IMU Sensor

An **Unmanned aerial vehicle** (UAV) is a Unmanned Aerial Vehicle. UAVs include both autonomous (means they can do it alone) drones and remotely piloted vehicles (RPVs). A UAV is capable of controlled, sustained level flight and is powered by a jet, reciprocating, or electric engine.

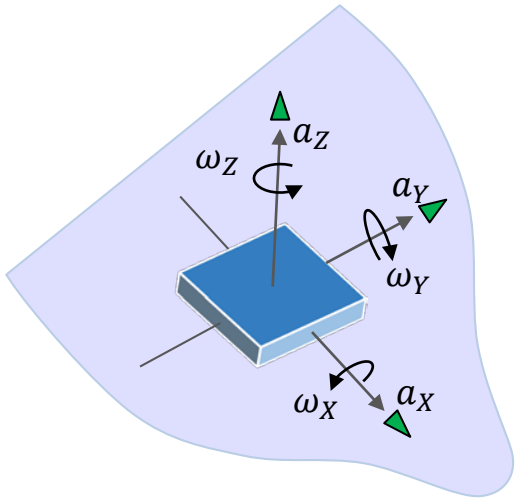
CONTENTS TITLE

IMU Sensor ?

IMU Sensor?

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

- **IMU** (Inertial Measurement Unit : 관성측정장치)
  - 구성:
    - 가속도센서(**Accelerometer**): 선가속도를 측정
    - 자이로(**Gyro**) 센서 : 각속도를 측정
    - magnetosensor도 포함하면 AHRS가 됨.
  - 이 2 센서 측정값을 융합하여 물체의 자세를 측정함.
  - 자세를 표시하는 방법은 여러가지 있지만, X, Y, Z 3개 축의 회전각도 Roll, Pitch, Yaw 값을 구하는 것이다.



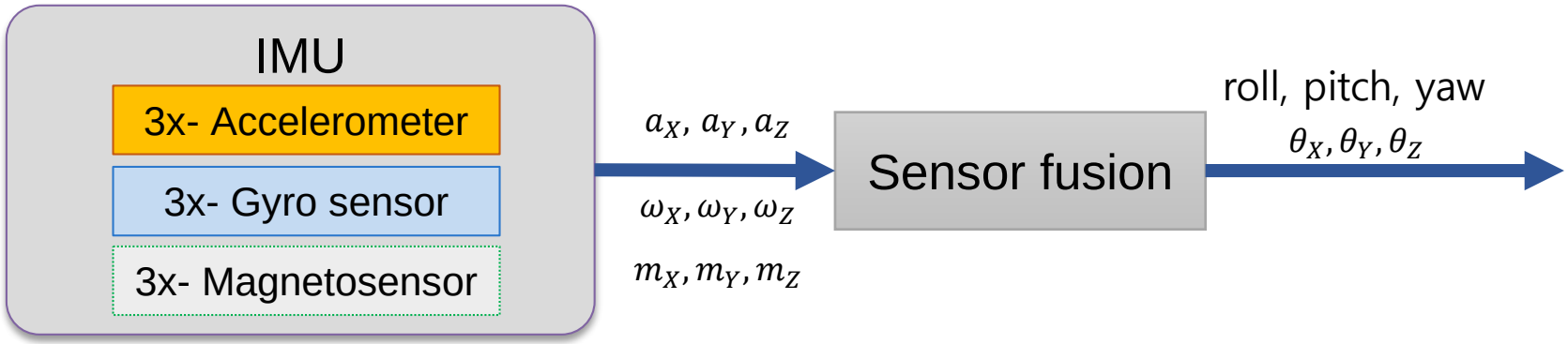
$\omega_X, \omega_Y, \omega_Z$: 회전 각속도

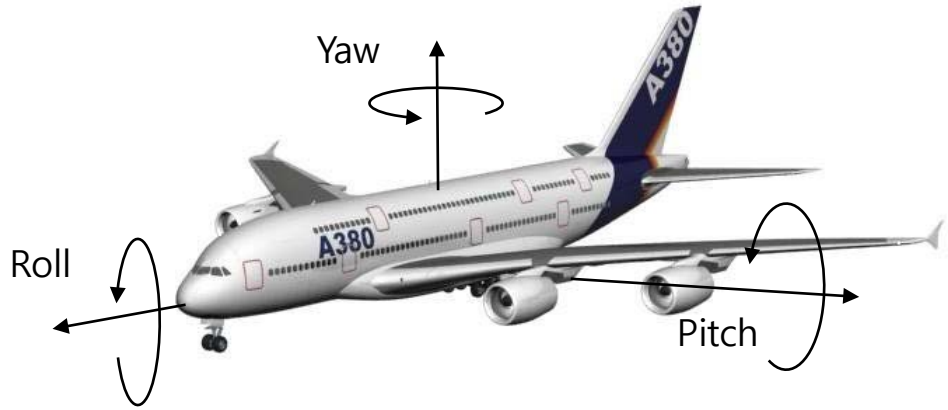
a_X, a_Y, a_Z: 직선 가속도

센서융합 (Fusion)

