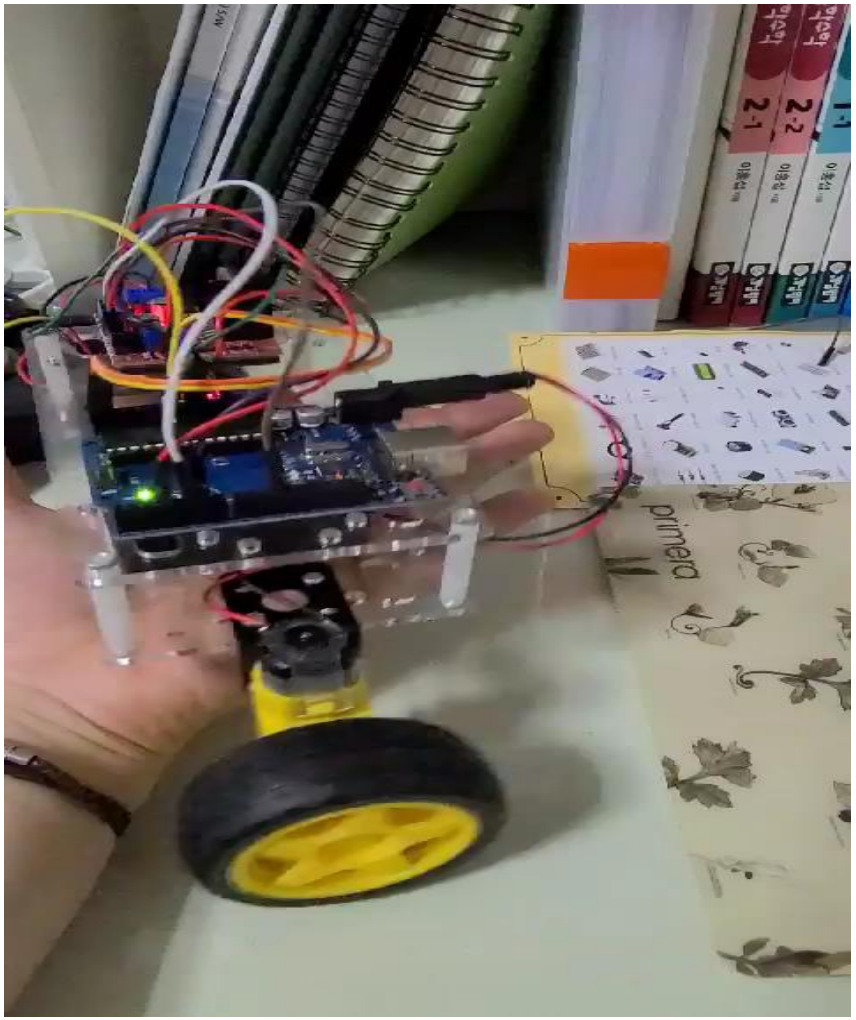
→$y'[n] = \frac{x[n]-x[n-1]}{T}$

* 적분 = 구분구적법(0 order hold)

