mbed프로젝트

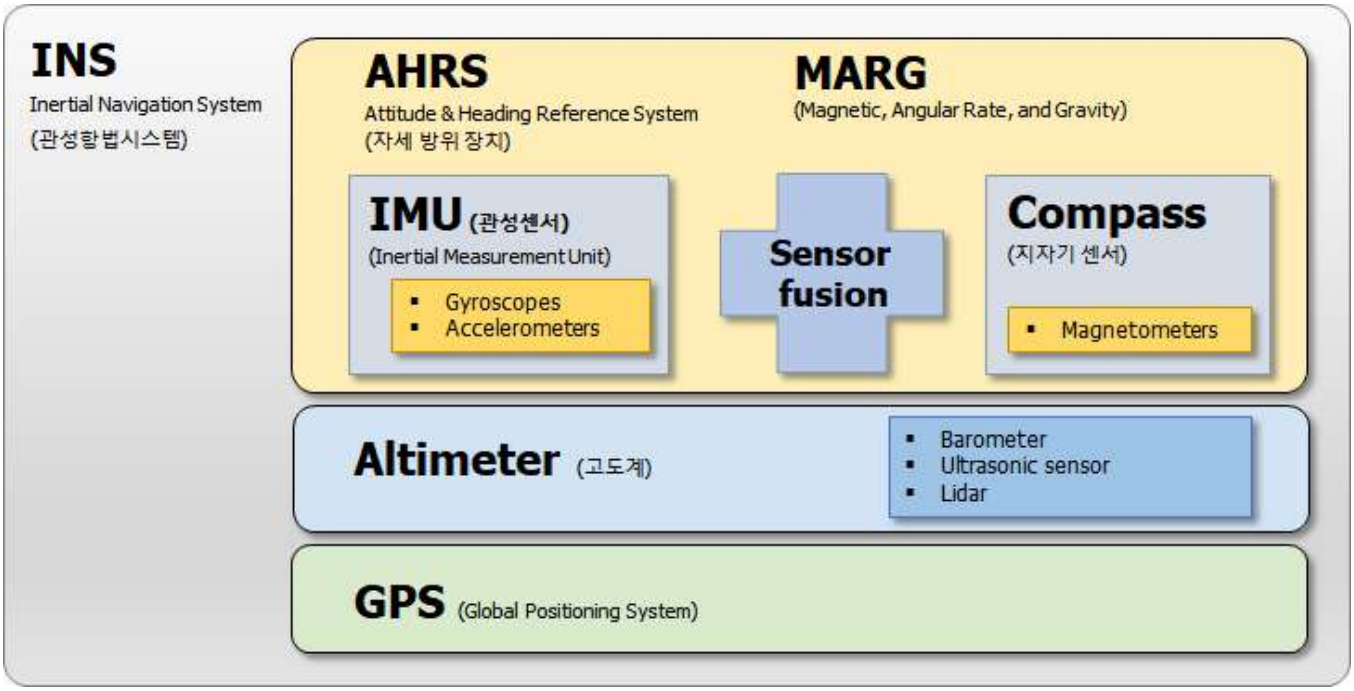
[Project] MPU9250으로 AHRS만들기

친절한알렉스 · 2019. 9. 17. 22:30

URL 복사 이웃추가

AHRS란?

자세방위시스템을 말하며 Attitude, Heading Reference System의 약자이다. 주로 IMU센서와 Magnetometer를 융합하여 자세와 절대 방위각을 찾을 수 있는 시스템 을 의미하여 종종 MARG(Magnetic, Angular Rate, and Gravity라고도 불리기도 한다. AHRS가 부족한 정보인 공간상의 위치정보를 얻기위한 고도계와 GPS를 추가적으로 융합시키면 관성항법시스템(INS: Inertial Navigation System)이 된다.



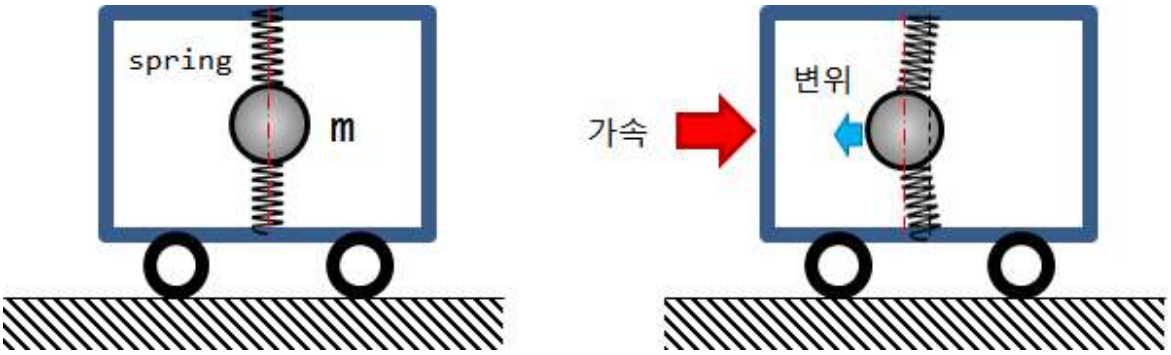
AHRS용 센서의 원리

1. IMU 센서

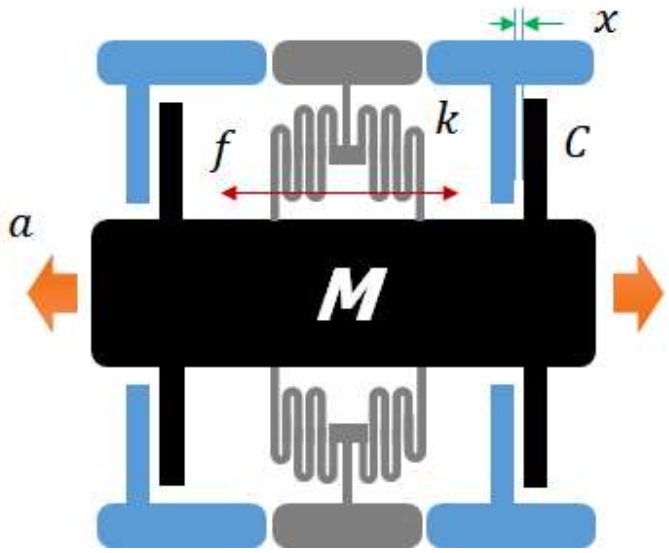
질량체의 관성을 이용하는 센서라 관성센서라고도 한다. 일반적으로 MEMS기술을 이용하여 아주 작게 가속도센서와 각속도센서를 구성한다.

□ 가속도센서 (Accelerometer)

가속도센서의 원리를 살펴보면 아래 그림과 같이 스프링에 묶여있는 질량체 m을 보면 자동차가 가속을 받으면 가속력에의하여 스프링에 변위가 발생된다. 이를 정교한 변위 센서로 측정하면 x 방향의 가속도를 비례적으로 알 수 있다.



