

I²C

Lesson 05

- **1980년대 초반 필립스에서 개발**
- **2의 입출력 핀을 사용하여 장치 간 양방향 통신을 지원**
- **하나의 마스터(master) 장치와 여러 개의 슬레이브(slave) 장치가 연결될 수 있는 버스(bus)**
- **빠른 속도를 요구하지 않는 간단한 주변 장치와의 통신에 적합**
 - 최대 속도는 평균 수백(100-400)KHz로 매우 낮음
- **CLOCK과 DATA 회선 두 개로 통신하기 때문에 두 가닥(two-wire) 프로토콜이라고도 불림**

I²C 통신 하드웨어 구성

- **CLOCK 신호(SCL):** 클럭 펄스가 생성되는 회선
- **DATA 신호(SDA):** 양방향 데이터 회선
- **CLOCK과 DATA 회선에 각각 풀업 저항을 연결해야 한다.**
 - 저항값은 연결하는 슬레이브 장치의 종류 및 개수에 따라 결정된다.
 - 보통 4.7k 저항으로 사용한다.

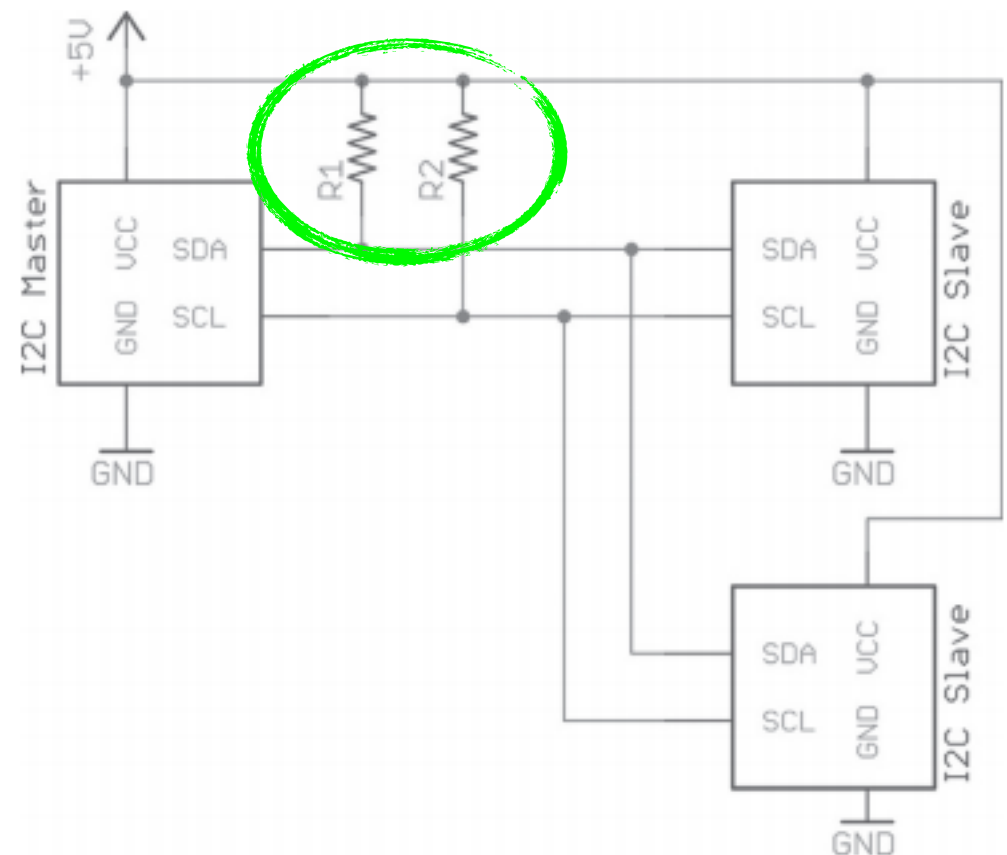
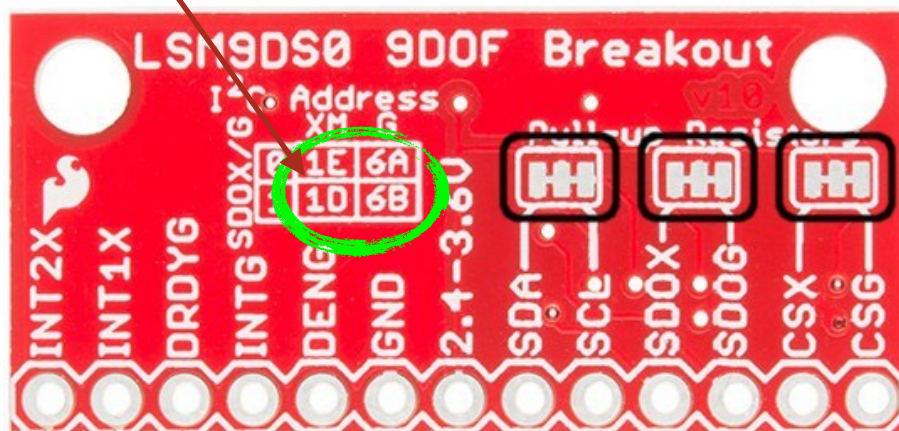


Image created with Eagle

I²C 장치 간 통신 주소 설정

- 각각의 I²C 장치는 7비트의 고유한 통신 주소를 가지고 있음
 - 버스에 연결된 모든 장치가 동일한 메시지를 전달 받기 때문에 고유 주소가 중복되어서는 안됨

I²C 주소



Set the SDO_G and SDO_XM to set the gyro and accel/mag I²C addresses:

SDOG/XM	XM Addr.	G Addr.
0	0x1E	0x6A
1	0x1D	0x6B

I²C통신 절차

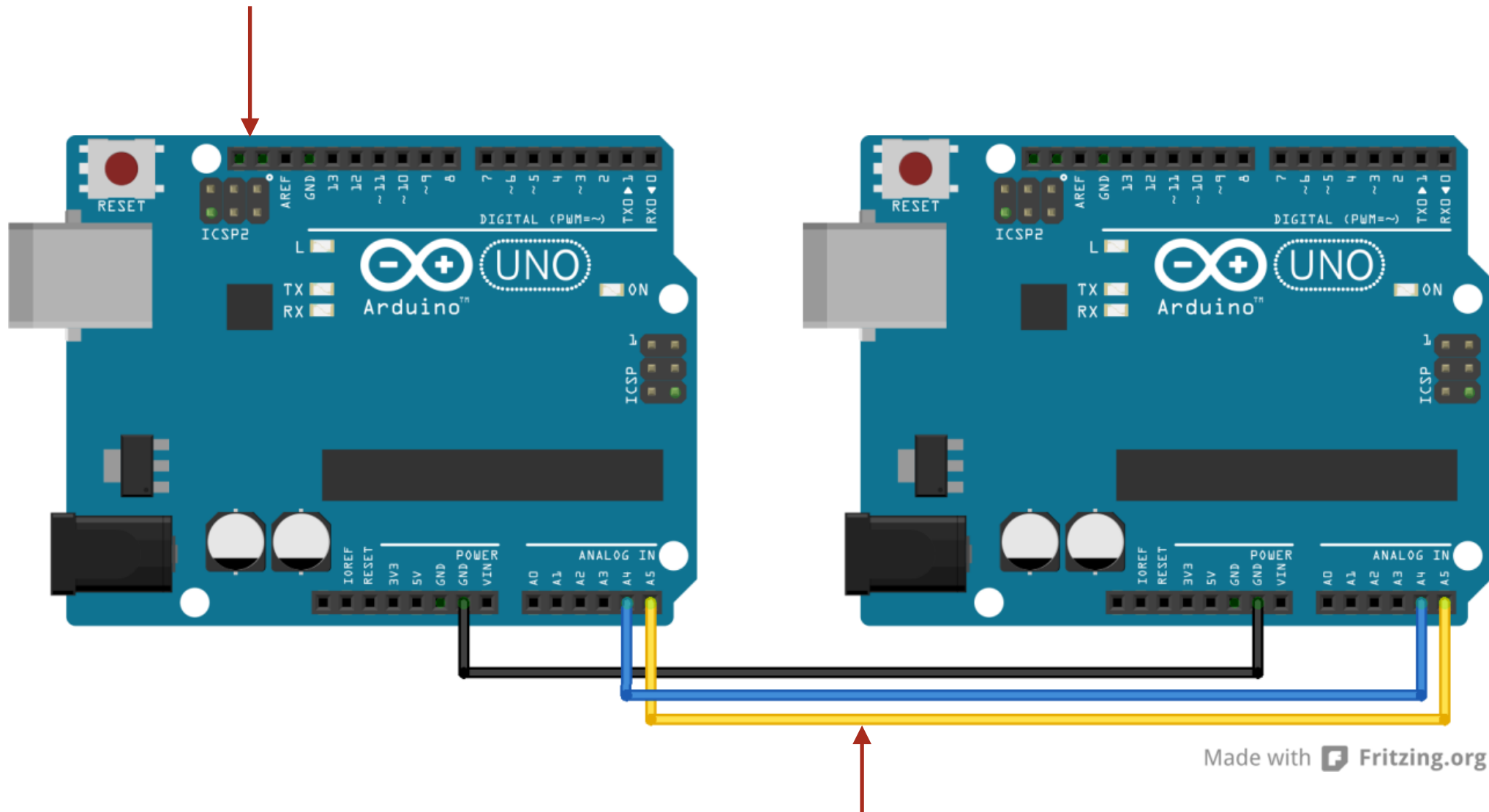
1. 마스터에서 시작 비트를 전송한다.
2. 마스터에서 통신하려는 슬레이브 장치의 7비트 주소를 보낸다.
3. I²C 장치의 레지스터에 데이터를 쓸 것인지 아니면 읽어 올 것인지에 따라 마스터에서 1비트의 읽기 명령(1)이나 쓰기 명령(0)을 보낸다.
4. 슬레이브에서는 acknowledge 또는 ACK(LOW 신호)로 응답한다.
5. 마스터는 I²C 장치의 레지스터를 읽겠다는 1바이트 명령을 쓰기 모드에서 전송하고, 슬레이브에서는 ACK 비트로 응답한다.
6. 마스터는 I²C 기기에서 전송되는 1바이트 정보를 읽기 모드에서 수신한다. 1바이트 씩 읽고 나면 슬레이브에 ACK 비트를 전송한다.
7. 마스터에서 STOP 비트를 전송하여 마스터와 슬레이브 기기 간 통신을 완료한다.

두 대의 Arduino간에 I²C통신하기

Arduino간의 I²C 통신

<https://www.arduino.cc/en/Tutorial/MasterWriter>

Uno R3에서는 A4, A5대신 AREF 옆의 전용 SDA, SCL 핀을 사용해도 된다.



A4가 SDA이고 A5가 SCL이다.

Wire Library를 사용하면 SDA와 SCL에 아두이노 내장 pull-up저항이 활성화된다.
따라서 pull-up 저항을 추가로 설치할 필요는 없다.

Arduino간의 I²C 통신

- **SDA와 SCL** 핀은 각각 아날로그 입력4번(**A4**)과 5번(**A5**) 핀에 연결한다. **Uno R3**버전에서는 **SDA**와 **SCL** 전용 핀이 **AREF** 핀 옆에 추가 되었다.
- 아날로그 **A4**와 **A5**는 아두이노의 하드웨어 **I²C** 인터페이스와 내장 **ADC**가 다중으로 분기된 회선이다.
- 아두이노 프로그램 코드에서 **Wire** 라이브러리를 초기화시키면 **A4**와 **A5** 핀은 **ATMega**의 **I²C** 컨트롤러에 연결된다.
- **Wire** 라이브러리를 초기화 시키면 **A4**와 **A5** 핀은 아날로그 입력으로는 사용할 수 없다.