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

IMU 센서의 Fusion



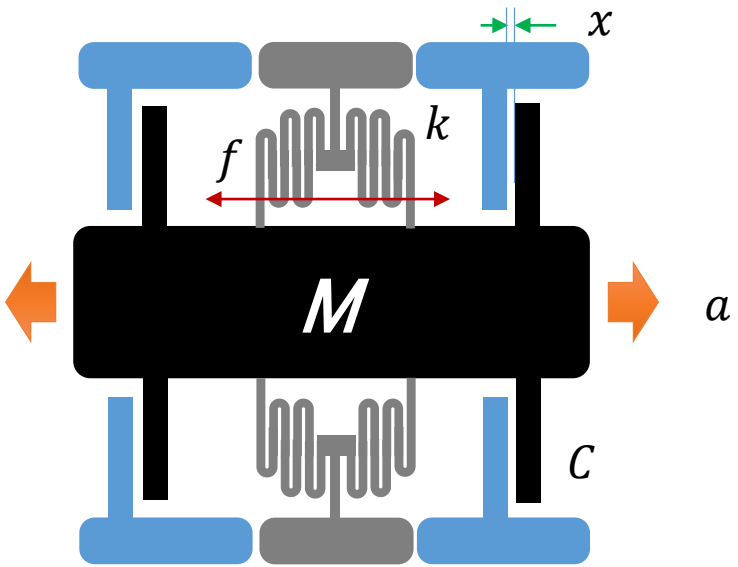


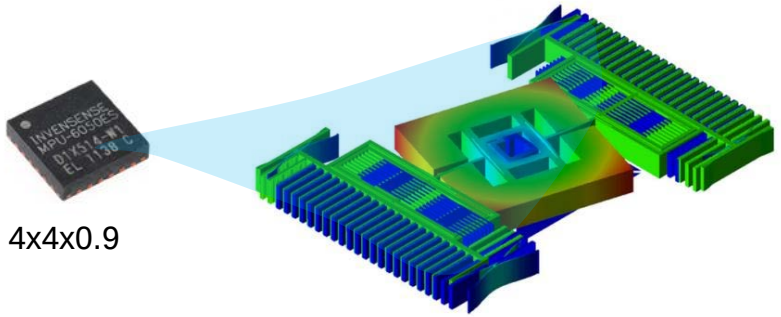
MEMS IMU 센서

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

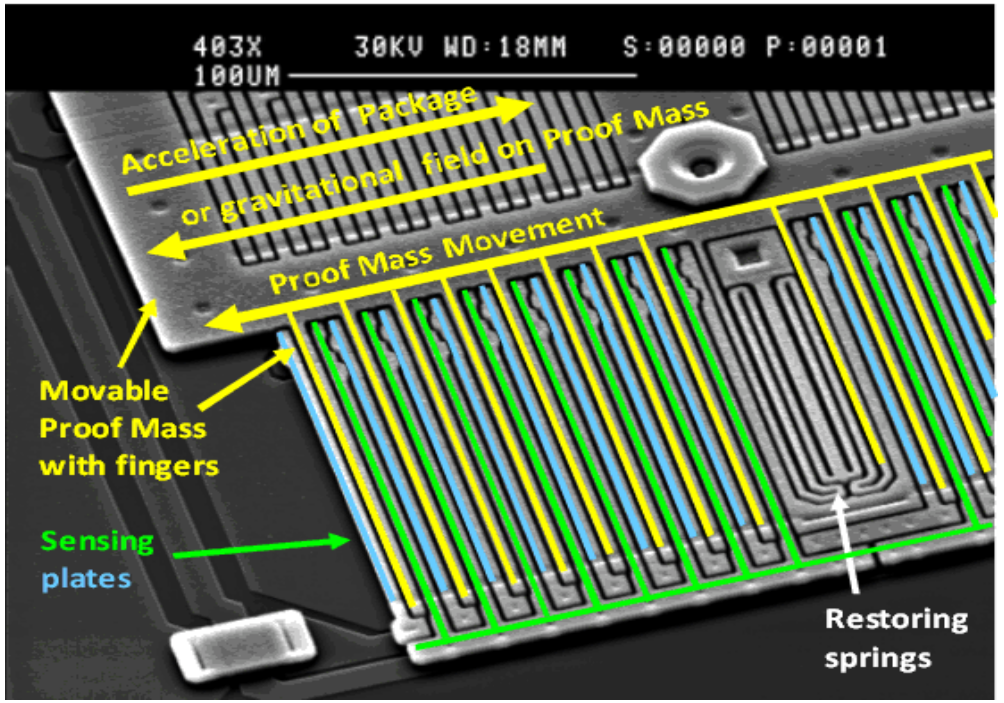
가속도 측정센서

- Mass-Spring + Capacitive sensor
  - $f = ma,$
  - $f \propto kx$
  - $a \propto x$





MEMS IMU



출처:
http://www.freescale.com/files/sensors/doc/app_note/AN3461.pdf

MEMS IMU 센서

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

- MEMS IMU 센서의 원리
- 물체가 직선운동을 시킬때 회전을 시키면 **Coriolis force**가 발생
  - $\boldsymbol{F_c} = -2m\boldsymbol{\Omega} \times \boldsymbol{v}$
  - $\boldsymbol{F_c}$: Coriolis force vector, $\boldsymbol{\Omega}$: rotation vector, $\boldsymbol{v}$: linear velocity vector
- $\boldsymbol{F_c}$ 에 의한 변위를 capacitive 센서로 측정

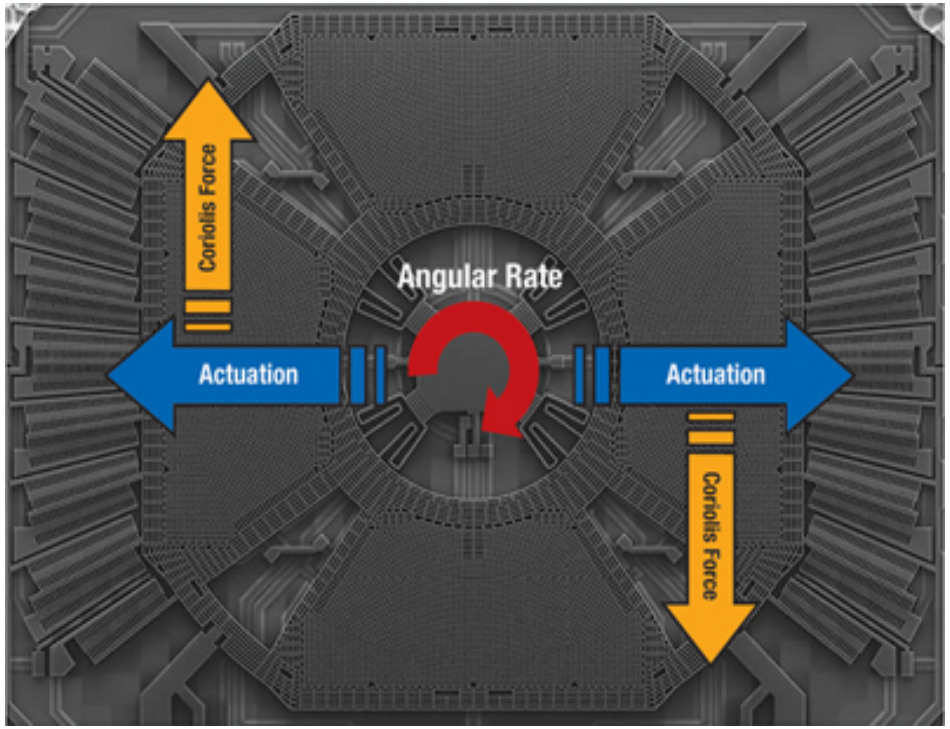


Image credit:
http://www.digikey.com/us/en/techzone/sensors/resources/articles/MEMS-Accelerometers.html

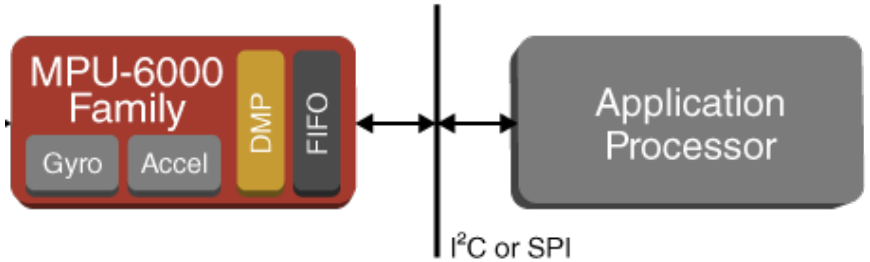
IMU MPU6000

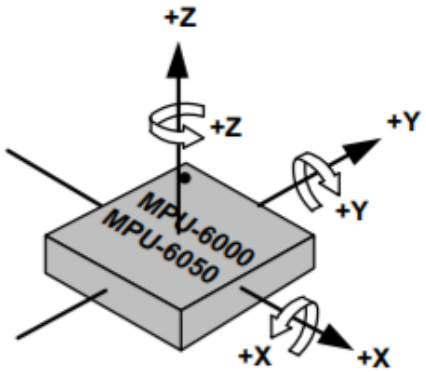
IMU MPU6000

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

Invensens IMU:

- **SPECIFICATIONS**
  - Gyro : 3축 자이로 센서의 감도 ±250, ±500, ±1000, ±2000°/sec
  - Accel: 3축 가속도 센서의 감도 ±2g, ±4g, ±8g, ±16g
  - DMP(Digital Motion Processing) 기능 내장:
    - 내부의 프로세서를 이용하여 자동으로 센서 융합 계산. ➔ 롤, 피치
  - SPI, I2C 버스 인터페이스
  - 16 비트 ADC 내장



