Software Guide for NS-RX231

<MPU-6050>

Contents

1 Load the Project

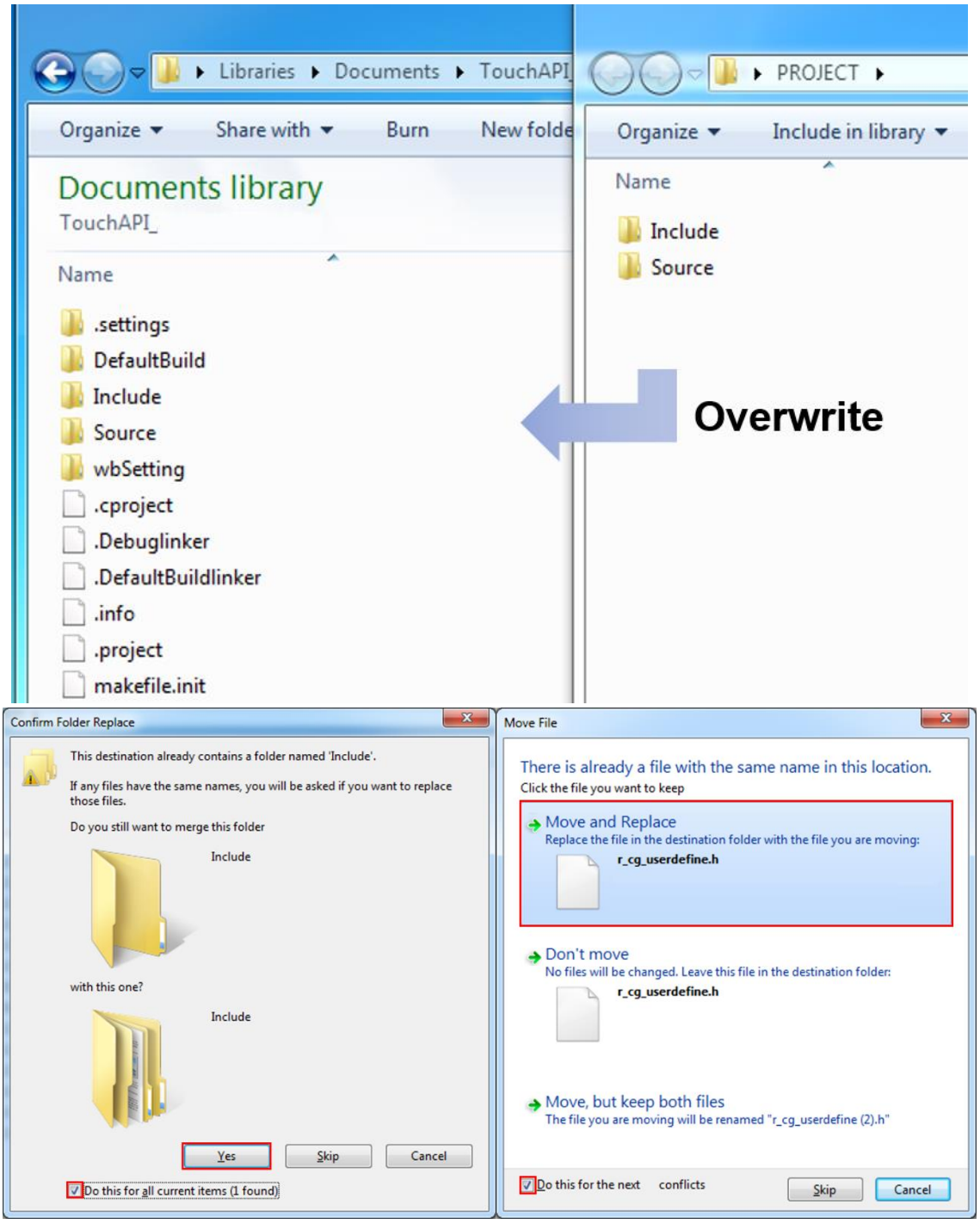


Figure 1-1 Overwrite the source file

Overwrite the attached source file with the project created by the Workbench6 First step guide wizard and run it in e2studio.

