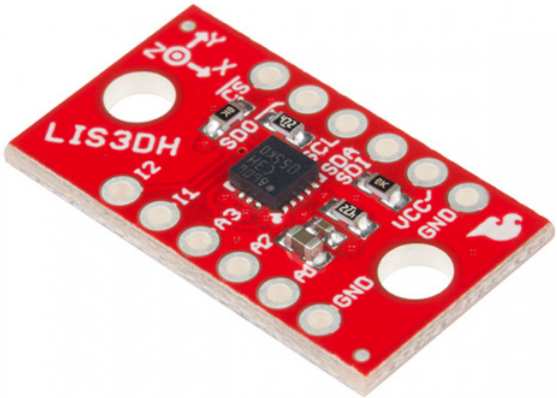
LIS3DH连接指南

贡献者： *MTAYLOR*

♡ 喜爱 1个

介绍



英国威廉希尔SparkFun三轴加速度计突破-LIS3DH

SEN-13963

$ 5.50

★★★½☆ 2

该LIS3DH是一个三轴加速度计可以使用翻译检测添加到项目中。LIS3DH中的“ 3D”是指它是3DoF或3自由度的事实。此外，它还具有一些模拟输入，并且具有一些内置的运动检测功能，可以检测诸如自由落体之类的东西，并指示FIFO缓冲区是否已满。

如果您正在寻找体积小且价格便宜的产品，并且仅测量加速度，那么这就是适合您的产品。其他惯性测量单元（或IMU），例如LSM9DS1；该LSM6DS3 ; LSM303C或LSM303C可以提供其他空间位置数据，例如陀螺仪或磁力计。

本指南介绍了将开发板插入RedBoard的基础知识，展示了如何使用Arduino库实时获取加速数据或通过FIFO收集，并介绍了库的用法。

所需材料

要继续学习，您需要以下材料：

- LIS3DH突破板
- Arduino UNO，RedBoard或其他Arduino兼容板

- 直公头-或电线。突破和面包板之间的联系。
- 面包板-任何大小（甚至迷你）都应该做。
- M / M跳线-用于在Arduino和面包板之间进行连接。

> **LIS3DH是3.3V器件！**高于~3.6V的电源电压可能会永久损坏IC。只要您的Arduino具有3.3V电源输出，并且使用I ^{2} C就可以了；您不需要任何额外的电平转换。但是，如果要使用SPI，则可能需要一个电平转换器。

任何运行5V的Arduino（UNO，RedBoard，Leonardo等）都需要一个逻辑电平转换器。如果您使用基于3.3V的'duino-像Arduino Pro 3.3V或3.3V Pro Mini-则不需要进行电平转换。

