


[Submit a Video](#)

- [Updates](#)
- [Forum](#)

[Search](#)

- [What is Arduino?](#)
- [Start Here](#)
 - [OSEPP Arduino Basic Companion Kit](#)
 - [OSEPP™ Bluetooth](#)
 - [OSEPP™ Fio](#)
 - [OSEPP™ Nano](#)
 - [OSEPP™ Mega 2560](#)
 - [OSEPP™ Mega 2560 R3 Plus](#)
 - [OSEPP™ Pro](#)
 - [OSEPP™ Pro Mini](#)
 - [OSEPP™ Uno](#)
 - [FTDI Breakout Board](#)
 - [OSEPP™ Uno R3 Plus](#)
 - [Accelerometer Sensor Module](#)
 - [Compass Sensor Module](#)
 - [Gyroscope Sensor Module](#)
 - [IR Proximity Sensor Module](#)
 - [IR Line Sensor Module](#)
 - [I2C Expansion Shield](#)
 - [101 Basic Starter Kit](#)
- [Example](#)

[Learning Centre](#) > [Start Here](#) > [Gyroscope Sensor Module](#)

share tweet email share  [print](#)

Gyroscope Sensor Module

Setup

1. Connect one end of the cable into either Molex connectors on the sensor
2. Connect the other end of the cable to the Arduino board:
 - RED: 5V
 - WHITE: I2C SDA (pin A4 on Uno; pin 20 on Mega)
 - BLACK: GND
 - GREY: I2C SCL (pin A5 on Uno; pin 21 on Mega)
3. Set the DIP switch on the sensor to set the sensor address (check back of sensor for possible addresses)

Example Sketch

```

// OSEPP Gyroscope Sensor Example Sketch
// by OSEPP <http://www.osepp.com>

// This sketch demonstrates interactions with the Gyroscope Sensor

#include <Wire.h>

// Possible sensor addresses (suffix correspond to DIP switch positions)
#define SENSOR_ADDR_OFF  (0x69)
#define SENSOR_ADDR_ON   (0x68)

// Set the sensor address here
const uint8_t sensorAddr = SENSOR_ADDR_OFF;

// One-time setup
void setup()
{
    // Start the serial port for output
    Serial.begin(9600);

    // Join the I2C bus as master
    Wire.begin();

    // Use the default configuration (see datasheet for more details)
}

// Main program loop
void loop()
{
    uint8_t x_msb;    // X-axis most significant byte
    uint8_t x_lsb;    // X-axis least significant byte
    uint8_t y_msb;    // Y-axis most significant byte
    uint8_t y_lsb;    // Y-axis least significant byte
    uint8_t z_msb;    // Z-axis most significant byte
    uint8_t z_lsb;    // Z-axis least significant byte

    uint16_t x;
    uint16_t y;
    uint16_t z;

    // Get the value from the sensor
    if ((ReadByte(sensorAddr, 0x1d, &x_msb) == 0) &&
        (ReadByte(sensorAddr, 0x1e, &x_lsb) == 0) &&
        (ReadByte(sensorAddr, 0x1f, &y_msb) == 0) &&
        (ReadByte(sensorAddr, 0x20, &y_lsb) == 0) &&
        (ReadByte(sensorAddr, 0x21, &z_msb) == 0) &&
        (ReadByte(sensorAddr, 0x22, &z_lsb) == 0))
    {
        x = (x_msb << 8) | x_lsb;
        y = (y_msb << 8) | y_lsb;
        z = (z_msb << 8) | z_lsb;

        // Perform 2's complement
        int16_t real_x = ~(x - 1);
        int16_t real_y = ~(y - 1);
        int16_t real_z = ~(z - 1);

        Serial.print("X: ");
        Serial.println(real_x);
    }
}

```

```

        Serial.print("Y: ");
        Serial.println(real_y);
        Serial.print("Z: ");
        Serial.println(real_z);
    }
    else
    {
        Serial.println("Failed to read from sensor");
    }

    // Run again in 1 s (1000 ms)
    delay(1000);
}

// Read a byte on the i2c interface
int ReadByte(uint8_t addr, uint8_t reg, uint8_t *data)
{
    // Do an i2c write to set the register that we want to read from
    Wire.beginTransmission(addr);
    Wire.write(reg);
    Wire.endTransmission();

    // Read a byte from the device
    Wire.requestFrom(addr, (uint8_t)1);
    if (Wire.available())
    {
        *data = Wire.read();
    }
    else
    {
        // Read nothing back
        return -1;
    }

    return 0;
}

// Write a byte on the i2c interface
void WriteByte(uint8_t addr, uint8_t reg, byte data)
{
    // Begin the write sequence
    Wire.beginTransmission(addr);

    // First byte is to set the register pointer
    Wire.write(reg);

    // Write the data byte
    Wire.write(data);

    // End the write sequence; bytes are actually transmitted now
    Wire.endTransmission();
}

```

• Newsletter

Subscribe to our newsletter to get the latest from Osepp!