Master가 전송하고 slave가 수신: Master Sketch

<https://www.arduino.cc/en/Tutorial/MasterWriter>

```
#include <Wire.h>

void setup()
{
  Wire.begin(); // i2c 버스에 참여 (마스터의 경우, 주소를 적지 않아도 된다)
}

byte x = 0;

void loop()
{
  Wire.beginTransaction(4); // 4번 슬레이브 장치로 전송 시작
  Wire.write("x is ");      // 5 바이트 전송
  Wire.write(x);             // 그 다음 1 바이트 전송
  Wire.endTransmission();    // 전송 종료

  x++;
  delay(500);
}
```

Master가 전송하고 slave가 수신: Master Sketch

<https://www.arduino.cc/en/Tutorial/MasterWriter>

```
#include <Wire.h>

void setup()
{
  Wire.begin(4); // 자신의 주소를 4번으로 설정하고 i2c 버스에 참여
  Wire.onReceive(receiveEvent); // 수신 이벤트 함수 등록
  Serial.begin(9600);
}

void loop()
{
  delay(100);
}

// 다음 함수는 마스터가 데이터를 전송해 올 때마다 호출됨
void receiveEvent(int howMany)
{
  while(1 < Wire.available()) // 마지막 한 개의 데이터를 제외하고 읽어 들임
  {
    char c = Wire.read(); // byte를 읽어 char로 변환
    Serial.print(c);
  }
  int x = Wire.read(); // byte를 읽어 int로 변환
  Serial.println(x);
}
```

Master가 수신하고 slave가 송신: Master Sketch

<https://www.arduino.cc/en/Tutorial/MasterReader>

```
#include <Wire.h>
```

```
void setup()
```

```
{
```

```
  Wire.begin(); // i2c 버스에 참여
```

```
  Serial.begin(9600);
```

```
}
```

```
void loop()
```

```
{
```

```
  Wire.requestFrom(2, 6); // 슬레이브 장치 #2에게 6 바이트의 데이터를 요청
```

```
  while(Wire.available()) // 슬레이브가 요청한 바이트 수를 보내라는 보장은 없음. 데이터가  
  있는지 점검하면서 읽음
```

```
  {
```

```
    char c = Wire.read(); // byte를 읽어 char로 변환
```

```
    Serial.print(c);
```

```
  }
```

```
  delay(500);
```

```
}
```

Master가 수신하고 slave가 송신: Master Sketch

<https://www.arduino.cc/en/Tutorial/MasterReader>

```
#include <Wire.h>
```

```
void setup()  
{  
  Wire.begin(2);           // 자신의 주소를 2번으로 설정하고 i2c 버스에 참여  
  Wire.onRequest(requestEvent); // 송신 요청을 받았을 때 호출할 함수 등록  
}
```

```
void loop()  
{  
  delay(100);  
}
```

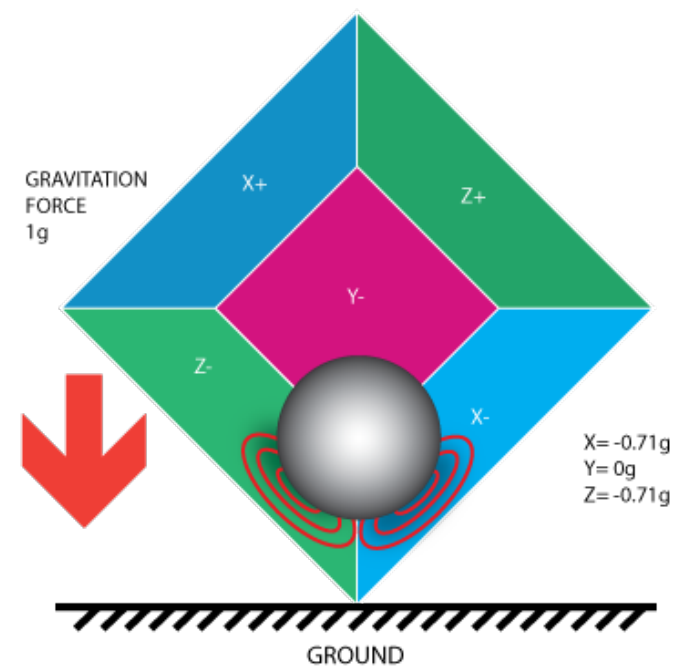
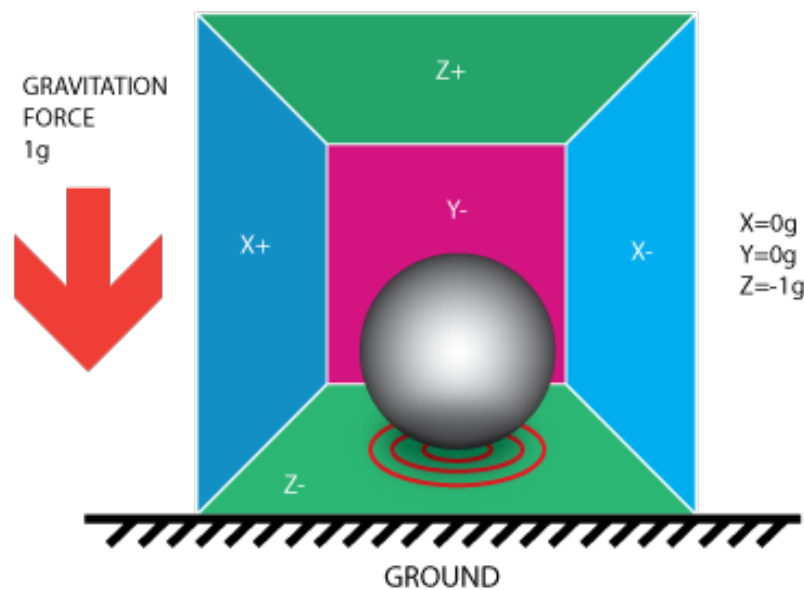
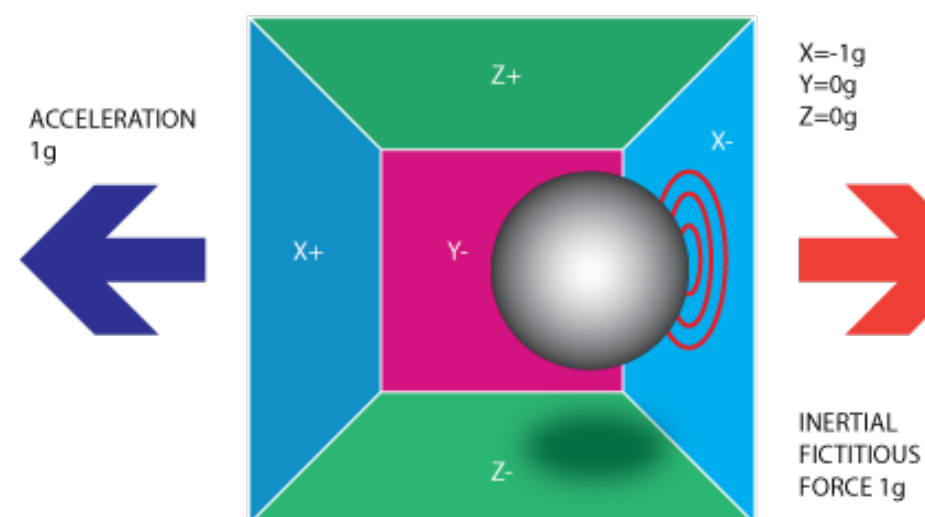
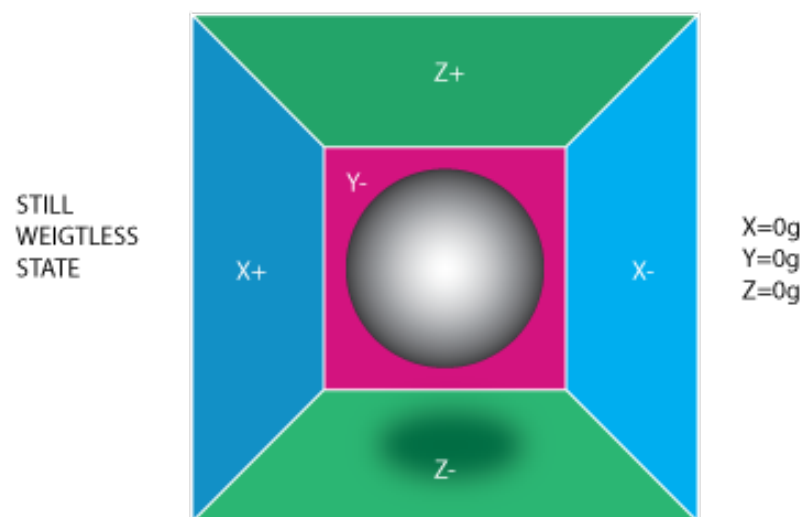
```
// 마스터가 자신에게 데이터를 요청할 때 호출됨
```

```
void requestEvent()  
{  
  Wire.write("hello");  
}
```

LSM9DS0 IMU 센서 사용하기

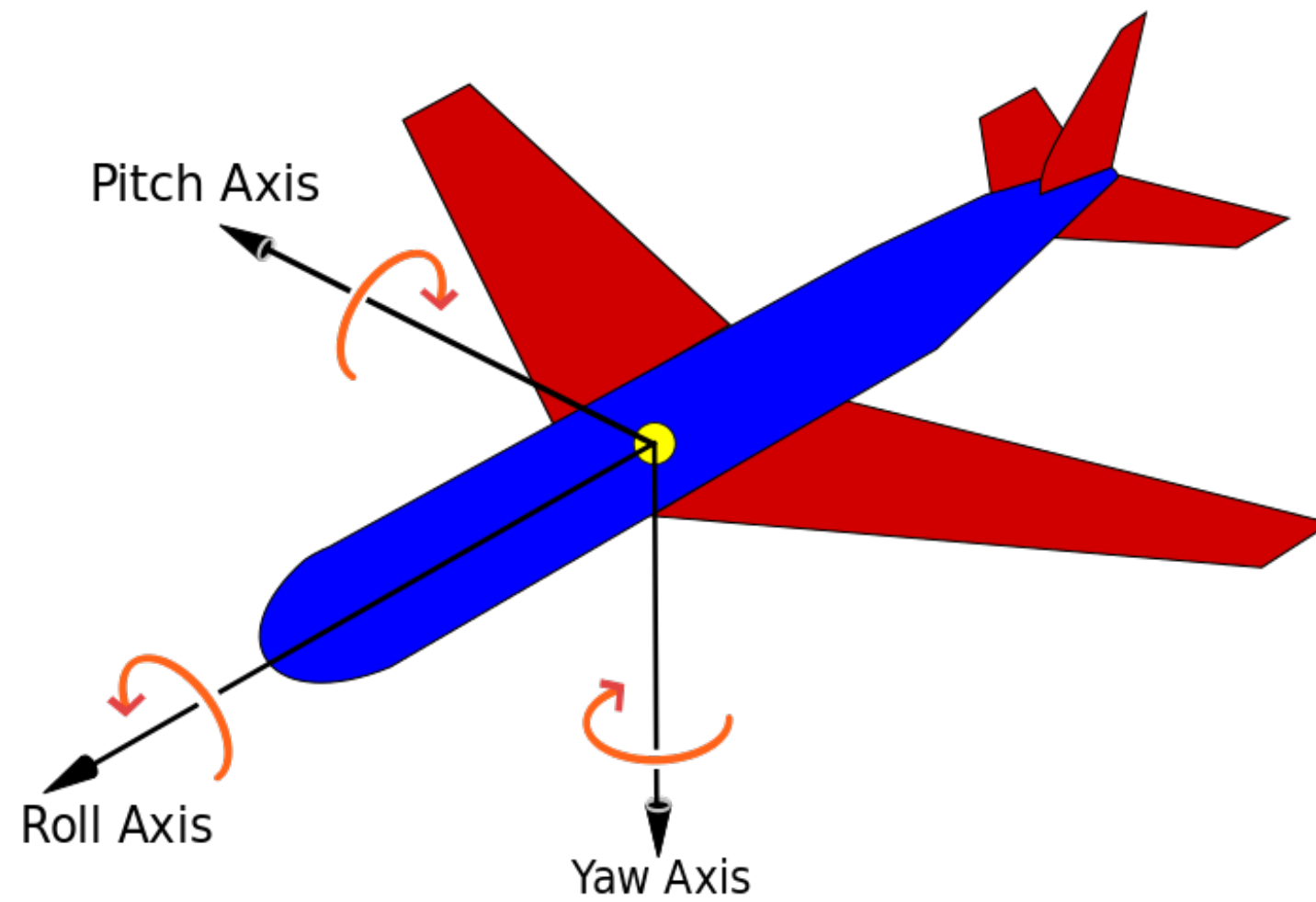
- **Inertial Measurement Unit**
- **MEMS 기술을 사용한 초소형의 저렴한 센서**
- **일반적 구성**
 - Magnetometer (자기센서): 나침반(compass)의 역할
 - Accelerometer (가속도 센서): 정지상태에서는 중력가속도를 측정
 - Gyroscope (자이로스코프, 각속도 센서)