실제로는 아래 그림 처럼 MEMS기술을 이용하여 웨이퍼 상에 질량체와 스프링을 한꺼번에 애칭 등의 방법으로 센서를 제작한다. 또한 작은 갭 x 는 두개의 전극을 이루고 있어 이를 전기용량 (Capcitive) 센서로 활용하여 변위 x를 측정하는 방법을 사용하고 있다.



□ 각속도센서 (Gyro 센서)

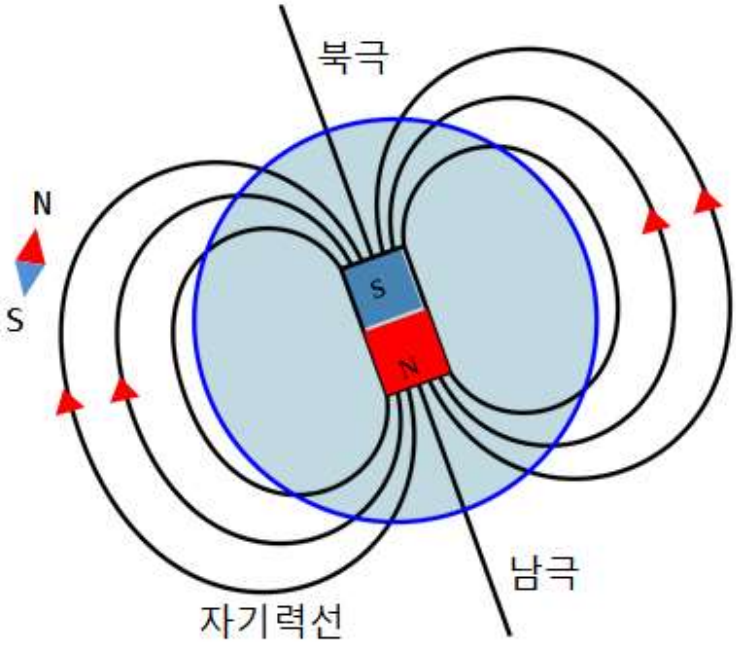
각속도 센서는 어떤 질량체가 직선 왕복운동 중 회전속도가 생기면 코리올리스(Coriolis) 힘이 발생하는 원리를 이용하고 있다.

$$f_c = -2m\omega \times v$$

가속도 센서와 마찬가지로 스프링과 전기용량 변위 센서를 활용하여 회전속도를 전기신호로 변환시키고 있다.

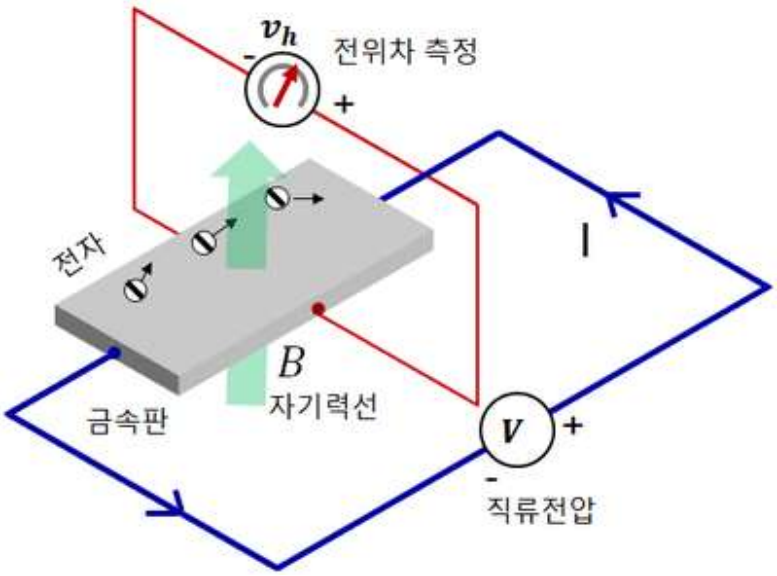
2. 지자기 센서(Magnetometer)

지자기 센서는 자기력을 측정할 수 있는 소자를 이용하여 지구의 자기력선의 방향과 세기를 측정하는 센서이다. 지구는 아래 그림처럼 거대한 영구자석으로서 남극에서 자기력선이 나와 공중을 지나 북극으로 향한다.



자기력선의 방향과 세기를 측정하는 원리는 크게 AMR과 홀효과가 있다.

□ 홀 효과(Hall Effect) 센서
금속판에 일정한 직류 전류를 흘리면 자기력의 세기에 따라 전류방향과 직각방향의 양끝단에서 전위차 발생현상을 말하고 이를 이용하면 자기센서를 만들 수 있다.

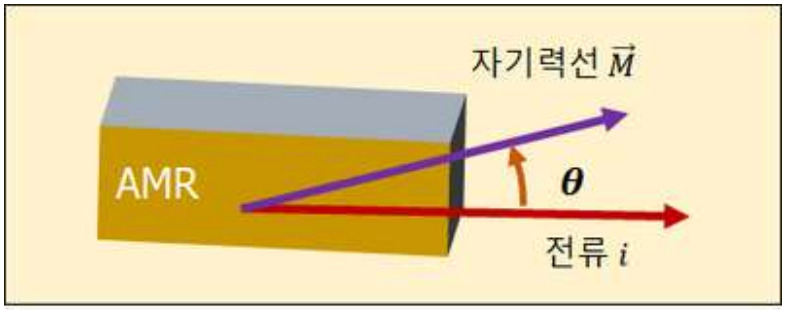


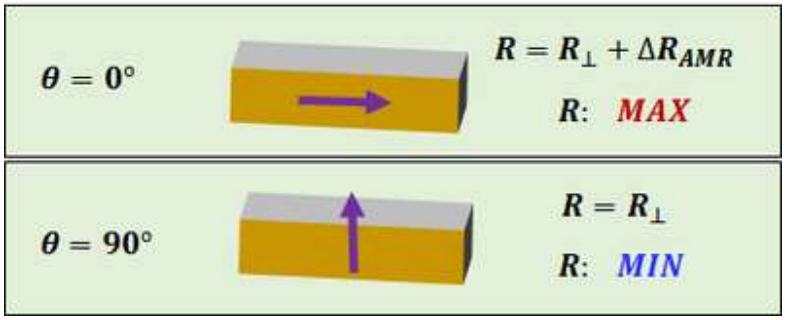
□ 이방성자기저항(AMR: Anisotropic Magneto Resistive) 소자

자기력세기에 따라 전기저항이 변화하는 AMR 소자를 이용한다. 이를 식으로 표현하면 다음과 같다.

$$R = R_{\perp} + \Delta R_{AMR} \cos^2 \theta$$

여기서 R은 소자의 전기저항이고 $R_{\perp}$는 자기력선이 전류방향에 직각일 때의 저항이며 θ 는 전류방향과 자기력선 사이의 각도이다.



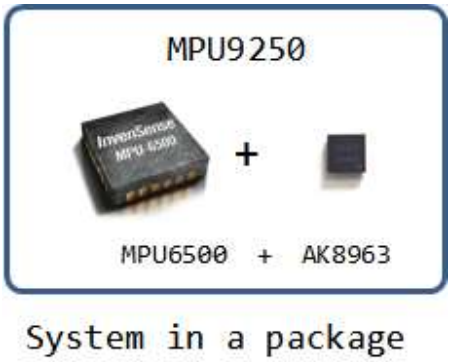


자기력선의 방향에 따라 저항이 바뀌어 휘스턴 브리지 회로를 이용하여 자기력선의 방향을 전압으로 측정할 수 있다.

