NS-RX231를 위한 소프트웨어가이드

<MPU-6050 6축센서>

목차

1 프로젝트 가져오기

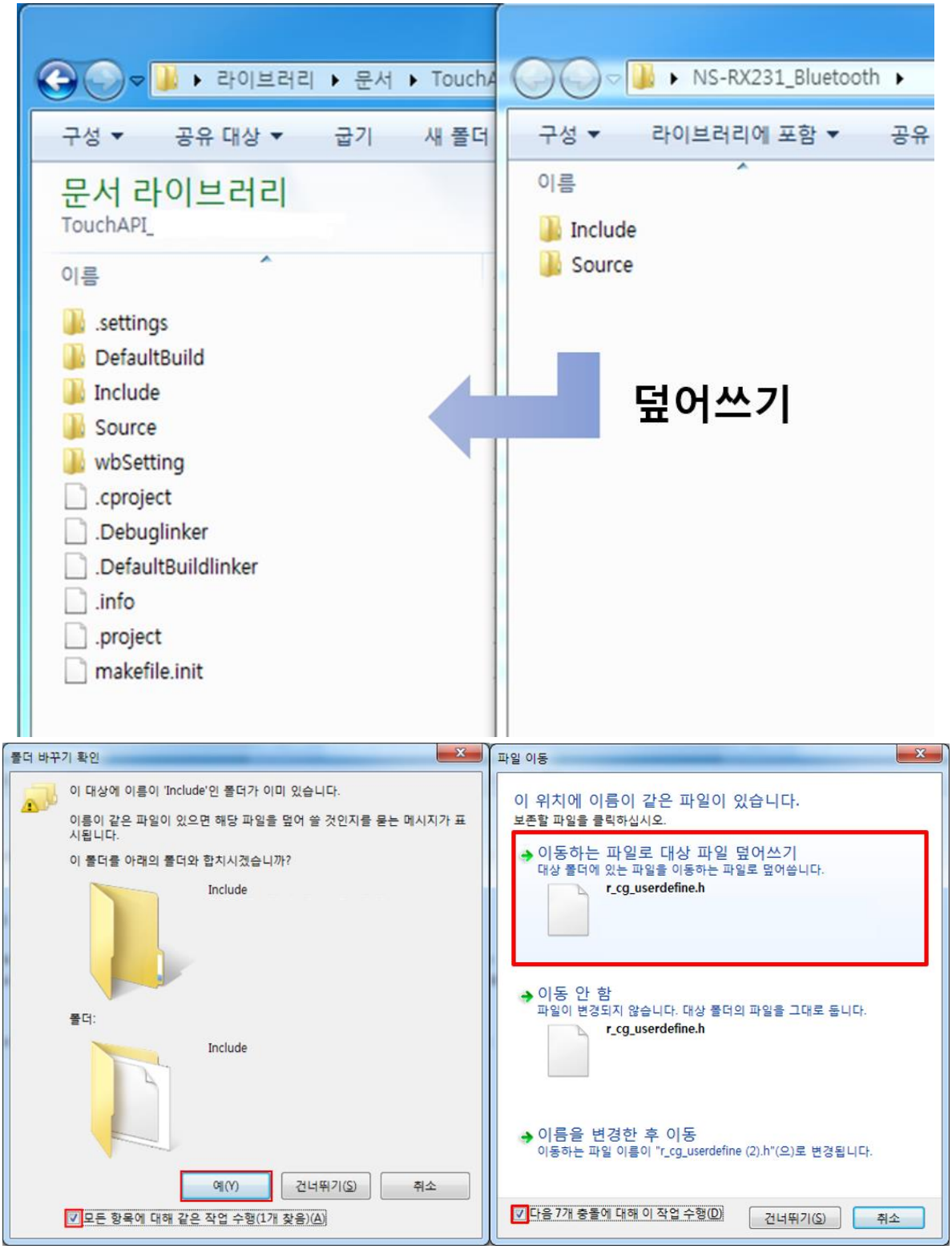


그림 1-1 소스파일 덮어쓰기

첨부된 소스파일을 Workbench6 First step guide 마법사로 만든 프로젝트에 덮어쓰기 한 후, e2studio에서 실행합니다.