2 Summary

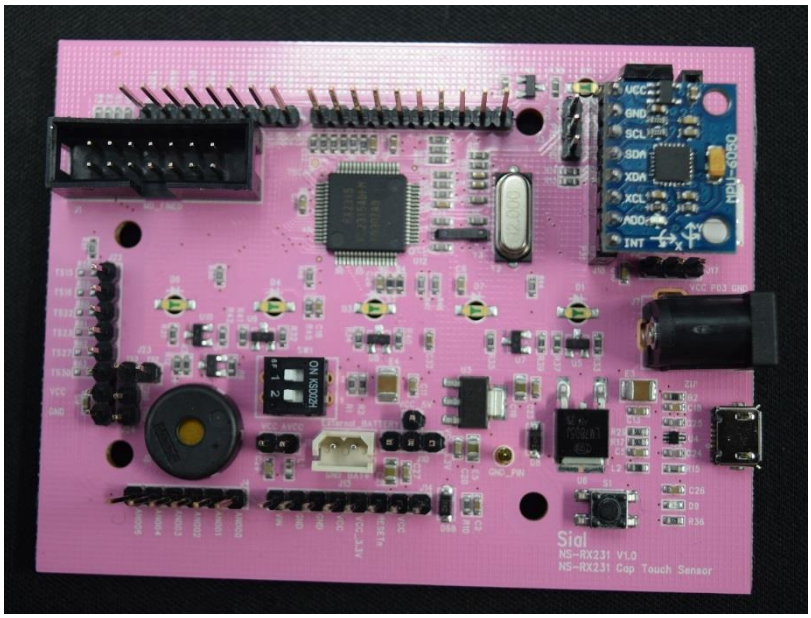


Figure 2-1 NS-RX231 with MPU6050

The MPU6050 can be used for attitude control system by merging data of 3-axis gyro sensor, 3-axis acceleration sensor and temperature sensor. There are Kalman filters, complementary filters, etc. for data fusion, but this document deals with I2C communication with MPU6050 using NS-RX231 board.

The MPU6050 uses I2C communication to send and receive data. I2C communication has the advantage of controlling many devices with simple configuration such as controlling multiple slave devices with one master using only two lines of SCL and SDA. However it has a disadvantage to be slow in communication compared to other communications, such as SPI and UART.

The SCL outputs the clock for synchronization between the master and the slave, and the communication data is transferred to the SDA pin according to the clock.

The address value of the MPU6050 is determined by the AD0 (Address0) pin. AD0 specifies the least significant bit,
1 when HIGH and 0 when LOW. The reason why ADDRESS value is 7 bits is that it is 8 bits if R / W bit is included. The R / W bit is described in section **7. Etc**.