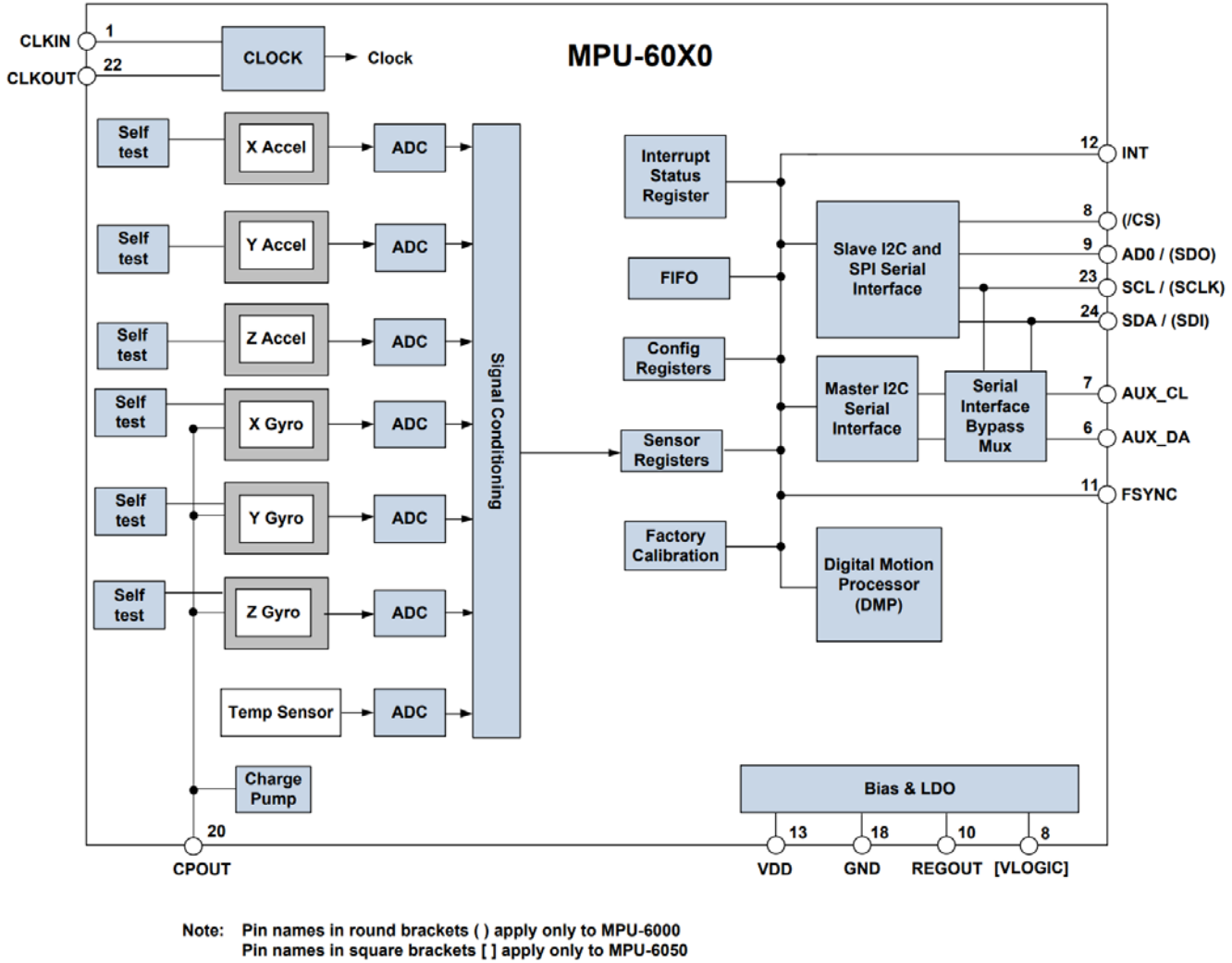


출처: https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf

IMU MPU6000

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

- **SPI connections**:
  - /CS: chip select
    - pin 53 in APM
  - SDI : MOSI
  - SDO: MISO
  - SCLK: clock
  - INT: Interrupt digital output
    - PE6 (AVR port)
  - FSYNC:
    - supports image, video and GPS synchronization
    - PJ3 (AVR port)



출처: https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf

IMU MPU6000

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

- **설정용 내부 레지스터**:
  - 측정 sample rate
  - DLP (low pass filter) setup
    - 0: 260Hz ~ 6: 5Hz
  - Gyro scale
    - 0: ± 250 °/s, 1:± 500 °/s,
    - 2:± 1000 °/s, 3: ± 2000 °/s
  - Accelerometer scale
    - 0: ± 2g, 1: ± 4g,
    - 2: ± 8g, 3: ± 16g
  - I2C disable
  - device reset, clock select
  - Interrupt enable

Adr.	Register	Purpose	Descriptions						
19	SMPLRT_DIV	setup	Gyro Output Rate(8kHz)를 기준으로 Sample rate를 설정						
			SMPLRT_DIV[7:0]						
1A	CONFIG	setup	저역필터 DLP 주파수 설정						
					EXT_SYNC_SET[2:0]			DLPF_CFG[2:0]	
1B	GYRO_CONFIG	setup							
						FS_SEL [1:0]			
1C	ACCEL_CONFIG	read							
						AFS_SEL[1:0]			
6A	USER_CTRL	setup	I2C_MST_EN						
						I2C_IF _DIS			
6B	PWR_MGMT_1	setup	DEVICE _RESET						
			DEVICE _RESET					CLKSEL[2:0]	
37	INT_PIN_CFG	setup							
						INT_RD _CLEAR			
38	INT_ENABLE	setup							
									RAW_R DY_EN

IMU MPU6000

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

내부 레지스터:

- 데이터 레지스터:
  - 주소: 0x3B~0x48
  - 데이터
    - Acceleration x, y, z
    - Temperature
    - Gyro x, y, z
  - big endian
    - "big end first"
  - 16bit data
    - data= High<<8 + Low;

Adr.	Register	Purpose	Descriptions
3B	ACCEL_XOUT_H	read	Acc. x in 16bit
3C	ACCEL_XOUT_L	read	
3D	ACCEL_YOUT_H	read	Acc. y in 16bit
3E	ACCEL_YOUT_L	read	
3F	ACCEL_ZOUT_H	read	Acc. 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 x in 16bit
44	GYRO_XOUT_L	read	
45	GYRO_YOUT_H	read	Gyro y in 16bit
46	GYRO_YOUT_L	read	
47	GYRO_ZOUT_H	read	Gyro z in 16bit
48	GYRO_ZOUT_L	read	

IMU MPU6000

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

Initialize sequence:

① Reset: reset the chip

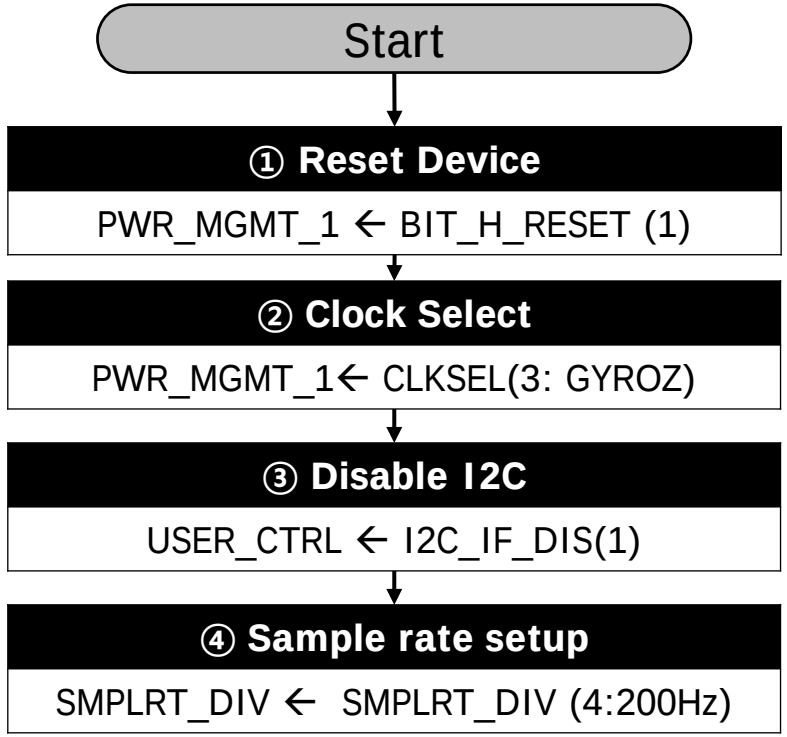
② Clock select:

CLKSEL	0	1	2	**3**	4	5
Source	Internal 8MHz	Gyro X	Gyro Y	**Gyro Z**	External 32.768	External 19.2MHz

③ Disable I2C

④ Sample rate setup

- Sample Rate = Gyro Output Rate/(1+SMPLRT_DIV)
- Gyro Output Rate= 1khz(DLP) ~8kHz



IMU MPU6000

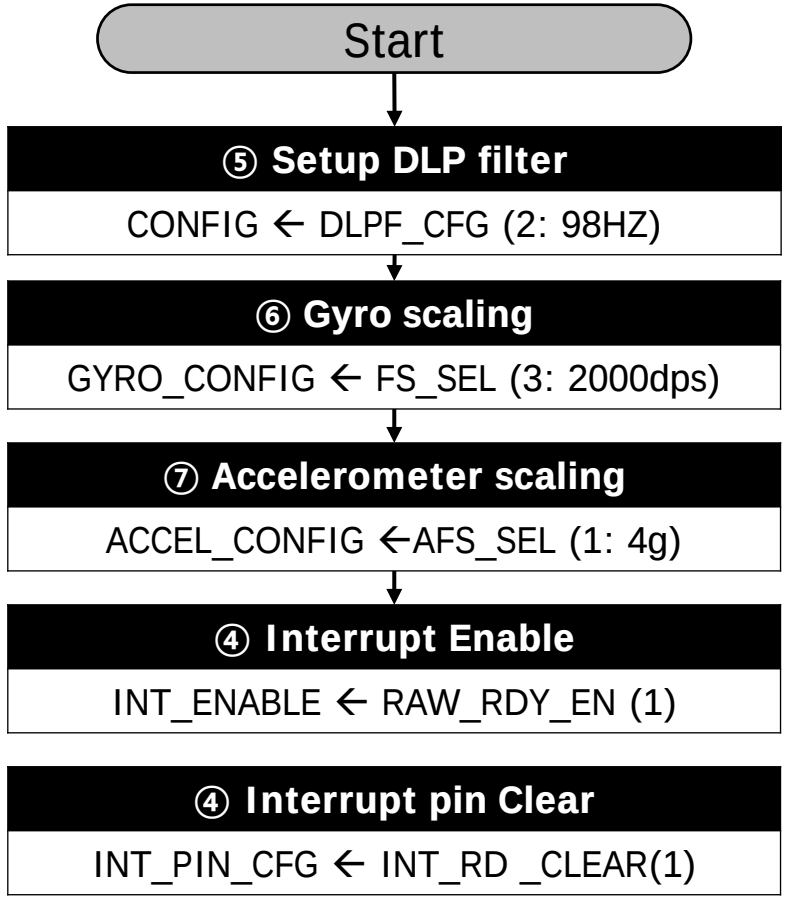
Dept. of Mechanical System Design, Seoul National University of Science and Technology.

Initialize sequence:

⑤ DLP(Digital Low Pass) filter 설정

CLKSEL	0	1	2	3	4	5	6
Accel. Bandwidth(Hz)	260	184	94	44	21	10	5
Gyro Bandwidth(Hz)	256	188	98	42	20	10	5
Gyro frequency (kHz)	8	1	1	1	1	1	1
Accel frequency (kHz)	1						

⑥ Gyro sensor scaling
⑦ Accelerometer scaling
⑧ Interrupt Enable
⑨ Interrupt pin Clear

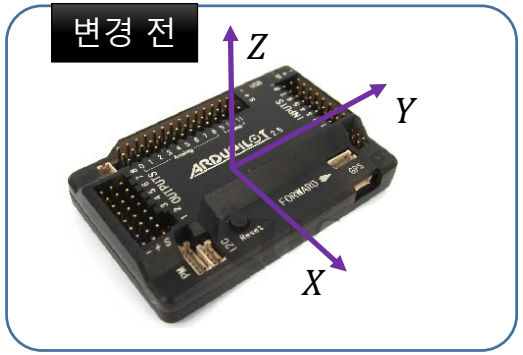


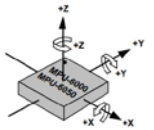
Algorithm for MPU6000

Redefining coordinate

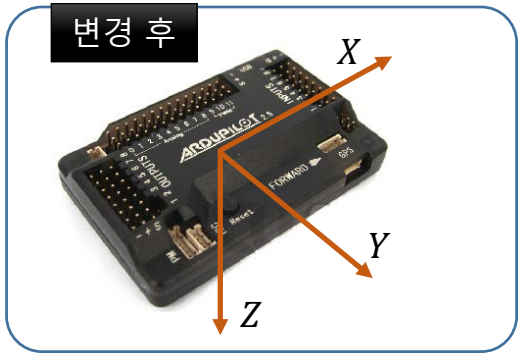
Dept. of Mechanical System Design, Seoul National University of Science and Technology.

- 우측 그림과 같이 좌표를 변경한다.
- 좌표 변경 이유
  - 원래는 전진 방향이 Y축으로 칩이 장착됨.
  - 그런데 항공기 제어에서는 주로
    - **NED : North, East, Down** 방향을 기준
    - 전진 방향이 +x
    - 전진방향의 오른쪽이 +y
    - 중력 방향이 +z축이다.
- 필요 내용
  - $y' = x,\ x' = -y,\ z' = -z$