建议阅读

如果您不熟悉以下某些概念，建议您在继续之前先阅读该教程。

- 加速度计基础
- 陀螺仪
- 串行外设接口（SPI）
- IC间通信（I ^{2} C）
- 逻辑水平
- 双向电平转换器连接指南

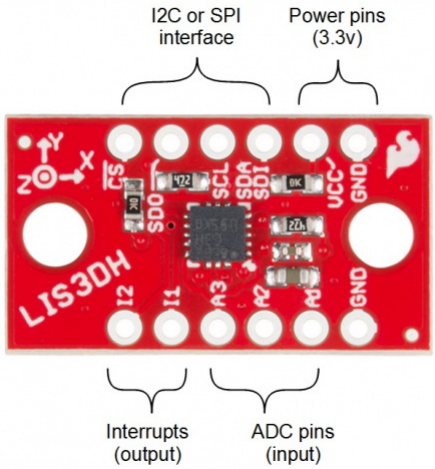
此外，以下ST文档对高级用户有帮助：

- LIS3DH_Datasheet-硬件信息和寄存器映射。
- LIS3DH_AppNote-描述性材料显示基本用法。

硬件概述和组装

您可以使用几种不同的方法来使用LIS3DH。

评估板的顶部具有LIS3DH传感器，一些旁路电容和上拉电阻。



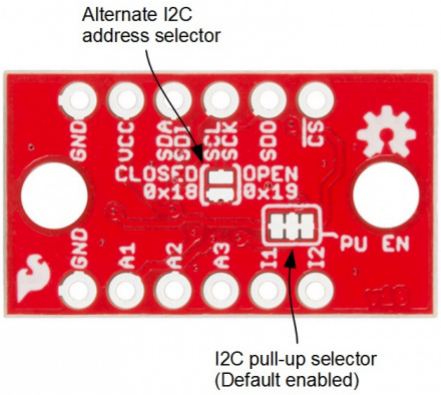
引脚连接

该表提供了有关每个引脚功能的更多信息。串行端口可以通过SPI或I2C连接，并且两者使用相同的物理引脚。要开始使用，只需连接您选择的接口，提供3.3v电压并接地。请注意，无论选择哪种通信方式，都无需使用所有引脚。

				连接

组	名称	方向	描述	I2C	SPI
序列号	！ CS	一世	片选（用于SPI）	数控	！ CS
	SDO	Ø	数据输出（用于SPI的MISO）	数控	味噌
	SCL	一世	数据时钟	SCL	SCK
	SDA / SDI	输入输出	数据输入（用于I2C的SDA，用于SPI的MOSI）	SDA	摩西
中断	I1	Ø	主整数具有FIFO +运动	可选的MCU	
	I2	Ø	次要int有运动	可选的MCU	
ADC	A1	一世	模拟输入	可选的	
	A2	一世	模拟输入	可选的	
	A3	一世	模拟输入（不用于温度读数）	可选的	
功率	VCC	一世	3.3V输入	供应	
	地线	一世	接地（PTH均可）	供应	

在底部，有两个跳线分别对应于I2C地址和上拉使能。



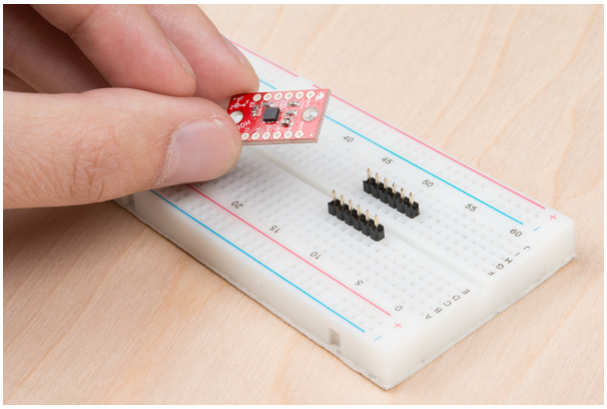
提供以下选项：

- **I2C地址跳线**-桥接以使用备用地址0x18，否则为0x19保持打开状态。保持开放状态以供SPI使用。
- **I2C上拉使能**-默认情况下处于关闭状态，这在I2C线和VCC之间连接了一个上拉电阻。通常这不会干扰SPI操作，但是，如果需要较少的功耗，请小心地切断铜走线。

使用面包板

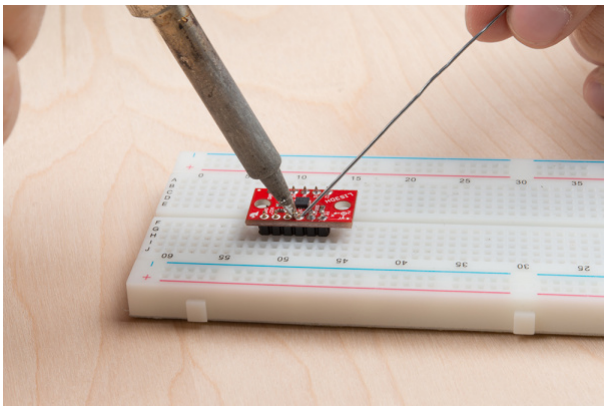
该传感器可以很好地与面包板配合使用，以方便连接，并且由于它给加速度计增加了一些质量，因此它与项目或手机的预期值更加接近。