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

Figure 2-2 Description of the AD0 pin

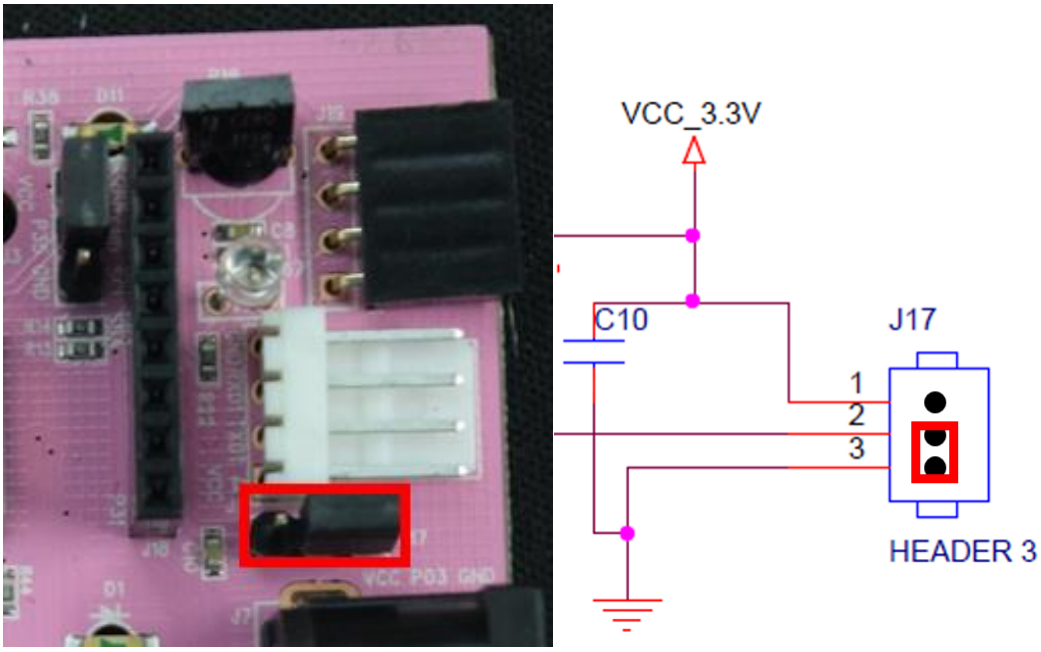


Figure 2-3 Jumper Connection Diagram

As shown above, the jumper connection of J17 is basically set to GND with 2-3.
To change the address value to 1101001, change the jumper connection of J17 to 1-2.

3 Source Code

```
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);
    }
}
```

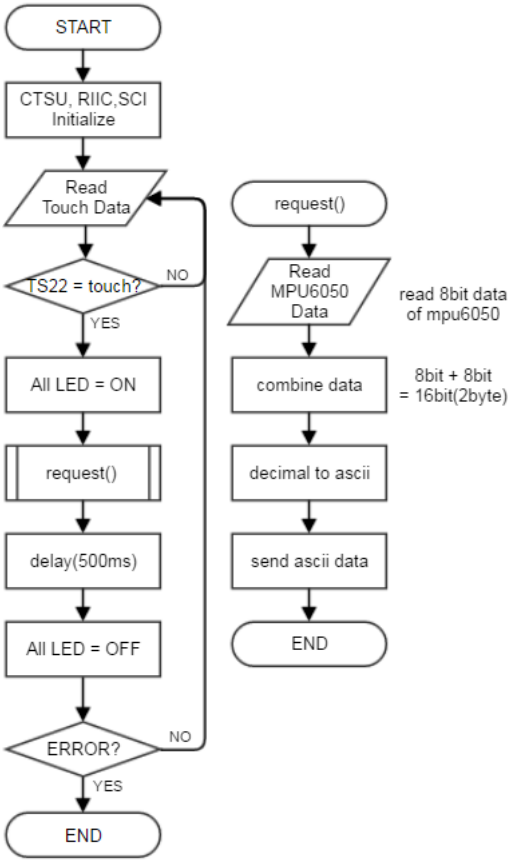


Figure 3-1 Source code and flowchart

4 Debugging

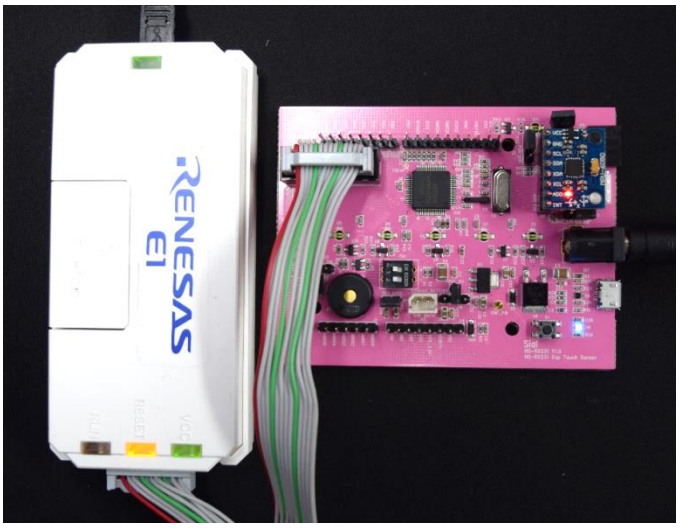


Figure 4-1 Connect NS-RX231 to power adapter and E1 debugger.