→y[n] = y[n-1] + x[n]*T

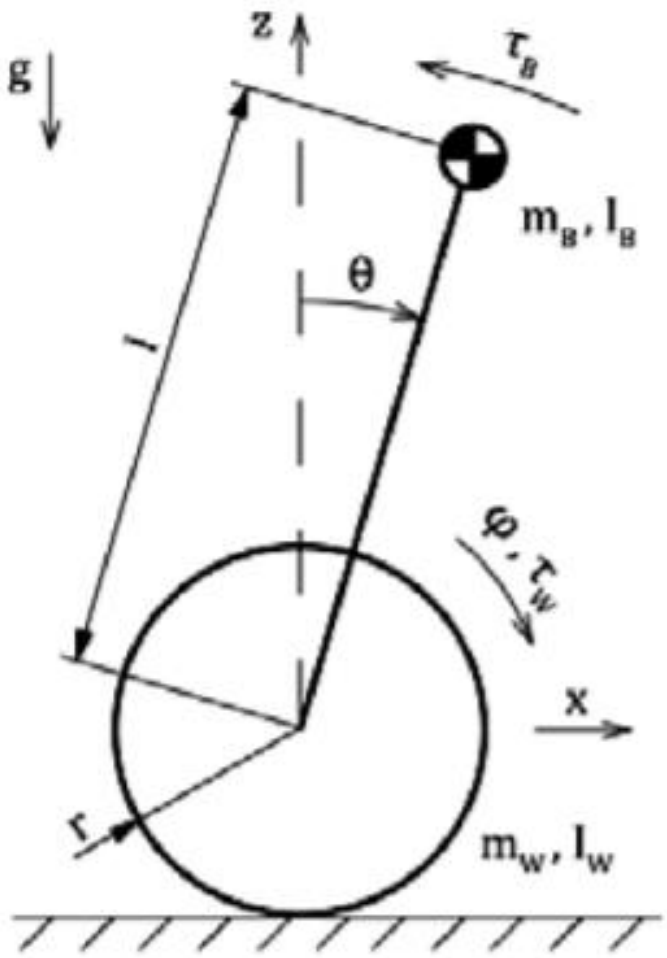
● 구동영상

➢ **밸런싱 로봇 구동영상**



3. 동역학 모델링

(1) 동역학 모델링

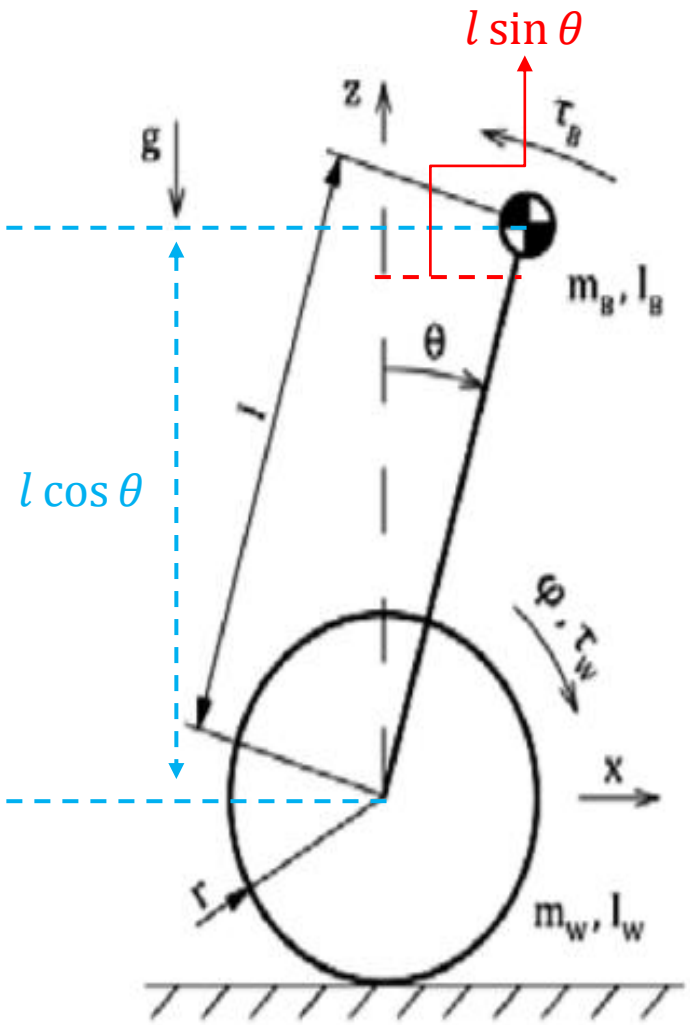


※기호 정리

- x : 로봇의 이동거리
- θ : 몸체의 각도
- φ : 바퀴의 회전각
- τ_B : 몸체의 회전토크
- τ_W : 바퀴의 회전토크
- m_W: 바퀴의 질량
- m_B: 몸체의 질량
- I_W: 바퀴의 관성모멘트
- I_B: 몸체의 관성모멘트
- l : 무게 중심간 거리
- r : 바퀴의 반지름

3. 동역학 모델링

(1) 동역학 모델링-로봇



(1) $x = (r\varphi)\vec{i}$: 바퀴의 X축 위치벡터(바퀴의 Y축 위치벡터는 작으므로 무시)

