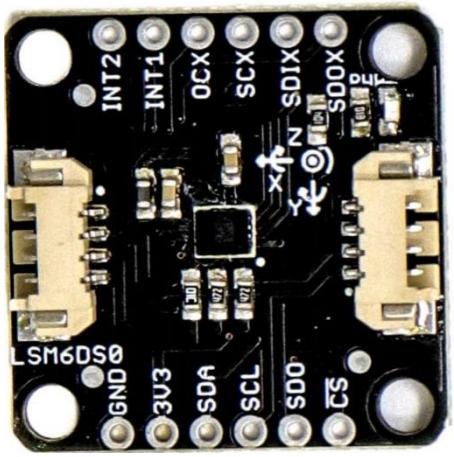
SmartElex 6 Degrees of Freedom Breakout - LSM6DSO

Note: This tutorial is for the LSM6DSO. It is important to note that last designation for the IC is the letter O as opposed to the number 0. There is also the LSM6DS0 that was released by STMicroelectronics but it is EOL.

The LSM6DSO is an accelerometer and gyroscope sensor with a giant 9 kbyte buffer and embedded processing interrupt functions, specifically targeted at the cellphone market. The sensor is super-flexible and can be configured specifically for an application. We've put together a driver and slew of examples to help you explore the possibilities.



Some of the things the LSM6DSO can do:

- Read accelerometer data up to 6.66 kilosamples per second, for super accurate movement sensing
- Read gyroscope data up to 6.66 kilosamples per second
- Operates at 0.55mA for up to 6.66 ksps modes
- Read temperature
- Buffer up to 9 kbytes of data between reads (built-in FIFO)
- Count steps (Pedometer)
- Detect shocks, tilt, motion, taps, double-taps
- Host other sensors into its FIFO

- Drive interrupt pins by embedded functions or by FIFO low-capacity/overflow warning.

Warning! The LSM6DSO is a 3.3V device! Supplying voltages greater than ~3.6V can permanently damage the IC. As long as your Arduino has a 3.3V supply output, and you're ok with using I^2C, you shouldn't need any extra level shifting. If you want to use SPI, you may need a level shifter.

Power and Logic Levels

We recommend powering the board through the connector when quickly prototyping. For a more secure connection, you can always solder to the PTH labeled **3V3** and **GND**. The recommended input voltage when using the board with a microcontroller is **3.3V** if you are using the connector. However, you can use a regulated supply voltage between 1.71V and 3.6V to power the sensor. The logic levels will match the input voltage (e.g. if the sensor is powered at 3.3V, the logic level will be 3.3V as well).

I^2C

The main method of reading the LSM6DSO is through the I^2C bus. The board includes two connectors for fast prototyping and removes the need for soldering. You can also solder to the PTHs labeled as **SDA** and **SCL** as an alternative. The default address for the IC is **0x6B**. However, you can adjust the jumper on the back of the board to change the address to **0x6A**.

SPI

If you decide to use a SPI bus, you will need to solder header pins or wires to the board.

- **SDA/SDI** - Device data in. Note that the SDA pin used for I^2C is also the SDI pin used for SPI. Flipping the board to the bottom side will show the label for the SPI pin.
- **SCL** - Serial clock for either I^2C or SPI.
- **SDO** - Device data out. By default, the SDO pin is connected to power to set the I^2C address. Make sure to cut the trace as explained below if you decide to use this sensor in SPI mode.
- **CS** - Chip select.

When using the board in SPI mode, you will need to cut the I^2C jumper for the default address (e.g. 0x6B) on the back and leave the jumper pads unconnected when using SPI.

Interrupt Pins

INT1 and INT2 are programmable interrupts for the accelerometer and gyroscope. They can be set to alert on over/under thresholds, data ready, or FIFO overruns. Make sure these are connected to an INPUT pin to prevent driving 5v back into the LSM6DSO.

There are a variety of interrupts on the LSM6DSO. While connecting these is not as critical as the communication or power supply pins, using them will help you get the most out of the chip.

