

제어용통신 수업자료

# **NUCLEO-F429 보드를 이용한 IMU 센서 (MPU6050) 읽기**

2018. 11 (Ver1.1)

한국산업기술대학교  
메카트로닉스공학과

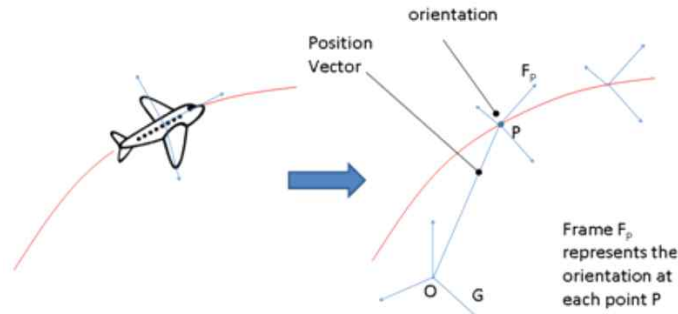
## 목 차

1. MPU-6050 : 물체의 자세(방향) 측정용 센서
2. I2C (Inter-Integrated Circuit) 통신
3. 보드와 MPU6050의 결선
4. 예제 프로그램 다운로드 및 실행
5. 주요 소스코드 설명

## 1. MPU-6050 : 물체의 자세(방향) 측정용 센서

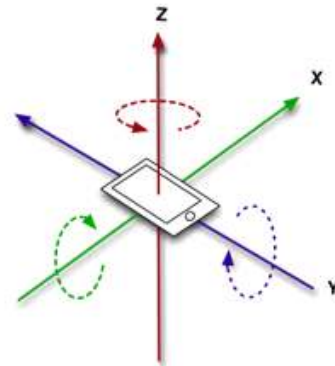
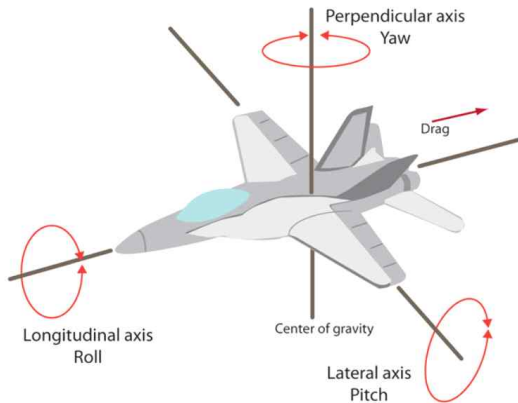
### 1.1 물체의 자세(방향)

공간상에서 물체의 위치 = 3 position + 3 orientation(방향, 자세)



Roll, Pitch, Yaw : 3 orientation을 표시하는 방법 중의 한가지

( Roll : 물체의 좌표계를 기준으로 x 방향의 회전, Pitch : y방향 , Yaw : z 방향의 회전 )





### 1.2 IMU(Inertial Measurement Unit : 관성측정장치) 센서

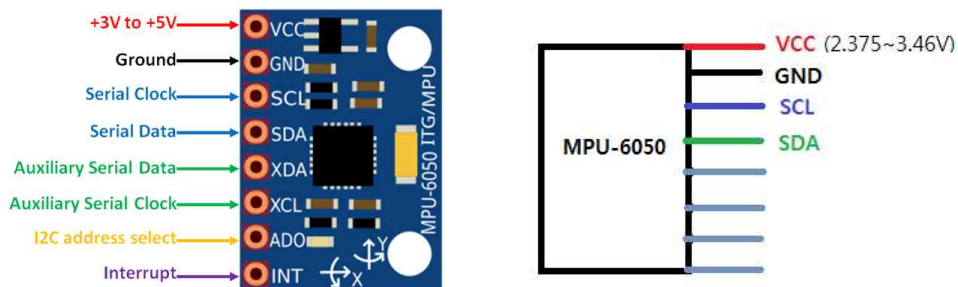
- IMU 센서 : 물체의 방향을 측정하기 위한 센서.
- 일반적으로 IMU는 자이로스코프 / 가속도계 / 지자기센서로 구성된다.
  - 종류에 따라 자이로스코프 가속도계만 있는 6축센서, 자이로스코프와 가속도계 지자기센서 까지 포함한 9축센서가 있다.

### 1.3 MPU-6050 센서

- IMU 센서 중에서 비교적 저가이며, 학습용으로 널리 사용됨
- [ 3축 가속도 센서 + 3축 자이로(회전속도) 센서] 를 내장한 6DOF(Degrees of Freedom) 모션센서
- 3축 가속도 센서 + 3축 자이로(회전속도) 센서값을 이용하여 3축의 Orientation 값을 구한다.
- I2C(Inter Integrated Circuit) 통신을 통해 데이터를 받는다.

MPU-6000, MPU-6050은 가속도 및 자이로 센서로 유명한 Invensense 사의 제품으로 사양은 다음과 같다.

Part #	Gyro Full Scale Range	Gyro Sensitivity	Gyro Rate Noise	Accel Full Scale Range	Accel Sensitivity	Digital Output	Logic Supply Voltage	Operating Voltage Supply
UNITS:	(°/sec)	(LSB/°/sec)	(dps/√Hz)	(g)	(LSB/g)		(V)	(V +/-5%)
 MPU-6000	±250	131	±2	16384	I²C or SPI	VDD	2.375V–3.46V	4x4x0.9
	±500	65.5	±4	8192				
	±1000	32.8	±8	4096				
	±2000	16.4	±16	2048				
 MPU-6050	±250	131	±2	16384	I²C	1.8V±5% or VDD	2.375V–3.46V	4x4x0.9
	±500	65.5	±4	8192				
	±1000	32.8	±8	4096				
	±2000	16.4	±16	2048				



- MCU와 통신하기 위해서는 옆의 4개의 선을 이용한다.