(2) $\dot{x} = (r\dot{\varphi})\vec{i}$: 바퀴의 X축 속도벡터

(3) $\vec{x}_{COG} = (x + l\sin\theta)\vec{i}$: 로봇무게중심의 X축 위치벡터

(4) $\vec{z}_{COG} = (l\cos\theta)\vec{j}$: 로봇무게중심의 Y축 위치벡터

(5) $\dot{x}_{COG} = (\dot{x} + l\dot{\theta}\cos\theta)\vec{i}$: X축 속도벡터

(6) $\dot{z}_{COG} = (-l\dot{\theta}\sin\theta)\vec{j}$: Y축 속도벡터

(7) $v_{COG} = \sqrt{\dot{x}_{COG}^{\ 2} + \dot{z}_{COG}^{\ 2}}$: 로봇의 속력

(8) $E_{kB} = \frac{1}{2}m_B v_{COG}^{\ 2} + \frac{1}{2}I_B\dot{\theta}^2 = \frac{1}{2}m_B(\dot{x}^2 + 2\dot{x}l\cos\theta\dot{\theta} + l^2\dot{\theta}^2) + \frac{1}{2}I_B\dot{\theta}^2$:로봇몸체의 운동에너지

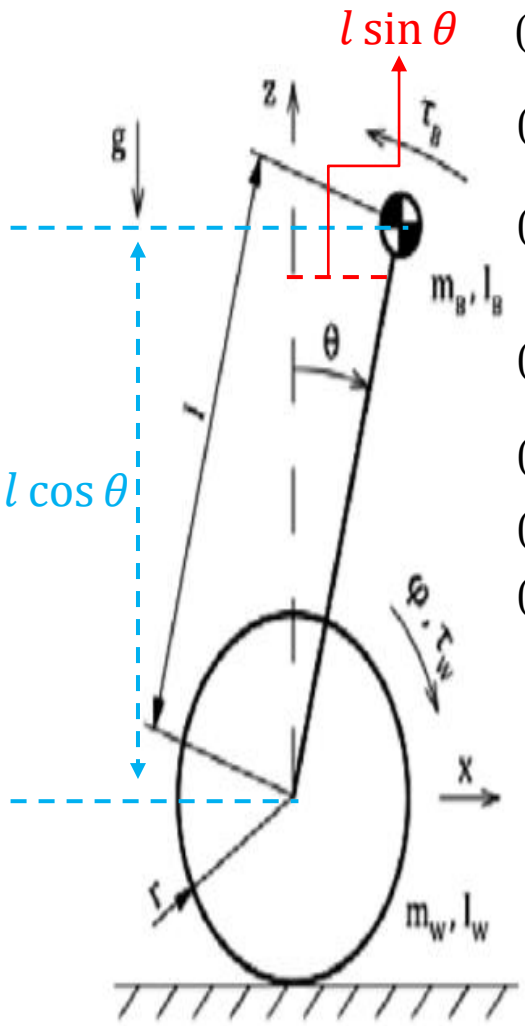
(9) $E_{kW} = \frac{1}{2}m_W\dot{x}^2 + \frac{1}{2}I_W\dot{\varphi}^2 = \frac{1}{2}m_W\dot{x}^2 + \frac{1}{2}\frac{I_W}{r^2}\dot{x}^2$:로봇바퀴의 운동에너지

(10) $E_{pB} = m_B gl\cos\theta$: 로봇 몸체의 위치에너지

(11) $E_{pW} = 0$: 로봇 바퀴의 위치에너지

3. 동역학 모델링

(1) 동역학 모델링-로봇



(12) $D = 2\frac{1}{2}b_W\dot{\varphi}^2 = \frac{b_W}{r^2}\dot{x}^2$: 바퀴마찰에 의한 에너지 소실(*Rayleigh's dissipation function*)

(13) $L = E_k - E_p = E_{kB} + 2E_{kW} - (E_{pB} - 2E_{pW})$: 라그랑지안(2를 곱한이유는 바퀴가 2개라..)

(14) $\frac{d}{dt}(\frac{dL}{d\dot{q}_{x,\theta}}) - (\frac{dL}{dq_{x,\theta}}) = Q_{x,\theta} - (\frac{dD}{d\dot{q}_{x,\theta}})$: 라그랑지방정식

(15) $(m_B + 2m_W + 2\frac{I_W}{r^2})\ddot{x} + 2\frac{b_W}{r^2}\dot{x} + m_B l\cos\theta\ddot{\theta} - m_B l\sin\theta\dot{\theta}^2 = \frac{\tau_R + \tau_L}{r^2}$: q가 x일 때, 오른쪽/왼쪽 바퀴의 토크