The interrupt pins are **INT1** and **INT2**. One or both pins can be software configured and mapped to the following conditions:

- Step detected
- Step detected after delta time
- Step counter overflowed
- Significant motion (shock, drop)
- FIFO full
- FIFO overrun
- FIFO threshold reached (Datasheet calls this the "watermark")
- Boot status
- Gyroscope data ready
- Accelerometer data ready
- Inactivity
- Single tap
- Wake-up
- Free-fall
- Double tap
- 6D (orientation)
- Tilt
- Timer
- Ironing interrupt

Only a few interrupt examples are provided. See the datasheet and application guide for using the advanced interrupt features.

Auxiliary Pins

The auxiliary serial data output pins are used to attach slave I^2C and auxiliary SPI 3/4-wire devices for FIFO data collection. This function is not covered in this tutorial.

- OCS - aux chip select

- SCX - aux serial clock
- SDIX - aux serial data input
- SDOX - aux serial data output

Reference Axis

For easy reference, we've documented the 6DoF's vectors with 3D Cartesian coordinate axes on the top and bottom side of the board. Make sure to orient and mount the board correctly for your application. Remember, it's all relative.

LED

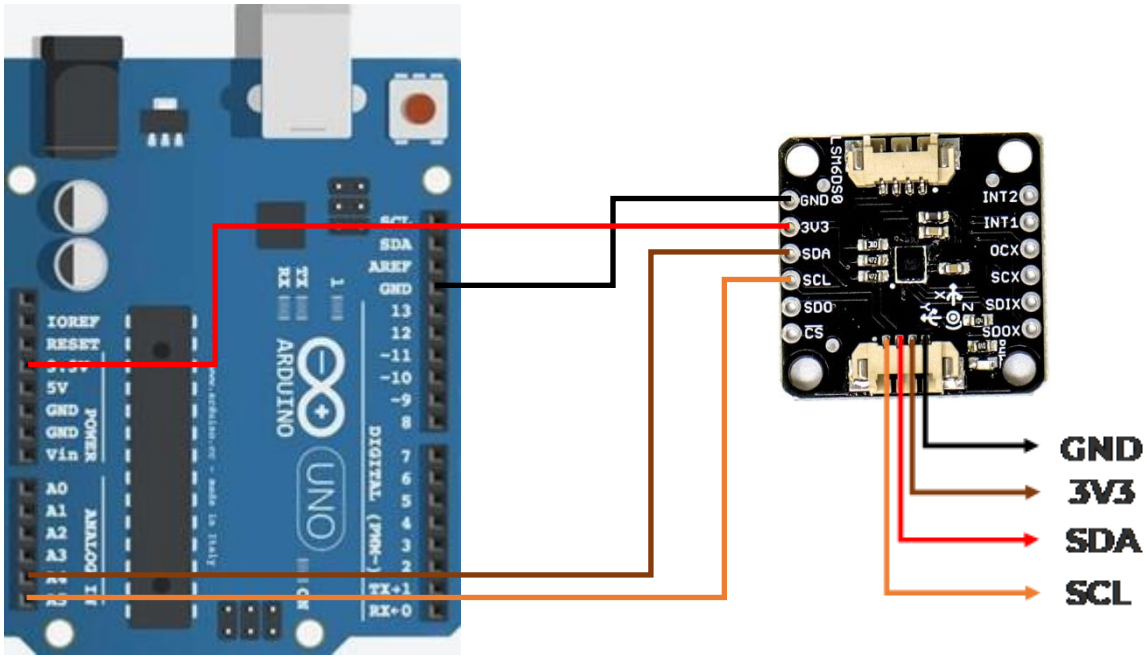
The board includes an LED indicator that lights up when there is power available.

Jumper Pins

There are five jumpers on the back of the board. **LED** - This is connected to the PWR LED on the top of the board. Cutting this disables the LED.