2 개요

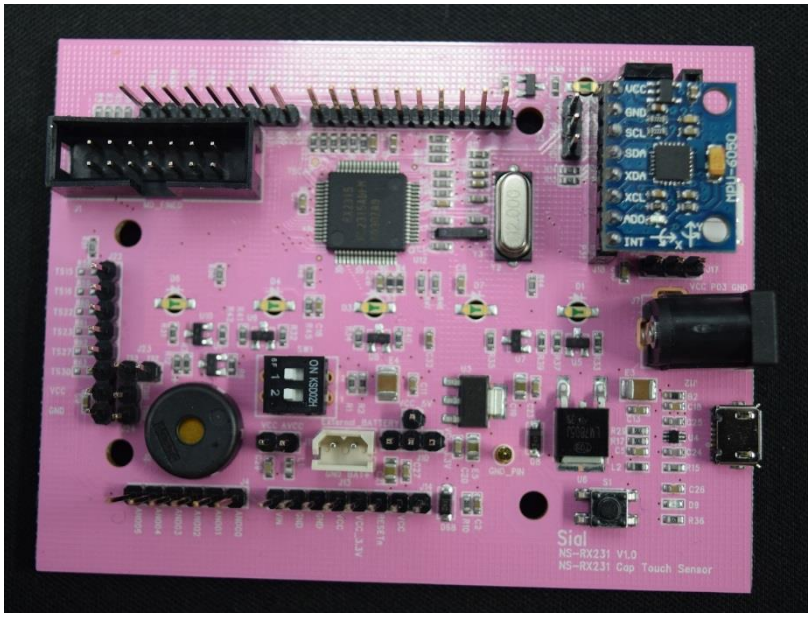


그림 2-1 MPU-6050을 장착한 NS-RX231

MPU-6050 은 MPU-6050 은 3 축 자이로 센서, 3 축 가속도 센서, 온도 센서의 데이터를 융합하여 자세 제어 시스템에 사용할 수 있습니다. 본 가이드에서는 NS-RX231 보드를 이용하여 MPU-6050 및 I2C 통신을하는 방법을 설명합니다.

MPU-6050 은 I2C 통신을 사용해 데이터를 주고받게 됩니다. I2C 통신은 SCL 과 SDA 두 선만을 이용해 한 개의 마스터로 여러 슬레이브 장치들을 제어하는 등, 간단한 구성으로 많은 장치를 제어할 수 있는 장점을 가지고 있습니다. 하지만 SPI, UART 같은 다른 통신들과 비교해보면 속도가 느리다는 단점이 있습니다.

SCL 은 마스터와 슬레이브 간의 동기화를 위한 클럭을 출력하며, 그 클럭에 맞춰 SDA 핀으로 통신 데이터가 교환하게 됩니다.

MPU-6050 의 I2C 버스 주소 값은 AD0(Address0)핀에 의해 정해집니다. AD0 는 최하위 비트를 지정하며 HIGH 일 때 1, LOW 일 때 0 이 되게 됩니다. 주소값이 7 비트인 이유는 R/W 비트를 포함하면 8 비트가 되기 때문입니다. R/W 비트는 밑에 서술할 **7. 기타** 부분에 설명되어 있습니다.