CHIP



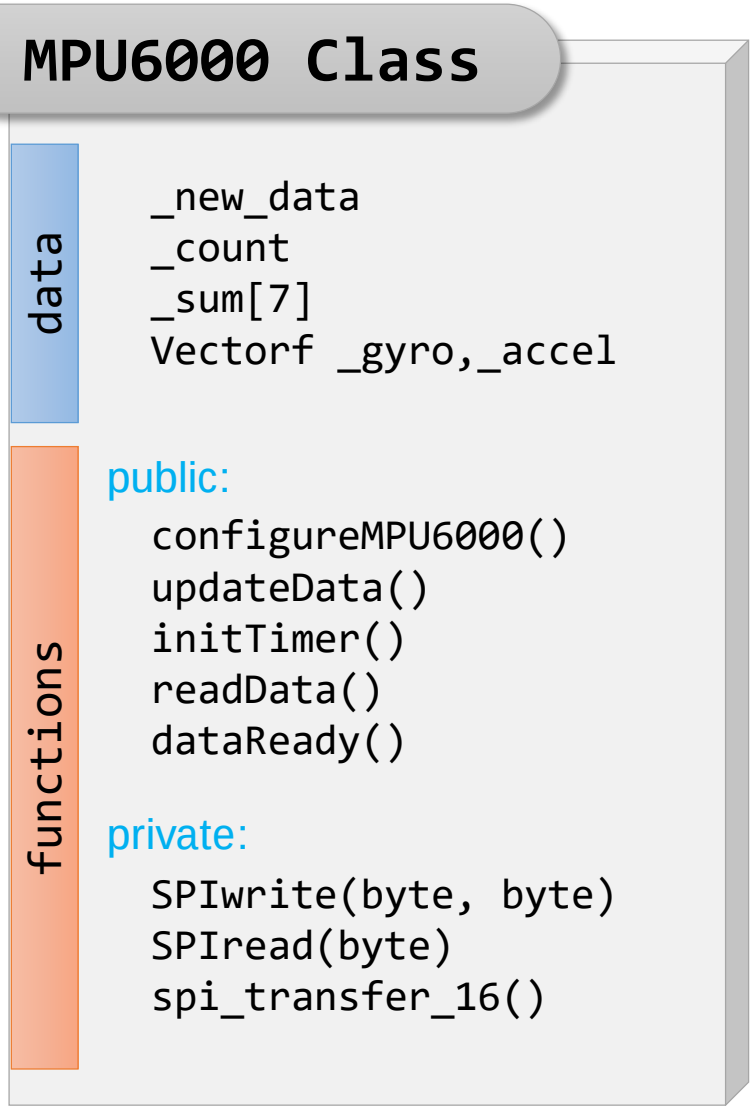
NED

MPU6000 선언

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

MPU6000 Class:

- Property
  - _new_data : 데이터 도착을 알림
  - _count: 데이터의 누적수를 보관
  - _sum[7]: 누적 평균용 배열
  - _gyro,_accel: gyro, accel데이터를 벡터로 보관
- Method
  - configureMPU6000(): 레지스터 초기화 설정
  - updateData(): 데이터를 업데이트
  - initTimer(): 타이머 초기화
  - readData(): 데이터 읽기
  - dataReady(): 외부 interrupt 처리

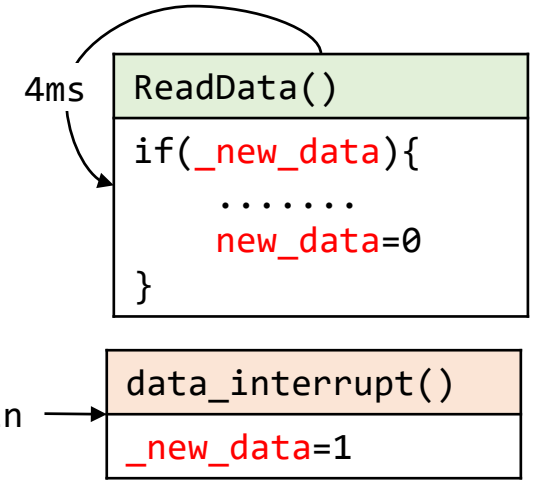


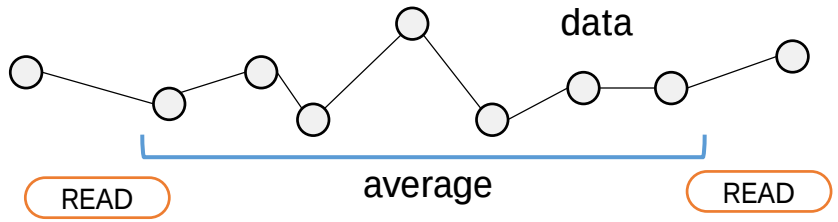
Algorithm

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

주요 특징

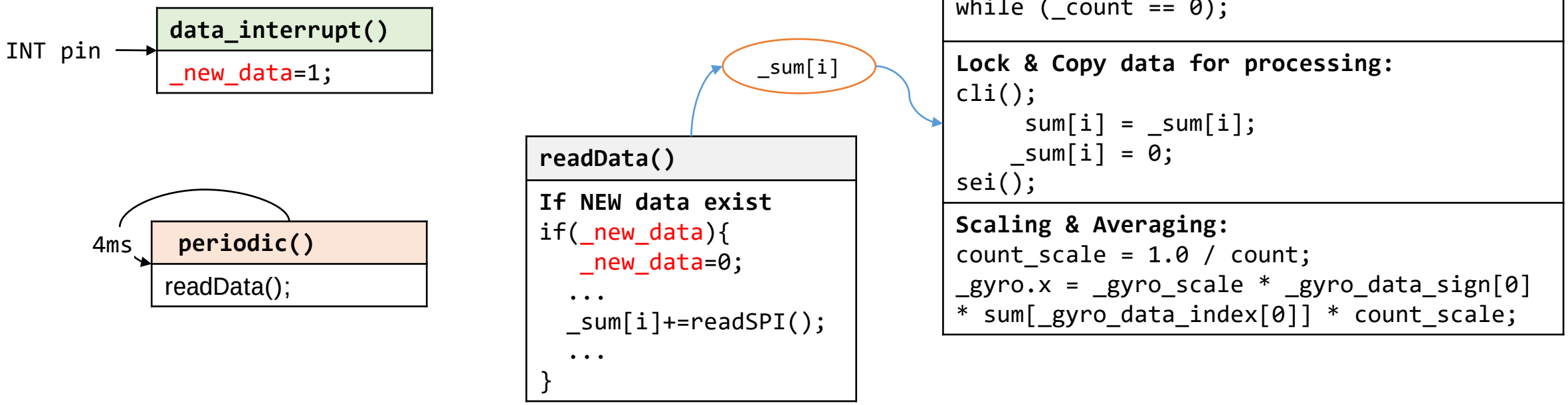
- SPI 통신 사용을 위하여 다른 칩을 비활성화 필요.
- Timer 인터럽트에 따라 일정시간 간격으로 동작
  - 4ms 주기 250Hz 정도
- INT핀의 rising 변화를 감지하는 외부 인터럽트사용
  - 유효한 데이터가 있을 때만 읽도록 한다.
  - 불필요한 읽기를 배제
- 데이터를 읽지 않아도 계속 데이터를 **누적평균**하는 기능





동작원리

Dept. of Mechanical System Design, Seoul National University of Science and Technology.