MPU9250 소개

1. MPU9250

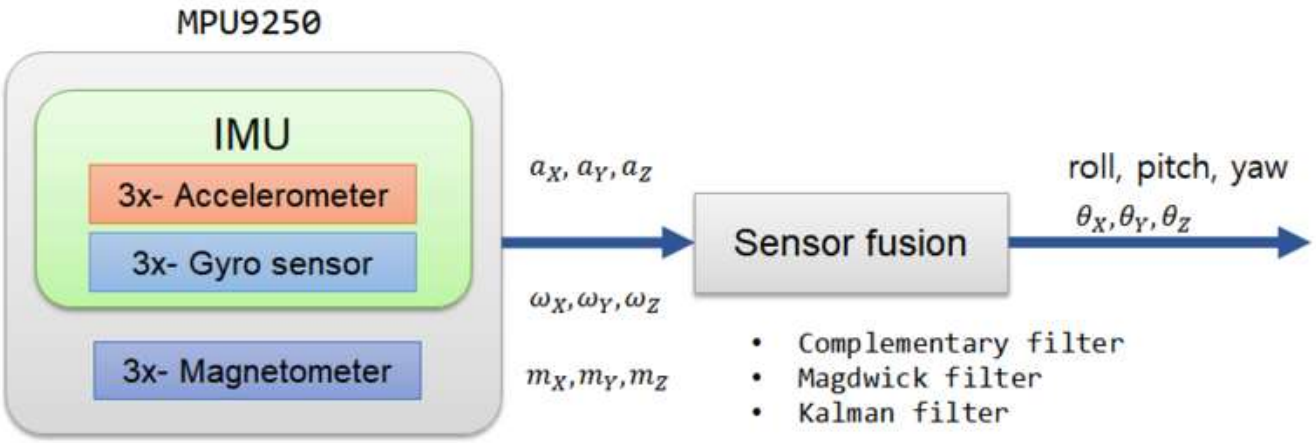
Invensens 사의 칩으로서 6축 IMU 센서(MPU6500)와 지자기 센서 3축(AK8963)이 한 패키지에 내장된 센서로서 드론, VR, 기타 웨어러블 장치에 많이 사용된다.



이 칩은 기본적으로 3.3V에서 동작하고 I2C와 SPI 모두를 지원한다.

2. MPU9250과 센서융합

MPU9250을 통하여 x,y,z축의 가속도 a_X, a_Y, a_Z, 각속도 ω_X, ω_Y, ω_Z, 그리고 자기력선 벡터 m_X, m_Y, m_Z 를 얻을 수 있다. 이렇게 9개의 성분으로부터 적절한 센서 융합(sensor fusion)방법을 통하여 3차원 공간상의 자세인 롤, 피치, 요 각도를 구하는 것이 AHRS의 목표가 된다.



3. 센서융합 방법

AHRS관련 대표적인 센서융합 방법은 다음과 같다.

□ 상보필터(Complementary Filter)
센서의 특성이 어떤 주파수를 경계로 서로 다른 경우 저역통과필터와 고역통과 필터를 이용하여 역할 분담을 시키는 필터이다.
□ Magdwick 필터
AHRS의 융합문제는 미지수가 3개이고 식이 9개인 과잉결정식이 되므로 최적화의 개념이 도입되어야 한다. 이 방법에서는 경사하강법(Steepest descent method)을 이용하여 실시간으로 최적의 롤, 피치,요를 구해준다. 이 방법은 짐벌락(gimbal lock) 현상을 극복하기 위하여 사원수(Quarternion)를 이용하여 3차원의 회전각도를 표현하였다.
□ Kalman 필터
칼만필터는 확률적 사전 지식을 가지고 동역학적 모델을가진 시스템에 대하여 전개된 필터이다. 동역학식을 알고 있으면 시간진행에 따른 상태의 예측이 가능한데 여기에 센서로 측정한 결과를 사용하여 보정을 진행하여 상태량을 추정한다.

MPU9250 레지스터

1. 기본 설정 레지스터

MPU9250 내부의 주요 레지스터를 살펴본다.

Adr.	Register	Purpose	Descriptions
19	SMPLRT_DIV	R/W	Sample rate selection (also depend on filter choice) **SMPLRT_DIV[7:0]**

1A	CONFIG	R/W	Gyro 저역필터 DLP frequency						
					EXT_SYNC_SET[2:0]			DLPF_CFG[2:0]	
1B	GYRO_CONFIG	R/W	Gyro range and filter choice						
						FS_SEL [1:0]		Fchoice_b[1:0]	
1C	ACCEL_CONFIG	R/W	acc 센서 full scale set						
						AFS_SEL[1:0]			
1D	ACCEL_CONFIG2	R/W	acc 센서 DLPF 필터 선택, 필터 주파수						
							ACCEL_FCHOICE_B	A_DLPF_CFG	
6A	USER_CTRL	R/W	I2C_MST_EN						
				FIFO_EN	I2C_MST_EN	I2C_IF_DIS			
6B	PWR_MGMT_1	R/W	chip reset and clock selection						
			DEVICE_RESET					CLKSEL[2:0]	
6C	PWR_MGMT_2	R/W	Enable/Disable sensor: 1-disable, 0-enable						
					DIS_XA	DIS_YA	DIS_ZA	DIS_XG	DIS_YG	DIS_ZG

설정용 내부 레지스터는 다음과 같다.

□ SMPLRT_DIV레지스터의 값으로 샘플레이트를 결정한다.

SAMPLE_RATE=1kHz/(1+SMPLRT_DIV)

□ CONFIG레지스터의 DLPF_CFG의 3개 비트는 자이로 센서의 디지털 저역통과 필터의 통과대역 설정한다. GYRO_CONFIG의 Fchoice_b와 함께 다음과 같은 설정이 가능하다. Fchoice_b는 역을 의미하므로 설정 시 반대로 지정해야한다.

FCHOICE		DLPF_CFG	Gyroscope			Temperature Sensor	
<1>	<0>		Bandwidth (Hz)	Delay (ms)	Fs (kHz)	Bandwidth (Hz)	Delay (ms)
x	0	x	8800	0.064	32	4000	0.04
0	1	x	3600	0.11	32	4000	0.04
1	1	0	250	0.97	8	4000	0.04
1	1	1	184	2.9	1	188	1.9
1	1	2	92	3.9	1	98	2.8
1	1	3	41	5.9	1	42	4.8
1	1	4	20	9.9	1	20	8.3
1	1	5	10	17.85	1	10	13.4
1	1	6	5	33.48	1	5	18.6
1	1	7	3600	0.17	8	4000	0.04

□ GYRO_CONFIG레지스터의 FS_SEL 비트는 자이로의 스케일을 지정한다.