참고: Accelerometer

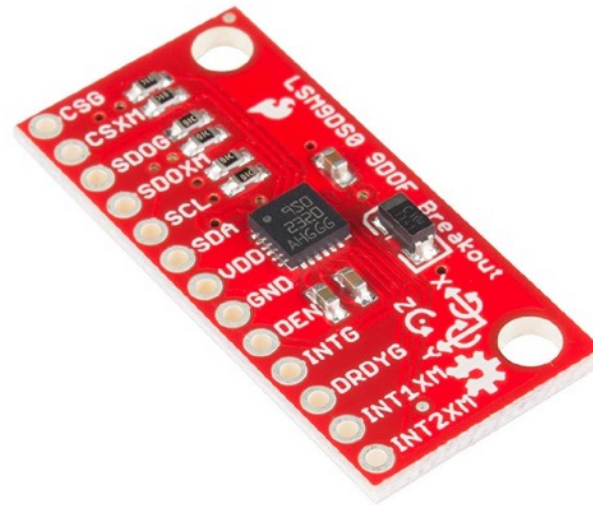


정지상태에서 accelerometer는 중력가속도를 출력한다.

Yaw, Pitch, Roll

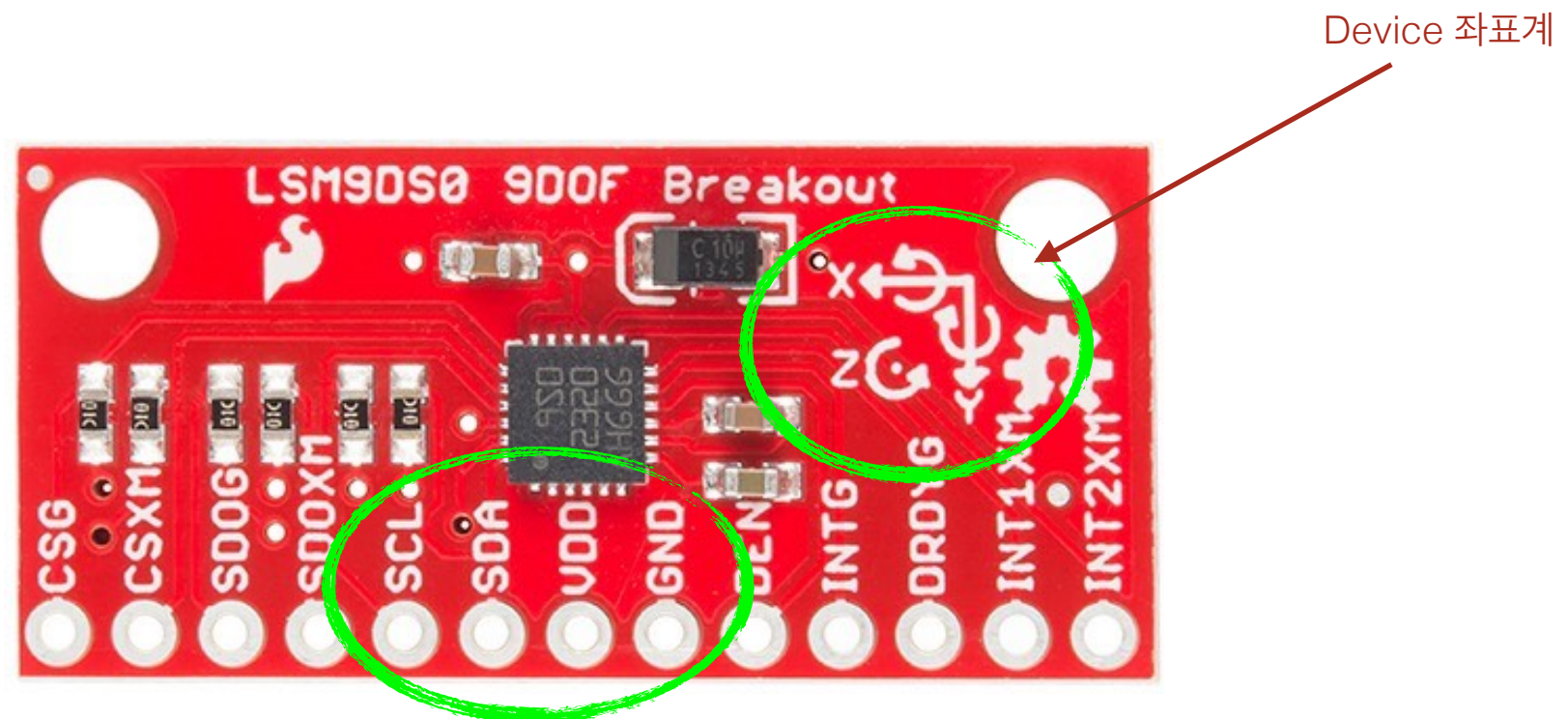
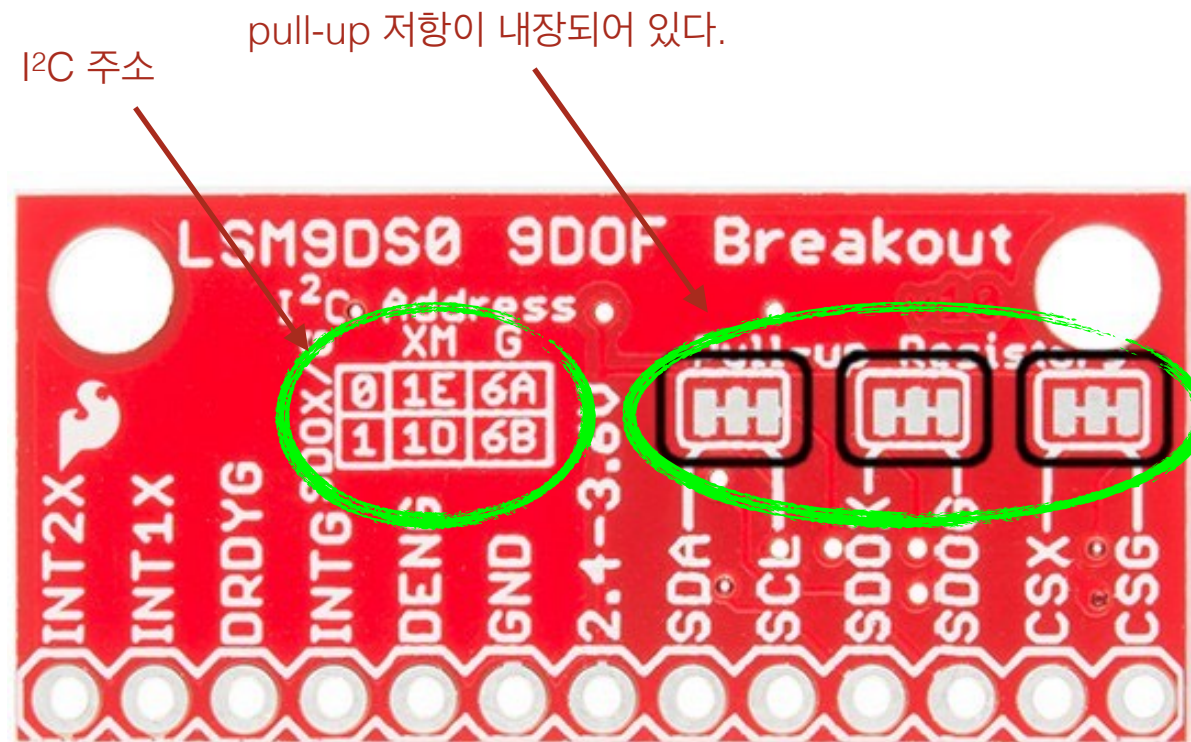


IMU: LSM9DS0

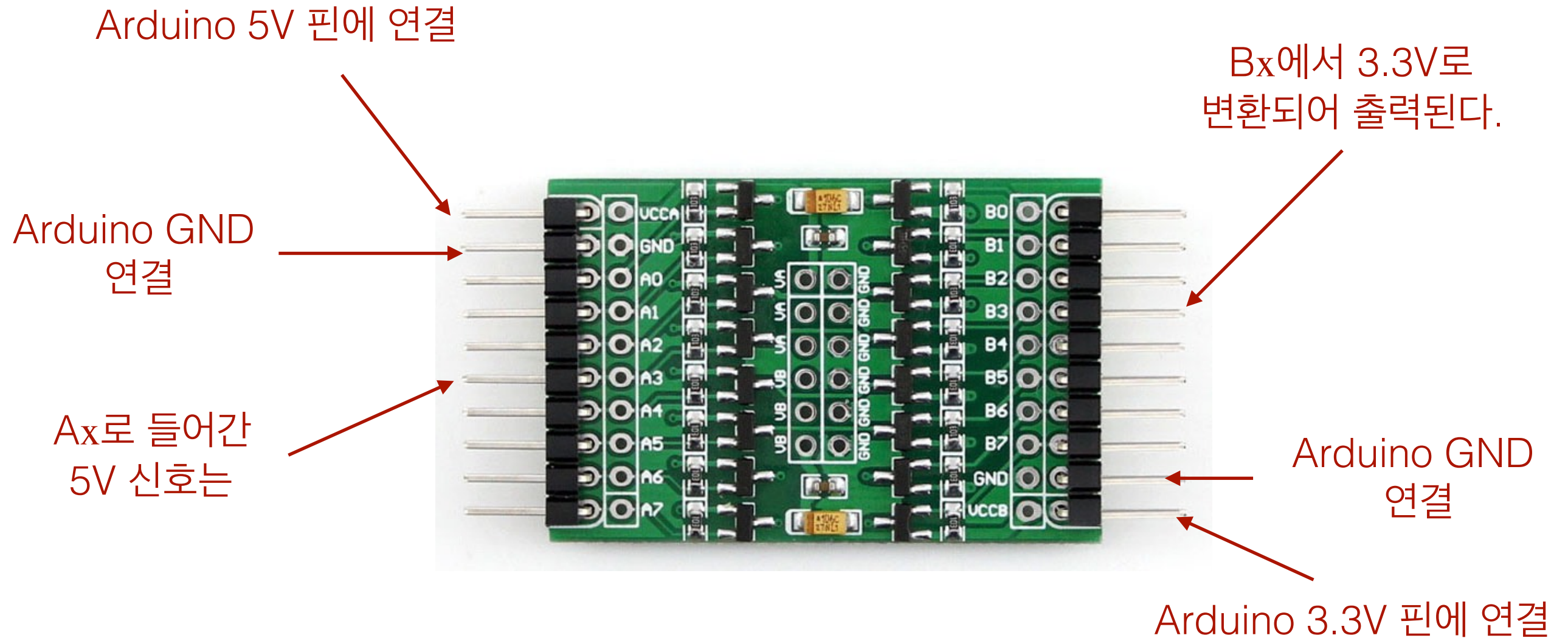


- **9 DOF(Degrees of Freedom) IMU**
 - 3축 자기센서 (magnetometer): 자기장의 방향과 크기를 측정
 - 3축 가속도센서 (accelerometer): 중력가속도의 크기와 방향을 측정하는 용도
 - 3축 자이로스코프(gyroscope): 회전의 각속도(angular velocity)를 측정
- **SPI/I²C serial interface를 동시에 지원**
- **Voltage supply: 2.4V ~ 3.6V (아두이노에서 사용하기 위해서는 Level converter가 필요)**
- **SDA와 SCL 라인에 10k Ω pull-up 저항이 이미 내장되어 있음**