要添加接头，请断开两个6针长的0.1英寸公接头，并将它们放入面包板中以用作焊接夹具。



放置两排接头，并准备焊接。

将分线板放到插针上，然后焊接各行。



焊接在排针上。

恭喜你！现在您可以将传感器连接到所选的微控制器。

获取Arduino库

本指南中的示例使用Arduino IDE和RedBoard与LIS3DH进行通信。

要获取Arduino库，请从Github下载，或使用Arduino库管理器。

下载Github存储库

访问GitHub存储库以下载该库的最新版本，或单击下面的链接：

下载ARDUINO库

图书馆经理

如需安装库的帮助，请查看我们的“如何安装Arduino库”教程。

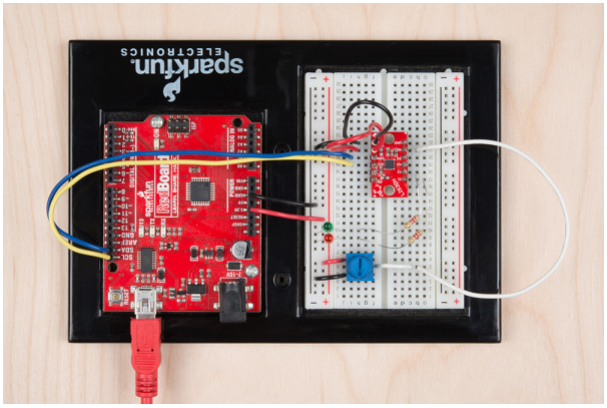
如果你没有最终使用的经理，你需要给移动*SparkFun_LIS3DH_Arduino_Library*文件夹到库您的Arduino速写本中的文件夹。

示例：I2C，模拟和中断

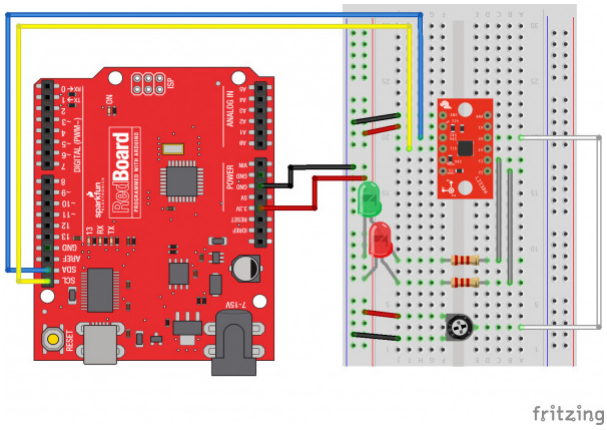
第一个电路允许RedBoard通过I2C与LIS3DH对话，并在中断和ADC引脚上提供连接。

中断对于指示极端Gs或自由落体等情况很有用，并告诉RedBoard FIFO已满并且需要服务。模拟输入引脚可用于测量各种电压，类似于RedBoard的模拟输入，但电压范围更受限制（约0.9V至1.8V）。如果不需要这些，只需连接电源，接地和通信引脚，而忽略中断和ADC示例。

使用这两张图片作为构建电路的指南。



建立在*RedBoard*上的电路



*Fritzing*中显示的连接

基本加速度计数据收集：

从基本的加速度计草图开始，该库在库中也称为“ MinimalistExample ”。这将定期对传感器进行采样并以检测到的G数的形式显示数据。请记住，垂直轴在静止时的读数为1G。

```
#include "SparkFunLIS3DH.h"
#include "Wire.h"
#include "SPI.h"

LIS3DH myIMU; //Default constructor is I2C, addr 0x19.

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  delay(1000); //relax...
  Serial.println("Processor came out of reset.\n");

  //Call .begin() to configure the IMU
  myIMU.begin();

}


void loop()
{
  //Get all parameters
  Serial.print("\nAccelerometer:\n");
  Serial.print(" X = ");
  Serial.println(myIMU.readFloatAccelX(), 4);
  Serial.print(" Y = ");
  Serial.println(myIMU.readFloatAccelY(), 4);
  Serial.print(" Z = ");
  Serial.println(myIMU.readFloatAccelZ(), 4);

  delay(1000);
}
```

输出示例:

```
Processor came out of reset.


Accelerometer:
 X = -0.1481
 Y = -0.1361
 Z = 0.9768

Accelerometer:
 X = -0.1481
 Y = -0.1361
 Z = 0.9768

Accelerometer:
 X = -0.1481
 Y = -0.1361
 Z = 0.9768

Accelerometer:
 X = -0.1481
 Y = -0.1361
 Z = 0.9768
```