00: ± 250 °/s, 01:± 500 °/s, 10:± 1000 °/s, 11: ± 2000 °/s

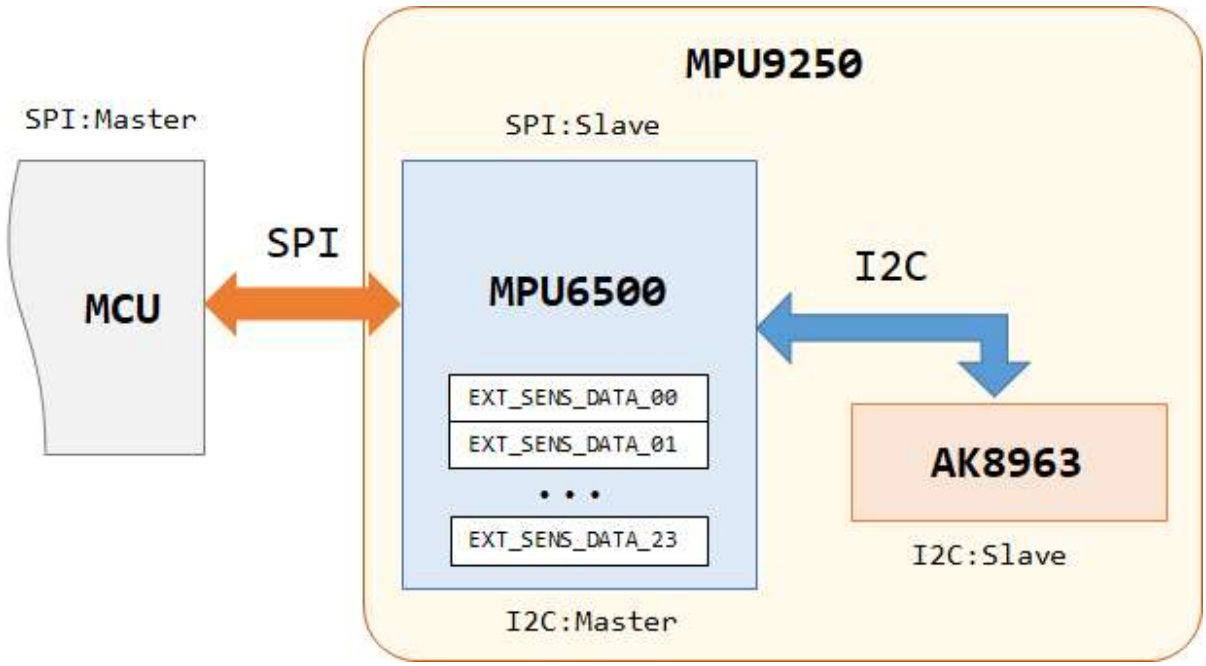
□ ACCEL_CONFIG 레지스터의 AFS_SEL 비트는 가속도의 스케일을 지정한다.

00: ± 2g, 01: ± 4g, 10: ± 8g, 11: ± 16g

□ ACCEL_CONFIG2 레지스터의 ACCEL_FCHOICE와 A_DLPF_CFG 비트를 설정하여 는 가속도의 데이터의 저역통과필터를 설정한다.

ACCEL_FCHOICE	A_DLPF_CFG	Output			
		Bandwidth (Hz)	Delay (ms)	Noise Density (ug/rtHz)	Rate (kHz)
0	X	1.13 K	0.75	250	4
1	0	460	1.94	250	1
1	1	184	5.80	250	1
1	2	92	7.80	250	1
1	3	41	11.80	250	1
1	4	20	19.80	250	1
1	5	10	35.70	250	1
1	6	5	66.96	250	1
1	7	460	1.94	250	1

□ USER_CTRL레지스터의 I2C_MAST_EN비트는 I2C의 마스터 모드를 활성화한다. MPU9250은 SPI 이지만 다음 그림 처럼 내부적으로는 지자기센서인 AK8963과 I2C 통신을 하기 위하여 I2C 마스터로 설정이 필요하다.



□ PWR_MGMT_1 레지스터의 DEVICE_RESET 비트는 장치의 리셋이고 CLKSE 은 기준 클락을 지정한다.

□ PWR_MGMT_2 레지스터의 6개 비트를 이용하여 각 센서들을 활성화 시킨다. 다만, 해당 비트가 1이면 비활성화이고 0 이면 할성화이다.

2. 슬레이브와 인터럽트 관련 설정

Adr.	Register	Purpose	Descriptions				
24	I2C_MST_CTRL	R/W	I2C master clock				
							I2C_MST_CLK[3:0]
25	I2C_SLV0_ADDR	R/W	I2C master clock				
			I2C_SLV 0 _RNW	I2C_ID_0 [6:0]			
26	I2C_SLV0_REG	R/W	I2C 슬레이브0 주소 지정				
			I2C_SLV0_REG[7:0]				
27	I2C_SLV0_CTRL	R/W	I2C 슬레이브0 로 부터의 데이터 설정				
			I2C_SLV 0 _EN	I2C_SLV 0_BYTE_ SW	I2C_SLV0_ REG_DIS	I2C_SLV0 _GRP	I2C_SLV0_LENG[3:0]
63	I2C_SLV0_DO	R/W	I2C 슬레이브0 에 보내는 송신 데이터				
			I2C_SLV0_DO[7:0]				

37	INT_PIN_CFG	R/W	INT핀 설정						
					LATCH_INT_EN	INT_ANYRD2_CLEAR			
			Interrupt활성화						
38	INT_ENABLE	R/W							**RAW_RDY_EN**
			칩 식별 번호						
75	WHO_AM_I	R	0x68						

□ I2C_MST_CTRL레지스터의 I2C_MST_CLK 4비트는 I2C 마스터 클락의 속도를 의미한다.

□ I2C_SLAVE0_ADDR레지스터의 I2C_ID_0 7비트는 I2C 슬레이브0 디바이스 주소 지정한다.

□ I2C_SLAVE0_REG의 I2C_SLV0_REG 8비트는 I2C 슬레이브0 디바이스의 레지스터 주소를 지정한다.

□ I2C_SLAVE0_CTRL 레지스터의 각각의 비트의 읨는 다음과 같다.

· I2C_SLV0 _EN 1이면 자동 슬레이브 자동데이터읽기 활성화

· I2C_SLV0_BYTE_SW 는 슬레이브로부터의 데이터 바이트 순서 앞뒤 교환

· I2C_SLV0_REG_DIS 비트가 1 이면 쓰기 금지,

· I2C_SLV0_LENG 는 4개의 비트로서 데이터의 바이트 수를 지정한다.

· 슬레이브의 데이터는 EXT_SENS_DATA_00~23의 24바이트의 레지스터에 저장된다.

□ I2C_SLV0_DO레지스터는 슬레이브로 보낼 데이터를 쓰는 레지스터이다. SPI로 여기에 데이터를 쓰면 I2C를 통하여 슬레이브로 간다.

□ INT_PIN_CFG 레지스터는 INT핀에 관련된 설정을 한다.

· LATCH_INT_EN 비트가 1이면 인터럽트 status 비트가 0일 때까지 INT핀이 레벨을 유지하고 0 이면

INT핀이 50us 동안만 펄스를 출력한다.