IMU: LSM9DS0



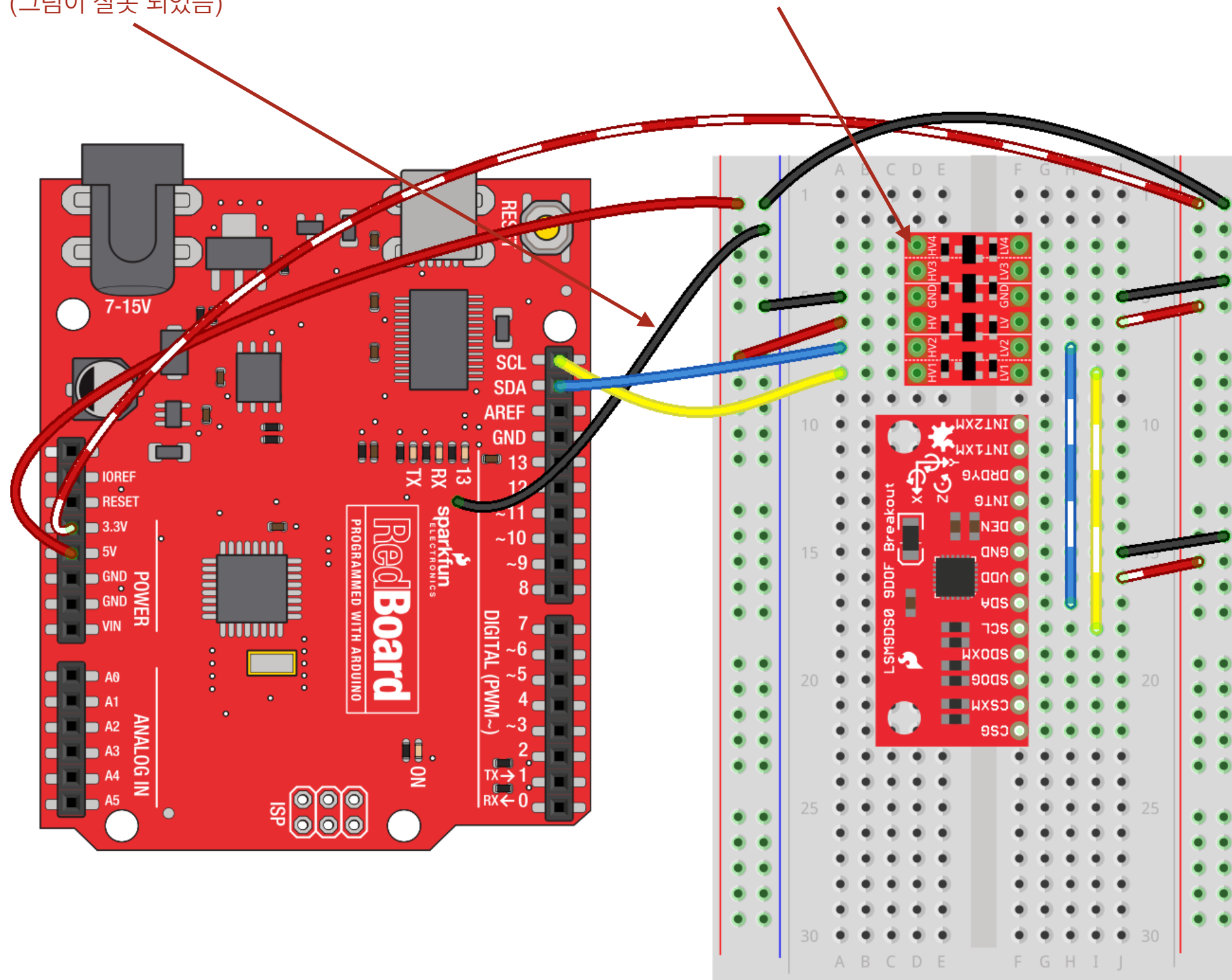
Logic Level Converter



I²C 연결

GND에 연결
(그림이 잘못 되었음)

Bidirectional Logic Level Converter



https://learn.sparkfun.com/tutorials/lsm9ds0-hookup-guide?_ga=1.78320541.841774440.1439019361

SFE_LSM9DS0 라이브러리

- 라이브러리 다운로드

- 아래 링크된 페이지에서 "Download ZIP" 클릭

- https://github.com/sparkfun/SparkFun_LSM9DS0_Arduino_Library

- **Arduino IDE에서 "Sketch->Include Library->Manage Libraries"**
메뉴를 이용하여 설치

Sketch

```
#include <SPI.h> // Included for SFE_LSM9DS0 library
#include <Wire.h>
#include <SFE_LSM9DS0.h>
```

```
#define LSM9DS0_XM 0x1D // I2C Address of Acc and Mag
#define LSM9DS0_G 0x6B // I2C Address of Gyro
```

```
LSM9DS0 dof(MODE_I2C, LSM9DS0_G, LSM9DS0_XM);
```

```
#define PRINT_CALCULATED
```

```
#define PRINT_SPEED 500 // 500 ms between prints
```

<https://learn.sparkfun.com/tutorials/lsm9ds0-hookup-guide?ga=1.247322509.841774440.1439019361>

```
void setup()
{
```

```
  Serial.begin(115200); // Start serial at 115200 bps
```

```
  uint16_t status = dof.begin();
```

```
  Serial.print("LSM9DS0 WHO_AM_I's returned: 0x");
```

```
  Serial.println(status, HEX);
```

← 16진수로 출력

```
  Serial.println("Should be 0x49D4"); // Device ID in WHO_AM_I registers
```

```
  Serial.println();
```

← status가 0x49D4가 아니면 오류


```
void loop()
{
    printGyro(); // Print "G: gx, gy, gz"
    printAccel(); // Print "A: ax, ay, az"
    printMag(); // Print "M: mx, my, mz"

    printHeading((float) dof.mx, (float) dof.my); ← Magnetometer로 부터 yaw를 계산

    printOrientation(dof.calcAccel(dof.ax), dof.calcAccel(dof.ay),
                    dof.calcAccel(dof.az)); ↑ Accelerometer로 부터 roll과 pitch를 계산
    Serial.println();

    delay(PRINT_SPEED);
}
```

```
void printGyro()
{
    dof.readGyro();
    Serial.print("G: ");
#ifdef PRINT_CALCULATED
    // If you want to print calculated values, you can use the
    // calcGyro helper function to convert a raw ADC value to DPS(°/s).
    Serial.print(dof.calcGyro(dof.gx), 2);
    Serial.print(", ");
    Serial.print(dof.calcGyro(dof.gy), 2);
    Serial.print(", ");
    Serial.println(dof.calcGyro(dof.gz), 2);
#elif defined PRINT_RAW
    Serial.print(dof.gx);
    Serial.print(", ");
    Serial.print(dof.gy);
    Serial.print(", ");
    Serial.println(dof.gz);
#endif
}
```



소수점 이하 2자리까지


```
void printAccel()
{
    dof.readAccel();
    // Now we can use the ax, ay, and az variables as we please.
    // Either print them as raw ADC values, or calculated in g's.
    Serial.print("A: ");
#ifdef PRINT_CALCULATED
    // If you want to print calculated values, you can use the
    // calcAccel helper function to convert a raw ADC value to g's.
    Serial.print(dof.calcAccel(dof.ax), 2);
    Serial.print(", ");
    Serial.print(dof.calcAccel(dof.ay), 2);
    Serial.print(", ");
    Serial.println(dof.calcAccel(dof.az), 2);
#elif defined PRINT_RAW
    Serial.print(dof.ax);
    Serial.print(", ");
    Serial.print(dof.ay);
    Serial.print(", ");
    Serial.println(dof.az);
#endif
}
```

```
void printMag()  
{  
    dof.readMag();  
    // Now we can use the mx, my, and mz variables as we please.  
    // Either print them as raw ADC values, or calculated in Gauss.  
    Serial.print("M: ");  
#ifdef PRINT_CALCULATED  
    // If you want to print calculated values, you can use the  
    // calcMag helper function to convert a raw ADC value to Gauss.  
    Serial.print(dof.calcMag(dof.mx), 2);  
    Serial.print(", ");  
    Serial.print(dof.calcMag(dof.my), 2);  
    Serial.print(", ");  
    Serial.println(dof.calcMag(dof.mz), 2);  
#elif defined PRINT_RAW  
    Serial.print(dof.mx);  
    Serial.print(", ");  
    Serial.print(dof.my);  
    Serial.print(", ");  
    Serial.println(dof.mz);  
#endif  
}
```