## 1.4 가속도 센서, 자이로 센서 및 상보 필터, 칼만 필터

### 1) 가속도 센서(accelerometer)

- 센서에 작용하는 3축 (x, y, z축) 방향의 중력가속도를 측정
- 가속도 값은 노이즈가 많으나, 누적오차가 생기지는 않음 (이리저리 자세가 변하더라도 원래 자세로 돌아오면 이전과 동일한 출력값이 나옴(노이즈 성분을 제외하면) )

### 2) 자이로 센서(gyro sensor)

- 센서가 회전할 때 3축(x, y, z축) 방향의 각속도 변화량을 측정
- 각속도를 적분하여 각도를 구하므로 누적오차가 생긴다.(센서의 노이즈도 적분이 됨)

### 3) 따라서 정확한 Orientation 값을 구하기 위해서는 가속도 센서와 자이로 센서의 장점만 모아서 보정을 해주어야 한다.

-> 이를 위해 필터를 사용 : 보통 상보 필터(Complementary filter) 또는 칼만 필터(Kalman filter)를 사용

### 4) [참고] 가속도 센서는 Yaw 방향의 회전은 검출할 수 없다.

- 예를 들어, 머리꼭대기에 센서를 수평으로 두고 몸을 좌우로 회전하면 중력가속도 방향의 변화가 없다, 즉, z 방향으로 회전하면 이때 가속도 센서의 값에 노이즈가 없다고 하면 값의 변화는 없다.

- 따라서 MPU-6050과 같은 센서로는 Yaw 값의 정확한 측정은 어렵다.

- Yaw 값의 정확한 측정을 위해서는 z축의 회전각을 알수 있는 지자기 센서(나침반)가 추가로 필요하다.

## 1.5 MPU-6050 데이터 시트 및 레지스터

### Electrical Specifications, Continued

VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T<sub>A</sub> = 25°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	Units	Notes
SERIAL INTERFACE SPI Operating Frequency, All Registers Read/Write	MPU-6000 only, Low Speed Characterization		100 ±10%		kHz	
	MPU-6000 only, High Speed Characterization		1 ±10%		MHz	
	SPI Operating Frequency, Sensor and Interrupt Registers Read Only		20 ±10%		MHz	
	I <sup>2</sup> C Operating Frequency			400 100	kHz kHz	
I <sup>2</sup> C ADDRESS	AD0 = 0		1101000			
	AD0 = 1		1101001			

0b11010000 = 0xD0

#define i2caddress 0xD0

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
67	103	I2C_MST_DELAY_CTRL	R/W	DELAY_ES_SHADOW	-	-	I2C_SLV4_DLY_EN	I2C_SLV3_DLY_EN	I2C_SLV2_DLY_EN	I2C_SLV1_DLY_EN	I2C_SLV0_DLY_EN
68	104	SIGNAL_PATH_RESET	R/W	-	-	-	-	-	GYRO_RESET	ACCEL_RESET	TEMP_RESET
6A	106	USER_CTRL	R/W	-	FIFO_EN	I2C_MST_EN	I2C_IF_DIS	-	FIFO_RESET	I2C_MST_RESET	SIG_COND_RESET
6B	107	PWR_MGMT_1	R/W	DEVICE_RESET	SLEEP	CYCLE	-	TEMP_DIS	CLKSEL[2:0]		
6C	108	PWR_MGMT_2	R/W	LP_WAKE_CTRL[1:0]		STBY_XA	STBY_YA	STBY_ZA	STBY_XG	STBY_YG	STBY_ZG
72	114	FIFO_COUNTH	R/W	FIFO_COUNT[15:8]							
73	115	FIFO_COUNTL	R/W	FIFO_COUNT[7:0]							
74	116	FIFO_R_W	R/W	FIFO_DATA[7:0]							
75	117	WHO_AM_I	R	-	WHO_AM_I[6:1]						-

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

```
int main(void)
{
    sei();

    Mpu6050.Write(0x6B, 0x01); // PLL with X axis gyroscope reference and disable sleep mode
                                // MPU-60XX 을 동작 시킨다. (모드 및 설정 값은 데이터시트 참조)

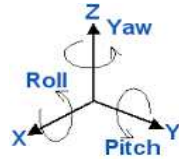
    _DELAY_MS(100);
}
```

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
3B	59	ACCEL_XOUT_H	R	ACCEL_XOUT[15:8]							
3C	60	ACCEL_XOUT_L	R	ACCEL_XOUT[7:0]							
3D	61	ACCEL_YOUT_H	R	ACCEL_YOUT[15:8]							
3E	62	ACCEL_YOUT_L	R	ACCEL_YOUT[7:0]							
3F	63	ACCEL_ZOUT_H	R	ACCEL_ZOUT[15:8]							
40	64	ACCEL_ZOUT_L	R	ACCEL_ZOUT[7:0]							
41	65	TEMP_OUT_H	R	TEMP_OUT[15:8]							
42	66	TEMP_OUT_L	R	TEMP_OUT[7:0]							
43	67	GYRO_XOUT_H	R	GYRO_XOUT[15:8]							
44	68	GYRO_XOUT_L	R	GYRO_XOUT[7:0]							
45	69	GYRO_YOUT_H	R	GYRO_YOUT[15:8]							
46	70	GYRO_YOUT_L	R	GYRO_YOUT[7:0]							
47	71	GYRO_ZOUT_H	R	GYRO_ZOUT[15:8]							
48	72	GYRO_ZOUT_L	R	GYRO_ZOUT[7:0]							

```
while(1)
{
    // 가속도 값과 각 가속도 값을 받아온다.
    buffer[0] = Mpu6050.Read(0x3B);
    buffer[1] = Mpu6050.Read(0x3C);
    buffer[2] = Mpu6050.Read(0x3D);
    buffer[3] = Mpu6050.Read(0x3E);
    buffer[4] = Mpu6050.Read(0x3F);
    buffer[5] = Mpu6050.Read(0x40);
    buffer[6] = Mpu6050.Read(0x43);
    buffer[7] = Mpu6050.Read(0x44);
    buffer[8] = Mpu6050.Read(0x45);
    buffer[9] = Mpu6050.Read(0x46);
    buffer[10] = Mpu6050.Read(0x47);
    buffer[11] = Mpu6050.Read(0x48);
}
```