· NT_ANYRD_2CLEAR 비트가 1이면 아무 바이트나 읽으면 인터럽트 상태가 해제되고 0이면 INT_STAUS

레지스터를 읽었을 때만 해제된다.

□ INT_ENABLE레지스터는 어떤 사건에 의하여 I인터럽트를 발생시킬지를 선택한다.

RAW_RDY_EN 비트가 1이되면 센서가 측정을 마치고 데이터 준비가 완료되었을 때 인터럽트를 발생시킨다.

□ WHO_AM_I 레지스터는 칩의 식별번호(0x68)이 저장되어있어 통신오류나 칩의 유무를 확인할 때 사용된다.

3. 데이터 레지스터

다음은 가속도와 자이로센서의 데이터와 온도 데이터에 해당하는 레지스터이다. 주의할 점은 빅엔디언으로 상위바이트가 하위바이트보다 낮은번호에 있다는 것이다.

Adr.	Register	Purpose	Descriptions
3B	ACCEL_XOUT_H	read	Acceleration x in 16bit
3C	ACCEL_XOUT_L	read	
3D	ACCEL_YOUT_H	read	Acceleration y in 16bit
3E	ACCEL_YOUT_L	read	
3F	ACCEL_ZOUT_H	read	Acceleration z in 16bit
40	ACCEL_ZOUT_L	read	
41	TEMP_OUT_H	read	Temperature in 16bit
42	TEMP_OUT_L	read	
43	GYRO_XOUT_H	read	Gyro data x in 16bit
44	GYRO_XOUT_L	read	
45	GYRO_YOUT_H	read	Gyro data y in 16bit
46	GYRO_YOUT_L	read	
47	GYRO_ZOUT_H	read	Gyro data z in 16bit
48	GYRO_ZOUT_L	read	

AK8963레지스터

Adr.	Register	Purpose	Descriptions
0	WIA Device ID(0x48)	R	identification **0x48**
1	INFO Device info	R	

2	ST1 status1	R	DRDY:1- Data Ready, DOR: 1-data over-run DOR, DRDY
3	HXL low byte of X	R	low byte of X data
4	HXH high byte of X	R	high byte of X data
5	HYL low byte of Y	R	low byte of Y data
6	HYH high byte of Y	R	high byte of Y data
7	HZL low byte of Z	R	low byte of Z data
8	HZH high byte of Z	R	high byte of Z data

□ WIA 레지스터는 칩의 식별 번호(0x48)를 보관하고 있어 칩의 유무를 확인할 때 사용된다.
□ ST1레지스터는 칩의 상태를 표시하면 DOR비트가 1이면 데이터 오버런이 일어났다는 뜻이고 DRDY는 데이터가 준비되었다는 플레그이다.
□ 나머지 HXL에서 HZH 까지는 2바이트씩 각각 x, y, z 방향의 자기력 벡터 성분이다.

Adr.	Register	Purpose	Descriptions
9	ST2 status 2	R	HOFL: 1-Magnetic sensor overflow, BITM: 1-16bit/0-14bit BITM, HOFL
A	CNTL1 control 1	R/W	BIT: setting output bit 1-16bit/0-14bit. MODE BIT, MODE[3:0]
B	CNTL2 control 2	R/W	SRST:1- soft reset SRST
C	ASTC self test control	read	SELF: 1- Generate magnetic field for self-test

			SEL F
F	I2CDIS I2C Disable	R	
10	ASAX Sensitivity Adjustment	R	Magnetic sensor X-axis sensitivity adjustment value
11	ASAY Sensitivity Adjustment	R	Magnetic sensor X-axis sensitivity adjustment value
12	ASAZ Sensitivity Adjustment	R	Magnetic sensor X-axis sensitivity adjustment value

□ ST1 은 상태레지스터로서 BITM은 현재 센서 ADC의 비트 수이며 1이면 16, 0이면 14비트이다. HOFL 비트는 1이면 센서의 오버플로우를 의미한다.

□ CNTL1레지스터의 BIT는 현재 센서 ADC의 비트 수를 설정하녀 1이면 16 0이면 14비트 이다. MODE의 4개비트는 다음과 같다.

- 0000: Power-down 모드
- 001: Single measurement 모드
- 0010: 연속측정모드 1
- 0110: 연속측정모드 2
- 0100 : 외부 트리거 신호에의한 측정모드
- 1000: 셀프 테스트 모드
- 1111: 퓨즈 ROM 접근모드

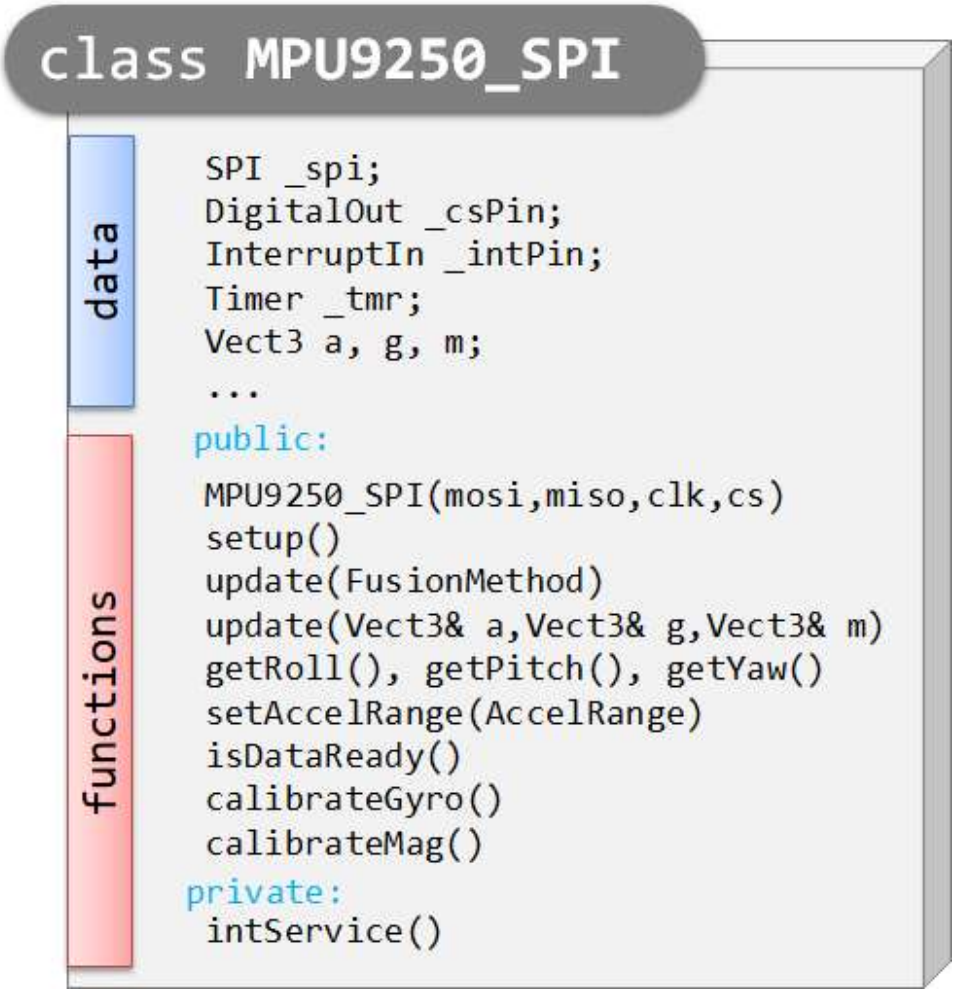
□ CNTL2레지스터의 SRST 비트가 1이되면 AK8963칩을 리셋시킨다.