I^2C ADDRESS	AD0 = 0 AD0 = 1	1101000 1101001

그림 2-2 AD0핀의 설명

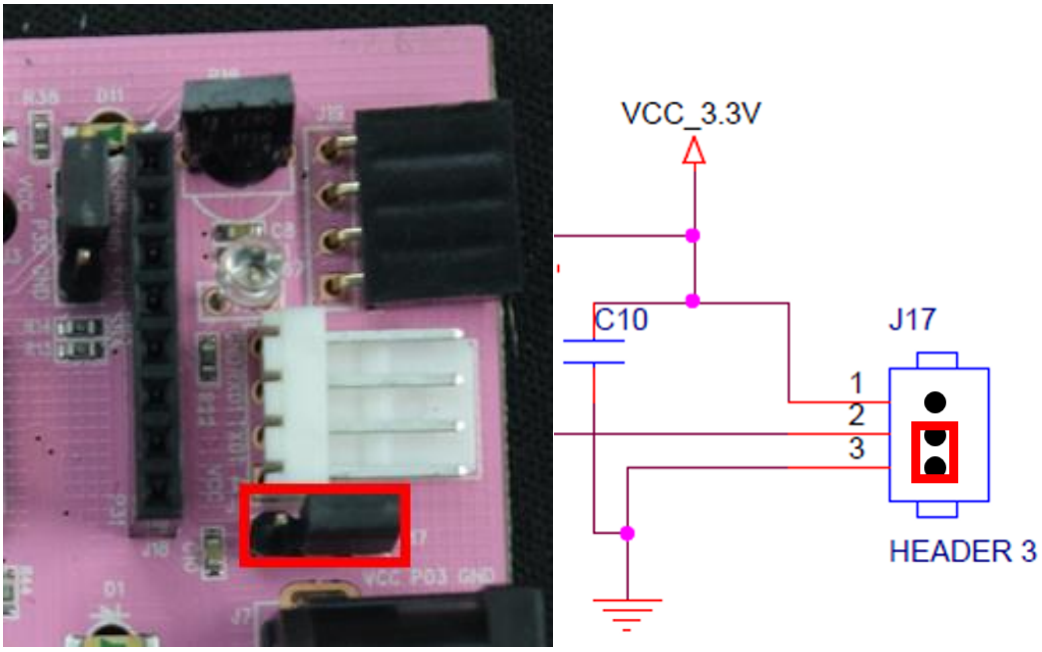


그림 2-3 점퍼 연결도

위 사진과 같이 기본적으로 J17 번의 점퍼 연결은 2-3 번으로, AD0 가 LOW 로 맞추어져 있습니다. 만약 주소 값을 1101001 로 바꾸고 싶다면, J17 번의 점퍼 연결을 1-2 번으로 바꿔주면 됩니다.

3 소스코드

```
void request(){
    int i,j;
    uint8_t chrArr[7][14] =
    {"AxisX : ","AxisY : ","AxisZ : ",
     "Temperature : ","GyroX : ","GyroY : ","GyroZ : "};
    uint8_t rx_buf[14] = {0,};

    R_RIIC0_Master_Receive(0x68, 0x3B, rx_buf, 14);

    for(i = 0 ; i < 7 ; i++){
        j = i*2;
        sci1_print(chrArr[i],i == 3 ? 14 : 8);
        sendAscii((rx_buf[j] << 8) | rx_buf[j+1]);
    }
    sci1_println("--------------------", 20);
}

void sendAscii(int value){
    int count, i = 0;
    int temp = value;
    uint8_t * arr = 0;
    if(value != 0){
        while( temp > 0 )
        {
            temp /= 10;
            i++;
        }
        count = i;
        while( i > 0 )
        {
            arr[--i] = (value % 10) + 48;
            value  /= 10;
        }
        sci1_println(arr,count);
    }
}
```

