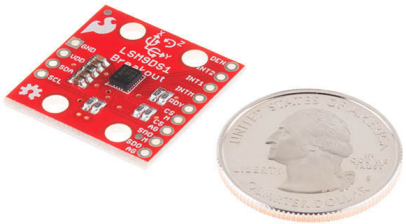
LSM9DS1 Breakout Hookup Guide

CONTRIBUTORS: *JIMB0*

♡ FAVORITE 2

Introduction

The LSM9DS1 is a versatile, motion-sensing system-in-a-chip. It houses a 3-axis accelerometer, 3-axis gyroscope, and 3-axis magnetometer – **nine degrees of freedom (9DOF)** in a single IC! Each sensor in the LSM9DS1 supports a wide range of…ranges: the accelerometer's scale can be set to ± 2, 4, 8, or 16 *g*, the gyroscope supports ± 245, 500, and 2000 °/s, and the magnetometer has full-scale ranges of ± 2, 4, 12, or 16 gauss. The IMU-in-a-chip is so cool we put it on a quarter-sized breakout board.



The LSM9DS1 is equipped with a digital interface, but even that is flexible: it supports both I^{2}C and SPI, so you'll be hard-pressed to find a microcontroller it doesn't work with.

Covered In This Tutorial

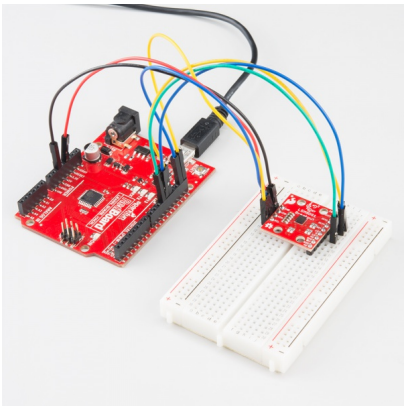
This tutorial is devoted to all things LSM9DS1. We'll introduce you to the chip itself, then the breakout board. Then we'll switch over to example code, and show you how to interface with the board using an Arduino and our LSM9DS1 Arduino library.

The tutorial is split into the following pages:

- LSM9DS1 Overview – An overview of the LSM9DS1, examining its features and capabilities.
- Breakout Board Overview – This page examines the LSM9DS1 Breakout Board – topics like the pinout, jumpers, and schematic are covered.
- Hardware Assembly – Assembly tips and tricks, plus some information about the breakout's dimensions.
- Hardware Hookup – Example I^{2}C and SPI wiring diagrams.
- Instlaling the Arduino Library – How to install the **Arduino library**, and use a simple example sketch to verify that your hookup works.
- Using the Arduino Library – An overview of the SFE_LSM9DS1 Arduino library's functions and variables.

Required Materials

This tutorial explains how to use the LSM9DS1 Breakout Board with an Arduino. To follow along, you'll need the following materials:



- LSM9DS1 Breakout Board
- Arduino UNO, RedBoard, or another Arduino-compatible board
- Straight Male Headers – Or wire. Something to connect between the breakout and a breadboard.
- Breadboard – Any size (even mini) should do.
- M/M Jumper Wires – To connect between Arduino and breadboard.

The LSM9DS1 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^{2}C, you shouldn't need any extra level shifting. But if you want to use SPI, you may need a level shifter.

A logic level shifter is required for any 5V-operating Arduino (UNO, RedBoard, Leonardo, etc). If you use a 3.3V-based 'duino – like the Arduino Pro 3.3V or 3.3V Pro Mini – there is no need for level shifting.

Suggested Reading

If you're not familiar with some of the concepts below, we recommend checking out that tutorial before continuing on.

- Accelerometer Basics
- Gyroscopes
- Serial Peripheral Interface (SPI)
- Inter-IC Communication (I^2C)
- Logic Levels
- Bi-Directional Level Shifter Hookup Guide

LSM9DS1 Overview

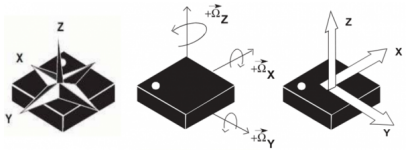
The LSM9DS1 is one of only a handful of IC's that can measure three key properties of movement – angular velocity, acceleration, and heading – in a single IC.