□ 마지막에 있는 ASAX, ASAY, ASAZ 레지스터는 AK8963칩을 생산하고 난 후 개별칩에 대한 감도 데이터를 ROM에 보관했는데 그 데이터를 읽어들이는 레지스터들이다. 각각의 레지스터 값 ASA과 각축의 자기력값 H를 가지고 다음 식으로 보정값을 얻는다.

$$H_{adj} = H\left(\frac{0.5 \times (ASA - 128)}{128} + 1\right)$$

MPU9250 클래스 만들기

1. MPU9250_SPI 클래스



□ member변수

▷ SPI _spi; : SPI 객체 선언

▷ DigitalOut _csPin : SPI chip select용 핀 DigitalOut 객체

▷ InterruptIn _intPin : 인터럽트 핀에 대한 InterruptIn 객체

▷ Timer _tmr : 샘플 시간 측정용 타이머 객체

▷ Vect3 a, g, m: 가속도, 자이로, 자기력선 벡터를 구조체 Vect3로 생성

struct Vect3 {float x, y, z;};

□ member함수

▷ 생성자

· MPU9250_SPI(mosi,miso,clk,cs, intr): 클래스의 객체를 SPI의 핀인 mosi, miso등의 핀과 INT핀을 지정

하면서 생성

▷ 장치 초기화 설정

· setup() : MPUS9250 을 초기화 한다.

▷ 데이터 업데이트

· update(FusionMethod) : AHRS 센서융합 알고리즘을 실행한다. 측정값으로부터 내부적으로 roll, pitch,

yaw 를 계산

- FusionMethod : MAGDWICK/COMPLEMENTARY 중 센서융합방법 선택

· update(Vect3& a,Vect3& g,Vect3& m)

- 센서융합없이 가속도 a, 자이로 각속도 g,자기력 m 벡터를 반환한다.

▷ 데이터 읽기:

· getRoll(), getPitch(), getYaw(): 데이터 업데이트 후 계산된 roll, pitch, yaw를 반환하는 함수들이다.

▷ 데이터 상태 확인

· isDataReady(): data ready interrupt 상태를 읽음. 1이면 유효한 데이터가 있음을 의미

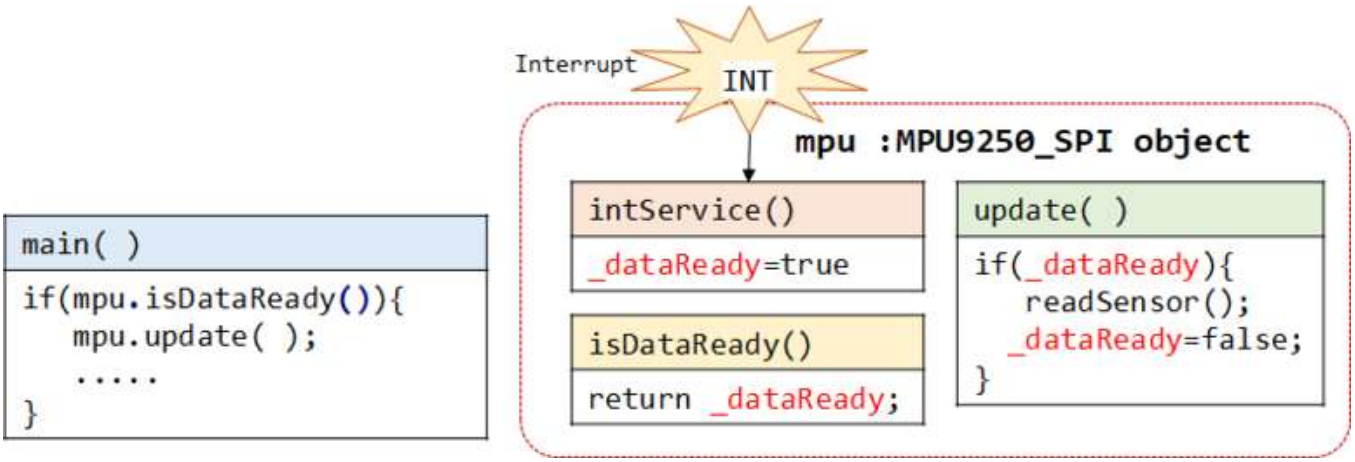
▷ 인터럽트 서비스

· intService() : _intPin 의 콜백함수이다.

▷ 센서 검정

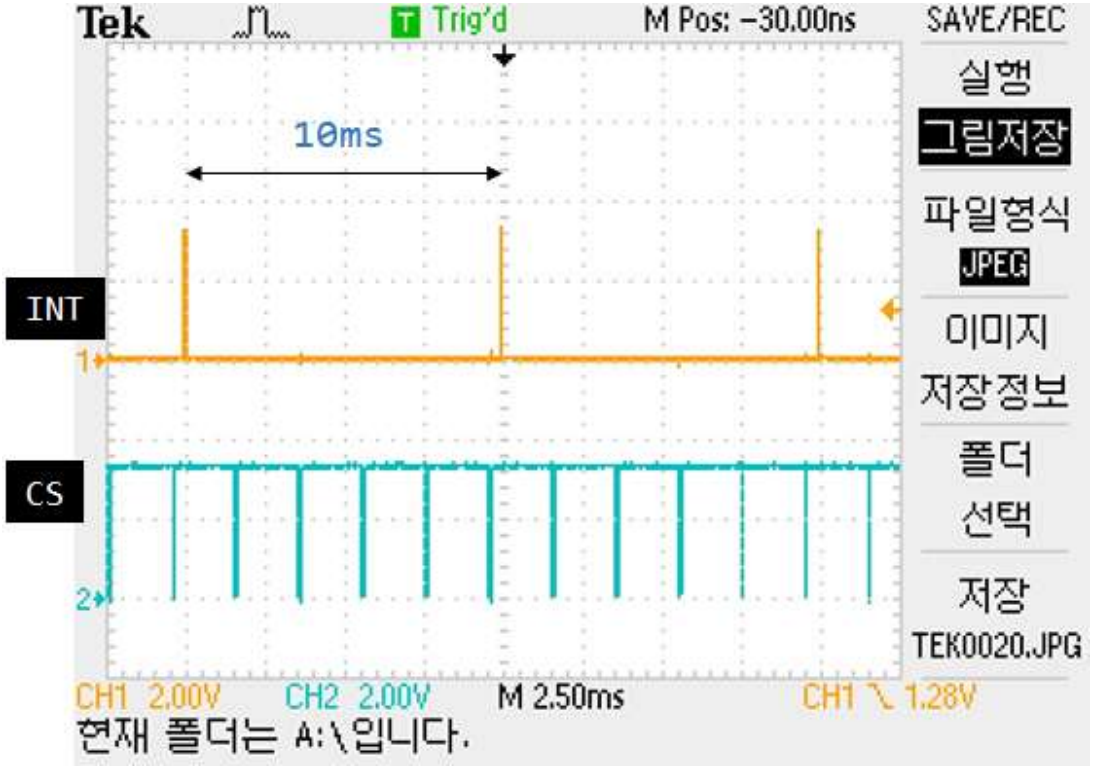
· calibrateGyro(), calibrateMag(): 각각 자이로센서, 지자기센서의 교정을 실시하는 함수이다.