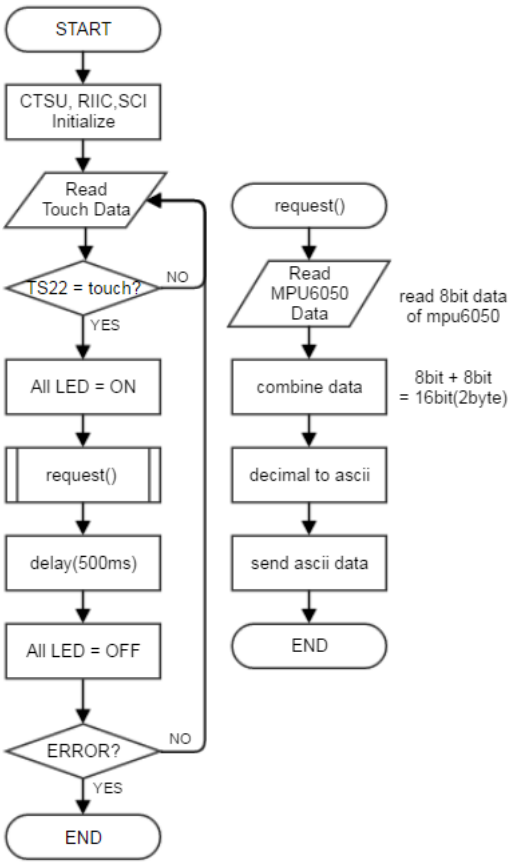


그림 3-1 소스코드 및 순서도

4 디버깅

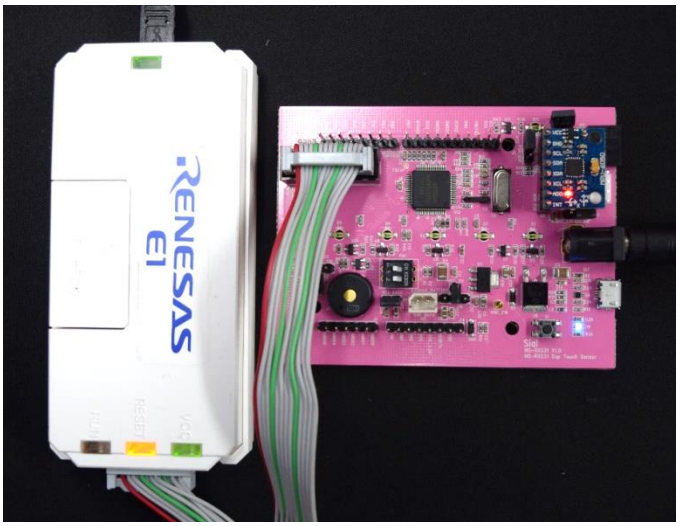


그림 4-1 NS-RX231을 전원 어댑터와 E1디버거를 연결한 모습.