The gyroscope can measure **angular velocity** – that is "how fast, and along which axis, am I rotating?" Angular velocities are measured in **degrees per second** – usually abbreviated to DPS or °/s. The LSM9DS1 can measure up to ± 2000 DPS, though that scale can also be set to either 245 or 500 DPS to get a finer resolution.

An accelerometer measures **acceleration**, which indicates how fast velocity is changing – "how fast am I speeding up or slowing down?" Acceleration is usually either measured in m/s^2 (meters per second per second) or *g*'s (gravities [about 9.8 m/s^2]). If an object is sitting motionless it feels about 1 *g* of acceleration towards the ground (assuming that ground is on earth, and the object is near sea-level). The LSM9DS1 measures its acceleration in *g*'s, and its scale can be set to either ± 2, 4, 8, or 16 *g*.

Finally, there's the magnetometer, which measures the power and direction of **magnetic fields**. Though they're not easily visible, magnetic fields exist all around us – whether you're holding a tiny ferromagnet or feeling an attraction to Earth's magnetic field. The LSM9DS1 measures magnetic fields in units of **gauss** (Gs), and can set its measurement scale to either ± 4, 8, 12, or 16 Gs.

By measuring these three properties, you can gain a great deal of knowledge about an object's movement and orientation. 9DOF's have tons and tons of applications. Measuring the force and direction of Earth's magnetic field with a magnetometer, you can approximate your **heading**. An accelerometer in your phone can measure the direction of the force of gravity, and estimate **orientation** (portrait, landscape, flat, etc.). Quadcopters with built-in gyroscopes can look out for sudden rolls or pitches, and correct their momentum before things get out of hand.



Axis orientations of the LSM9DS1 IC. Note the IC's polarity-marking dot (for some reason they rotated the magnetometer in the datasheet).

The LSM9DS1 measures each of these movement properties in three dimensions. That means it produces **nine pieces of data**: acceleration in x/y/z, angular rotation in x/y/z, and magnetic force in x/y/z. The LSM9DS1 Breakout has labels indicating the accelerometer and gyroscope axis orientations, which share a right-hand rule relationship with each other. Note that, according to the datasheet, the x and y axes of the magnetic sensor are flipped (the image above is copied from the datasheet).

The LSM9DS1 is, in a sense, two IC's smashed into one package – like if you combined an LSM6DS3 accel/gryo with an LSM303DLMTR accel/mag. One half of the device takes care of all-things gyroscope and accelerometer, and the other half manages solely the magnetometer. In fact, a few of the control pins are dedicated to a single sensor – there are **two chip select pins** (CS_AG for the accel/gyro and CS_M for the mag) and **two serial data out pins** (SDO_AG and SDO_M).

Choose Your Own Adventure: SPI or I^2C

In addition to being able to measure a wide variety of movement vectors, the LSM9DS1 is also multi-featured on the communication interface end. It supports both SPI and I^2C, so you should have no difficulty finding a microcontroller that can talk to it.

SPI is generally the easier of the two to implement, but it also requires more wires – four versus I^2C's two.

Because the LSM9DS1 supports both methods of communication, **some pins have to pull double-duty**. The *Serial Data Out* (SDO) pin for example, does just that for SPI mode, but if you're using the device over I^2C it becomes an address selector. The *chip select* (CS_M and CS_AG) pins activate SPI mode when low, but if they're pulled high the device assumes I^2C communication. In the section below we discuss each of the LSM9DS1's pins, watch out for those pins that support both interfaces.