MPU6000 Programming

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

- MPU6k.h
  - register

MPU6k.h

```
#ifndef __MPU6K_H__
#define __MPU6K_H__
#include <stdint.h>
#include <Arduino.h>
//===============Registers of MPU6000
#define MPUREG_WHOAMI       0x75
#define MPUREG_SMPLRT_DIV   0x19
#define MPUREG_CONFIG       0x1A
#define MPUREG_GYRO_CONFIG  0x1B
#define MPUREG_ACCEL_CONFIG 0x1C
#define MPUREG_FIFO_EN      0x23
#define MPUREG_INT_PIN_CFG  0x37
#define MPUREG_INT_ENABLE   0x38
#define MPUREG_INT_STATUS   0x3A
#define MPUREG_ACCEL_XOUT_H 0x3B
#define MPUREG_ACCEL_XOUT_L 0x3C
#define MPUREG_ACCEL_YOUT_H 0x3D
#define MPUREG_ACCEL_YOUT_L 0x3E
#define MPUREG_ACCEL_ZOUT_H 0x3F
#define MPUREG_ACCEL_ZOUT_L 0x40
#define MPUREG_TEMP_OUT_H   0x41
#define MPUREG_TEMP_OUT_L   0x42
#define MPUREG_GYRO_XOUT_H  0x43
#define MPUREG_GYRO_XOUT_L  0x44
#define MPUREG_GYRO_YOUT_H  0x45
#define MPUREG_GYRO_YOUT_L  0x46
#define MPUREG_GYRO_ZOUT_H  0x47
#define MPUREG_GYRO_ZOUT_L  0x48
#define MPUREG_USER_CTRL    0x6A
#define MPUREG_PWR_MGMT_1   0x6B
#define MPUREG_PWR_MGMT_2   0x6C
#define MPUREG_FIFO_COUNTH  0x72
#define MPUREG_FIFO_COUNTL  0x73
#define MPUREG_FIFO_R_W     0x74
//============= Configuration BITS
#define BIT_SLEEP 0x40
#define BIT_H_RESET 0x80
#define BITS_CLKSEL 0x07
#define MPU_CLK_SEL_PLLGYROX 0x01
#define MPU_CLK_SEL_PLLGYROZ 0x03
#define MPU_EXT_SYNC_GYROX 0x02
#define BITS_FS_250DPS              0x00
#define BITS_FS_500DPS              0x08
#define BITS_FS_1000DPS             0x10
#define BITS_FS_2000DPS             0x18
#define BITS_FS_MASK                0x18
#define BITS_AFS_2G                 0x00
#define BITS_AFS_4G                 0x08
#define BITS_AFS_8G                 0x10
#define BITS_AFS_16G                0x18
#define BITS_DLPF_CFG_256HZ_NOLPF2  0x00
#define BITS_DLPF_CFG_188HZ         0x01
#define BITS_DLPF_CFG_98HZ          0x02
#define BITS_DLPF_CFG_42HZ          0x03
#define BITS_DLPF_CFG_20HZ          0x04
#define BITS_DLPF_CFG_10HZ          0x05
#define BITS_DLPF_CFG_5HZ           0x06
#define BITS_DLPF_CFG_2100HZ_NOLPF  0x07
#define BITS_DLPF_CFG_MASK          0x07
#define BIT_INT_ANYRD_2CLEAR        0x10
#define BIT_RAW_RDY_EN              0x01
#define BIT_I2C_IF_DIS              0x10
#define BIT_INT_STATUS_DATA         0x01
```

MPU6000 Programming

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

- MPU6k.h
  - class

```
//====================================
#define MPU6K_CS_PIN 53
#define EI_INT_6 6
const float _gyro_scale = PI/180*2000./32768.0;
const float _accel_scale = 4*9.81 / 32768.0;
const uint8_t _gyro_data_index[3]  = { 5, 4, 6 };
const int8_t  _gyro_data_sign[3]   = { 1, 1, -1 };
const uint8_t _accel_data_index[3] = { 1, 0, 2 };
const int8_t  _accel_data_sign[3]  = { 1, 1, -1 };
const uint8_t _temp_data_index = 3;
struct Vectorf{float x;float y; float z;};
class MPU6000{
  protected:
    static volatile uint8_t _new_data;
    static volatile uint16_t _count;
    static volatile int32_t _sum[7];
    void SPIwrite(byte reg, byte data);
    uint8_t SPIread(byte reg);
    static int16_t spi_transfer_16(void);
  public:
   Vectorf _gyro,_accel;
   void configureMPU6000();
   bool updateData();
   void initTimer();
   void readData();
   static void dataReady(void);
};
#endif
```

MPU6000 Programming

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

MPU6k.cpp

MPU6k.cpp <1/5>

```
#include "MPU6k.h"
#include <SPI.h>
#include <Arduino.h>
static volatile uint8_t MPU6000::_new_data=0;
static volatile uint16_t MPU6000::_count=0;
static volatile int32_t MPU6000::_sum[7]={0};
MPU6000 mpu;
//================= Configuration of MPU6k
void MPU6000::configureMPU6000() {
  SPIwrite(MPUREG_PWR_MGMT_1,BIT_H_RESET);             delay(100);// eset device
  SPIwrite(MPUREG_PWR_MGMT_1,MPU_CLK_SEL_PLLGYROZ);    delay(1); //select GyroZ clock
  SPIwrite(MPUREG_USER_CTRL,BIT_I2C_IF_DIS);           delay(1); //disable I2C interface
  SPIwrite(MPUREG_SMPLRT_DIV,0x04);                    delay(1); //Fsample=1Khz/(4+1)=200Hz
  SPIwrite(MPUREG_CONFIG,BITS_DLPF_CFG_98HZ);          delay(1); //DLPF = 98Hz (Gyro)
  SPIwrite(MPUREG_GYRO_CONFIG,BITS_FS_2000DPS);        delay(1); //Gyro scale 2000deg/s
  SPIwrite(MPUREG_ACCEL_CONFIG,BITS_AFS_4G);           delay(1); //accel scale at 4g(4096LSB/g)
  SPIwrite(MPUREG_INT_ENABLE,BIT_RAW_RDY_EN);          delay(1); //INT: Raw data ready
  SPIwrite(MPUREG_INT_PIN_CFG,BIT_INT_ANYRD_2CLEAR);   delay(1); //INT: Clear on any read
  attachInterrupt(EI_INT_6,dataReady,RISING);  // External Interrupt pin PE6 INT6
  initTimer();
}
```

MPU6000 Programming

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

MPU6k.cpp <2/5>

```
bool MPU6000::updateData( ) {
  int32_t sum[7];
  uint16_t count;
  float count_scale;
  while (_count == 0);   // wait for at least one sample
  cli();   // disable interrupts
  for (int i=0; i<7; i++) {
    sum[i] = _sum[i];
    _sum[i] = 0;
  }
  count = _count;
  _count = 0;
  sei();   //enable interrupts
  count_scale = 1.0 / count;
  _gyro.x = _gyro_scale * _gyro_data_sign[0] * sum[_gyro_data_index[0]] * count_scale;
  _gyro.y = _gyro_scale * _gyro_data_sign[1] * sum[_gyro_data_index[1]] * count_scale;
  _gyro.z = _gyro_scale * _gyro_data_sign[2] * sum[_gyro_data_index[2]] * count_scale;
  _accel.x = _accel_scale * _accel_data_sign[0] * sum[_accel_data_index[0]] * count_scale;
  _accel.y = _accel_scale * _accel_data_sign[1] * sum[_accel_data_index[1]] * count_scale;
  _accel.z = _accel_scale * _accel_data_sign[2] * sum[_accel_data_index[2]] * count_scale;
  return true;
}
```

MPU6000 Programming

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

MPU6k.cpp <3/5>

```
void MPU6000::readData( ){
    if (_new_data == 0) return; // no new data then exit
    _new_data = 0;
    digitalWrite(MPU6K_CS_PIN, LOW);
    byte addr = MPUREG_ACCEL_XOUT_H | 0x80;// R/W bit
    SPI.transfer(addr);
    for (uint8_t i=0; i<7; i++) {
        _sum[i] += spi_transfer_16();
    }
   _count++;
    digitalWrite(MPU6K_CS_PIN, HIGH);
}

//=================== SPI Read/Write ==================
void MPU6000::SPIwrite(byte reg, byte data) {
  uint8_t dump;
  digitalWrite(MPU6K_CS_PIN,LOW);
  dump=SPI.transfer(reg);
  dump=SPI.transfer(data);
  digitalWrite(MPU6K_CS_PIN,HIGH);
}
```

MPU6000 Programming

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

MPU6k.cpp <4/5>

```
uint8_t MPU6000::SPIread(byte reg) {
  uint8_t dump, return_value, addr=reg|0x80; // R/W bit
  digitalWrite(MPU6K_CS_PIN,LOW);
  dump=SPI.transfer(addr);
  return_value=SPI.transfer(0x00);
  digitalWrite(MPU6K_CS_PIN,HIGH);
  return return_value;
}
static int16_t MPU6000::spi_transfer_16(void) {
  uint8_t byte_H, byte_L;
  byte_H = SPI.transfer(0);
  byte_L = SPI.transfer(0);
  return (((int16_t)byte_H)<<8) | byte_L;
}
```