(16) $m_B l\cos\theta\ddot{x} + (m_B l^2 + I_B)\ddot{\theta} - m_B gl\sin\theta = -(\tau_R + \tau_L)$: q가 θ일 때

(17) 선형화를 위해 비선형 함수를 근사화하면, $\cos\theta \approx l,\ \sin\theta \approx \theta, \dot{\theta}^2 \approx 0$

(18) 식(17)에 의해 선형화된 동역학식은

$$(m_B + 2m_W + 2\frac{I_W}{r^2})\ddot{x} + 2\frac{b_W}{r^2}\dot{x} + m_B l^2\ddot{\theta} = \frac{\tau_R + \tau_L}{r^2},$$

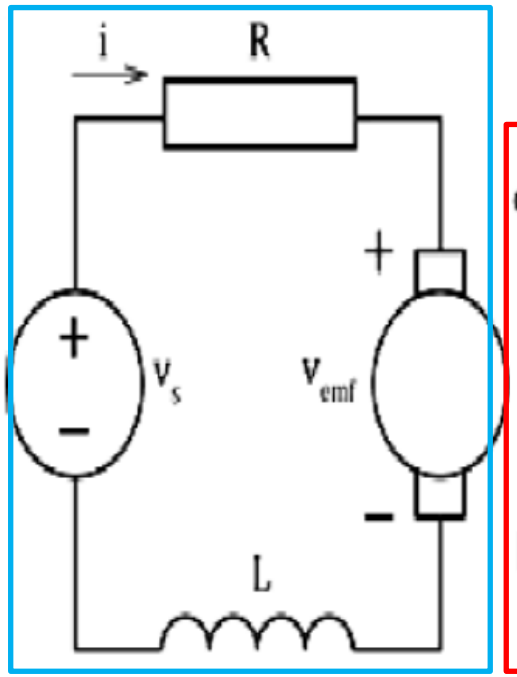
$$m_B l^2\ddot{x} + (m_B l^2 + I_B)\ddot{\theta} - m_B gl\theta = -(\tau_R + \tau_L)$$

다음장에선 라그랑지 방정식의 입력인 우변의 토크를 발생시키는 모터의 모델링과 최종 동역학 모델링에 대해 알아보자..

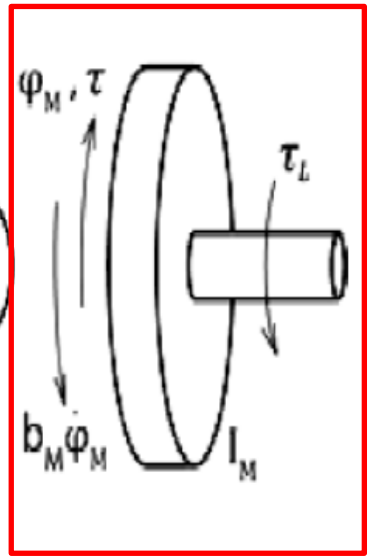
3. 동역학 모델링

(2) 동역학 모델링-DC모터

모터의 등가회로



모터의 기계부



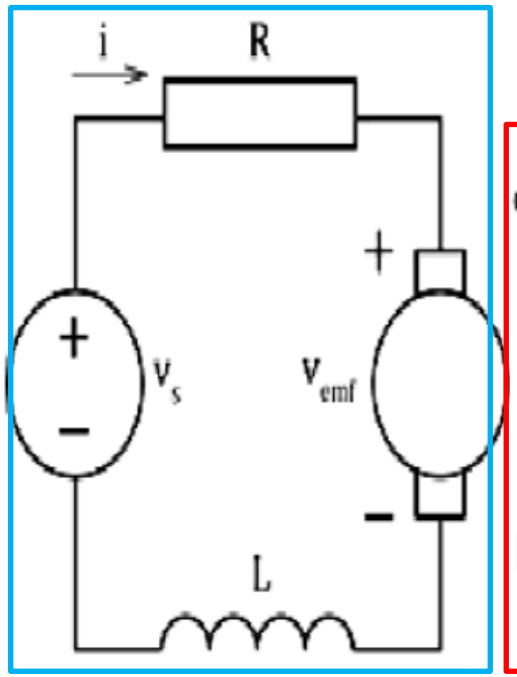
※기호 정리

- v_s : 모터의 입력전압
- I : 모터의 전류
- R : 모터 권선저항
- L : 모터의 권선 인덕턴스
- v_{emf} : 역기전력
- φ_M: 모터 회전각
- $b_M\dot{\varphi}$: 회전 마찰력
- I_M: 모터의 관성모멘트
- τ : 모터의 토크
- τ_L : 부하토크