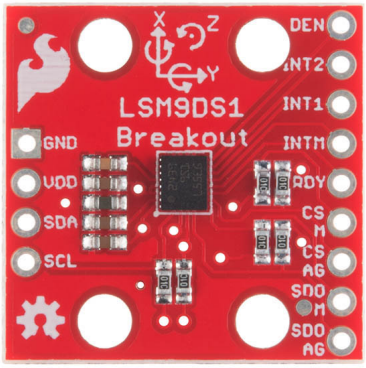
For much more detailed information about the IC, we encourage you to check out the datasheet!

Breakout Board Overview

Now that you know everything you need to about the LSM9DS1 IC, let's talk a bit about the breakout board it's resting on. On this page we'll discuss the pins that are broken out, and some of the other features on the board.

The Pinout

In total, the LSM9DS1 Breakout breaks out 13 pins.



The bare-minimum connections required are broken out on the left side of the board. These are the power and I²C pins (the communication interface the board defaults to):

Pin Label	Pin Function	Notes
GND	Ground	0V voltage supply
VDD	Power Supply	Supply voltage to the chip. Should be regulated between **2.4V and 3.6V.**
SDA	SPI: MOSI I²C: Serial Data	SPI: Device data in (MOSI) I²C: Serial data (bi-directional)
SCL	Serial Clock	I²C and SPI serial clock.

The remaining pins are broken out on the other side. These pins break out SPI functionality and interrupt outputs:

Pin Label	Pin Function	Notes
DEN	Gyroscope Data Enable	Mostly unknown. The LSM9DS1 datasheet doesn't have much to say about this pin.
INT2	Accel/Gyro Interrupt 2	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.
INT1	Accel/Gyro Interrupt 1	
INTM	Magnetometer Interrupt	A programmable interrupt for the magnetometer. Can be set to alert on over-under thresholds.
RDY	Magnetometer Data Ready	An interrupt indicating new magnetometer data is available. Non-programmable.
CS M	Magnetometer Chip Select	This pin selects between I²C and SPI on the magnetometer. Keep it HIGH for I²C, or use it as an (active-low) chip select for SPI. **HIGH (1)**: SPI idle mode / **I²C enabled** **LOW (0)**: **SPI enabled** / I²C disabled.
CS AG	Accel/Gyro Chip Select	This pin selects between I²C and SPI on the accel/gyro. Keep it HIGH for I²C, or use it as an (active-low) chip select for SPI. **HIGH (1)**: SPI idle mode / **I²C enabled** **LOW (0)**: **SPI enabled** / I²C disabled.
SDO M	SPI: Magnetometer MISO I²C: Magnetometer Address Select	In SPI mode, this is the magnetometer data output (SDO_M). In I²C mode, this selects the LSb of the I²C address (SA0_M)
SDO AG	SPI: Accel/Gyro MISO I²C: Accel/Gryo Address Select	In SPI mode, this is the accel/gryo data output (SDO_AG). In I²C mode, this selects the LSb of the I²C address (SA0_AG)

Power Supply

The VDD and GND pins are where you'll supply a voltage and 0V reference to the IC. The breakout board does not regulate this voltage, so make sure it falls within the allowed supply voltage range of the LSM9DS1: **2.4V to 3.6V**. Below is the electrical characteristics table from the datasheet.

Table 4. Electrical characteristics

Symbol	Parameter	Test conditions	Min.	Typ.[1]	Max.	Unit
Vdd	Supply voltage		1.9		3.6	V
Vdd_IO	Module power supply for I/O		1.71		Vdd+0.1	
Idd_XM	Current consumption of the accelerometer and magnetic sensor in normal mode [2]			600		µA
Idd_G	Gyroscope current consumption in normal mode[3]			4.0		mA

The communication pins are not 5V tolerant, so they'll need to be regulated to within a few mV of VDD.

Another very cool thing about this sensor is how **low-power** it is. In normal operation – with every sensor turned on – it'll pull around **4.5mA**.

Communication

CS_AG, CS_M, SDO_AG, SDO_M, SCL, and SDA are all used for the I²C and SPI interfaces. The function of these pins depends upon which of the two interfaces you're using.