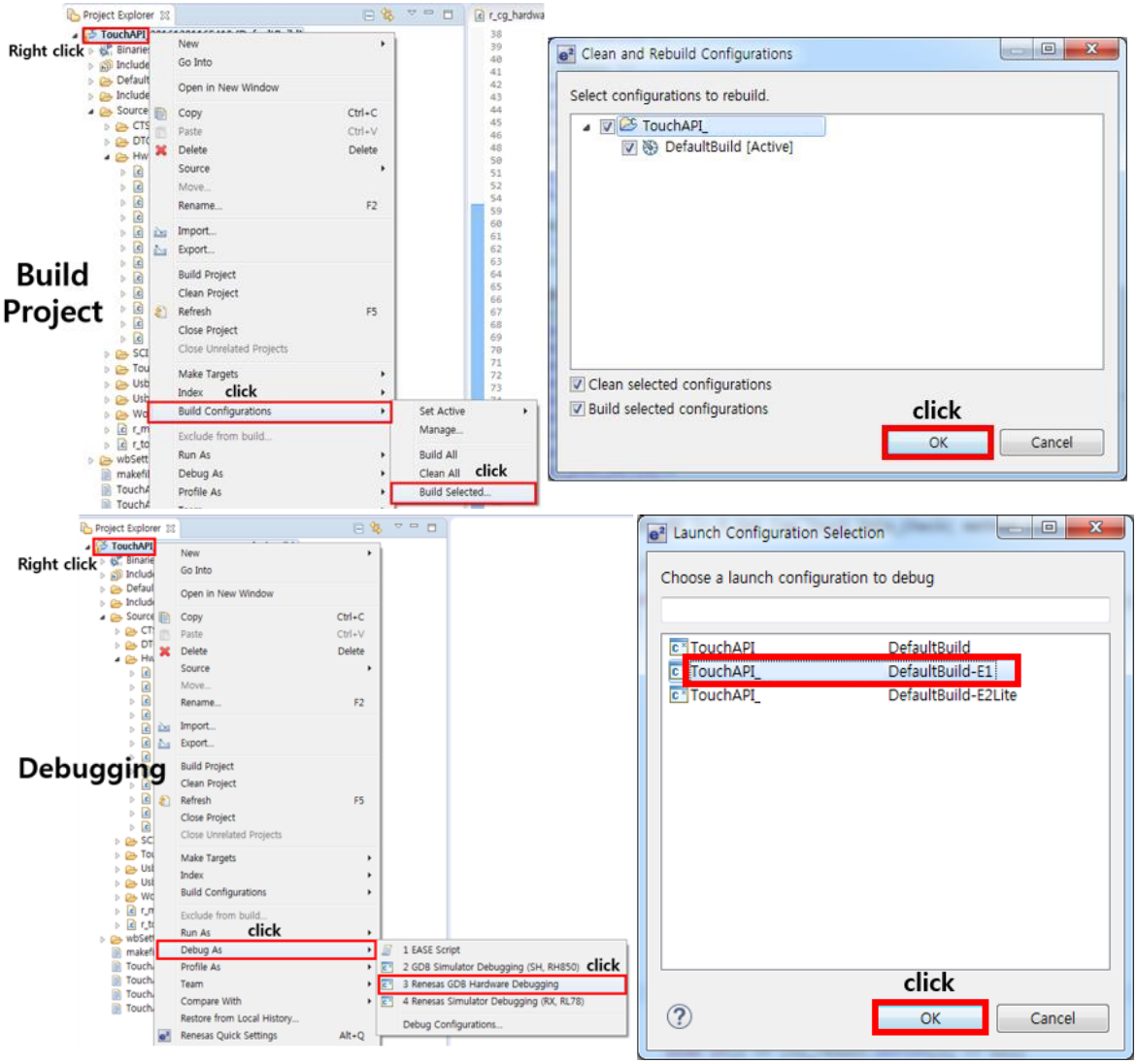
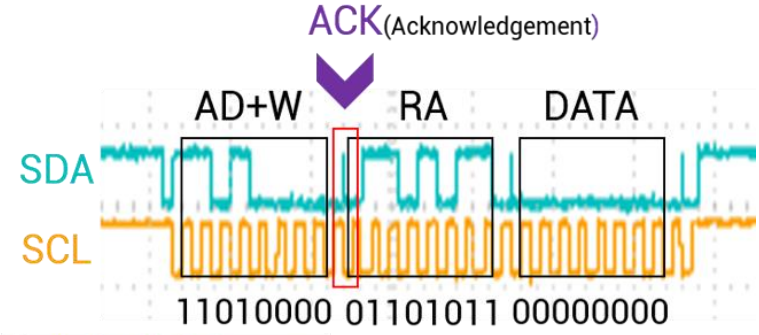


그림 4-2 프로젝트 빌드 및 디버깅

5 실행

우선, 데이터를 받아오기 위해서는 MPU-6050의 107(0x6B)번지 주소에 있는 SLEEP비트를 0으로 만들어 MPU-6050을 활성화 해야 합니다.



Single-Byte Write Sequence

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

그림 5-1 1바이트 쓰기 순서도

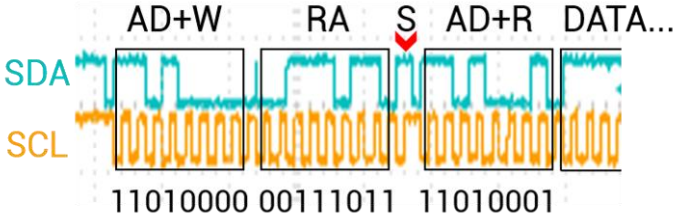
ACK는 확인 응답 신호로써, 장치에서 데이터를 제대로 받았다는 의미를 가집니다.
신호의 크기는 다를 수 있습니다.

MPU-6050을 활성화 시킨 이후, 데이터를 수신하기 시작합니다.

