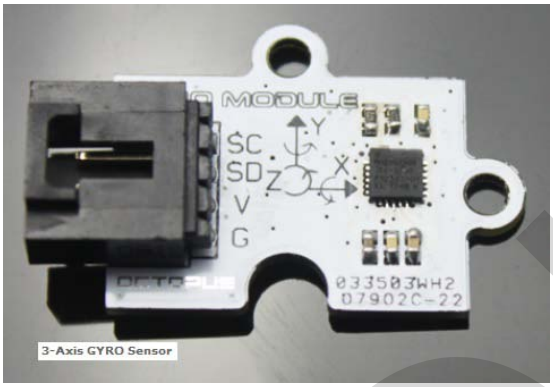
3-Axis GYRO Sensor



1. 사양

1.1. Description

3-Axis GYRO Sensor는 IC ITG3205 axis gyro에 기반한 센서모듈이다. ITG3205는 digital output MEMS gyroscope로 3개의 16-bit analog-digital converter(ADC)와 digital gyro-scope output을 가진다. 고속의 I2C serial Interface(400KHZ)를 사용하며, temperature sensor와 2% accurate internal oscillator를 내장한다.

1.2. Feature

the digital output X, Y, Z-axis angular value velocity sensor in a single circuit, with the least significant 16 bits (LSBs) ° / sec sensitivity and ± 2000 ° / sec full scale range.
the programmable digital low-pass filter
less than 6.5mA operating current, greatly extending battery life, standby current of only 5uA.
VDD power supply range 2.1V-3.6V
the digital output temperature sensor
high-speed I2C serial interface (400KHZ)
Power supply needs: 3.3V/5V
Fast response and High sensitivity
Simple drive circuit
Stable and long life

1.3. 구성

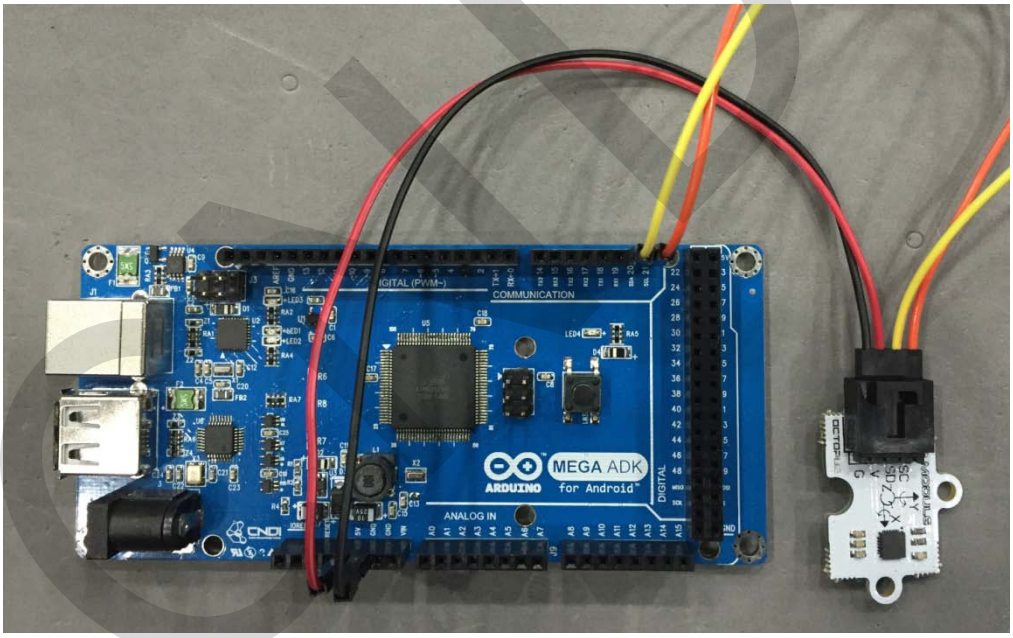
1 X 3-Axis GYRO Sensor

1 x Analog Sensor Cable

2. User Guide

2.1. 결선

모듈 간 결선 방법은 다음의 그림 및 표와 같다.



Module	MEGA ADK	3-Axis GYRO Sensor
Pin	SCL(21)	SC
Pin	SDA(20)	SD
Pin	3.3V	V
Pin	GND	G

모듈의 장치 및 회로에 대한 상세한 내용은 데이터시트 및 회로도를 참고한다.

2.2. 예제프로그램

GYRO 센서를 사용하여 3축 가속도를 측정한다. ITG3205 칩을 사용하므로 관련 데이터 시트를 참고하여 프로그래밍 한다. 먼저, 해당하는 칩의 id를 확인하여 통신 여부를 체크하고, 센싱 주기를 설정한다.

각 축에 해당하는 측정 값을 읽고, 시리얼 모니터를 통하여 표시한다.