If you're using using **I²C** here's how you might configure these pins:

- Pull CS_AG and CS_M HIGH. This will set both the accel/gyro and magnetometer to I²C mode.
- Set SDO_AG and SDO_M either HIGH or LOW. These pins set the I²C address of the gyro and accel/mag sensors.
- Connect SCL to your microcontroller's SCL pin.
- Connect SDA to your microcontroller's SDA pin.
- The board has a built-in 10kΩ pull-up resistor on both SDA and SCL lines. If that value is too high, you can add a second resistor in parallel to divide the pull-up resistance down (another 10kΩ in parallel, for example, would create an equivalent 5kΩ resistance).

Or, if you're using **SPI**:

- Connect CS_AG and CS_M to two individually controllable pins on your microcontroller. These chip-selects are active-low – when the pin goes LOW, SPI communication with either the accel/gyro (CS_AG) or magnetometer (CS_M) is enabled.
- SDO_AG and SDO_M are the serial data out pins. In many cases you'll want to connect them together, and wire them to your microcontroller's **MISO** (master-in, slave-out) pin.
- Connect SCL to your microcontroller's **SCLK** (serial clock) pin.

- Connect SDA to your microcontroller's **MOSI** (master-out, slave-in) pin.

Interrupts

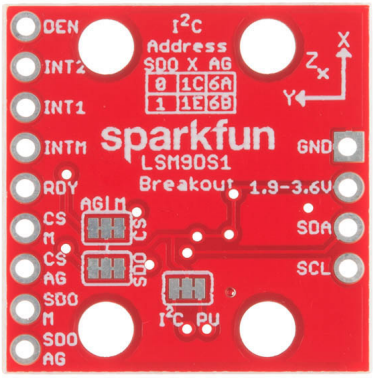
There are a variety of interrupts on the LSM9DS1. While connecting up to these is not as critical as the communication or power supply pins, using them will help you get the most out of the chip.

The accelerometer- and gyroscope-specific interrupts are **INT1** and **INT2**. These can both be programmed to interrupt as either active-high or active-low, triggering on events like data ready or when an acceleration or angular rotation exceeds a set threshold.

DRDY and **INTM** are devoted magnetometer interrupts. DRDY will go low when new magnetometer readings are available. INTM is a little more customizable – it can be used to trigger whenever magnetic field readings exceed a threshold on a set axis.

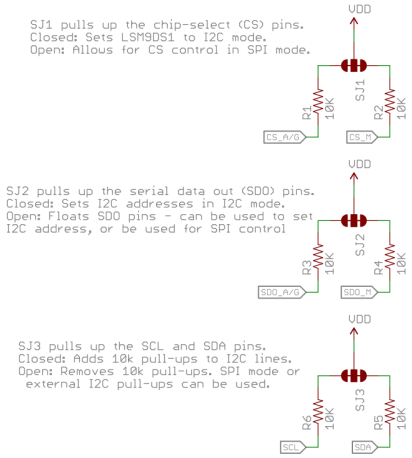
The Jumpers

Flipping the LSM9DS1 breakout over reveals a trio of two-way, surface mount jumpers. Each of these jumpers comes **closed**. Their purpose is to **automatically put the LSM9DS1 into I^{2}C mode**.



The three two-way jumpers on the back of the board. Follow the labels to see which pin they pull up.

Each of these jumpers pulls a pair of pins up to VDD, through a 10kΩ resistor. The middle pad of the jumper connects to the resistor, and the edge pads connect to a pin (follow the labels to find out which one). You can see how those jumpers match up on the schematic:



The top jumper connects CS_AG and CS_M to a pull-up – this'll set the LSM9DS1 to I^{2}C mode. The middle jumper pulls up SDO_AG and SDO_M, which sets the I^{2}C address of the chip. Finally, the far-left jumper adds pull-up resistors to the I^{2}C communication pins – SDA and SCL.