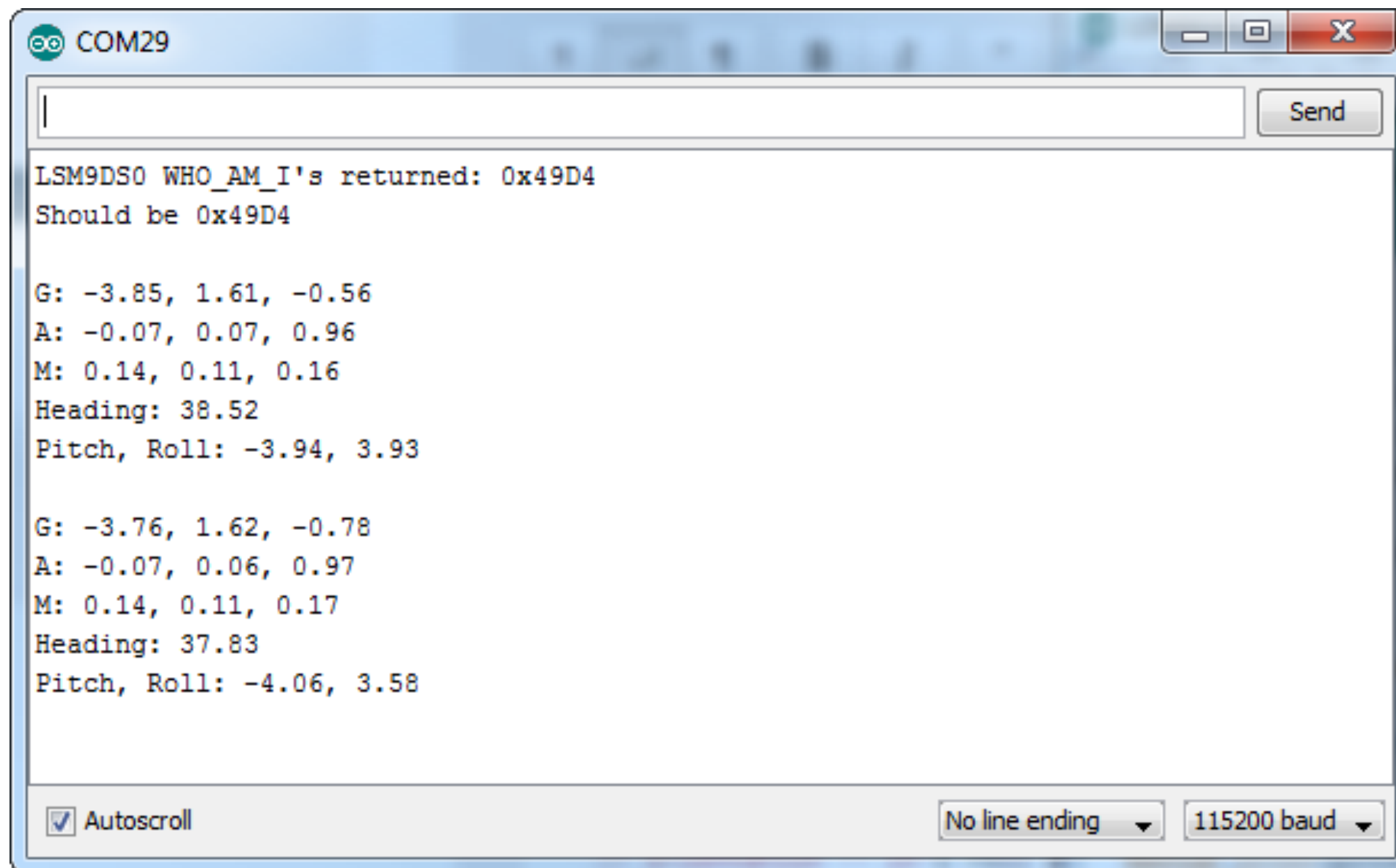
```
void printHeading(float hx, float hy)
{
    float heading;
    if (hy > 0)
    {
        heading = 90 - (atan(hx / hy) * (180 / PI));
    }
    else if (hy < 0)
    {
        heading = - (atan(hx / hy) * (180 / PI));
    }
    else // hy = 0
    {
        if (hx < 0) heading = 180;
        else heading = 0;
    }
    Serial.print("Heading: ");
    Serial.println(heading, 2);
}
```

```
void printOrientation(float x, float y, float z)
{
    float pitch, roll;

    pitch = atan2(x, sqrt(y * y) + (z * z));
    roll = atan2(y, sqrt(x * x) + (z * z));
    pitch *= 180.0 / PI;
    roll *= 180.0 / PI;

    Serial.print("Pitch, Roll: ");
    Serial.print(pitch, 2);
    Serial.print(", ");
    Serial.println(roll, 2);
}
```

실행결과



The screenshot shows a serial terminal window titled 'COM29'. It has a text input field at the top with a 'Send' button. The main area displays two sets of sensor data. The first set includes a status message, a gravity vector (G), acceleration vector (A), magnetic field vector (M), heading, and pitch/roll values. The second set follows a similar format with slightly different numerical values. At the bottom, there are checkboxes for 'Autoscroll' (checked) and dropdown menus for 'No line ending' and '115200 baud'.

```
COM29

LSM9DS0 WHO_AM_I's returned: 0x49D4
Should be 0x49D4

G: -3.85, 1.61, -0.56
A: -0.07, 0.07, 0.96
M: 0.14, 0.11, 0.16
Heading: 38.52
Pitch, Roll: -3.94, 3.93

G: -3.76, 1.62, -0.78
A: -0.07, 0.06, 0.97
M: 0.14, 0.11, 0.17
Heading: 37.83
Pitch, Roll: -4.06, 3.58

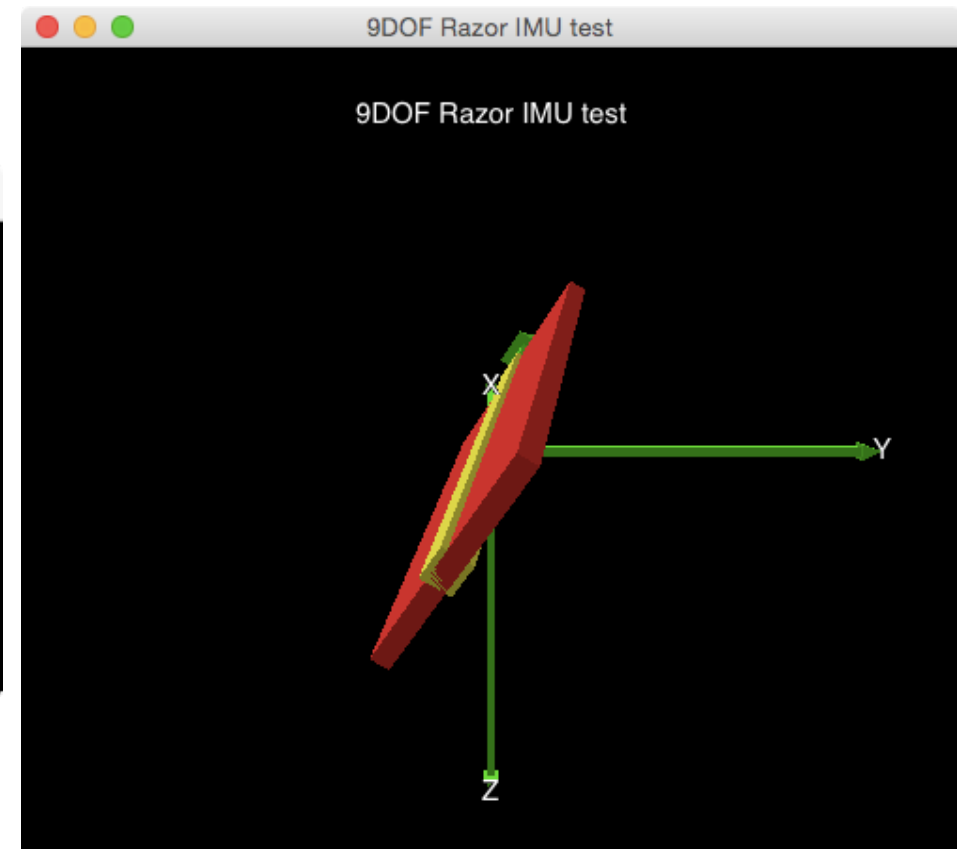
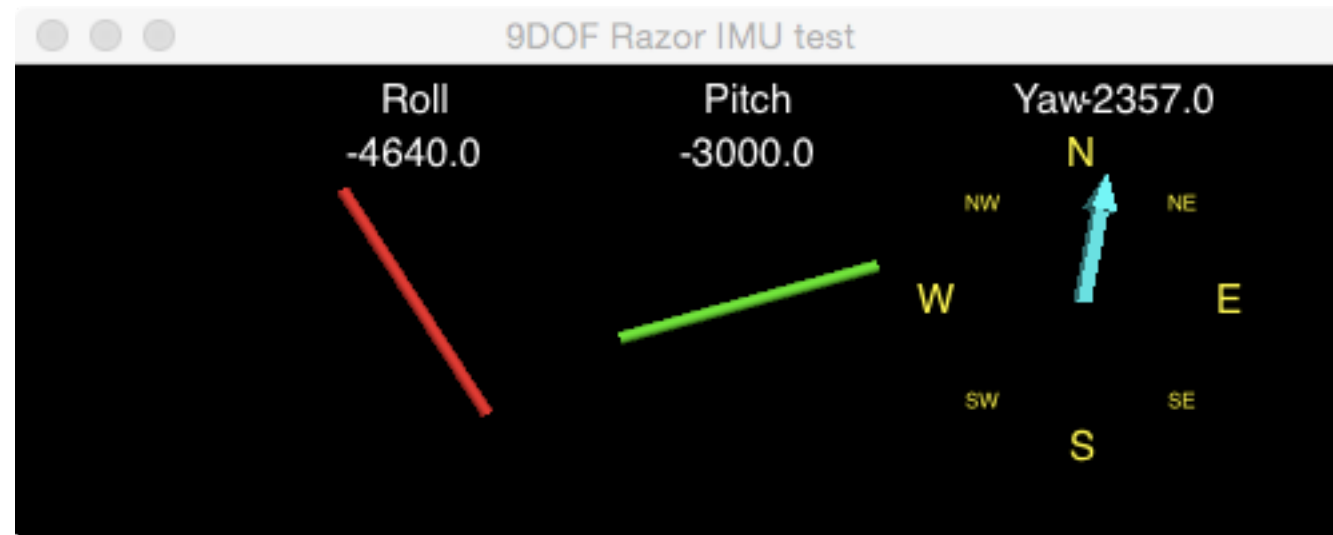
☒ Autoscroll
No line ending ▼ 115200 baud ▼
```

SFE_LSM9DS0 라이브러리 살펴보기

SFE_LSM9DS0 라이브러리 살펴보기

- **LSM9DS0 Data Sheet 참조**
- **SFE_LSM9DS0 라이브러리**
 - SFE_LSM9DS0.h와 SFE_LSM9DS0.cpp 파일로 구성
- **온도를 읽는 코드를 스케치에 추가하라.**

IMU Visualizer



• **ahrs-visualizer**

- <http://edge.rit.edu/content/P11015/public/Visual%20Python%20for%20Gyroscope%20Utilization.py>