2, 주요동작



주요동작을 보면 MPU9250이 측정을 마치고 데이터 준비를 완료하면 INT핀에 펄스가 발생되는데 이를 InterruptIn객체로 rising변화를 감지한여 콜백함수로 지정된 intServe함수가 동작되고 여기서 _dataReady 가 true가 된다. main 에서 isDataReady 함수로 데이터 준비 상태를 확인하고 update 함수는 계속 실행시키지만 _dataReady가 true인 경우만 센서 데이터를 읽도록하여 준비가 되지 않은 경우 불필요하게 읽기 작업을하지 않도록하여 효율을 높였다. 현재는 main 함수에서 update함수를 주기적으로 호출하는 구조이지만 Ticker 객체를 사용하면 객체 내부에서 모든 과정을 주도하도록할 수도 있을 것이다.

3. 동작파형 확인



오실로스코프의 파형은 INT핀과 SPI의 CS핀이다. INT핀은 정확하게 10ms 이며 CS는 2ms 단위로 mpu9250에 접근하는 것을 알 수 있다.

MPU9250 AHRS 실습

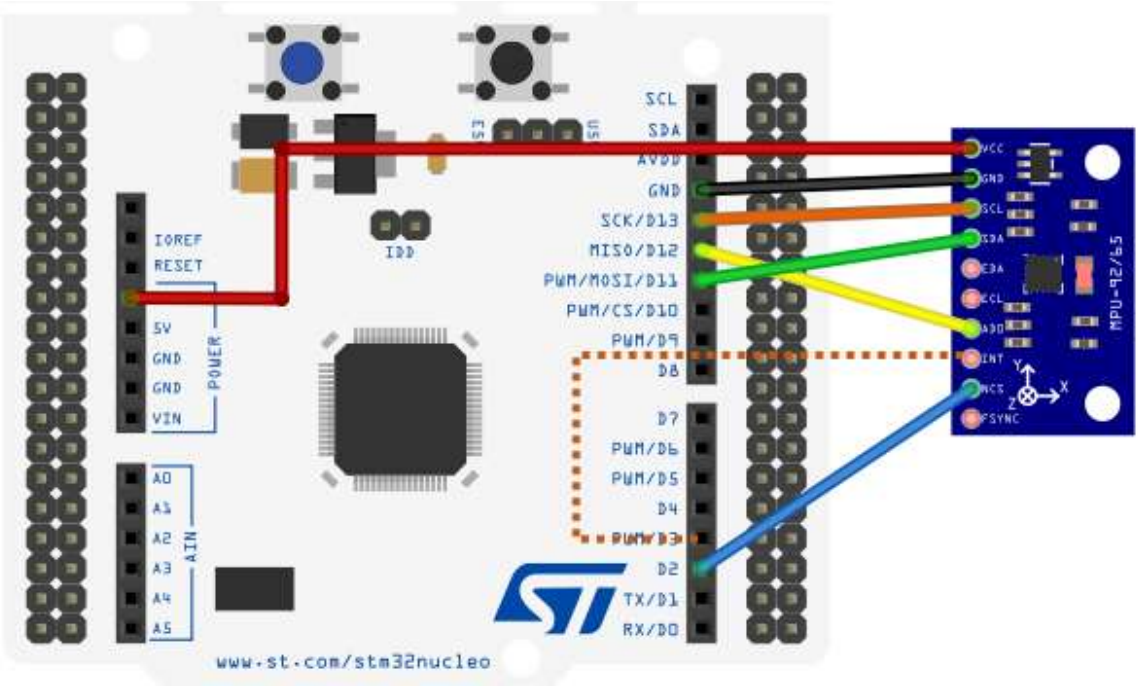
1. 목표

MPU9250을 이용하여 AHRS알고리즘을 적용하고 이를 PC의 그래픽 프로그램으로 롤, 피치, 요 각도가 적절하게 표현되도록 한다.

2. 실습장치

□ 실습재료

NUCLEO F401RE, MPU9250모듈



□ 결선

SCL ⇒ D13 [SPI: SCLK]

SDA ⇒ D11 [SPI: MOSI]

ADO ⇒ D12 [SPI: MISO]

NCS ⇒ D2 DigitalOut 적용

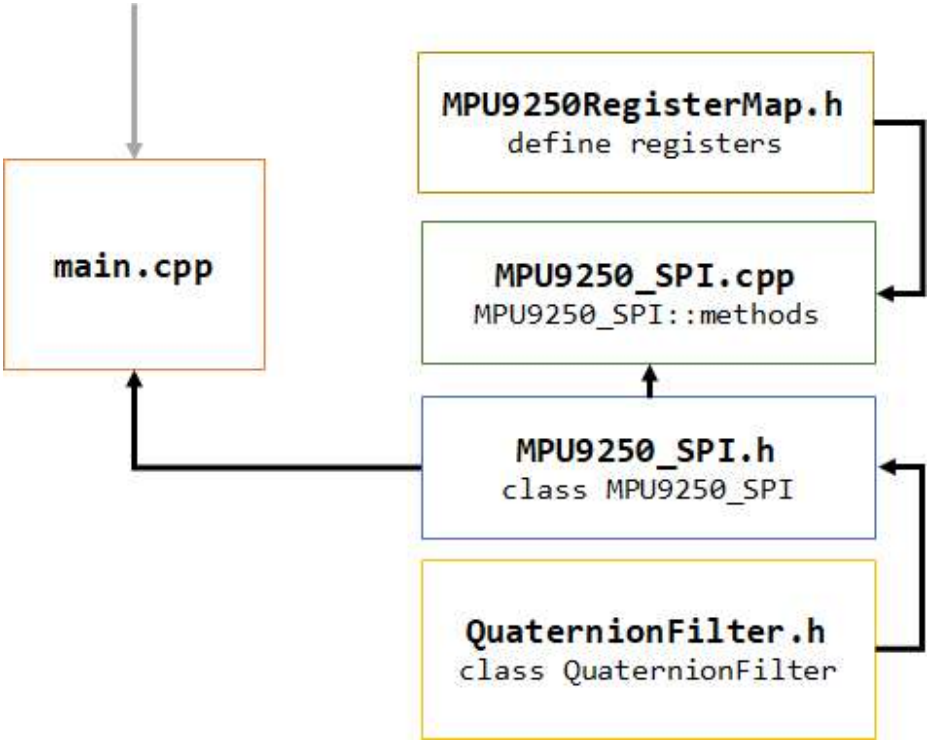
INT ⇒ D3 InterruptIn 적용

VCC ⇒ 3V3 전원

GND ⇒ GND

3. MPU9250 라이브러리 코드

□ 라이브러리 구조



main.cpp : main 함수

MPU9250_SPI.h : MPU9250_SPI 클래스의 정의

MPU9250RegisterMap.h : MPU9250의 내부레지스터의 주소 및 설정 상수값 정의

QuaternionFilter.h : Magdwick 필터 함수

MPU9250_SPI.cpp : MPU9250_SPI 클래스의 멤버함수 정의

drawPlane2 폴더: Processing 3차원 그래픽 프로그램 코드

□ 소스 코드 링크

https://github.com/AlexParkSeoultech/mBed_MPU9250_SPI

AlexParkSeoultech/mBed_MPU9250_SPI

MPU9250 with SPI interface for mBed. Contrib...