MPU6000 Programming

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

MPU6k.cpp <5/5>

```
//================= Interrupts ==================
void MPU6000::dataReady(void) {
    _new_data = 1;  // new data arrived for read
}
void MPU6000::initTimer(){
  TIMSK2 = 0;                     // Disable interrupts
  TCCR2A = 0;                     // normal counting mode
  TCCR2B = _BV(CS21) | _BV(CS22); // Set prescaler of 256 -> 16usec
  TCNT2  = 0;                     // Set count=0
  TIFR2  = _BV(TOV2);             // clear previous interrupts;
  TIMSK2 = _BV(TOIE2);            // enable overflow interrupt
}
ISR(TIMER2_OVF_vect) {
    mpu.readData();
}
```

MPU6000 Programming

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

- test_MPU6k.ino

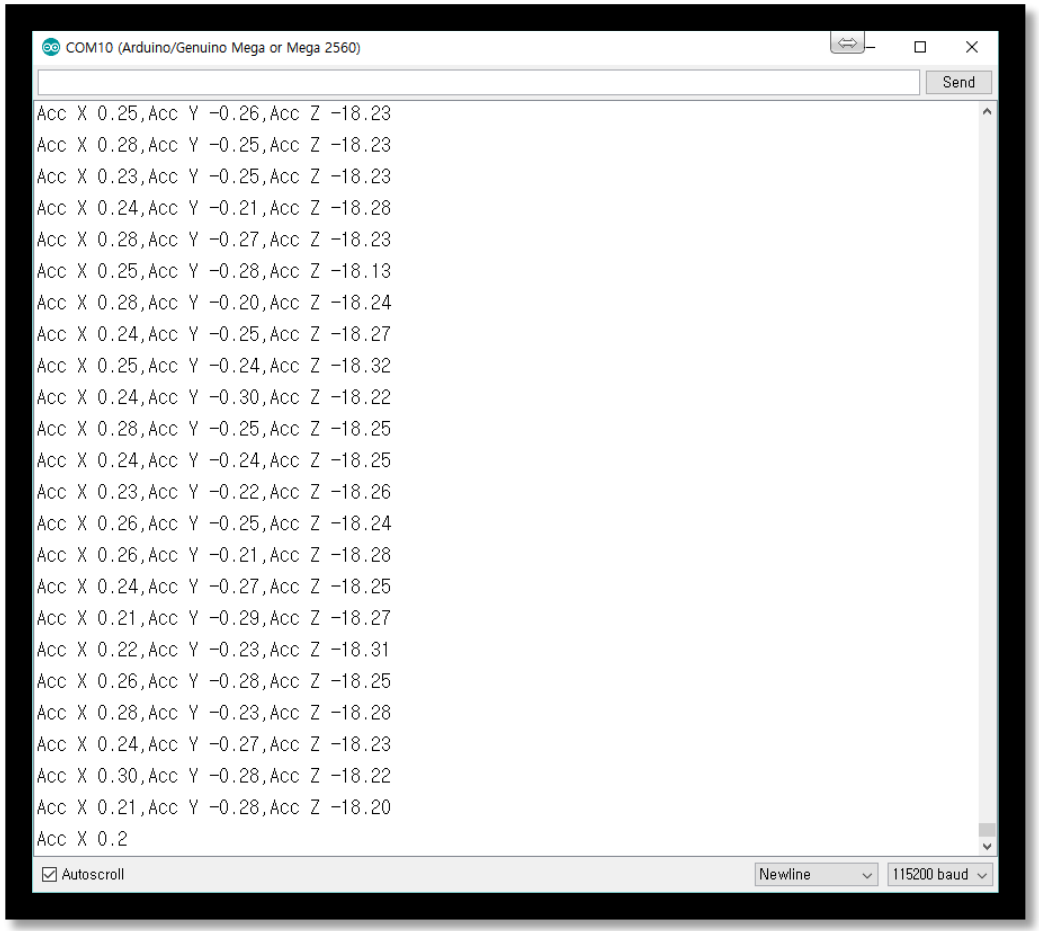
test_MPU6k.ino

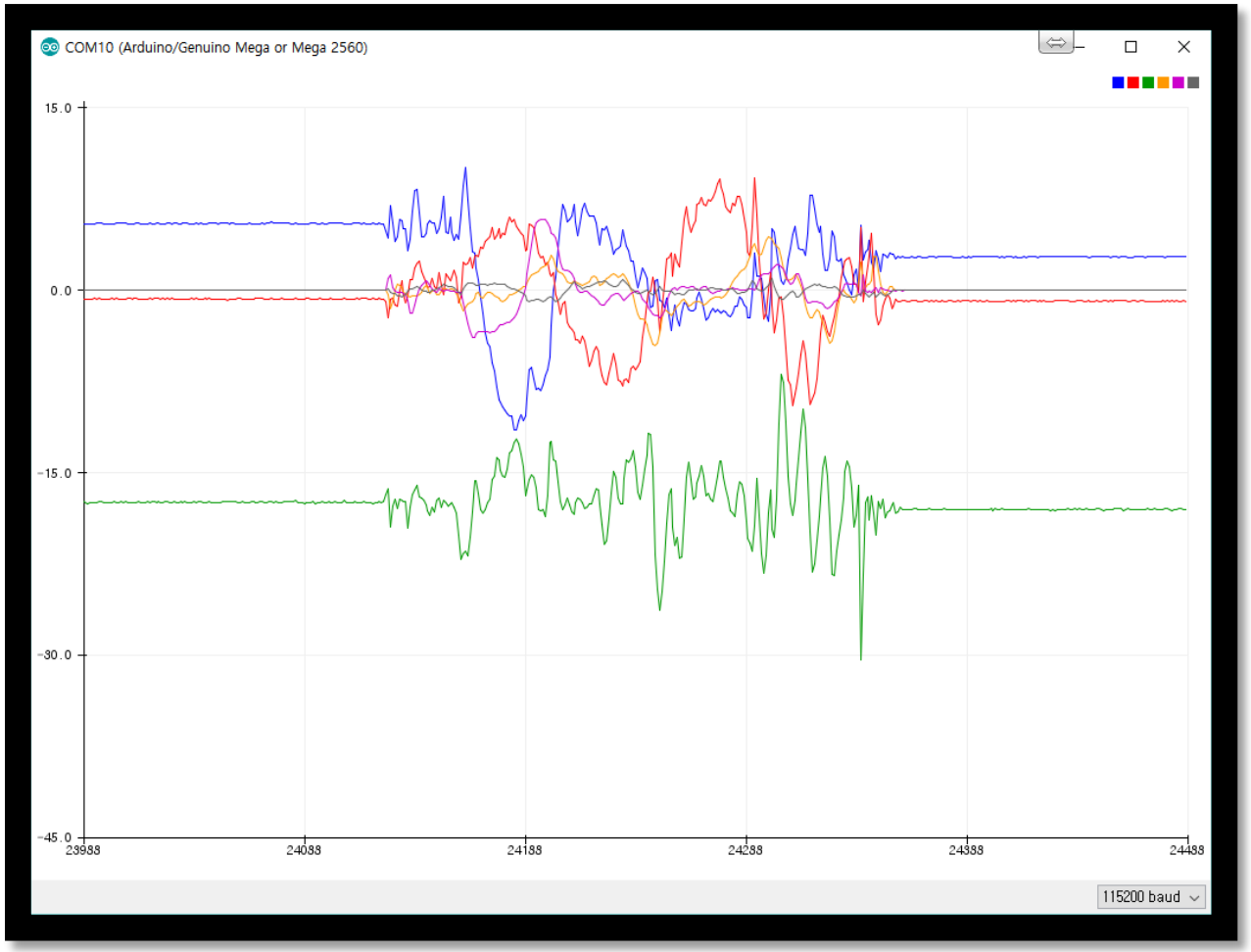
```
#include <SPI.h>
#include "MPU6k.h"
#define BARO_CS_PIN 40
extern MPU6000 mpu;
void setup() {
  Serial.begin(115200);
  pinMode(BARO_CS_PIN, OUTPUT);  pinMode(12, OUTPUT);
  pinMode(MPU6K_CS_PIN, OUTPUT); digitalWrite(MPU6K_CS_PIN, HIGH);
  digitalWrite(BARO_CS_PIN, HIGH);   //Diselect Barometer
  SPI.begin();
  SPI.beginTransaction(SPISettings(8000000, MSBFIRST, SPI_MODE0));
  delay(100);
  mpu.configureMPU6000();  // configure chip
}
void loop() {
   mpu.updateData( );
  Serial.print(mpu._accel.x); Serial.print(",");
  Serial.print(mpu._accel.y); Serial.print(",");
  Serial.print(mpu._accel.z); Serial.print(",");
  Serial.print(mpu._gyro.x);Serial.print(",");
  Serial.print(mpu._gyro.y);  Serial.print(",");
  Serial.print(mpu._gyro.z);
  Serial.print("\n");
}
```