- GyroSensor3Axis_Example.ino

```
// The Wire library is used for I2C communication
#include <Wire.h>

// This is a list of registers in the ITG-3200.
// Registers are parameters that determine how the sensor will behave,
// or they can hold data that represent the sensors current status.
// To learn more about the registers on the ITG-3200, read the datasheet.
char WHO_AM_I = 0x00;
char SMPLRT_DIV= 0x15;
char DLPF_FS = 0x16;
char GYRO_XOUT_H = 0x1D;
char GYRO_XOUT_L = 0x1E;
char GYRO_YOUT_H = 0x1F;
char GYRO_YOUT_L = 0x20;
char GYRO_ZOUT_H = 0x21;
char GYRO_ZOUT_L = 0x22;

//This is a list of settings that can be loaded into the registers.
//DLPF, Full Scale Register Bits
//FS_SEL must be set to 3 for proper operation
//Set DLPF_CFG to 3 for 1kHz Fint and 42 Hz Low Pass Filter
char DLPF_CFG_0 = 1<<0;
char DLPF_CFG_1 = 1<<1;
char DLPF_CFG_2 = 1<<2;
char DLPF_FS_SEL_0 = 1<<3;
```

```
char DLPF_FS_SEL_1 = 1<<4;

// I2C devices each have an address. The address is defined in the datasheet for the device.
// The ITG-3200 breakout board can have different address depending on how the jumper on top of the board is configured.
// By default, the jumper is connected to the VDD pin. When the jumper is connected to the VDD pin the I2C address is 0x69.
// If AD0 = 0, I2C address is 0x68.
char itgAddress = 0x68;

// In the setup section of the sketch the serial port will be configured,
// the i2c communication will be initialized, and the itg-3200 will be configured.
void setup()
{
  // Create a serial connection using a 9600bps baud rate.
  Serial.begin(9600);

  // Initialize the I2C communication. This will set the Arduino up as the 'Master' device.
  Wire.begin();

  // Read the WHO_AM_I register and print the result
  char id=0;
  id = itgRead(itgAddress, WHO_AM_I);      // id confirmation
  Serial.print("ID: ");
  Serial.println(id, HEX);

  // Configure the gyroscope
  // FS_SEL=3 : Set the gyroscope scale for the outputs to +/-2000 degrees per second
  // DLPF_CFG=1 : Low Pass Filter Bandwidth = 188Hz, Internal Sample Rate = 1kHz
  itgWrite(itgAddress, DLPF_FS, (DLPF_FS_SEL_0|DLPF_FS_SEL_1|DLPF_CFG_0));
  // F_sample = F_internal / ( divider + 1 )
```

```
  // sample rate = 1kHz / ( 9 + 1 ) = 100Hz
  // Set the sample rate to 100 hz, 10ms per sample
  itgWrite(itgAddress, SMPLRT_DIV, 9);
}

// The loop section of the sketch will read the X,Y and Z output rates
// from the gyroscope and output them in the Serial Terminal
void loop()
{
  // Create variables to hold the output rates.
  int xRate, yRate, zRate;

  // Read the x,y and z output rates from the gyroscope.
  xRate = readX();
  yRate = readY();
  zRate = readZ();

  // Print the output rates to the terminal, seperated by a TAB character.
  Serial.print(xRate);
  Serial.print('₩t');
  Serial.print(yRate);
  Serial.print('₩t');
  Serial.println(zRate);

  // Wait 10ms before reading the values again.
  // (Remember, the output rate was set to 100hz and 1reading per 10ms = 100hz.)
  delay(10);
}
// This function will write a value to a register on the itg-3200.
// Parameters:
//   char address: The I2C address of the sensor. For the ITG-3200 breakout the address
is 0x68.
//   char registerAddress: The address of the register on the sensor that should be
written to.
```

```
//   char data: The value to be written to the specified register.
void itgWrite(char address, char registerAddress, char data)
{
  // Initiate a communication sequence with the desired i2c device
  Wire.beginTransmission(address);
  // Tell the I2C address which register we are writing to
  Wire.write(registerAddress);
  // Send the value to write to the specified register
  Wire.write(data);
  // End the communication sequence
  Wire.endTransmission();
}

// This function will read the data from a specified register on the ITG-3200 and return the value.
// Parameters:
//   char address: The I2C address of the sensor. For the ITG-3200 breakout the address is 0x68.
//   char registerAddress: The address of the register on the sensor that should be read
// Return:
//   unsigned char: The value currently residing in the specified register
unsigned char itgRead(char address, char registerAddress)
{
  // This variable will hold the contents read from the i2c device.
  unsigned char data=0;

  // Send the register address to be read.
  Wire.beginTransmission(address);
  // Send the Register Address
  Wire.write(registerAddress);
  // End the communication sequence.
  Wire.endTransmission();

  // Ask the I2C device for data
```

```
  Wire.beginTransmission(address);
  Wire.requestFrom(address, 1);
  delay(100);
  // Wait for a response from the I2C device
  if(Wire.available()){
    // Save the data sent from the I2C device
    data = Wire.read();
  }

  // End the communication sequence.
  Wire.endTransmission();

  //Return the data read during the operation
  return data;
}

// This function is used to read the X-Axis rate of the gyroscope.
// The function returns the ADC value from the Gyroscope
// NOTE: This value is NOT in degrees per second.
// Usage: int xRate = readX();
int readX(void)
{
  int data=0;
  data = itgRead(itgAddress, GYRO_XOUT_H)<<8;
  data |= itgRead(itgAddress, GYRO_XOUT_L);

  return data;
}

// This function is used to read the Y-Axis rate of the gyroscope.
// The function returns the ADC value from the Gyroscope
// NOTE: This value is NOT in degrees per second.
// Usage: int yRate = readY();
int readY(void)
```

```
{
   int data=0;
   data = itgRead(itgAddress, GYRO_YOUT_H)<<8;
   data |= itgRead(itgAddress, GYRO_YOUT_L);

   return data;
}

// This function is used to read the Z-Axis rate of the gyroscope.
// The function returns the ADC value from the Gyroscope
// NOTE: This value is NOT in degrees per second.
// Usage: int zRate = readZ();
int readZ(void)
{
   int data=0;
   data = itgRead(itgAddress, GYRO_ZOUT_H)<<8;
   data |= itgRead(itgAddress, GYRO_ZOUT_L);

   return data;
}
```