## 1.5 각도 구하기

### 1.5.1 각도 구하는 법



$$\rho = \arctan\left(\frac{A_X}{\sqrt{A_Y^2 + A_Z^2}}\right)$$

$$\phi = \arctan\left(\frac{A_Y}{\sqrt{A_X^2 + A_Z^2}}\right)$$

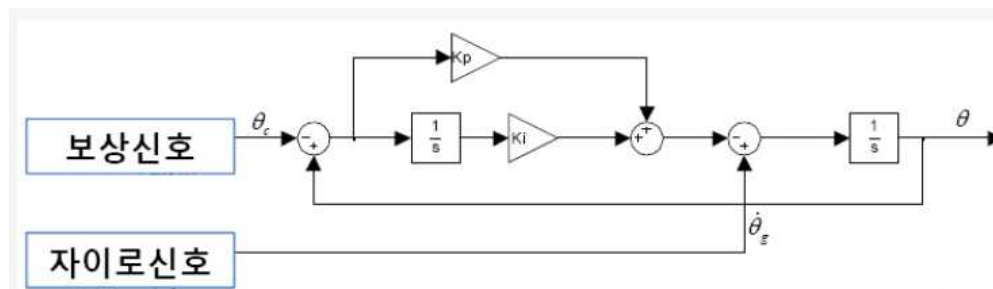
$$\theta = \arctan\left(\frac{\sqrt{A_X^2 + A_Y^2}}{A_Z}\right)$$

```
#include <math.h>
#define PI 3.14159265
#define RAD_TO_DEG 57.2957786
```

```
// 가속도값으로 각도를 구한다.
accXangle = (atan2(accY, accZ)+PI)*RAD_TO_DEG; // roll
accYangle = (atan2(accX, accZ)+PI)*RAD_TO_DEG; // pitch
```

```
// 가속도값으로 구한 각도 + 가속비 + 시간 을 이용한 상보 필터로 각도를 구한다.
// Calculate the angle using a Complimentary filter
compAngleX = (0.93*(compAngleX+(gyroXrate*(double)(TO.time-timer)/1000)))+(0.07*accXangle);
compAngleY = (0.93*(compAngleY+(gyroYrate*(double)(TO.time-timer)/1000)))+(0.07*accYangle);
```

### 1.5.2 상보 필터



## 1.5.3 [참고] 칼만 필터

### 칼만 필터

위키백과, 우리 모두의 백과사전.

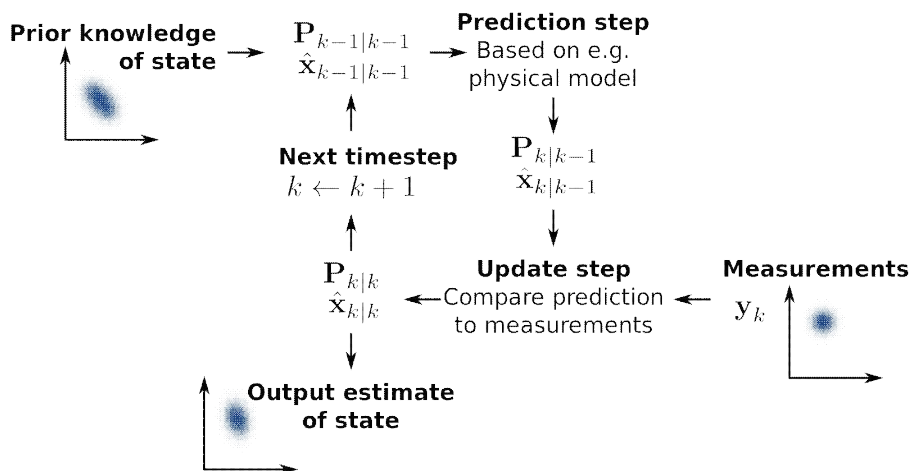
칼만 필터(Kalman filter)는 잡음이 포함되어 있는 선형 역학계의 상태를 추적하는 **재귀 필터**로, **루돌프 칼만**이 개발하였다. 칼만 필터는 컴퓨터 비전, 로봇 공학, 레이더 등의 여러 분야에 사용되며, 많은 경우에 매우 효율적인 성능을 보여준다.

이 알고리즘은 시간에 따라 진행된 측정을 기반으로 한다. 해당 순간에만 측정된 결과만 사용한 것보다는 좀 더 정확한 결과를 기대할 수 있다. 잡음까지 포함된 입력 데이터를 재귀적으로 처리하는 이 필터로서, 현재 상태에 대한 최적의 통계적 예측을 진행할 수 있다.

알고리즘 전체는 예측과 업데이트의 두 가지로 나눌 수 있다. 예측은 현재 상태의 예측을 말하며, 업데이트는 현재 상태에서 관측된 측정까지 포함한 값을 통해서 더 정확한 예측을 할 수 있는 것을 말한다.

#### \* 칼만 필터 계산 과정

0. 초기값 선정
1. 추정값과 오차 공분산 예측
2. 칼만 이득 계산
3. 추정값 계산
4. 오차 공분산 계산



#### [칼만 필터의 소스코드 예]

```
// 가속도값으로 구한 각도 + 가속비 + 시간 을 이용한 칼만 필터로 각도를 구한다.  
kalAngleX = kalmanX.getAngle(accXangle, gyroXrate, (double)(TO.time-timer)/1000);  
kalAngleY = kalmanY.getAngle(accYangle, gyroYrate, (double)(TO.time-timer)/1000);
```



## [ 칼만 필터 소스 코드 ] : 클래스(C++)를 이용

```
class Kalman {
public:
    Kalman() {
        Q_angle = 0.001;
        Q_bias = 0.003;
        R_measure = 0.03;

        bias = 0;
        P[0][0] = 0;
        P[0][1] = 0;
        P[1][0] = 0;
        P[1][1] = 0;
    };

    double getAngle(double newAngle, double newRate, double dt) {
        rate = newRate - bias;
        angle += dt * rate;

        P[0][0] += dt * (dt * P[1][1] - P[0][1] - P[1][0] + Q_angle);
        P[0][1] -= dt * P[1][1];
        P[1][0] -= dt * P[1][1];
        P[1][1] += Q_bias * dt;

        S = P[0][0] + R_measure;

        K[0] = P[0][0] / S;
        K[1] = P[1][0] / S;

        y = newAngle - angle;

        angle += K[0] * y;
        bias += K[1] * y;

        P[0][0] -= K[0] * P[0][0];
        P[0][1] -= K[0] * P[0][1];
        P[1][0] -= K[1] * P[0][0];
        P[1][1] -= K[1] * P[0][1];

        return angle;
    };
    void setAngle(double newAngle) { angle = newAngle; };
    double getRate() { return rate; };

    void setQangle(double newQ_angle) { Q_angle = newQ_angle; };
    void setQbias(double newQ_bias) { Q_bias = newQ_bias; };
    void setRmeasure(double newR_measure) { R_measure = newR_measure; };

private:
    double Q_angle;
    double Q_bias;
    double R_measure;

    double angle;
    double bias;
    double rate;

    double P[2][2];
    double K[2];
    double y;
    double S;
};
```

## 2. I2C (Inter-Integrated Circuit) 통신

### 2.1 I2C 통신이란?

I2C란 필립스(Philips)사에서 개발한 직렬 컴퓨터 버스 규격으로 빠른 속도를 요구하지 않는 간단하고 저가의 주변 장치들에 적합하다.