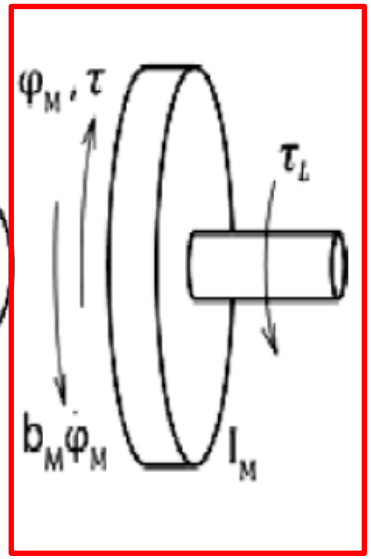
3. 동역학 모델링

(2) 동역학 모델링-DC모터

모터의 등가회로



모터의 기계부



(1) $\tau = k_t i$: 모터의 토크

(2) $v_{emf} = k_e \dot{\varphi}_M$: 모터의 역기전압

(3) $v_s - Ri - L\frac{di}{dt} - v_{emf} = 0$: 모터전압 kvl

(4) 모터의 L은 매우 작으므로, 식(3)에서 L의 값을 0으로 보면...

(5) $v_s = Ri + v_{emf} = Ri + k_e \dot{\varphi}_M$, $\therefore i = \frac{v_s - k_e \dot{\varphi}_M}{R}$

(6) $\tau = k_t i = \frac{k_t (v_s - k_e \dot{\varphi}_M)}{R}$: 토크와 입력전압의 관계식

(7) $\tau = I_M \ddot{\varphi}_M + b_M \dot{\varphi}_M + \tau_L$

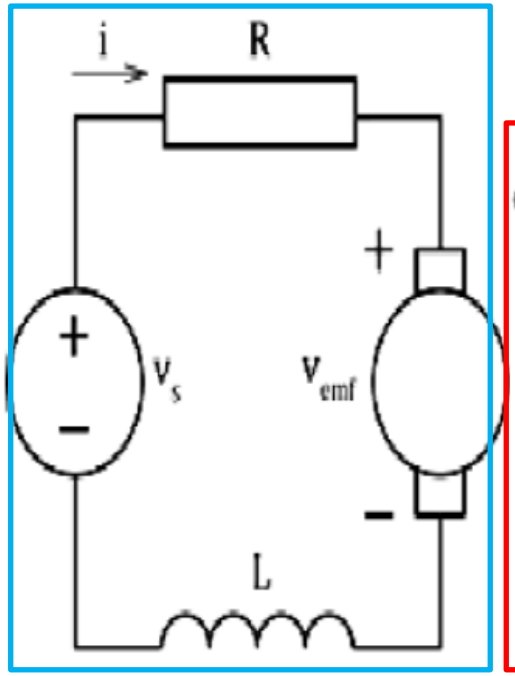
(8) 식(6)을 식(7)에 대입하면

$$\frac{k_t}{R} v_s = I_M \ddot{\varphi}_M + (b_M + \frac{k_e k_t}{R}) \dot{\varphi}_M + \tau_L$$