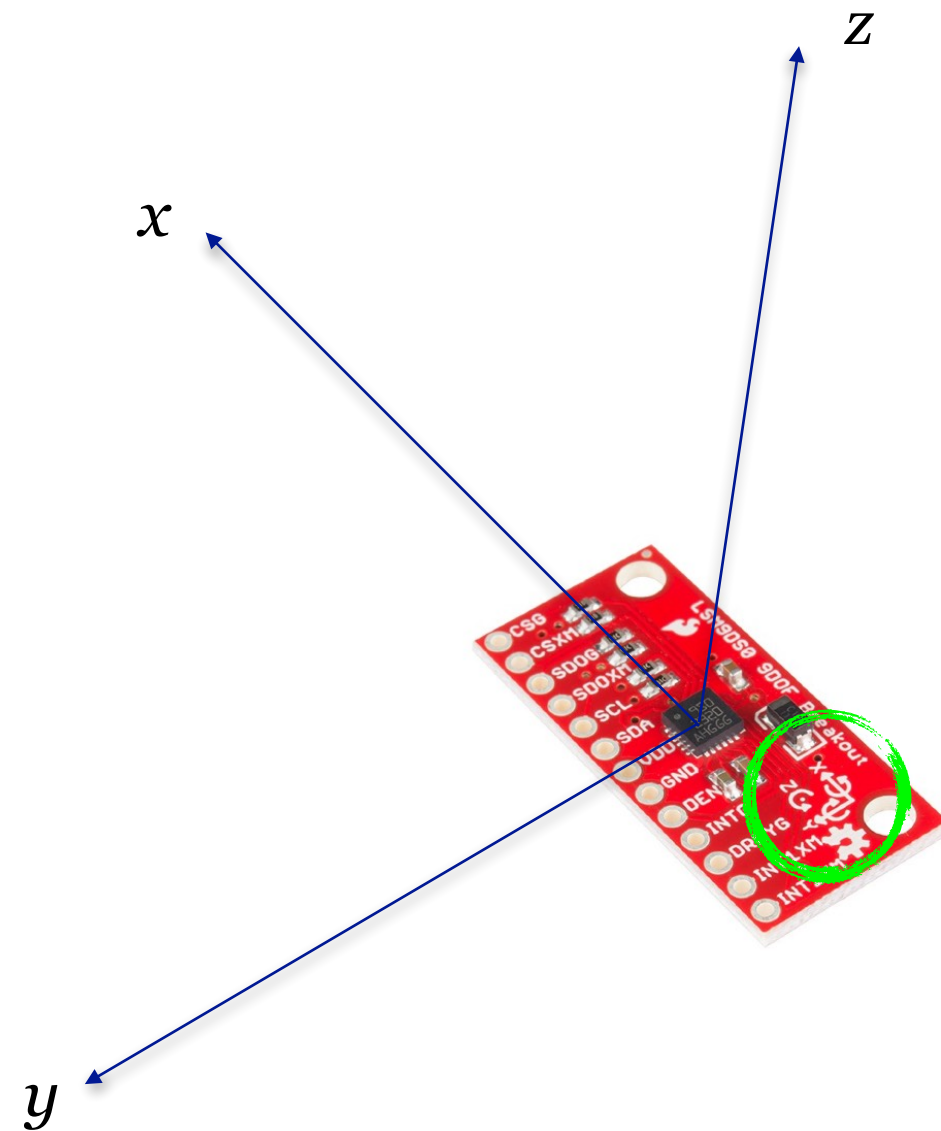
• **sketch:**

- <https://dl.dropboxusercontent.com/u/16142350/ESP2015/imu.ino>

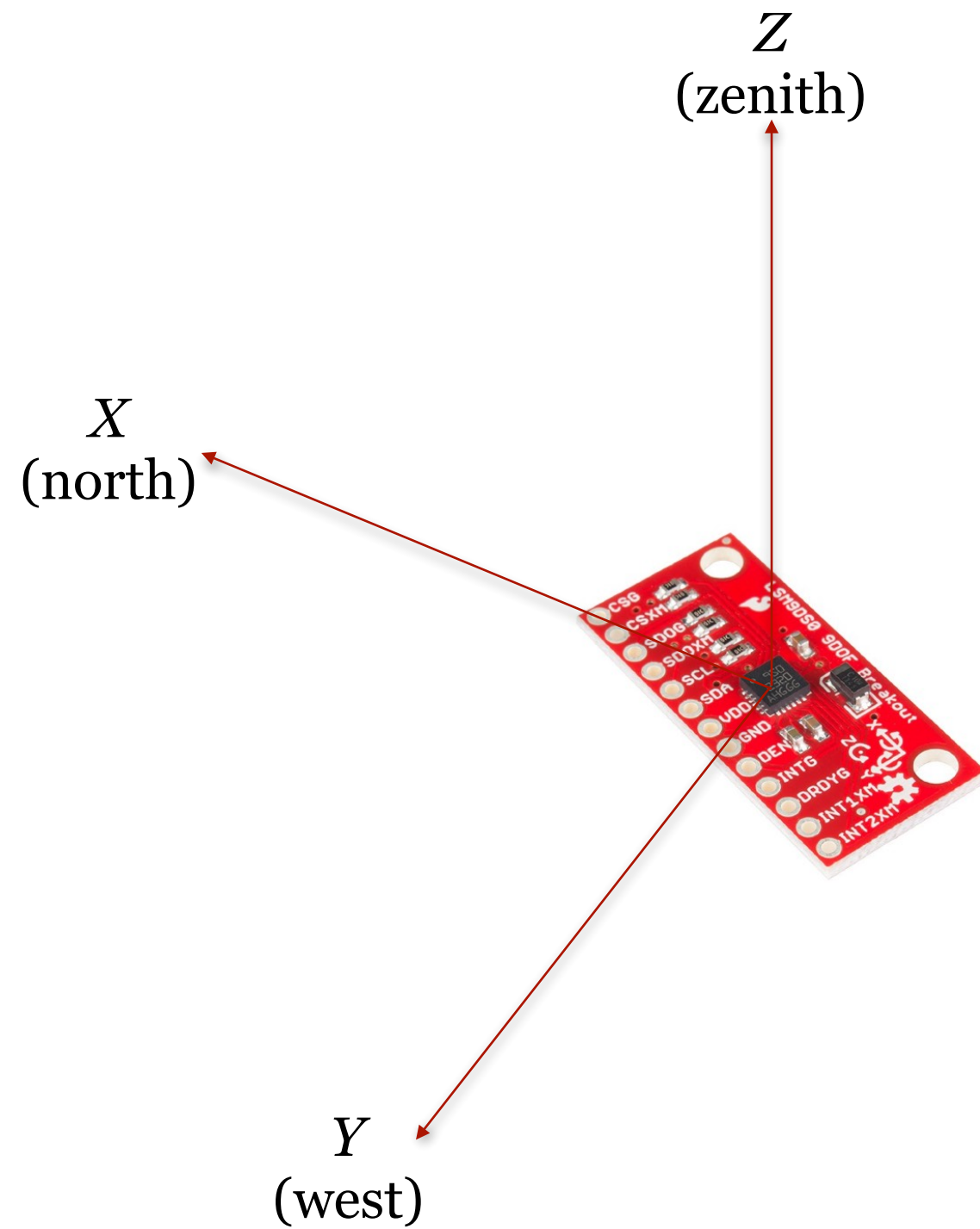
참조: Inertial Measurement Unit (IMU)

- **Inertial Measurement Unit**
- **MEMS 기술을 사용한 초소형의 저렴한 센서**
- **일반적 구성**
 - Magnetometer (자기센서)
 - Accelerometer (가속도 센서)
 - Gyroscope (자이로스코프, 각속도 센서)

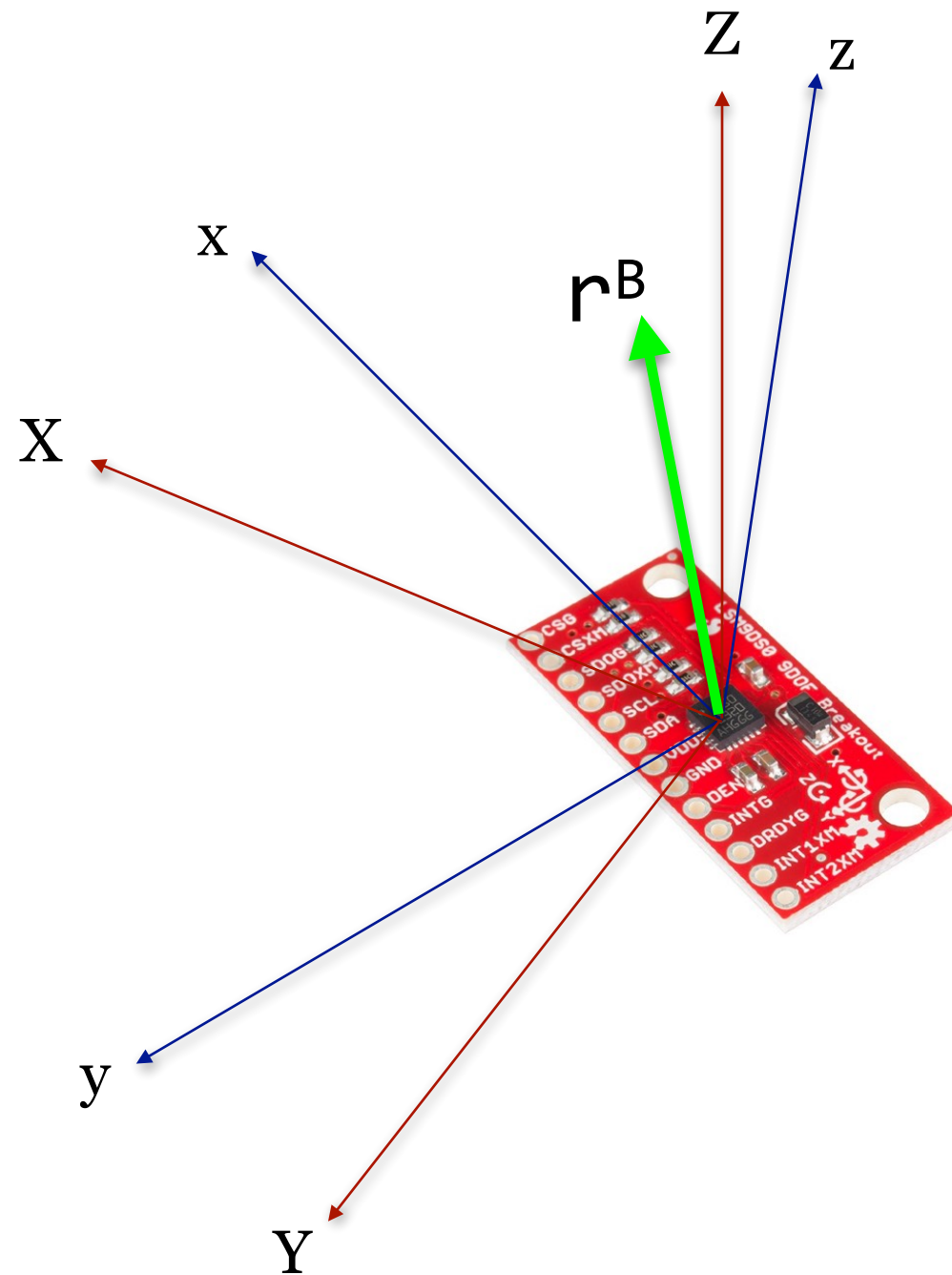
Body Coordinate System



Global Coordinate System



Attitude (Pose) Estimation



Body coordinate system 상의 임의의 한 벡터 r^B 의 global coordinate system 상의 좌표 r^G 를 알아내는 것