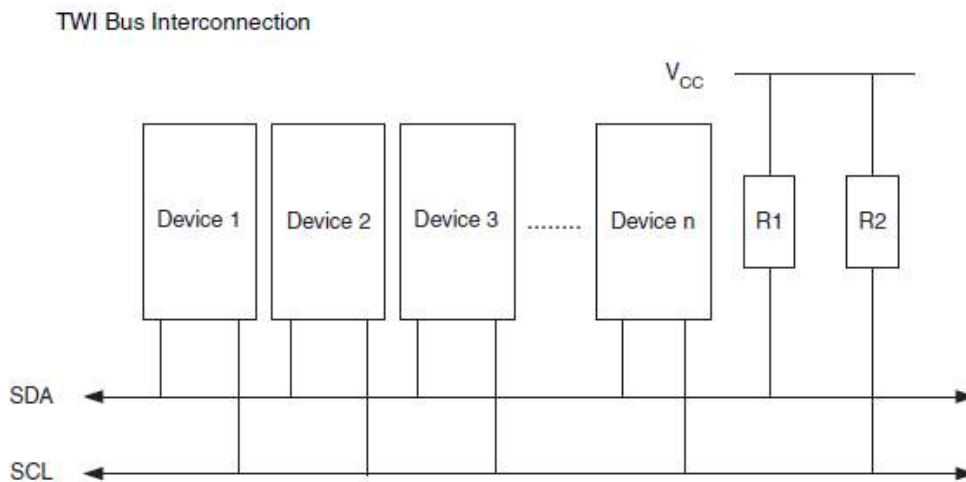
I2C는 SCL(Serial Clock Line)과 SDA(Serial Data Line)만 필요하므로 TWI (2-wire Serial Interface) 라고도 한다.

[참고] I2C의 속도

Standard mode : 100Kbit/s

Fast mode : 400Kbit/s

High speed mode : 3.4Mbit/s



- 주변장치 중 I2C는 SPI 보다 느리다.

v.d.e.h	컴퓨터 버스 기술 표준과 사실상 표준 (유선)	[숨기기]
일반	시스템 버스 · 프론트 사이드 버스 · 백사이드 버스 · 데이터 체인 · 제어 버스 · 주소 버스 · Bus contention · 플러그 앤 플레이 · 버스 대역폭 목록	
표준	S-100 bus · Unibus · VAXBI · MBus · STD Bus · SMBus · Q-Bus · ISA · Zorro II · Zorro III · CAMAC · FASTBUS · LPC · HP Precision Bus · EISA · VME · VXI · VXS · NuBus · TURBOchannel · MCA · SBus · VLB · PCI · PXI · HP GSC bus · CoreConnect · 인피니밴드 · UPA · PCI-X · AGP · PCIe · 하이퍼트랜스포트 · QPI	
휴대용	PC 카드 · 익스프레스카드	
임베디드	Multidrop bus · AMBA · Wishbone	
스토리지	ST-506 · ESDI · SMD · 병렬 ATA · DMA · SSA · HIPPI · USB MSC · IEEE 1394 · SATA · eSATA · eSATAp · SCSI · Parallel SCSI · SAS	
주변장치	Apple Desktop Bus · HPI · MIDI · Multibus · RS-232 · RS-422 · RS-423 · RS-485 · DMX512 · IEEE-488 (GPIB) · IEEE-1284 (parallel port) · UNI/O · ACCESS.bus · 1-Wire · <b>I2C</b> · SPI · Parallel SCSI · Profibus · USB · IEEE 1394 (FireWire) · Camera Link · External PCIe · 선더볼트	
참고: 나열된 인터페이스들은 개략적으로 속도 순서대로 정렬되었음. 각 부분의 마지막이 가장 빠름.		
🔍 컴퓨터 버스		

## Speed of various connectivity methods (bits/sec)

CAN (1 Wire)	33 kHz (typ)
I <sup>2</sup> C ('Industrial', and SMBus)	100 kHz
SPI	110 kHz (original speed)
CAN (fault tolerant)	125 kHz
I <sup>2</sup> C	400 kHz
CAN (high speed)	1 MHz
I <sup>2</sup> C 'High Speed mode'	3.4 MHz
USB (1.1)	1.5 MHz or 12 MHz
SCSI (parallel bus)	40 MHz
Fast SCSI	8-80 MHz
Ultra SCSI-3	18-160 MHz
Firewire / IEEE1394	400 MHz
Hi-Speed USB (2.0)	480 MHz

DesignCon 2003 TecForum I<sup>2</sup>C Bus Overview

9

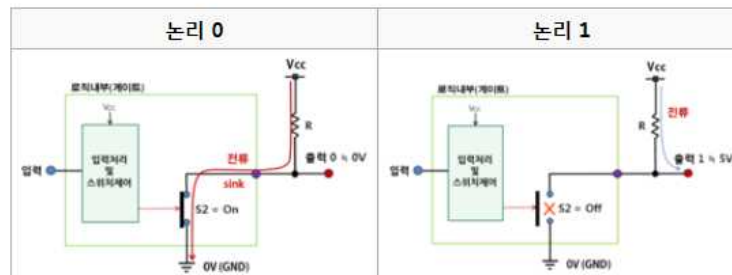
- I2C는 직렬 통신 중의 하나이다

- I2C는 풀업 저항이 연결된 직렬 데이터(SDA)와 직렬 클럭(SCL)이라는 두 개의 양 방향 오픈 컬렉터 라인을 사용한다.

### 오픈-컬렉터 출력 [편집]

위의 스위치를 제거하고 다음과 같이 출력하는 것을 오픈-컬렉터 (Open-Collector, TTL 게이트) 또는 오픈-드레인(Open-Drain, FET 게이트)라고 한다. 논리 0을 출력할 때는 GND 쪽의 스위치를 닫아 0V를 유지하고 논리1일 때는 Z 상태를 유지한다. 논리1일 때 외부에서 전압을 결정하도록 회로를 구성해야 한다.