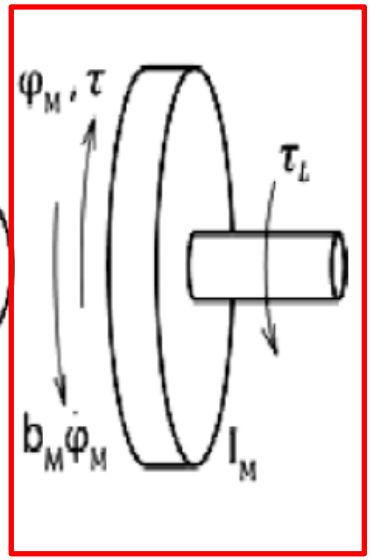
3. 동역학 모델링

(2) 동역학 모델링-DC모터

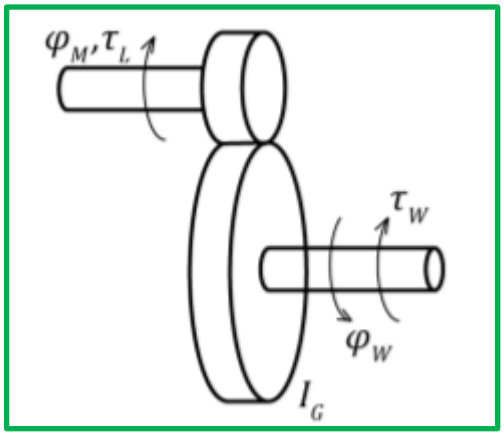
모터의 등가회로



모터의 기계부



모터의 기어



(9) $\tau_L \dot{\varphi}_M = \tau_W \dot{\varphi}_W$: 모터와 바퀴의 토크관계

(10) $\dot{\varphi}_M = \dot{\varphi}_W n$: n은 기어비, 기어비와 각속도관계

(11) $\tau_L = \frac{\tau_W}{n}$: 기어비와 각속도의 관계

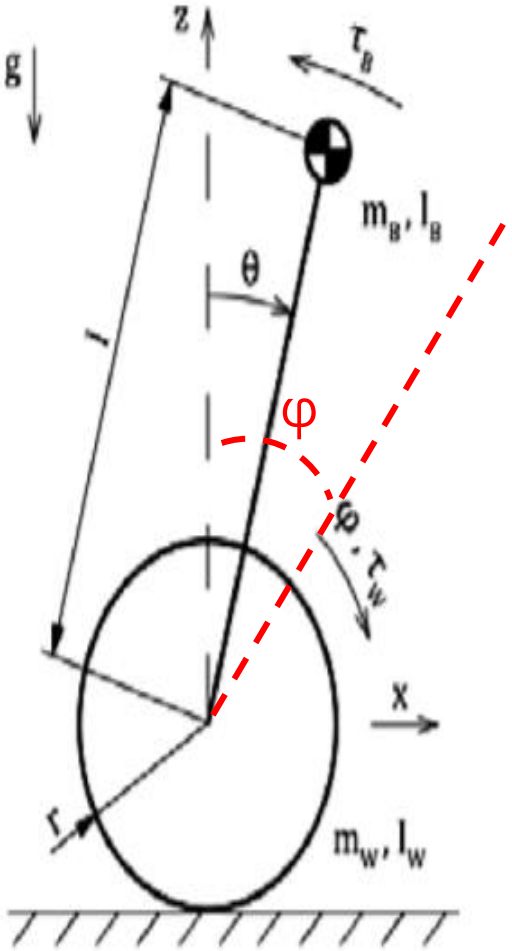
(12) 식(10),(11)을 식(9)에 대입하여 정리하면

$$\frac{k_t}{R} v_s = (I_M n^2 + I_G)\ddot{\varphi}_W + ((b_M + \frac{k_e k_t}{R})n^2 + b_G)\dot{\varphi}_W + \tau_W$$

3. 동역학 모델링

(2) 동역학 모델링-DC모터

- 기어의 회전각 $\varphi_W = \varphi$(바퀴 회전각) $- \theta$



로봇의 이동거리 x에 대한 운동 방정식

로봇의 몸체의 각도 θ에 대한 운동 방정식

(13) $\varphi_W = \varphi - \theta$

(14) $\tau_R + \tau_L = 2\tau_W$: 두 바퀴의 토크가 같다

(15) 앞서 정리한 식을 토대로 다음과 같이 정리된다.

(16) $$(\frac{m_B r^2}{2} + m_W r^2 + I_W + I_M n^2 + I_G)\ddot{\varphi} + (\frac{m_B rl}{2} - I_M n^2 - I_G)\ddot{\theta}$$
$$+ (b_W + b_M n^2 + \frac{K_e K_t n^2}{2} + b_G)\dot{\varphi} - (b_M n^2 + \frac{K_e K_t n^2}{2} + b_G)\dot{\theta} = \frac{K_t n}{R} v_s$$

(17) $$(\frac{m_B rl}{2} - I_M n^2 - I_G)\ddot{\varphi} + (\frac{m_B l^2}{2} + \frac{I_B}{2} + I_M n^2 + I_G)\ddot{\theta}$$
$$- (b_M n^2 + \frac{K_e K_t n^2}{2} - b_G)\dot{\varphi} - (b_M n^2 + \frac{K_e K_t n^2}{2} + b_G)\dot{\theta} - \frac{m_B gl}{2}\theta = -\frac{K_t n}{R} v_s$$