가령 pitch는 $r^B=(0,0,1)$ 에 해당하는 r^G 에 의해서 결정

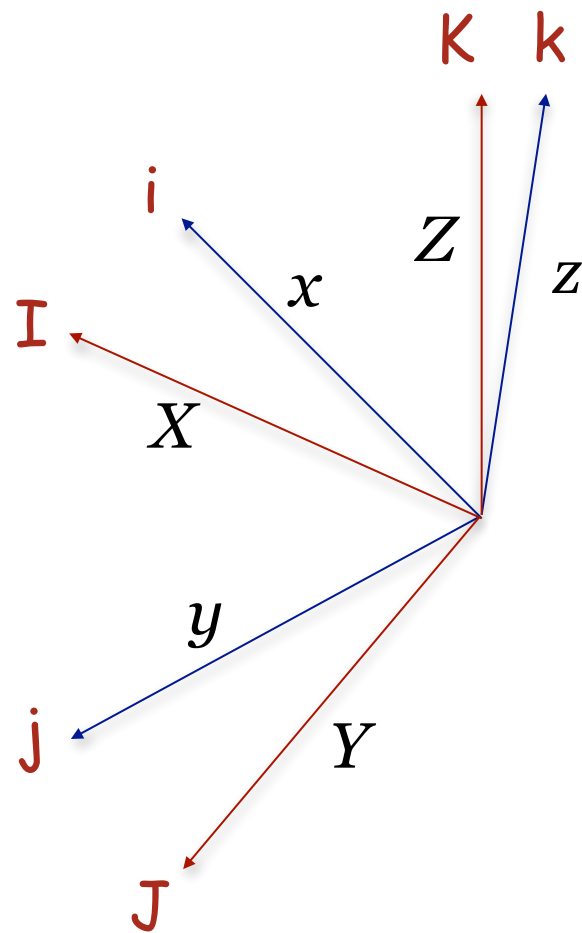
$$r^G = (r_x^G, r_y^G, r_z^G)$$

$$\tan \theta = \frac{r_x^G}{\sqrt{(r_y^G)^2 + (r_z^G)^2}}$$

$$\theta = \tan^{-1} \frac{r_x^G}{\sqrt{(r_y^G)^2 + (r_z^G)^2}}$$

pitch

Direction Cosine Matrix

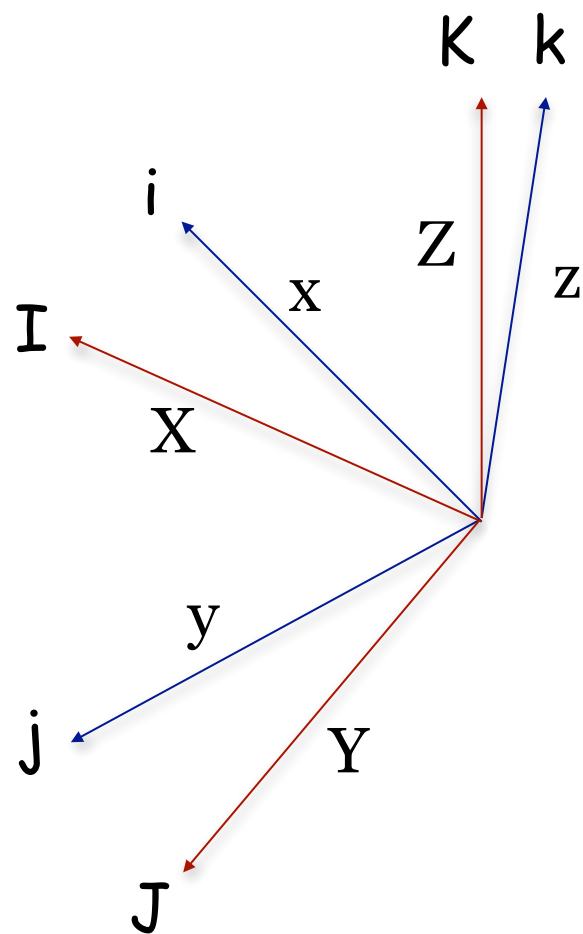


i, I, j, J, k, K 를 각각 6개의 축방향의 단위 벡터라고 하자.

$$I^G = \{1, 0, 0\}^T, J^G = \{0, 1, 0\}^T, K^G = \{0, 0, 1\}^T$$

$$i^B = \{1, 0, 0\}^T, j^B = \{0, 1, 0\}^T, k^B = \{0, 0, 1\}^T$$

Direction Cosine Matrix



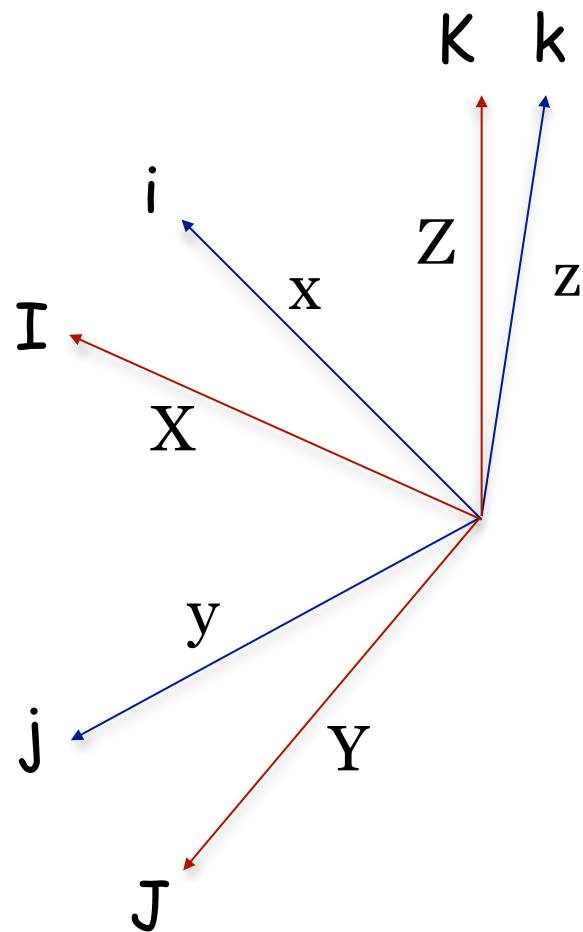
$$\mathbf{i}^G = (i_x^G, i_y^G, i_z^G)^T$$

$$\begin{aligned} i_x^G &= |\mathbf{i}| \cos(X, \mathbf{i}) = \cos(I, \mathbf{i}) \\ &= |\mathbf{I}| |\mathbf{i}| \cos(I, \mathbf{i}) = \mathbf{I} \cdot \mathbf{i} \end{aligned}$$

$$\begin{aligned} i_y^G &= |\mathbf{i}| \cos(Y, \mathbf{i}) = \cos(J, \mathbf{i}) \\ &= |\mathbf{J}| |\mathbf{i}| \cos(J, \mathbf{i}) = \mathbf{J} \cdot \mathbf{i} \end{aligned}$$

$$\begin{aligned} i_z^G &= |\mathbf{i}| \cos(Z, \mathbf{i}) = \cos(K, \mathbf{i}) \\ &= |\mathbf{K}| |\mathbf{i}| \cos(K, \mathbf{i}) = \mathbf{K} \cdot \mathbf{i} \end{aligned}$$

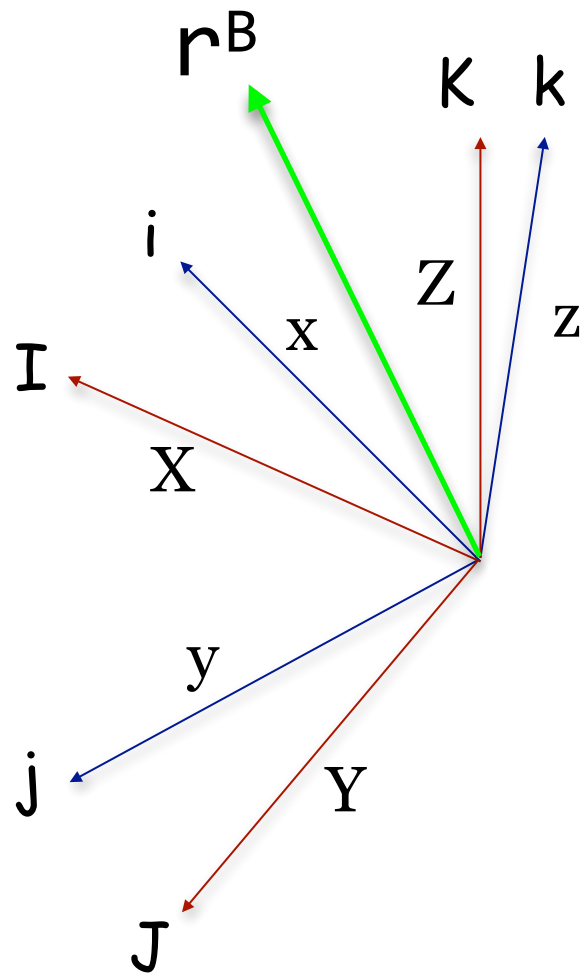
Direction Cosine Matrix



$$i^G = (I \cdot i, J \cdot i, K \cdot i)^T$$

$$\begin{aligned} [i^G, j^G, k^G] &= \begin{bmatrix} I \cdot i & I \cdot j & I \cdot k \\ J \cdot i & J \cdot j & J \cdot k \\ K \cdot i & K \cdot j & K \cdot k \end{bmatrix} \\ &= \begin{bmatrix} \cos(I, i) & \cos(I, j) & \cos(I, k) \\ \cos(J, i) & \cos(J, j) & \cos(J, k) \\ \cos(K, i) & \cos(K, j) & \cos(K, k) \end{bmatrix} \\ &= \text{DCM}^G \end{aligned}$$