运行时，草图将以Gs显示数据到串行终端。每秒都会收集并打印数据。

使用ADC

要尝试模拟输入，请加载名为“ ADCUsage ”的示例，或从以下部分复制粘贴。此示例还显示了可以在 `begin()` 功能内应用的一些其他设置。

```
#include "SparkFunLIS3DH.h"
#include "Wire.h"
#include "SPI.h"

LIS3DH myIMU; //Default constructor is I2C, addr 0x19.

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  delay(1000); //relax...
  Serial.println("Processor came out of reset.\n");

  myIMU.settings.adcEnabled = 1;
  //Note:  By also setting tempEnabled = 1, temperature data is available
  //on ADC3.  Temperature *differences* can be read at a rate of
  //1 degree C per unit of ADC3
  myIMU.settings.tempEnabled = 0;
  myIMU.settings.accelSampleRate = 50;  //Hz.  Can be: 0,1,10,25,50,100,200,400,1600,5000 Hz
  myIMU.settings.accelRange = 2;      //Max G force readable.  Can be: 2, 4, 8, 16
  myIMU.settings.xAccelEnabled = 0;
  myIMU.settings.yAccelEnabled = 0;
  myIMU.settings.zAccelEnabled = 0;

  //Call .begin() to configure the IMU
  myIMU.begin();

}

void loop()
{
  //Get all parameters
  Serial.print("\nADC:\n");
  Serial.print(" 1 = ");
  Serial.println(myIMU.read10bitADC1());
  Serial.print(" 2 = ");
  Serial.println(myIMU.read10bitADC2());
  Serial.print(" 3 = ");
  Serial.println(myIMU.read10bitADC3());

  delay(300);
}
```

输出示例:

```
Processor came out of reset.


ADC:
 1 = 1020
 2 = 522
 3 = 506

ADC:
 1 = 1020
 2 = 544
 3 = 516

ADC:
 1 = 1020
 2 = 540
 3 = 517
```

草图每300ms打印三个ADC值。移动旋钮，查看值如何变化以及有效电压范围如何在整个范围的中间。将电线从ADC引脚上移到另一根，以查看控制值的变化。

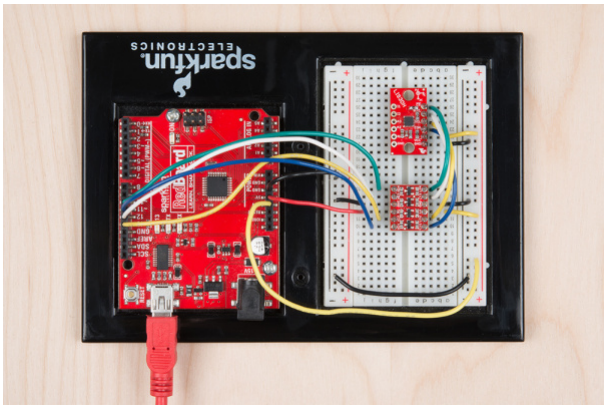
使用中断引脚

中断行为是高度可配置的，因此作为基本库函数被忽略。而是，LIS3DH寄存器根据数据表直接写入。

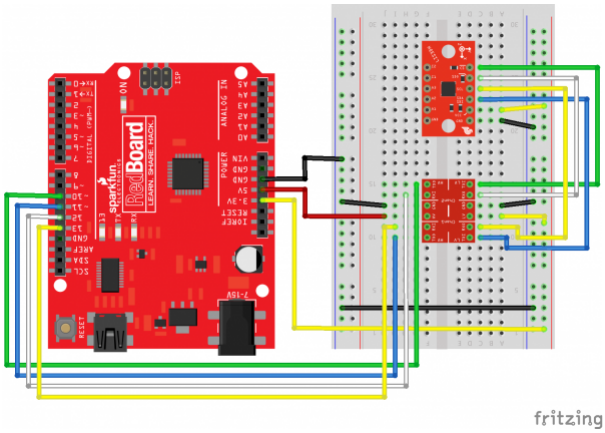
提供了一个示例，该示例在模板功能中使用注释配置了相关寄存器，可以将其复制到项目中并进行修改。运行名为IntUsage的示例，当检测到加速度超过极限时，它将在一个引脚上引发中断，而在检测到敲击时将在另一引脚上引发脉冲。