github.com

4. 메인코드

```
#include "mbed.h"
#include "MPU9250_SPI.h"
//#define CALIBRATION_MODE
//#define RAW_DATA
MPU9250_SPI mpu(D11, D12, D13, D2,D3); // mosi, miso, sclk, cs, intr
//DigitalOut led1(LED1);  // be careful it is D13 too
```

```
Timer tmr;
Serial pc(USBTX,USBRX, 115200);
float dt;
void getDt();
void calibrationProcess();
void applyCalbratedValue();
int main(){
    mpu.setup();
    mpu.setMagneticDeclination(8.5);
    mpu.setSampleRate(SR_100HZ);
    mpu.setGyroRange(GYRO_RANGE_2000DPS);
    mpu.setAccelRange(ACCEL_RANGE_16G);
    mpu.setDlpfBandwidth( DLPF_BANDWIDTH_184HZ);
    mpu.enableDataReadyInterrupt();
    #ifdef CALIBRATION_MODE
    calibrationProcess();
    #endif
    applyCalbratedValue();
    tmr.start(); tmr.reset();
    while(true){
        if (mpu.isDataReady()){
            getDt();
        #ifndef RAW_DATA
            mpu.update(MAGDWICK ); //  MAGDWICK  /COMPLEMENTARY
            pc.printf("YPR,%5.2f,%5.2f,%5.2f\n", mpu.getYaw()*RAD_TO_DEG,
                   mpu.getPitch()*RAD_TO_DEG, mpu.getRoll()*RAD_TO_DEG );
        #else
            Vect3  a, g, m;  // acc/gyro/mag vectors
            mpu.update(a,g,m);
            pc.printf("%5.2f, %5.2f, %5.2f,  %5.2f, %5.2f, %5.2f,
                       %5.2f, %5.2f, %5.2f\n", a.x, a.y, a.z, g.x, g.y,
             g.z,m.x, m.y, m.z);
        #endif
        }
    }
}
void getDt(){
  dt=tmr.read_us()/1000000.0;
  tmr.reset();
}
void calibrationProcess(){
  mpu.calibrateGyro();
  mpu.calibrateMag();
  pc.printf("Calibration Completed !!!!!!!!\n");
    while(true);   //stop here
```

```
}
void applyCalbratedValue(){
  Vect3 gBias ={ 1.881,   -2.630  ,   0.226};
  mpu.setGyroBias(gBias);
  Vect3 mBias ={12.885,  314.122  , -594.508};
  mpu.setMagBias(mBias);
  Vect3 mScale={ 1.852,    1.176  ,   0.621};
  mpu.setMagScale(mScale);
}
```

CALIBRATION_MODE를 정의하면 자이로와 지자기센서의 캘리브레이션을 실행하게 되고 주석을하면 정상적 실행을 한다. RAW_DATA를 정의하면 AHRS를 실행하지 않고 가속도, 자이로, 자기력 벡터를 구하는 모드이고 이를 주석처리하면 AHRS를 실행한다. 다음 MPU9250_SPI클래스의 객체 mpu를 생성하면서 SPI핀, CS핀 그리고 INT핀을 지정한다. 다음은 일정한 샘플 시간을 만들기 위하여 Timer 객체를 선언하였다. Serial 객체 pc를 생성하였다. main함수에서는 먼저 mpu의 초기화와 함께 센서 샘플주파수, 자이로 측정 레인지, 가속도 측정 레인지, 센서의 저주파 필터 대역폭, 인터럽트 설정을 하였다. 그다음에 #ifdef 를 이용하여 캘리브레이션을 할지를 정하고 applyCalbratedValue함수를 이용하여 캐리브레이션 값의 반영을 한다. Timer 객체인 tmr을 기동시키고 0으로 초기화하였다. while루프에서는 isDataReady함수를 통하여 센서 측정 데이터가 갱신되었는지를 확인한다. 데이터가 준비되었다면 RAW_DATA 정의에 따라 Magdwick 또는 Complementary 필터를 이용한 AHRS 알고리즘을 선택하거나 원시 데이터인 a,g,m 벡터를 구할 수 있다. AHRS를 선택한 경우 MAGWICK 또는 COMPLEMENTARY를 선택하여 roll, pitch, yaw 값을 계산하게 할 수있고 그 결과를 도단위로 바꾸어 PC에 전송한다. 코드에 포함된 darawPlane2라는 Processing 코드를 이용하여 이 데이터를 받아 3차원 애니메이션을 진행한다. calibrateGyro(), calibrateMag() 함수는 각각 자이로센서, 지자기센서의 교정을 실시하는 함수이다. 여기서 참고할 내용은 우리가 사용하고있는 누클래오 보드는 내부에 EEPROM이 없기 때문에 교정값을 저장하여 사용할 수 없어 교정 후 직접 applyCalibratedValue함수에 하드 코딩으로 입력시켜야 한다.

5. 실행 동영상

MPU9250-Magdwick filter
179 0

00:00 00:18 자동

MPU9250-Magdwick filter

6. 교정 절차

① 소스코드에서 main 파일 상단의 //#define CALIBRATION_MODE에서 주석표시 **//**를 지운다.
② 자이로 센서 교정단계로서 보드를 만지지 말고 그대로 둔다.

```
Please, don't move!
Gyro Calibration done!
MPU9250 Gyro biases (deg/s)
  1.893,   -2.551  ,   0.227

Mag Calibration: Wave device in a figure eight until done!
Mag Calibration done!
AK8963 mag biases (mG)_numSamples
 12.885,  314.122  , -594.508
AK8963 mag scale (mG)
  1.852,    1.176  ,   0.621
Calibration Completed !!!!!!!!
```

교정 결과 출력

③ 지자기 센서 교정은 자이로센서 교정이 끝나고 시리얼 모니터에 "Wave device in a figure eight until done!" 지시가 나오면 20초 정도 그림처럼 8자를 그려준다.

④ 교정 종료 후에 mbed 는 실행을 종료한다 이 때 교정 결과 값을 main 파일의 applyCalbratedValue()함수에 반영한다.
⑤ 교정된 결과를 가지고 AHRS를 실행하기 위하여 소스에서 #define CALIBRATION_MODE의 앞에 // 를 붙여 다시 주석처리한다