다음과 같이 예로 개념적 도식할 수 있다:



논리 1 출력할 때 하이 임피던스가 되므로 여러 오픈 컬렉터 출력들을 단순히 연결한 다음에 풀업을 하면, 출력의 어떤 것이라도 논리 0이 되었을 때 L가 되므로 논리곱 연산을 하게된다. 이와 같이 논리합 디지털 회로 소자 없이 논리곱을 구현한 회로를 「와이어드 AND」라고 부른다.

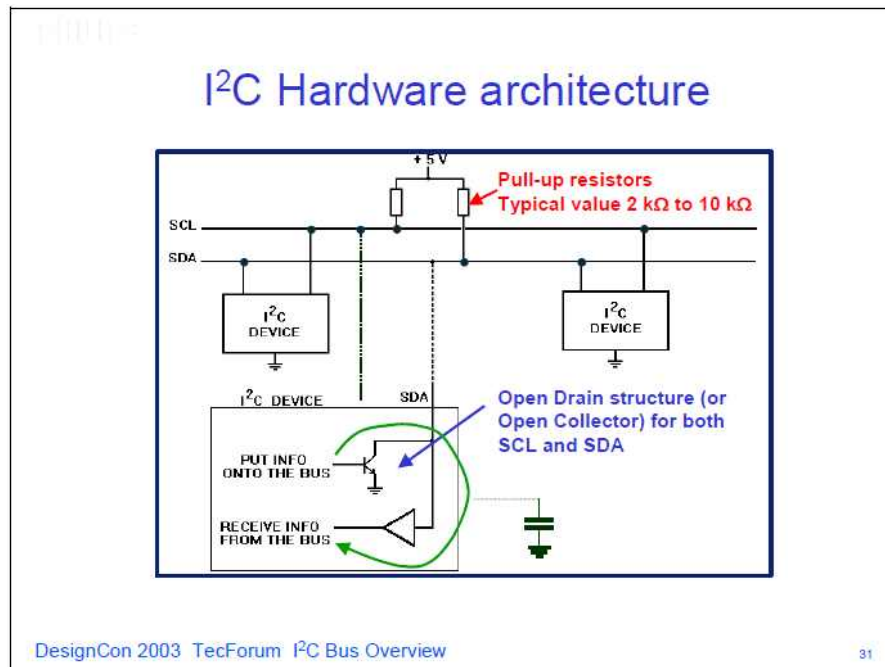
오픈 컬렉터를 이용하는 경우는:

- 단일전원을 사용하는 논리 게이트 동작에서 다른 전압의 출력을 원할 때
- 일반 출력 단자보다 더 많은 전류를 흘릴 필요가 있을 때
- 와이어드 AND 구현할 때

사용하는 경우가 일반적이다.

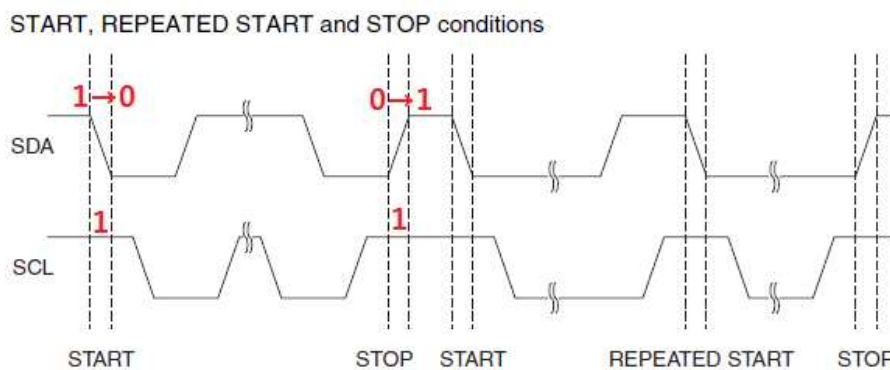
아날로그 회로에 의한 전류 증폭을 하지 않고도 직접 LED를 점등시키는 주변 장치와 연결할 수 있다.

마이크로프로세서에서 다른 칩과의 통신을 위해서 이 회로를 많이 사용한다. I<sup>2</sup>C등의 회로가 이 경우이다.



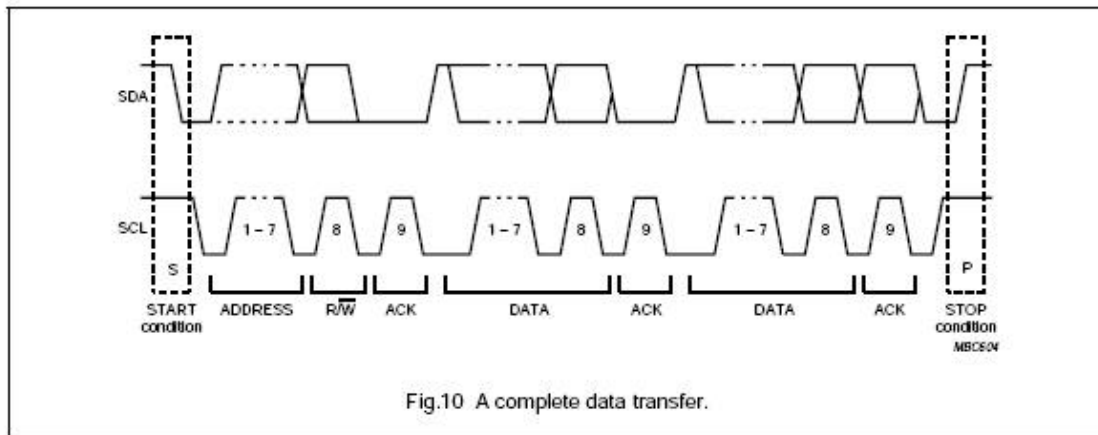
- I2C 의 장점은 마이크로컨트롤러에서 단지 2개의 입출력(I/O) 핀과 소프트웨어만을 이용하여 여러 장치들을 제어할 수 있다는 점이다.

## 2.2 I2C 통신 규격



<I2C 통신 원리>

- SCL의 신호에 맞춰 SDA에 데이터가 실려서 보내진다.
- SCL이 0이될 때 마스터는 슬레이브로부터 Ack 를 받게 된다.



- SCL이 1일 때를 주목해서 보자.

## I<sup>2</sup>C Address, 7-bit and 10-bit formats

- The 1st byte after START determines the Slave to be addressed
- Some exceptions to the rule:
  - “General Call” address: all devices are addressed : 0000 000 + R/W = 0
  - 10-bit slave addressing : 1111 0XX + R/W = X

**• 7-bit addressing**

S | XXXXXX | R/W | A | DATA | ...