示例：SPI和FIFO使用

与LIS3DH通信的第二种方法是使用SPI接口。SPI接口的工作电压为3.3v，因此请使用逻辑电平转换器或工作电压为3.3v的MCU。使用以下图片帮助构建电路。



建立在*RedBoard*上的电路



*Fritzing*中显示的连接

基本加速度计数据收集：

SPI不是默认配置，因此您必须通过使用参数构造将额外的信息传递给库。通过改变修改“MinimalistExample” `LIS3DH myIMU;` 到 `LIS3DH myIMU(SPI_MODE, 10);` 了SPI模式连接到引脚10！CS引脚。

修改后的“ MinimalistExample”在此处列出：

```
#include "SparkFunLIS3DH.h"
#include "Wire.h"
#include "SPI.h"

LIS3DH myIMU(SPI_MODE, 10); // constructed with parameters for SPI and cs pin number

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  delay(1000); //relax...
  Serial.println("Processor came out of reset.\n");

  //Call .begin() to configure the IMU
  myIMU.begin();

}


void loop()
{
  //Get all parameters
  Serial.print("\nAccelerometer:\n");
  Serial.print(" X = ");
  Serial.println(myIMU.readFloatAccelX(), 4);
  Serial.print(" Y = ");
  Serial.println(myIMU.readFloatAccelY(), 4);
  Serial.print(" Z = ");
  Serial.println(myIMU.readFloatAccelZ(), 4);

  delay(1000);
}
```

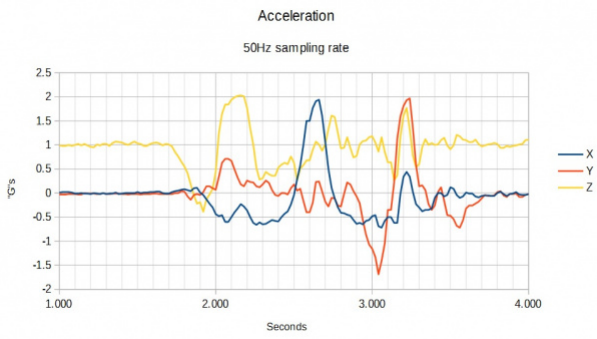
输出示例：

```
Processor came out of reset.


Accelerometer:
 X = -0.1481
 Y = -0.1361
 Z = 0.9768

Accelerometer:
 X = -0.1481
 Y = -0.1361
 Z = 0.9768

Accelerometer:
 X = -0.1481
 Y = -0.1361
 Z = 0.9768

Accelerometer:
 X = -0.1481
 Y = -0.1361
 Z = 0.9768
```