3B	59	ACCEL_XOUT_H
3C	60	ACCEL_XOUT_L

그림 5-2 MPU-6050장치의 내부 레지스터 주소 값

MPU-6050으로부터 가속도 X, Y, Z / 온도 / 자이로 X, Y, Z 총 7개의 데이터를 받아올 수 있습니다. 각각의 데이터는 2바이트(16비트)이지만, 데이터를 수신 할 때에는 상위 8비트인 _H와 하위 8비트인 _L두 개로 나뉘어져 있기 때문에, 버퍼에 데이터를 담은 후 합쳐 정리합니다.



Burst Read Sequence

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

그림 5-3 여러 바이트 읽기 순서도

본 소스코드는 NS-RX231에서 MPU-6050의 데이터를 수신 한 후 UART통신으로 데이터를 전송함으로써, 핀헤더의 TXD1, RXD1핀에 연결된 블루투스 모듈을 꽂아 데이터를 받을 수 있습니다.

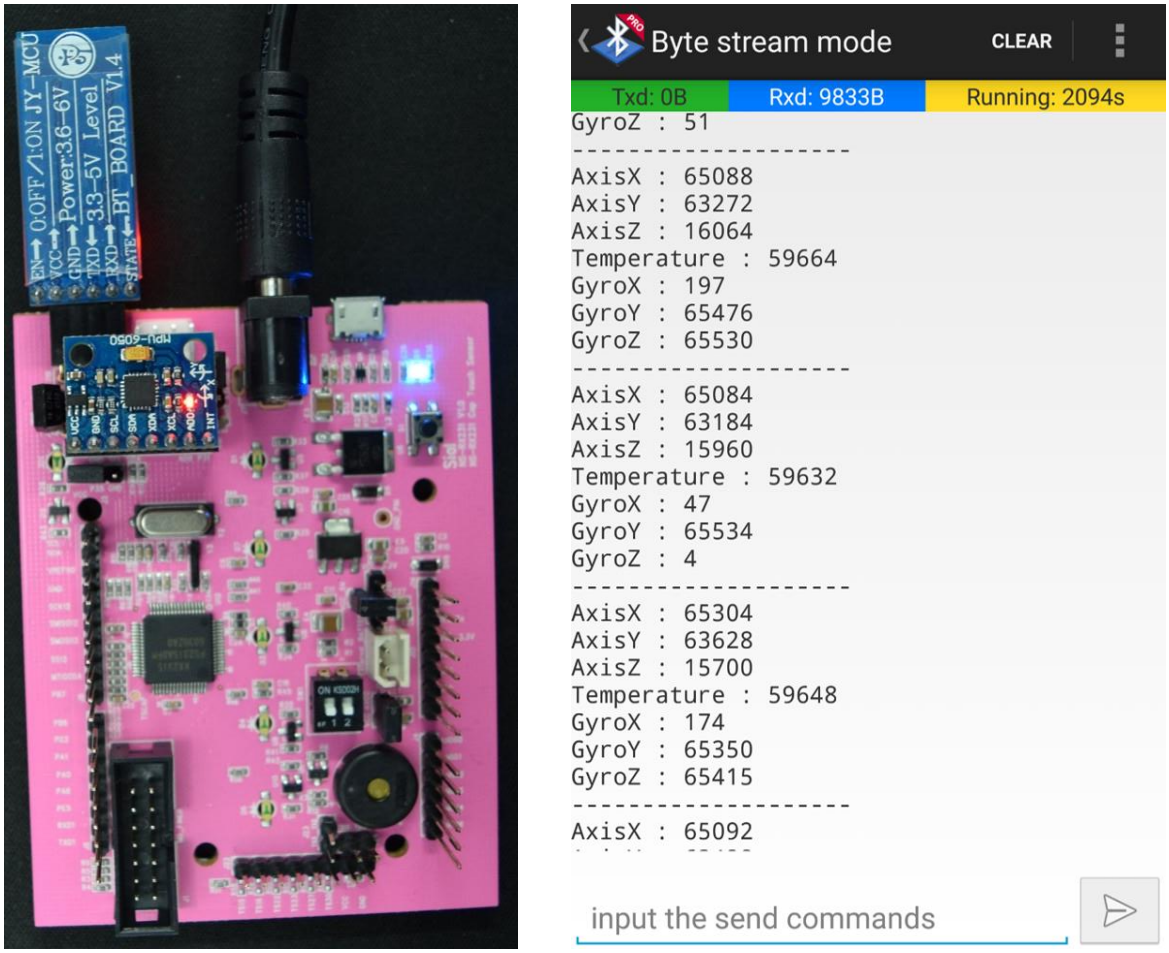


그림 5-4 MPU-6050 동작

블루투스 모듈을 꽂아 핸드폰과 연결해 MPU-6050의 데이터를 받아보았습니다.