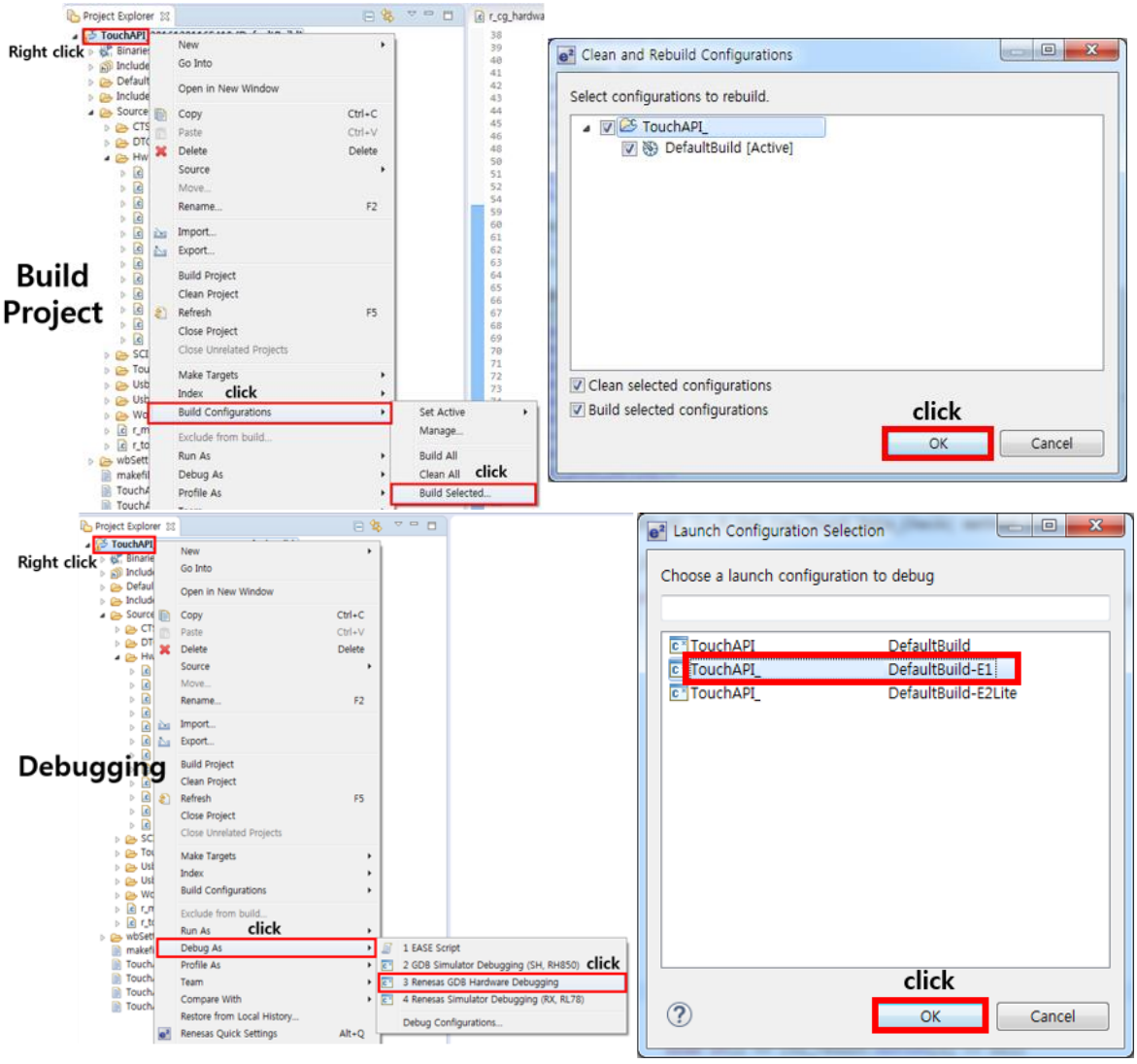
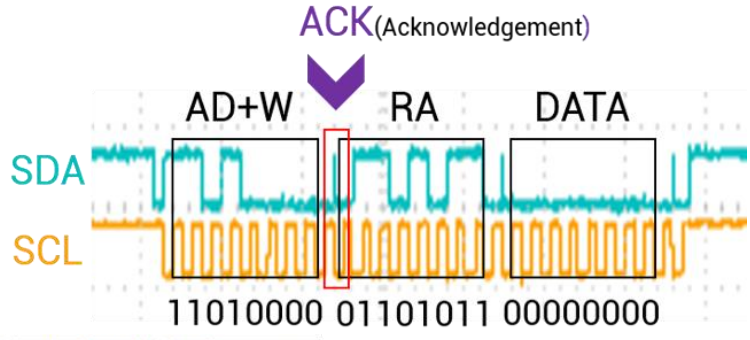


Figure 4-2 Building and debugging projects

5 Execution

Firstly, in order to receive data, the MPU6050 must be activated by setting the SLEEP bit at address 107 of MPU6050 to 0.