The 7 bits

↑ Only one device will acknowledge

**• 10-bit addressing**

S | 11110XX | R/W | A1 | XXXXXXXX | A2 | DATA | ...

XX = the 2 MSBs  
More than one device can acknowledge

The 8 remaining bits

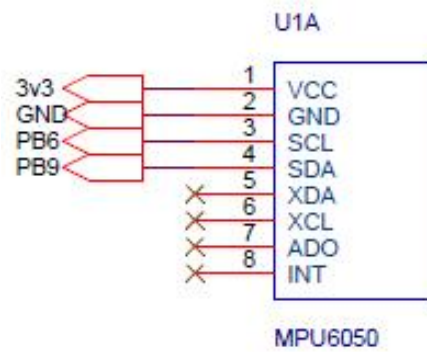
↑ Only one device will acknowledge

DesignCon 2003 TecForum I<sup>2</sup>C Bus Overview 34

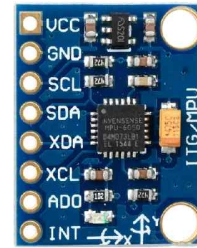
< I2C 데이터 형식 >

시작 비트 + 주소 7비트 + R/W 비트(0쓰기/1읽기) + Ack

### 3. 보드와 MPU6050의 결선

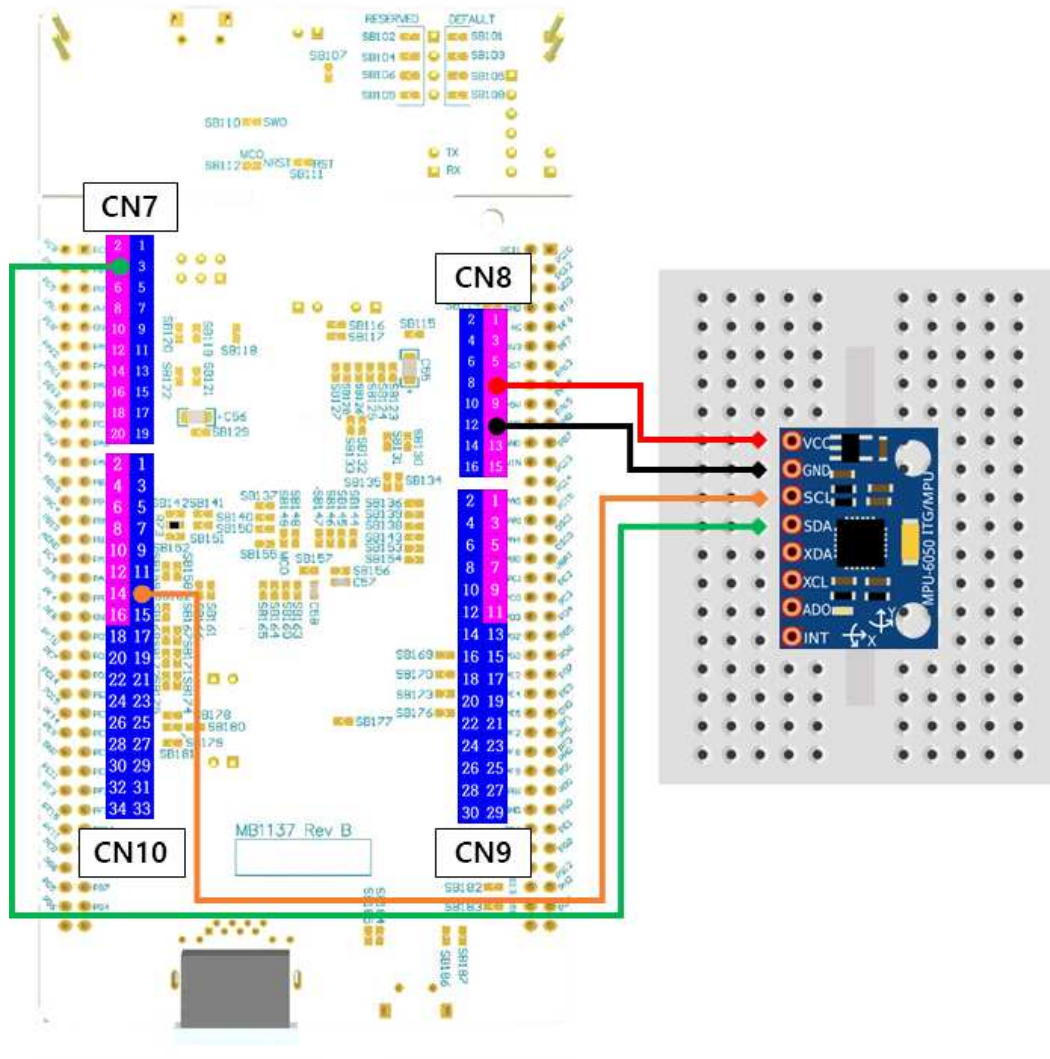


[ 결선 회로도 ]



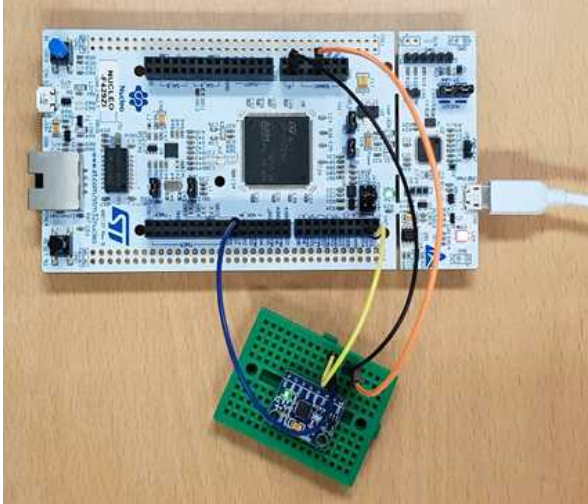
[ MPU6050

모듈의 모양 ]