The intention of these jumpers is to make it as easy-as-possible to use the board; using as few wires as possible. If you're using the breakout with I^{2}C, you can ignore the four SDO and CS pins.

To disable any of these jumpers, whip out your handy hobby knife, and carefully cut the small traces between middle pad and edge pads. Even if you're using SPI, though, the jumpers shouldn't hinder your ability to communicate with the chip.

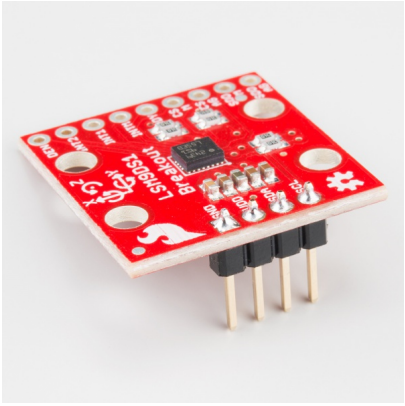
Hardware Assembly

On this page we'll discuss assembly hints. There's really not much to assembling the breakout board – the real key is soldering *something* into the breakout holes.

Solder Something

To get a solid electrical and physical connection to the LSM9DS1 Breakout, you'll need to solder either connectors or wires to the break-out pins. What, exactly, you solder into the board depends on how you're going to use it.

If you're going to use the breakout board in a breadboard or similar 0.1"-spaced perfboard, we recommend soldering straight male headers into the pins (there are also long headers if you need 'em).

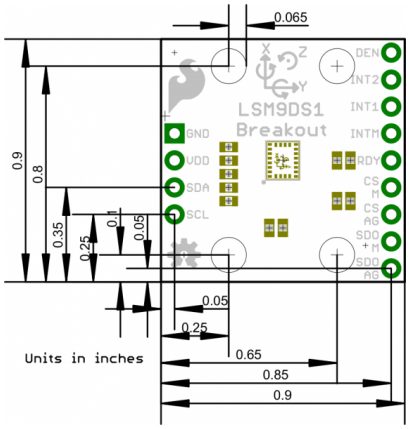


If you're only going to use the I²C interface – and ignore the interrupts – you can get away with soldering just the four-pin header.

If you're going to mount the breakout into a tight place, you may want to opt for soldering wires (stranded or solid-core) into the pins.

Mounting the Breakout

Because the LSM9DS1 senses motion, it's important (for most applications, at least) to keep it pinned in place. So the boards have four mounting holes toward the corners. The drill holes are 0.13" in diameter, so they should accommodate any 4/40 screw.



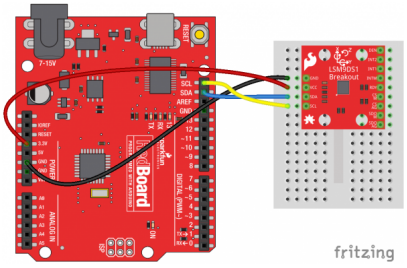
Consult the EAGLE PCB design files to find out more about the Breakout's dimensions.

Hardware Hookup

The LSM9DS1 will work over either I²C or SPI. Here are some example wiring diagrams, demonstrating how to wire up each interface.

I²C Hardware Hookup

Out-of-the-box, the board is configured for an I²C interface, as such we recommend using this hookup if you're otherwise agnostic. All you need is four wires!



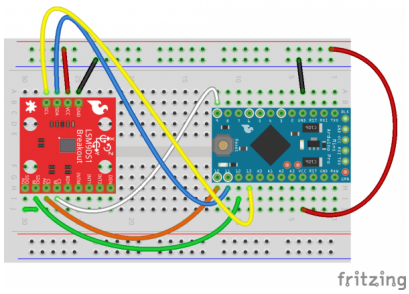
Connecting the LSM9DS1 to a RedBoard via I²C.

This hookup relies on all of the **jumpers** on the back of the board being set (as they should be, unless they've been sliced). If the jumpers have been cut, connect all four CS and SDO pins to 3.3V.