Direction Cosine Matrix



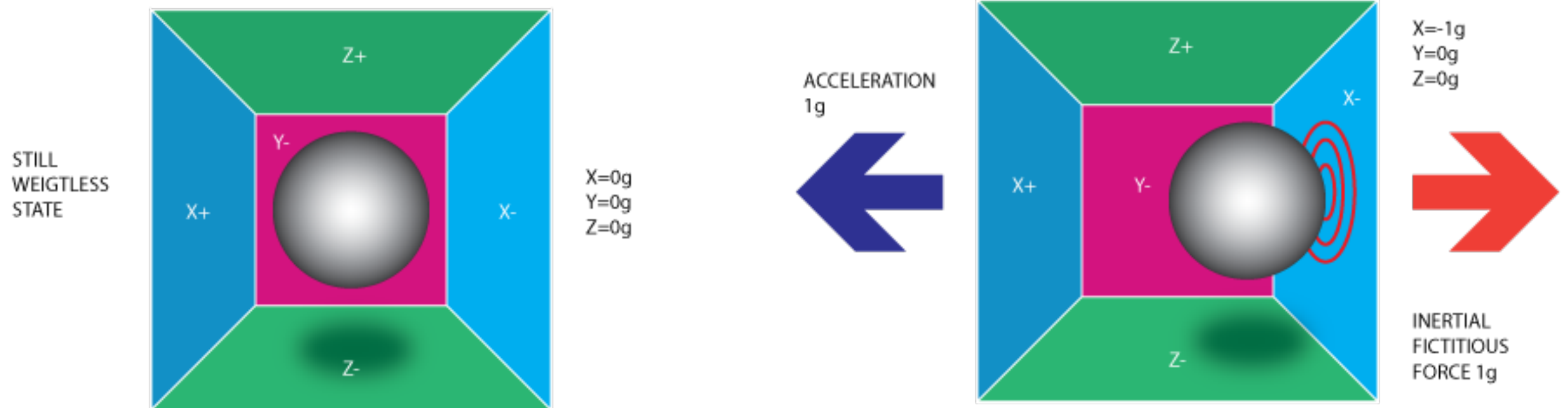
$$r^G = \text{DCM}^G r^B$$

$$r^B = (\text{DCM}^G)^{-1} r^G = (\text{DCM}^G)^T r^G = \text{DCM}^B r^G$$

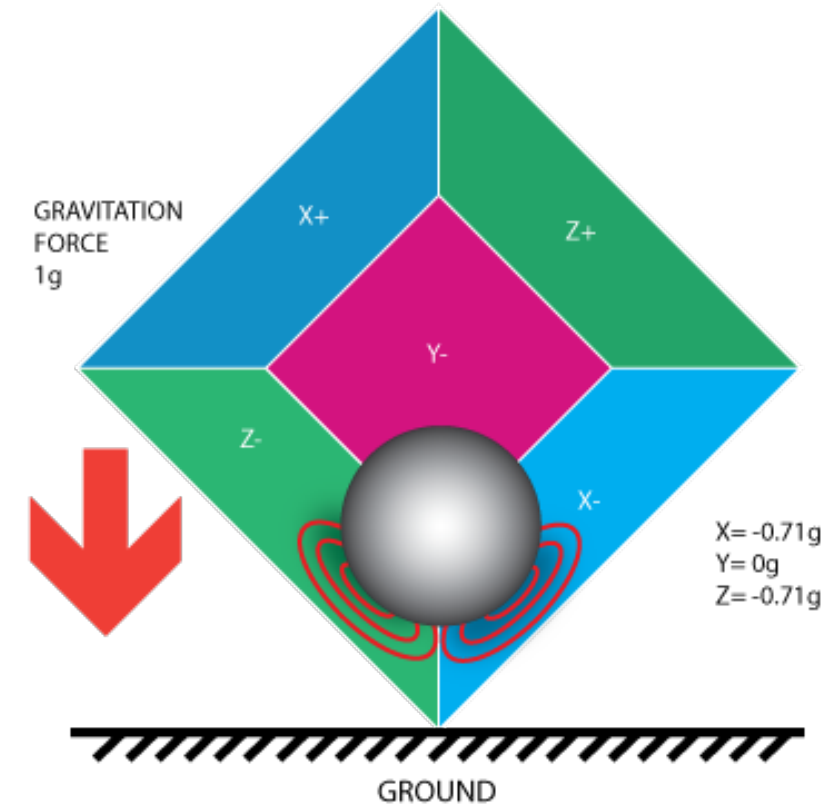
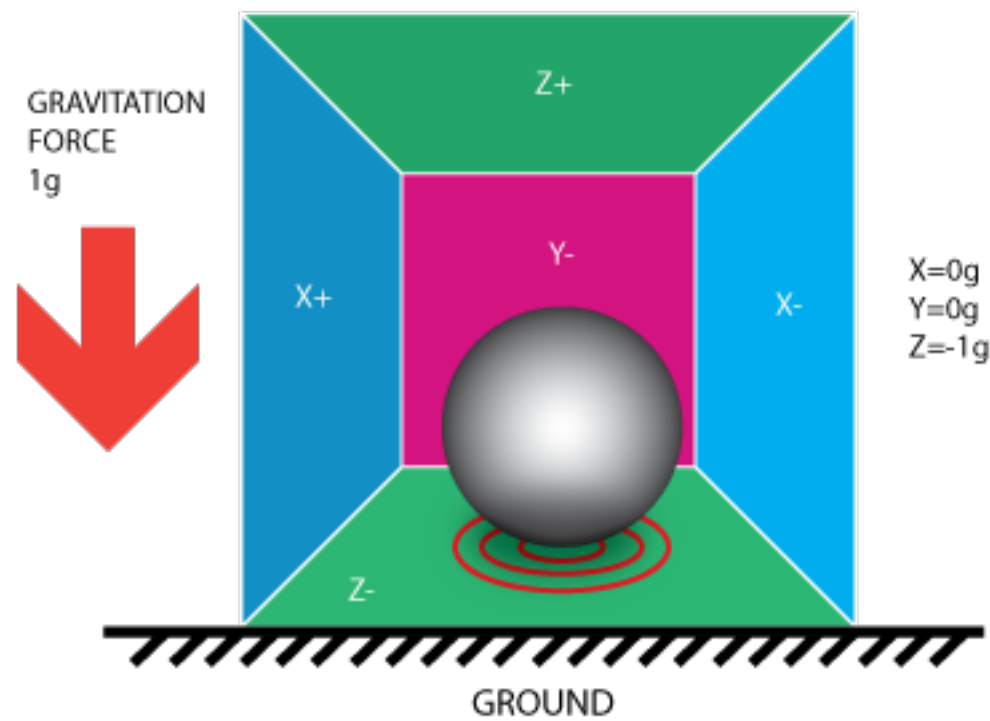
Magnetometer

- Pose estimation에서 magnetometer는 나침반(compass)의 역할
- 즉 global coordinate system상에서 정북방향(X축)의 벡터 I^G 에 해당하는 body coordinate system 상에서의 벡터 I^B 를 계산해 줌.
- 즉, magnetometer의 출력은 I^B

Accelerometer



Accelerometer



정지상태에서 accelerometer는 $-K^B$ 를 출력한다.

- \mathbf{I}^B 와 \mathbf{K}^B 로부터 오른손 법칙에 따라 \mathbf{J}^B 를 계산한다.

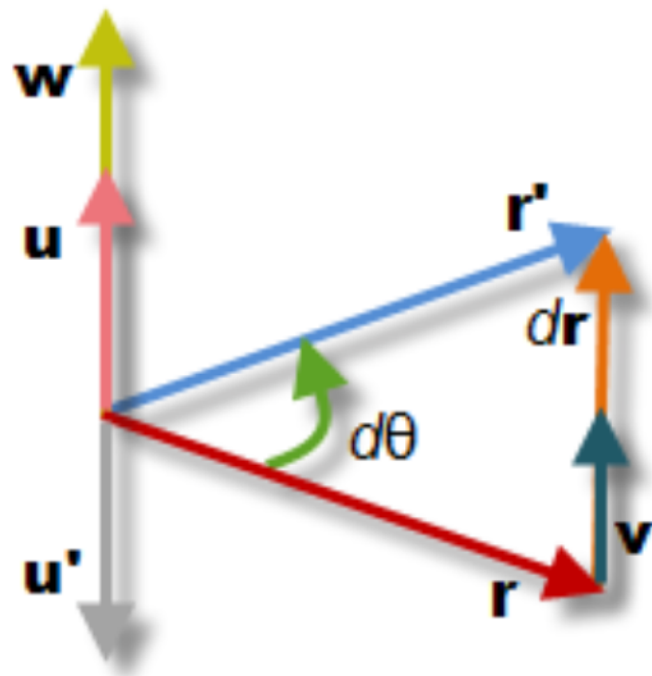
- $\mathbf{J}^B = \mathbf{K}^B \times \mathbf{I}^B$

- $\mathbf{I}^B, \mathbf{J}^B, \mathbf{K}^B$ 로 부터 \mathbf{DCM}^G 를 계산한다.

- $[\mathbf{I}^B, \mathbf{J}^B, \mathbf{K}^B] = \mathbf{DCM}^B = (\mathbf{DCM}^G)^T$

- Gyroscope ?

각속도(angular velocity)를 측정



definition of
angular velocity
(gyro의 출력)

$$u = \frac{(r \times r')}{|r \times r'|} = \frac{(r \times r')}{|r||r'| \sin(d\theta)} = \frac{(r \times r')}{|r|^2 \sin(d\theta)}$$

$$v = \frac{dr}{dt} = \frac{(r' - r)}{dt}$$

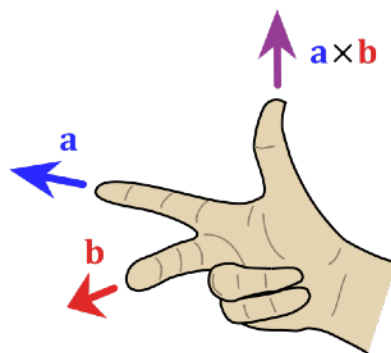
$$w = \frac{d\theta}{dt} u = \left(\frac{d\theta}{dt} \right) \frac{(r \times r')}{|r|^2 \sin(d\theta)}$$

$$\approx \frac{(r \times r')}{|r|^2 dt} \quad \leftarrow \sin d\theta \approx d\theta \text{ as } d\theta \rightarrow 0$$

$$= \frac{(r \times (r + dr))}{|r|^2 dt} = \frac{r \times (dr/dt)}{|r|^2}$$

$$w = r \times v / |r|^2 \quad \leftarrow (1)$$

$$v = w \times r$$



Sensor Fusion

$$K_{1G}^B \approx K_0^B + dtv = K_0^B + dt(w_g \times K_0^B) = K_0^B + (d\theta_g \times K_0^B) \quad \longleftarrow \text{여기서 } d\theta_g = dtw_g$$

$$w_a = K_0^B \times \frac{v_a}{|K_0^B|^2} \quad \longleftarrow \text{from (1)}$$

$$d\theta_a = dtw_a = K_0^B \times (K_{1A}^B - K_0^B) \quad \longleftarrow \text{since } v_a = (K_{1A}^B - K_0^B)/dt \text{ and } |K_0^B| = 1$$

$$d\theta_m = dtw_m = I_0^B \times (I_{1M}^B - I_0^B)$$

$$d\theta = (s_a d\theta_a + s_2 d\theta_g + s_m d\theta_m) / (s_a + s_g + s_m)$$

$$K_1^B \approx K_0^B + (d\theta \times K_0^B)$$

$$I_1^B \approx I_0^B + (d\theta \times I_0^B)$$

$$J_1^B \approx J_0^B + (d\theta \times J_0^B)$$

Correction

$$e = (I_1^B \cdot J_1^B)/2$$

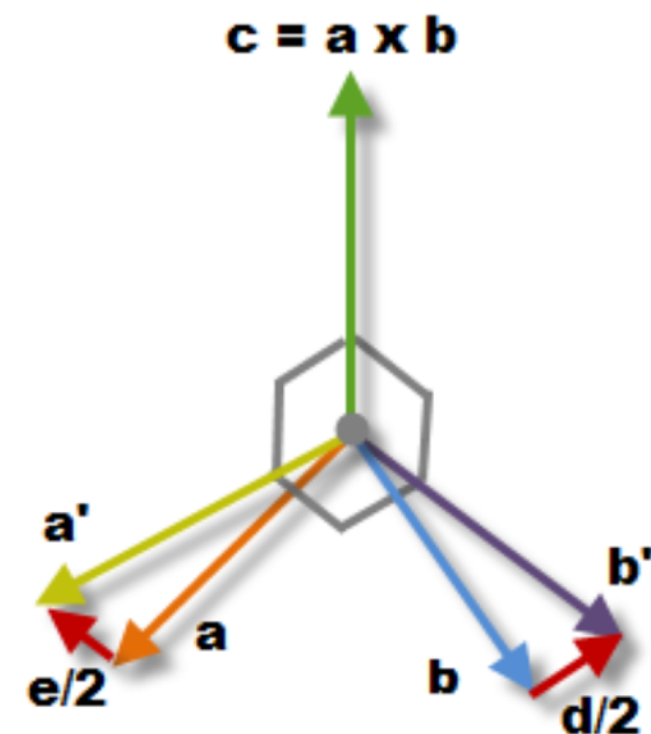
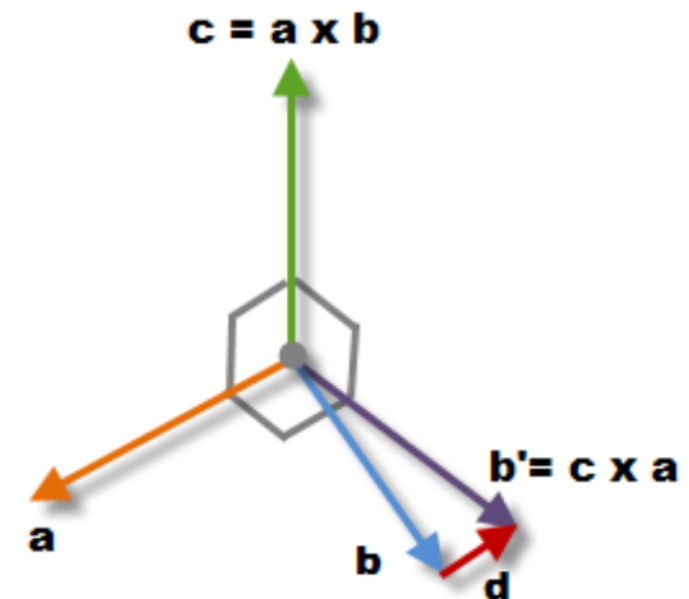
$$I_1^{B'} = I_1^B - eJ_1^B$$

$$J_1^{B'} = J_1^B - eI_1^B$$

$$I_1^{B''} = \text{Normalize}(I_1^{B'})$$

$$J_1^{B''} = \text{Normalize}(J_1^{B'})$$

$$K_1^{B''} = I_1^{B''} \times J_1^{B''}$$



Yaw, Pitch, Roll

$$\mathbf{A} = \begin{bmatrix} \cos \theta \cos \psi & \cos \phi \sin \psi + \sin \phi \sin \theta \cos \psi & \sin \phi \sin \psi - \cos \phi \sin \theta \cos \psi \\ -\cos \theta \sin \psi & \cos \phi \cos \psi - \sin \phi \sin \theta \sin \psi & \sin \phi \cos \psi + \cos \phi \sin \theta \sin \psi \\ \sin \theta & -\sin \phi \cos \theta & \cos \phi \cos \theta \end{bmatrix}$$

$$\phi = \arctan2(A_{31}, A_{32})$$

$$\theta = \arccos(A_{33})$$

$$\psi = -\arctan2(A_{13}, A_{23})$$

Magnetometer Calibration