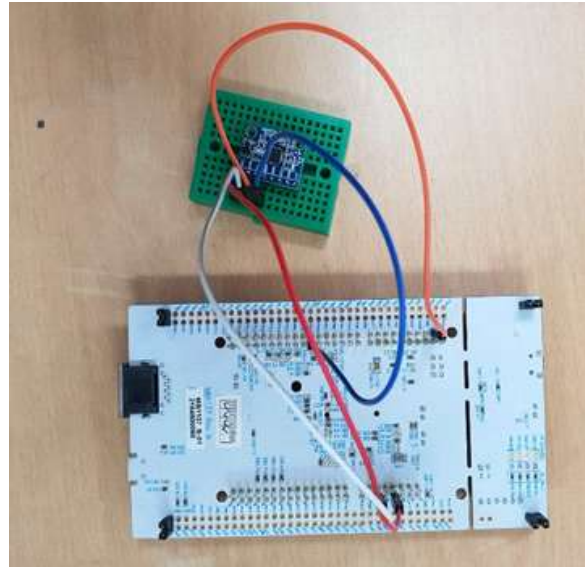


[ 점퍼선을 이용한 결선 예 ]





< 보드의 윗편으로 결선한 경우 >



< 보드의 아랫편으로 결선한 경우 >

MPU6050	NUCLEO-F429ZI
VCC	3V3
GND	GND
SCL	PB6
SDA	PB9

[ MPU6050, 보드 핀 연결 표 ]

#### 4. 예제 프로그램 다운로드 및 실행

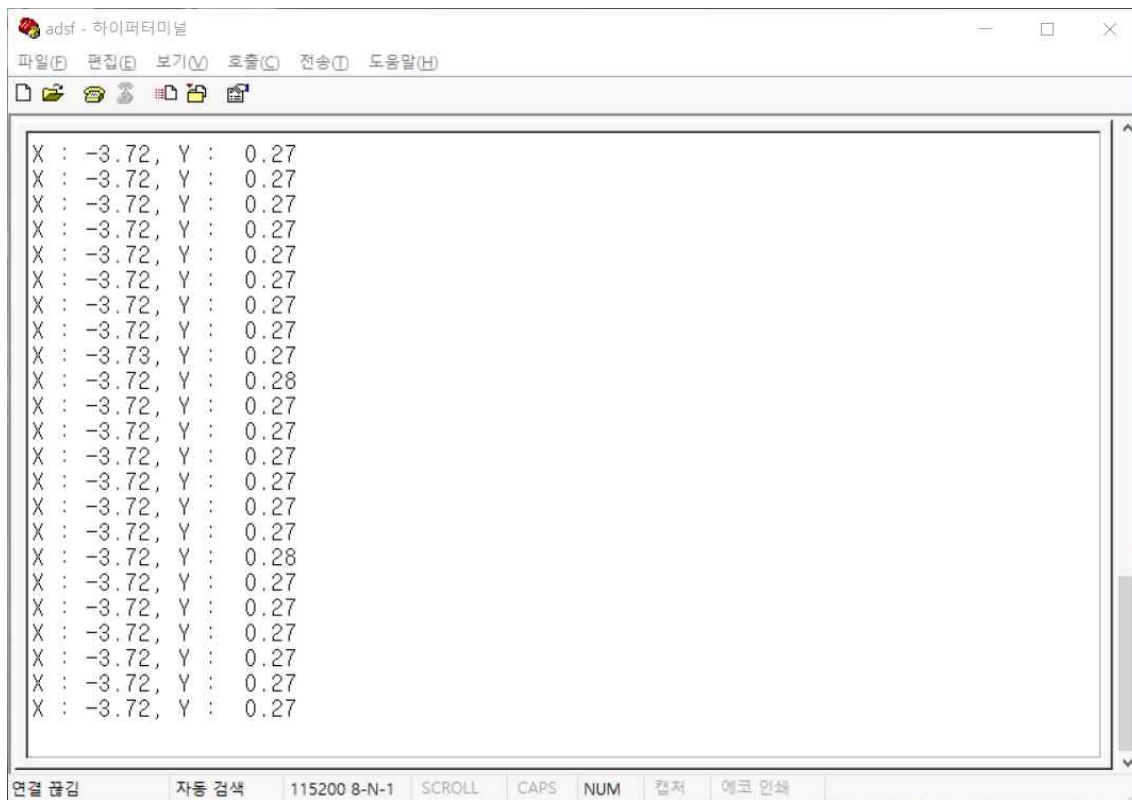
가. 네이버 카페 (Cortexworld)에서 예제를 다운받는다.

(게시글 1037 : F429보드를 이용한 IMU 센서 읽기)

나. 다운받은 파일의 압축을 푼다.

다. 압축을 푼 파일 중에서 F429ZI\_I2C\_Example\MDK-ARM\STM32F429ZI\_Example.uvprojx 파일 실행해서 빌드 후 보드에 다운로드

라. BaudRate 115200으로 설정 후 하이퍼터미널에서 동작 확인



The screenshot shows a HyperTerminal window titled 'adstf - 하이퍼터미널'. The window contains a list of sensor data readings. Each line displays 'X' and 'Y' coordinates. The 'X' values are mostly -3.72, with one instance of -3.73. The 'Y' values are mostly 0.27, with two instances of 0.28. The window has a standard menu bar (파일(F), 편집(E), 보기(V), 호출(C), 전송(T), 도움말(H)) and a toolbar. At the bottom, there is a status bar with various settings: 연결 끊김, 자동 검색, 115200 8-N-1, SCROLL, CAPS, NUM, 캡처, and 에코 인쇄.

```
X : -3.72, Y : 0.27
X : -3.72, Y : 0.27
X : -3.72, Y : 0.27
X : -3.72, Y : 0.27
X : -3.72, Y : 0.27
X : -3.72, Y : 0.27
X : -3.72, Y : 0.27
X : -3.72, Y : 0.27
X : -3.73, Y : 0.27
X : -3.72, Y : 0.28
X : -3.72, Y : 0.27
X : -3.72, Y : 0.27
X : -3.72, Y : 0.27
X : -3.72, Y : 0.27
X : -3.72, Y : 0.27
X : -3.72, Y : 0.28
X : -3.72, Y : 0.27
X : -3.72, Y : 0.27
X : -3.72, Y : 0.27
X : -3.72, Y : 0.27
X : -3.72, Y : 0.27
X : -3.72, Y : 0.27
```



## 5. 주요 소스코드 설명

### [ main.c ]

```
int main(void)
{
    /* -- <1> 초기 설정용 함수 -- */
    HAL_Init();
    SystemClock_Config();
    MX_GPIO_Init();
    MX_USART3_UART_Init();

    MX_TIM2_Init();
    MX_I2C1_Init();
    MX_TIM5_Init();
    MX_NVIC_Init();

    /* -- <2> 타이머 시작 -- */
    HAL_TIM_Base_Start_IT(&htim2);
    HAL_TIM_Base_Start_IT(&htim5);

    /* -- <3> I2C 디바이스(MPU6050) 부팅 대기 -- */
    HAL_I2C_IsDeviceReady(&hi2c1, I2C_DEVICE_ADDRESS, 1, 1000);

    while (1)
    {
        /* -- <4> 각도값 계산-- */
        getAngle();
    }
}
```