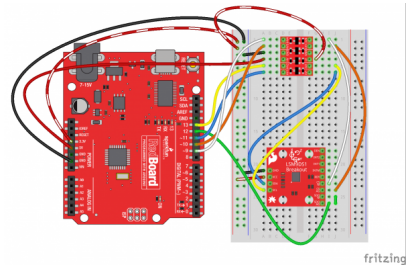
No level shifters even though the Arduino's I/O pins are 5V? Well, I²C is a funny interface: pins aren't directly pulled high by a GPIO, instead an open-drain MOSFET relies on pull-up resistors to create a logic high. Because the breakout board pull-up resistors are stronger (less resistance) than the Arduino's internal pull-ups, the voltage on the logic pins will be much closer to 3.3V (though a little higher) than 5V – within a tolerable range. Just make sure you power the breakout off of the Arduino's 3.3V power rail!

SPI Hardware Hookup

For the SPI hookup, we'll use the Arduino's hardware SPI pins – 11 (MOSI), 12 (MISO), and 13 (SCLK) – in addition to two digital pins of your choice (for the chip selects). If you're using a 3.3V Arduino, like the 3.3V/8MHz Pro Mini, you can hook the microcontroller pins directly up to the LSM9DS1.



If you're using a 5V Arduino, you'll also need to grab a level shifter to keep the SPI signals within the tolerable voltage range. You could use the SparkFun Bi-Directional Logic Level Converter for example:



(Unless you enjoy wire-tangles, the I^2C or 3.3V SPI hookups are recommended.)

Installing the Arduino Library

We've written a full-featured Arduino library to help make interfacing with the LSM9DS1's gyro, accelerometer, and magnetometer as easy-as-possible. Visit the GitHub repository to download the most recent version of the library, or click the link below:

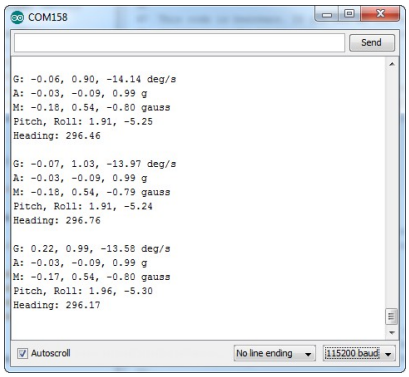
DOWNLOAD THE SPARKFUN LSM9DS1 ARDUINO LIBRARY

For help installing the library, check out our How To Install An Arduino Library tutorial. You'll need to move the *SparkFun_LSM9DS1_Arduino_Library* folder into a *libraries* folder within your Arduino sketchbook.

The LSM9DS1_Basic_I2C Example

To verify that your hookup works, load up the LSM9DS1_Basic_I2C example by going to **File** > **Examples** > **LSM9DS1 Breakout** > **LSM9DS1_Basic_I2C**. (If you're using the SPI hookup, load the LSM9DS1_Basic_SPI example instead.)

The default values set by this sketch should work for a fresh, out-of-the-box LSM9DS1 Breakout – it assumes all of the jumpers on the backside are closed. Upload the sketch, then open up your serial monitor, setting the baud rate to **115200**. You should see something like this:



The current reading from each axis on each sensor is printed out, then those values are used to estimate the sensor's orientation. Pitch is the angle rotated around the y-axis, roll is the board's rotation around the x-axis, and heading (i.e. yaw) is the sensor's rotation around the z-axis. Try rotating the board (without pulling out any wires!) to see how the values change.

Using the Arduino Library

To help demonstrate the library's functionality, a handful of examples are included which range from basic to more advanced. To begin to get a feel for the library's API, try loading up some of the other examples. The comments at the top of the sketch will instruct you on any extra hookup that may be required to use interrupts or other features.

On this page we'll quickly run down some of the more basic, fundamental concepts implemented by the library.