运行时，草图将以Gs显示数据到串行端子。每秒都会收集并打印数据。

FIFO使用：

SPI总线的运行速度比I2C快，因此对于需要定期采样的高速数据收集，建议使用SPI。



通过获取示例的输出并将其复制粘贴到电子表格程序中，然后创建图表来制作此图。在数据收集过程中，传感器在每个轴上绕一只脚来回移动。

```
#include "SparkFunLIS3DH.h"
#include "Wire.h"
#include "SPI.h"

LIS3DH myIMU(SPI_MODE, 10); //Constructing with SPI interface information
//LIS3DH myIMU(I2C_MODE, 0x19); //Alternate constructor for I2C

uint32_t sampleNumber = 0; //Used to make CSV output row numbers

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  delay(1000); //relax...
  Serial.println("Processor came out of reset.\n");

  myIMU.settings.adcEnabled = 0;
  //Note:  By also setting tempEnabled = 1, temperature data is available
  //instead of ADC3 in.  Temperature *differences* can be read at a rate of
  //1 degree C per unit of ADC3 data.
  myIMU.settings.tempEnabled = 0;
  myIMU.settings.accelSampleRate = 10;  //Hz.  Can be: 0,1,10,25,50,100,200,400,1600,5000 Hz
  myIMU.settings.accelRange = 2;      //Max G force readable.  Can be: 2, 4, 8, 16
  myIMU.settings.xAccelEnabled = 1;
  myIMU.settings.yAccelEnabled = 1;
  myIMU.settings.zAccelEnabled = 1;

  //FIFO control settings
  myIMU.settings.fifoEnabled = 1;
  myIMU.settings.fifoThreshold = 20;  //Can be 0 to 31
  myIMU.settings.fifoMode = 1;  //FIFO mode.
  //fifoMode can be:
  //  0 (Bypass mode, FIFO off)
  //  1 (FIFO mode)
  //  3 (FIFO until full)
  //  4 (FIFO when trigger)

  //Call .begin() to configure the IMU (except for the fifo)
  myIMU.begin();

  Serial.print("Configuring FIFO with no error checking...");
  myIMU.fifoBegin(); //Configure fifo
  Serial.print("Done!\n");

  Serial.print("Clearing out the FIFO...");
  myIMU.fifoClear();
  Serial.print("Done!\n");
  myIMU.fifoStartRec(); //cause fifo to start taking data (re-applies mode bits)

}

void loop()
{
  //float temp;  //This is to hold read data
```

```
  //uint16_t tempUnsigned;
  //
  while(( myIMU.fifoGetStatus() & 0x80 ) == 0) {};  //Wait for watermark

  //Now loop until FIFO is empty.
  //If having problems with the fifo not restarting after reading data, use the watermark
  //bits (b5 to b0) instead.
  //while(( myIMU.fifoGetStatus() & 0x1F ) > 2) //This checks that there is only a couple entrie
s left
  while(( myIMU.fifoGetStatus() & 0x20 ) == 0) //This checks for the 'empty' flag
  {
      Serial.print(sampleNumber);
      Serial.print(",");
      Serial.print(myIMU.readFloatAccelX());
      Serial.print(",");
      Serial.print(myIMU.readFloatAccelY());
      Serial.print(",");
      Serial.print(myIMU.readFloatAccelZ());
      Serial.println();
      sampleNumber++;
  }

}
```

输出示例：

```
Processor came out of reset.

Configuring FIFO with no error checking...Done!
Clearing out the FIFO...Done!
0,-0.15,-0.14,1.04
1,-0.17,-0.12,1.02
2,-0.21,-0.10,0.95
3,-0.21,-0.10,1.01
4,-0.22,-0.12,1.07
5,-0.17,-0.12,0.99
6,-0.12,-0.15,0.96
7,-0.18,-0.12,0.94
8,-0.19,-0.10,0.98
9,-0.20,-0.14,1.04
10,-0.19,-0.12,0.99
11,-0.20,-0.10,0.95
12,-0.21,-0.12,1.06
13,-0.14,-0.12,0.98
14,-0.10,-0.11,0.95
15,-0.12,-0.10,0.94
16,-0.14,-0.09,0.90
...
```

请注意，输出会定期产生大量数据。即使等待收集数据，仍会定期对其进行采样。当FIFO超过行中配置的水印时收集数据 `myIMU.settings.fifoThreshold = 20;` 。

额外示例和Arduino库参考

Arduino库中包含以下示例：

- **ADCUsage-**在读取中演示模拟，并有关于温度收集的注释
- **FifoExample-**演示使用内置缓冲区突发收集数据-**设置的良好演示**
- **FullSettingExample-**显示所有设置，并注释掉未使用的选项
- **IntUsage-**显示中断位的配置
- **LowLevelExample-**演示仅使用核心驱动程序而没有数学和设置开销
- **MinimalistExample** -的**最简单的**配置
- **MultiI2C** -使用过I2C 2个LIS3DHs
- **MultiSPI** -使用过SPI 2个LIS3DHs

图书馆使用

请按照以下步骤使用库

- 使用这些构造之一在全局空间中构造一个对象
  - 无参数-地址为0x19的I2C模式
  - I2C_MODE，地址
  - SPI_MODE，引脚号
- 使用in开始，设置.settings。价值观
- 跑 `.begin()`

例：

```
LIS3DH myIMU; //This creates an instance the library object.

void setup()
{
    myIMU.settings.adcEnabled = 1;
    myIMU.settings.tempEnabled = 0;
    myIMU.settings.accelSampleRate = 50;  //Hz.  Can be: 0,1,10,25,50,100,200,400,1600,5000 Hz
    myIMU.settings.accelRange = 2;      //Max G force readable.  Can be: 2, 4, 8, 16
    myIMU.begin();
}
```