- **I2C** - The I2C jumper is connected to the 4.7kΩ pull-up resistors for the I^2C bus. Most of the time you can leave these alone unless your project requires you to disconnect the pull-up resistors. SPI works with these connected but really should be cut apart for better signal shape at high speeds and to lower power consumption.
- **0x6B/0x6A** - These jumpers are used to select the address 0x6B (default) or 0x6A for I^2C communication. This jumper must be **opened for SPI mode** or the SDO line will not supply data.
- **SCX** - By default, this pin is connected to GND since ST recommends pulling the unused SCX to power or ground when not in use. For most users, you can leave this jumper alone. If your project requires connecting slave devices to the auxiliary pin, cut this trace.
- **SDIX** - By default, this pin is connected to GND since ST recommends pulling the unused SDIX to power or ground when not in use. For most users, you can leave this jumper alone. If your project requires connecting slave devices to the auxiliary pin, cut this trace.

I²C Mode



Arduino	LSM6DSO
A5(SCL)	SCL
A4(SDA)	SDA
3.3V	3V3
GND	GND

SPI Mode

NOTE: Use logic converter between microcontroller and the LSM6DSO while using SPI.

Arduino	LSM6DSO
D13(SCK)	SCL
D12(MISO)	SDO
D11(MOSI)	SDA
D10(SS)	CS
5V(THROUGH LOGIC CONVERTER)	3V3
GND	GND

Note: When using the LSM6DSO, make sure to firmly attach the thing that is being measured to filter movement noise. To secure the board using its mounting holes, you will need screws and standoffs.

Installing the Arduino Library

SparkFun have written an Arduino library to help make interfacing with the LSM6DSO's gyro, accelerometer, and temperature sensor as easy-as-possible. Download using the Arduino library manager by searching for '**SparkFun Qwiic 6DoF LSM6DSO Arduino**

Note: The LSM6DSO library is based on the LSM6DS3's Arduino Library. While the libraries are similar, the LSM6DS3's library will not work with the LSM6DSO. Make sure to download the correct library for your IC!

Example - Basic Readings

There are a few examples in the library but we recommend using the Basic Readings in I^2C mode to get started.

Hook up the LSM6DSO to the I^2C bus, and click "**File** > **Examples** > **SparkFun Qwiic 6 DoF - LSM6DSO** > **Basic_Readings**". This example demonstrates the highest level of

usage. Besides setting up the Wire library and bus, you will you have to do is create a variable of the type "LSM6DSO", set it to .begin();, and initialize the BASIC_SETTINGS.
To read the accelerometer, gyro, or temperature sensor using the Arduino Serial Monitor.
We'll assume that you have selected the board (in this case the **Arduino Uno**), COM port at this point. If you have the code open, hit the upload button. Otherwise, copy and paste the following into the Arduino IDE.

```
#include "SparkFunLSM6DSO.h"
#include "Wire.h"
//#include "SPI.h"

LSM6DSO myIMU; //Default constructor is I2C, addr 0x6B

void setup() {


  Serial.begin(115200);
  delay(500);

  Wire.begin();
  delay(10);
  if( myIMU.begin() )
    Serial.println("Ready.");
  else {
    Serial.println("Could not connect to IMU.");
    Serial.println("Freezing");
  }

  if( myIMU.initialize(BASIC_SETTINGS) )
    Serial.println("Loaded Settings.");

}


void loop()
{
  //Get all parameters
  Serial.print("\nAccelerometer:\n");
  Serial.print(" X = ");
  Serial.println(myIMU.readFloatAccelX(), 3);
  Serial.print(" Y = ");
  Serial.println(myIMU.readFloatAccelY(), 3);
  Serial.print(" Z = ");
  Serial.println(myIMU.readFloatAccelZ(), 3);
```

```
  Serial.print("\nGyroscope:\n");
  Serial.print(" X = ");
  Serial.println(myIMU.readFloatGyroX(), 3);
  Serial.print(" Y = ");
  Serial.println(myIMU.readFloatGyroY(), 3);
  Serial.print(" Z = ");
  Serial.println(myIMU.readFloatGyroZ(), 3);

  Serial.print("\nThermometer:\n");
  Serial.print(" Degrees F = ");
  Serial.println(myIMU.readTempF(), 3);

  delay(1000);
}
//////////////////////////////////////////////////////////END//////////////////////////////////////////////////////////
```

After uploading, open the Serial Monitor and set it at **115200**.