Single-Byte Write Sequence

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

Figure 5-1 Single-Byte Write Sequence

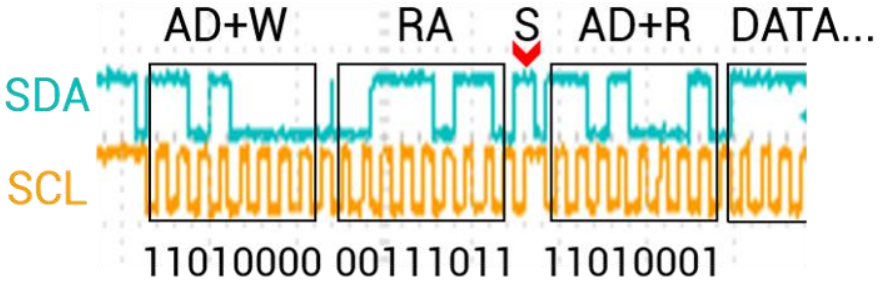
ACK A is an acknowledgment signal, meaning that the device has received the data correctly. The size of the signal may vary.

After activating the MPU6050, it starts to receive data.

3B	59	ACCEL_XOUT_H
3C	60	ACCEL_XOUT_L

Figure 5-2 Internal register address value of MPU6050 device

From the MPU6050, you can get 7 data of acceleration X, Y, Z / temperature / gyro X, Y, Z in total. Each data is 2 bytes (16 bits), but when receiving data, it is divided into _H, which is the upper 8 bits, and _L, which is the lower 8 bits.



Burst Read Sequence

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

Figure 5-3 Burst Data Read Sequence

This source code is to receive data of MPU6050 from NS-RX231 and then to distribute data by UART communication, so you can receive data by inserting TXD1, RXD1 pin of pin header or Bluetooth module.