Setup Stuff

To enable the library, you'll need to **include** it, and you also need to include the SPI and Wire libraries:

```
#include <SPI.h> // SPI library included for SparkFunLSM9DS1
#include <Wire.h> // I2C library included for SparkFunLSM9DS1
#include <SparkFunLSM9DS1.h> // SparkFun LSM9DS1 library
```

Make sure the SPI and Wire includes are above the "SparkFunLSM9DS1" (even though you'll only be using one of the two interfaces).

Constructor

The constructor creates an instance of the LSM9DS1 class. Once you've created the instance, that's what you'll use to control the breakout from there on. This single line of code is usually placed in the **global** area of your sketch.

The constructor should be left without any parameters:

```
// Use the LSM9DS1 class to create an object. [imu] can be
// named anything, we'll refer to that throught the sketch.
LSM9DS1 imu;
```

Setting Up the Interface

The LSM9DS1 has tons of settings to be configured. Some are minute, others are more critical. The three most critical settings we'll need to configure are the **communication interface** and the device addresses. To configure these values, we'll make calls to the IMU's `settings` struct.

Here's an example that sets the IMU up for I^2C mode, with the default (high) I^2C addresses:

```
// SDO_XM and SDO_G are both pulled high, so our addresses are:
#define LSM9DS1_M   0x1E // Would be 0x1C if SDO_M is LOW
#define LSM9DS1_AG  0x6B // Would be 0x6A if SDO_AG is LOW
...
imu.settings.device.commInterface = IMU_MODE_I2C; // Set mode to I2C
imu.settings.device.mAddress = LSM9DS1_M; // Set mag address to 0x1E
imu.settings.device.agAddress = LSM9DS1_AG; // Set ag address to 0x6B
```

Alternatively, if you're using SPI mode, the `imu.settings.device.mAddress` and `imu.settings.device.agAddress` values become the **chip select pins**. For example, if you're using SPI mode with CS_AG connected to D10 and CS_M connected to D9, your setting configuration would look like this:

```
imu.settings.device.commInterface = IMU_MODE_SPI; // Set mode to SPI
imu.settings.device.mAddress = 9; // Mag CS pin connected to D9
imu.settings.device.agAddress = 10; // AG CS pin connected to D10
```

Configuring any value from the `imu.settings.device` can't take place in the global are of a sketch. If you get a compilation error, like `'imu' does not name a type`, you may have those in the wrong place – put them in `setup()`.

begin()-ing and Setting Sensor Ranges

Once you've created the LSM9DS1 object, and configured its interface, call the `begin()` member function to initialize the IMU.

The `begin()` function will take the settings you've adjusted in the previous step, and attempt to communicate with and initialize the sensors. You can check the return value of `begin()` to verify whether or not the set up was successful – it will return `0` if something goes wrong.

```
if (!imu.begin())
{
    Serial.println("Failed to communicate with LSM9DS1.");
    Serial.println("Looping to infinity.");
    while (1)
      ;
}
```

Once `begin()` has returned a success, you can start reading some sensor values!

Reading and Interpreting the Sensors

What good is the sensor if you can't get any data from it!? Here are the functions you'll need to get acceleration, rotation speed, and magnetic field strength data from the library.

readAccel(), readGyro(), and readMag()

These three functions – `readAccel()`, `readGyro()`, and `readMag()` – poll the LSM9DS1 to get the most up-to-date readings from each of the three sensors.

The read functions don't take any parameters, and they don't return anything, so how do you get that data? After the function runs its course, it'll update a set of **three class variables**, which will have the sensor data you desire. `readAccel()` will update `ax`, `ay`, and `az`, `readGyro()` will update `gx`, `gy`, and `gz`, and `readMag()` will update `mx`, `my`, and `mz`. Here's an example:

```
imu.readAccel(); // Update the accelerometer data
Serial.print(imu.ax); // Print x-axis data
Serial.print(", ");
Serial.print(imu.ay); // print y-axis data
Serial.print(", ");
Serial.println(imu.az); // print z-axis data
```

An example of reading and printing all three axes of accelerometer data.