### [ interrupt.c ]

```
#include "interrupt.h"

unsigned long tim_counter;
unsigned long milliseconds;

void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    /* -- <1> 0.1sec 주기로 플래그 SET, 0.1sec마다 PC로 출력하기 위함 -- */
    if(htim->Instance == TIM2){
        flag.timer2 = 1;
        tim_counter++;
    }
    /* -- <2> msec 단위로 counting -- */
    else if(htim->Instance == TIM5){
        milliseconds++;
    }
}
```

## [ app.c ] getAccel( )

```
void getAccel(double *accXangle, double *accYangle)
{
    uint8_t buffer[6] = {0, };
    int16_t accX, accY, accZ;

    /* -- <1> -I2C 디바이스의 메모리에 접근해서 값 읽어오기- */
    HAL_I2C_Mem_Read(&hi2c1, I2C_DEVICE_ADDRESS, 0x3B, I2C_MEMADD_SIZE_8BIT, &buffer[0], 1, 500);
    HAL_I2C_Mem_Read(&hi2c1, I2C_DEVICE_ADDRESS, 0x3C, I2C_MEMADD_SIZE_8BIT, &buffer[1], 1, 500);
    HAL_I2C_Mem_Read(&hi2c1, I2C_DEVICE_ADDRESS, 0x3D, I2C_MEMADD_SIZE_8BIT, &buffer[2], 1, 500);
    HAL_I2C_Mem_Read(&hi2c1, I2C_DEVICE_ADDRESS, 0x3E, I2C_MEMADD_SIZE_8BIT, &buffer[3], 1, 500);
    HAL_I2C_Mem_Read(&hi2c1, I2C_DEVICE_ADDRESS, 0x3F, I2C_MEMADD_SIZE_8BIT, &buffer[4], 1, 500);
    HAL_I2C_Mem_Read(&hi2c1, I2C_DEVICE_ADDRESS, 0x40, I2C_MEMADD_SIZE_8BIT, &buffer[5], 1, 500);

    /* -- <2> 8비트로 나누어져 있는 값을 16비트로 변환 -- */
    accX = (int)buffer[0] << 8 | (int)buffer[1];
    accY = (int)buffer[2] << 8 | (int)buffer[3];
    accZ = (int)buffer[4] << 8 | (int)buffer[5];

    /* -- <3> 가속도 값을 이용해 각도를 계산-- */
    *accXangle = (atan2(accY, accZ)+PI)*RAD2DEG;
    *accYangle = (atan2(accX, accZ)+PI)*RAD2DEG;
}
```

## [ app.c ] getGyro( )

```
void getGyro(double *gyroXrate, double *gyroYrate)
{
    uint8_t buffer[6] = {0, };
    int16_t gyroX, gyroY, gyroZ;

    /* -- <1> I2C 디바이스의 메모리에 접근해서 값 읽어오기 -- */
    HAL_I2C_Mem_Read(&hi2c1, I2C_DEVICE_ADDRESS, 0x43, I2C_MEMADD_SIZE_8BIT, &buffer[0], 1, 500);
    HAL_I2C_Mem_Read(&hi2c1, I2C_DEVICE_ADDRESS, 0x44, I2C_MEMADD_SIZE_8BIT, &buffer[1], 1, 500);
    HAL_I2C_Mem_Read(&hi2c1, I2C_DEVICE_ADDRESS, 0x45, I2C_MEMADD_SIZE_8BIT, &buffer[2], 1, 500);
    HAL_I2C_Mem_Read(&hi2c1, I2C_DEVICE_ADDRESS, 0x46, I2C_MEMADD_SIZE_8BIT, &buffer[3], 1, 500);
    HAL_I2C_Mem_Read(&hi2c1, I2C_DEVICE_ADDRESS, 0x47, I2C_MEMADD_SIZE_8BIT, &buffer[4], 1, 500);
    HAL_I2C_Mem_Read(&hi2c1, I2C_DEVICE_ADDRESS, 0x48, I2C_MEMADD_SIZE_8BIT, &buffer[5], 1, 500);

    /* -- <2> 8비트로 나누어져 있는 값을 16비트로 변환-- */
    gyroX = (int)buffer[0] << 8 | (int)buffer[1];
    gyroY = (int)buffer[2] << 8 | (int)buffer[3];
    gyroZ = (int)buffer[4] << 8 | (int)buffer[5];

    /* -- <3> MPU6050의 경우 1[degree]/[sec] = 131이므로 131로 나누기 -- */
    *gyroXrate = (double)gyroX/131.0;
    *gyroYrate = -((double)gyroY/131.0);
}
```

## [ app.c ] getAngle( )

```
void getAngle(void)
{
    double accXangle, accYangle;
    double gyroXrate, gyroYrate;
    double timePass;
    unsigned long present;
    double A;
    static const double k = 3.0;

    HAL_I2C_Mem_Write(&hi2c1, I2C_DEVICE_ADDRESS, 0x6B, I2C_MEMADD_SIZE_8BIT, pData, 1, 500);
    getAccel(&accXangle, &accYangle);
    getGyro(&gyroXrate, &gyroYrate);

    present = milliseconds;
    timePass = (present - preTimeMPU)/1000.0;
    preTimeMPU = present;
    A = k/(k+timePass);

    /* -- <1> 초기값은 가속도 값으로 계산한 각도만 고려-- */
    if(flag.angle_offset) {
        flag.angle_offset = 0;
        gotXangle = accXangle;
        gotYangle = accYangle;
    }
    else {
        /* -- <2> 상보필터-- */
        gotXangle = A*(gotXangle + gyroXrate*timePass) + (1 - A)*accXangle;
        gotYangle = A*(gotYangle + gyroYrate*timePass) + (1 - A)*accYangle;
    }

    /* -- <3> 180도 기준을 0도로 바꾸기-- */
    angleX = gotXangle - 180.0;
    angleY = gotYangle - 180.0;

    /* -- <4> 각도값 출력(0.1sec 주기)-- */
    printAngle();
}
```

상보필터 참고 : <http://alnova2.tistory.com/1085>