Results

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

Results

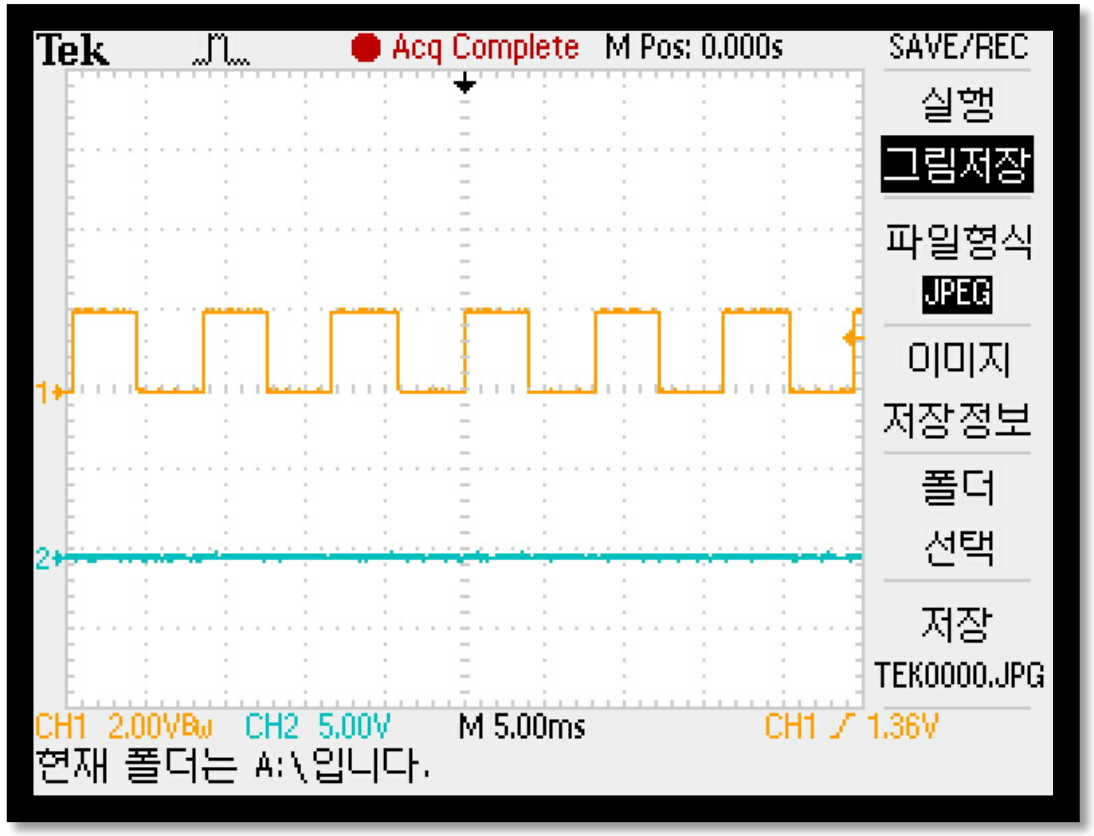




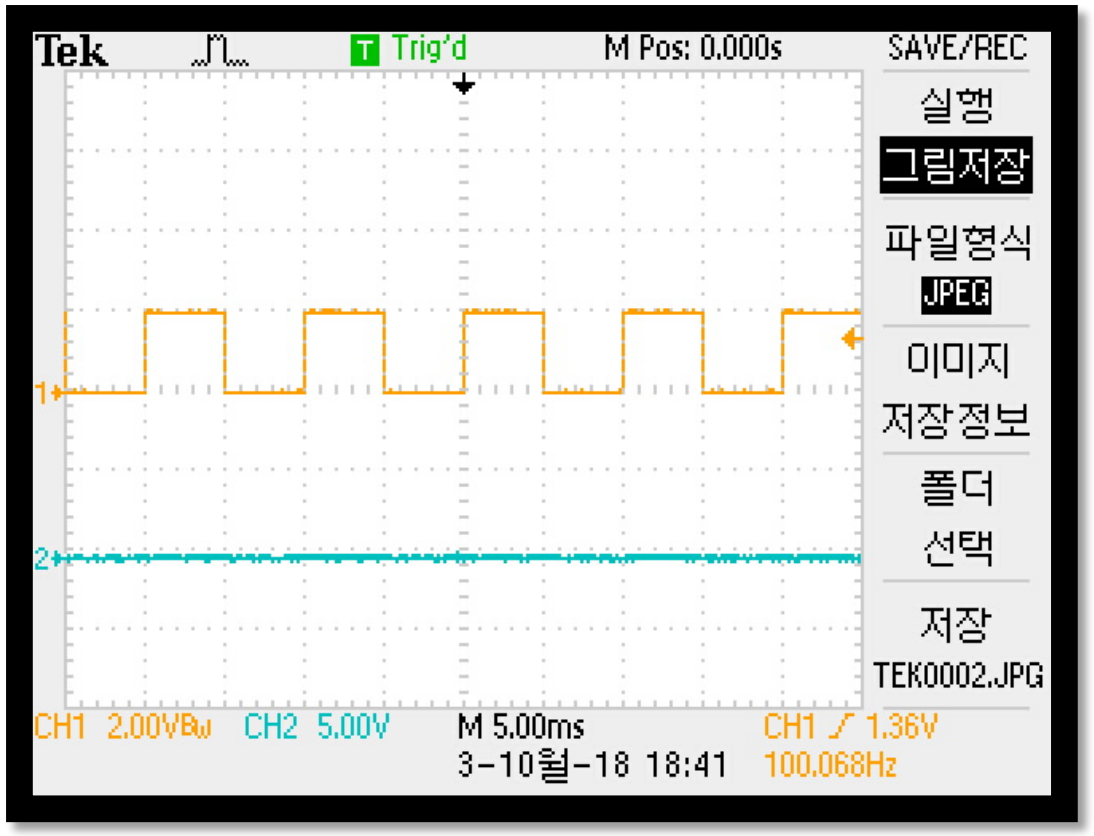
Timing

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

Timer 4ms



Timer 4ms



EI interrupt 5ms

THANK YOU

Powerpoint is a complete presentation graphic package it gives you everything you need to produce a professional-looking presentation