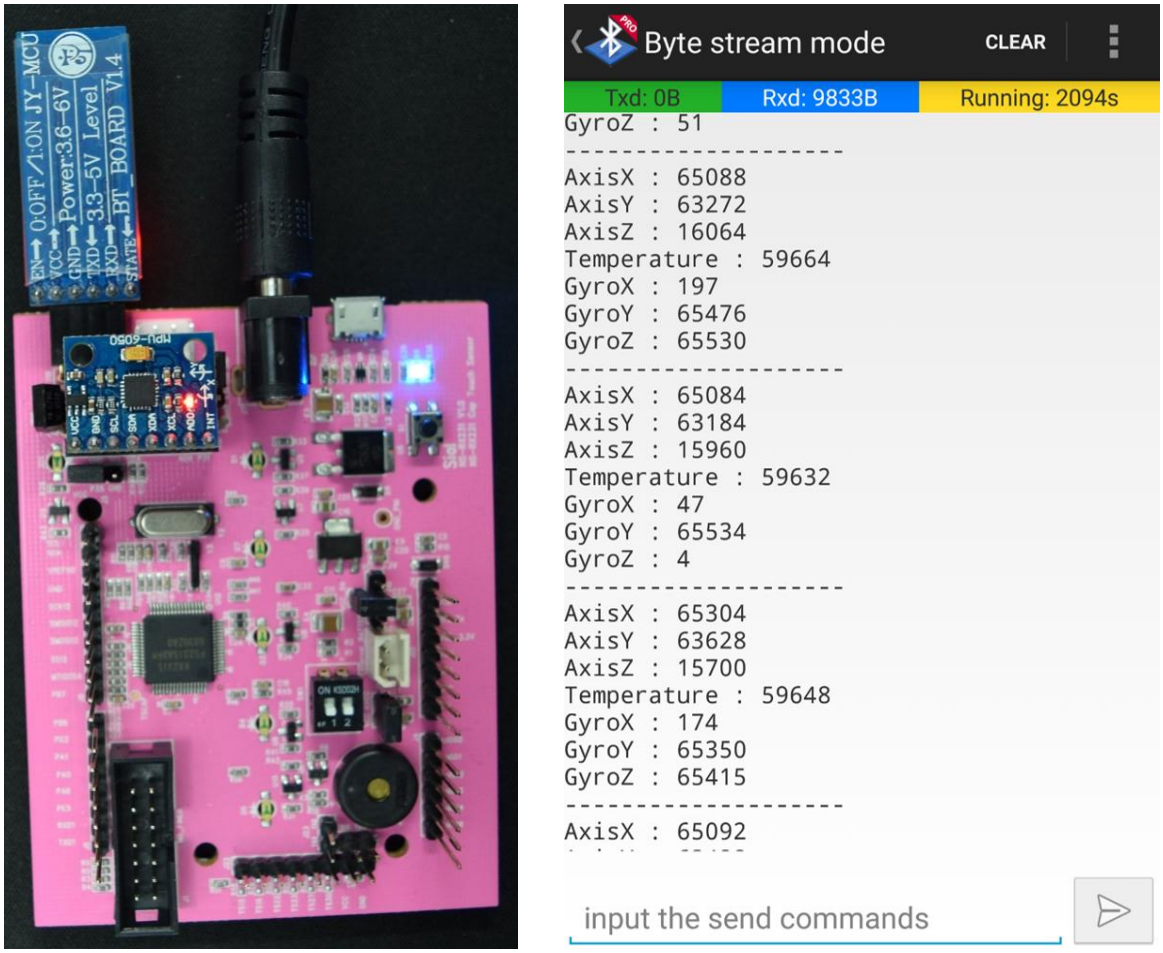


Figure 5-4 MPU6050 Operation

The Bluetooth module is plugged in and connected the cell phone to receive data of the MPU6050.

6 Schematic

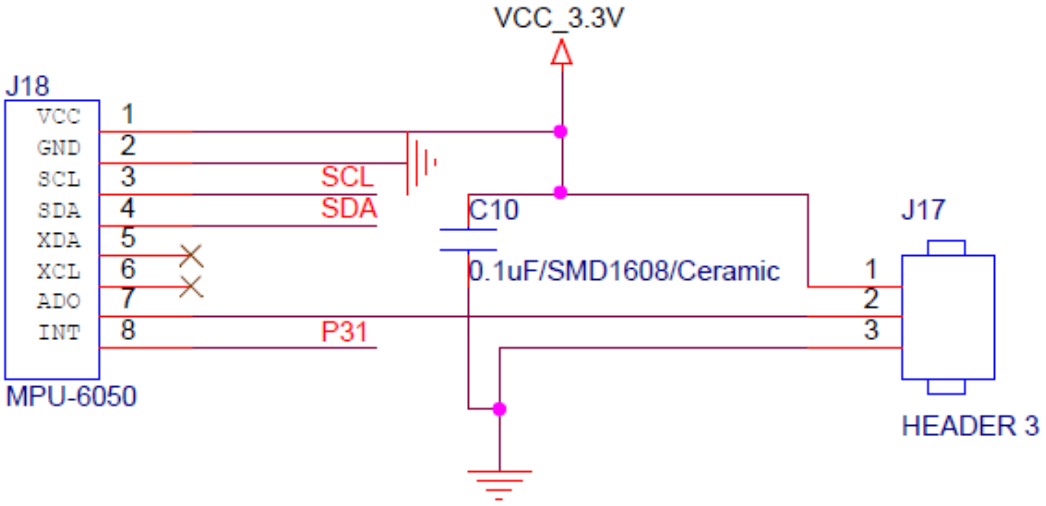


Figure 6-1 MPU-6050 terminal circuit of NS-RX231

7 Added

Burst Read Sequence

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

Figure 7-1 Burst Read Sequence

As shown in the above figure, the master reads the device address value (AD + W) and puts the address value (RA) of the device it wants to read after the Start Condition (S). And then there is Start Condition (S) again, which means Restart Condition.

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

Figure 7-2 A detailed description of the flowchart signal

The R / W bit consists of AD (7 bits) + R / W (1 bit) and is transmitted in 8 bits.

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;
```

Figure 7-3 AD+W & AD+R