6 회로도

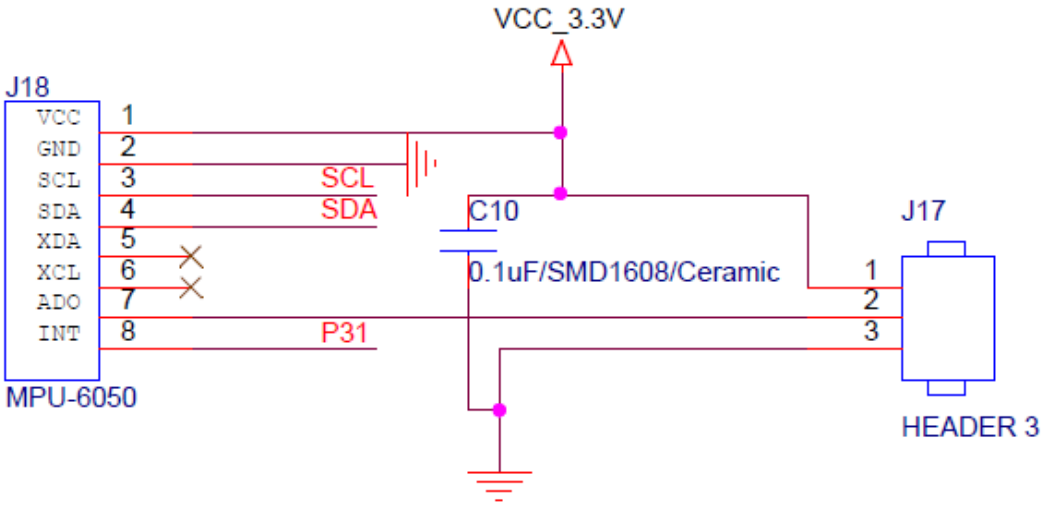


그림 6-1 NS-RX231 MPU-6050단자 회로도

7 기타

Burst Read Sequence

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

그림 7-1 데이터 연속 읽기 순서도

위의 사진인 여러 개 읽기 순서도를 보면, 마스터는 통신 시작을 알리는 Start Condition(S) 이후, 장치 주소 값(AD+W)을 보내고, 읽어오고 싶은 장치의 주소 값(RA)을 전송합니다. 그리고 통신 방향의 전환을 의미하는 Restart Condition(S)을 출력합니다.

Signal	Description
S	Start Condition: SDA goes from high to low while SCL is high
AD	Slave I_2C address
W	Write bit (0)
R	Read bit (1)
ACK	Acknowledge: SDA line is low while the SCL line is high at the 9_{th} clock cycle
NACK	Not-Acknowledge: SDA line stays high at the 9_{th} clock cycle
RA	MPU-60X0 internal register address
DATA	Transmit or received data
P	Stop condition: SDA going from low to high while SCL is high

그림 7-2 순서도 신호의 자세한 설명

R/W비트는 AD(7비트) + R/W(1비트)로 구성된 8비트 데이터로 전송되게 됩니다.

AD+W

```
RIIC0.ICDRT = (uint8_t)(g_riic0_slave_address << 1);
g_riic0_state = _04_IIC_MASTER_SENDS_ADR_10B;
```

AD+R

```
RIIC0.ICDRT = (uint8_t)((g_riic0_slave_address << 1) | 0x0001U);
g_riic0_state = _08_IIC_MASTER_RECEIVES_START;
```

그림 7-3 AD+W & AD+R