Those values are all **signed 16-bit integers**, meaning they'll range from -32,768 to 32,767. That value doesn't mean much unless you know the scale of your sensor, which is where the next functions come into play.

calcAccel(), calcGyro(), and calcMag()

The library keeps track of each sensor's scale, and it implements these helper functions to make translating between the raw ADC readings of the sensor to actual units easy.

`calcAccel()`, `calcGyro()`, and `calcMag()` all take a single parameter – a signed 16-bit integer – and convert to their respective units. They all **return a float value**, which you can do with as you please.

Here's an example of printing calculated gyroscope values:

```
imu.readGyro(); // Update gyroscope data
Serial.print(imu.calcGyro(imu.gx)); // Print x-axis rotation in DPS
Serial.print(", ");
Serial.print(imu.calcGyro(imu.gy)); // Print y-axis rotation in DPS
Serial.print(", ");
Serial.println(imu.calcGyro(imu.gz)); // Print z-axis rotation in DPS
```

Setting Sensor Ranges and Sample Rates

Some of the more commonly altered attributes in the IMU are the sensor ranges and output data rates. These values can be configured, once again, by setting a value in the `settings` struct.

For example, to set the IMU's accelerometer range to ±16g, gyroscope to ±2000 °/s, and magnetometer to ±8 Gs, do something like this:

```
imu.settings.accel.scale = 16; // Set accel range to +/-16g
imu.settings.gyro.scale = 2000; // Set gyro range to +/-2000dps
imu.settings.mag.scale = 8; // Set mag range to +/-8Gs
imu.begin(); // Call begin to update the sensor's new settings
```

The output data rates are a bit more abstract. These values can range from 1-6, where 1 is the slowest update rate and 6 is the fastest.

Resources and Going Further

Hopefully that info dump was enough to get you rolling with the LSM9DS1. If you need any more information, here are some more resources:

- LSM9DS1 Product GitHub Repository – Your revision-controlled source for all things LSM9DS1. Here you'll find our most up-to-date hardware layouts and code.
- LSM9DS1 Datasheet – This datasheet covers everything from the hardware and pinout of the IC, to the register mapping of the gyroscope and accelerometer/magnetometer.
- LSM9DS1 Breakout Schematic
- LSM9DS1 Breakout EAGLE Files

Going Further

Now that you've got the LSM9DS1 up-and-running, what project are you going to incorporate motion-sensing into? Need a little inspiration? Check out some of these tutorials!

- Dungeons and Dragons Dice Gauntlet – This project uses an accelerometer to sense a "rolling the dice" motion. You could swap in the LSM9DS1 to add more functionality – like compass-based damage multipliers!
- Are You Okay? Widget – Use an Electric Imp and accelerometer to create an "Are You OK" widget. A cozy piece of technology your friend or loved one can nudge to let you know they're OK from half-a-world away.
- Leap Motion Teardown – An IMU sensor is cool, but image-based motion sensing is the future. Check out this teardown of the miniature-Kinect-like Leap Motion!
- Pushing Data to Data.SparkFun.com – Need an online place to store your IMU data? Check out data.sparkfun.com! This tutorial demonstrates how to use a handful of Arduino shields to post your data online.

Leap Motion Teardown
Let's see what's inside the amazing new Leap Motion input device!

Dungeons and Dragons Dice Gauntlet
A playful, geeky tutorial for a leather bracer that uses a LilyPad Arduino, LilyPad accelerometer, and seven segment display to roll virtual 4, 6, 8, 10, 12, 20, and 100 side dice for gaming.

Are You Okay? Widget
Use an Electric Imp and accelerometer to create an "Are You OK" widget. A cozy piece of technology your friend or loved one can nudge to let you know they're OK from half-a-world away.

Pushing Data to Data.SparkFun.com
A grab bag of examples to show off the variety of routes your data can take on its way to a Data.SparkFun.com stream.