设定值

LIS3DH主类有一个公共成员，称为设置。要配置设置，请使用格式 `myIMU.settings.accelSampleRate = (...);` 。然后，致电 `.begin()` 申请。

设置包含以下成员：

- `uint8_t adcEnabled` -设置为1以启用ADC
- `uint8_t tempEnabled` -设置为1以使用增量温度信息覆盖ADC3
- `uint16_t accelSampleRate` -可以是：0,1,10,25,50,100,200,400,1600,5000 Hz
- `uint8_t accelRange` -最大G力可读 可以是：2、4、8、16
- `uint8_t xAccelEnabled` -设置为1以启用x轴
- `uint8_t yAccelEnabled` -设置为1以启用y轴
- `uint8_t zAccelEnabled` -设置为1以启用z轴
- `uint8_t fifoEnabled` -设置为1以启用FIFO
- `uint8_t fifoMode` -可以是0x0,0x1,0x2,0x3
- `uint8_t fifoThreshold` -检测到水印之前读取的字节数（0到31）

职能

> **高级程序员：** LIS3DH类继承了LIS3DHCore，可以在没有所有这些功能的情况下进行通信，因此您可以编写自己的函数。此联播指南中未涵盖此类。

uint8_t begin（void）；

在提供设置后调用，以启动线或SPI库（如构造所示）并运行 `applySettings()` 。返回0表示成功。

void applySettings（void）；

这将根据.settings的内容配置IMU的寄存器。

int16_t readRawAccelX（void）；

int16_t readRawAccelY（void）；

int16_t readRawAccelZ（void）；

这些函数以16位带符号整数形式返回轴加速度信息。

浮点型readFloatAccelX（void）；

浮点型readFloatAccelY（void）；

浮点型readFloatAccelZ（void）；

这些函数调用Raw函数，然后应用数学运算将其转换为以Gs表示加速度的浮点数。

uint16_t read10bitADC1（void）；

uint16_t read10bitADC2（void）；

uint16_t read10bitADC3（void）；

这些功能返回从引脚读取的ADC值。值将为10位，可检测范围约为0.9V至1.8V。

注：当时 `tempEnabled == 1` ，ADC3读数为未参考温度（以摄氏度为单位）。读取两次并计算温度变化。

void fifoBegin（void）；

这通过将适当的值写入FIFO控制寄存器和控制寄存器5来启用FIFO。这不会启动到FIFO的数据收集，而是 `fifoStartRec()` 在准备就绪时运行。

采样率取决于.settings中选择的数据率。

void fifoClear（void）；

这将读取所有数据，直到状态表明没有可用数据为止，然后丢弃数据。如果FIFO用旧数据填充，则用于从新数据开始。

void fifoStartRec（void）

这将启用FIFO数据收集。在开始检查数据是否可用之前，请运行此命令。

使用fifoStartRec之后，来自X，Y，Z寄存器的数据不是实时的，而是下一个可用的样本。

uint8_t fifoGetStatus（void）

这将返回FIFO状态字节。字节的内容如下：

- bit 7：超出水印
- 位6：FIFO溢出

- 位5：FIFO为空
- 位4到0：可用样本数（0到31）

void fifoEnd（void）；

这将停止FIFO，并使设备恢复正常操作。

资源和进一步发展

现在，您应该对如何使用LIS3DH有基本的了解，但是如果需要更多信息，请查看以下链接：

- LIS3DH Breakout Github存储库-设计文件。
- 英国威廉希尔SparkFun LIS3DH Arduino库Github回购-arduino库。
- LIS3DH数据表-硬件信息和寄存器映射
- LIS3DH AppNote-描述性材料，显示基本用法。

走得更远

需要一点灵感吗？查看其他一些很棒的SparkFun教程。

红板圣诞老人陷阱

一个有趣的假期项目，适合任何希望在圣诞节赶上圣诞老人的人！

串行控制电机驱动器连接指南

串行控制电机驱动器的连接指南

奇特的例子：超级马里奥兄弟西洋镜

使用眼镜电子设备制作动画西洋镜（有声音！）的研究。

英国威廉希尔SparkFun 9DoF IMU（ICM-20948）突破连接指南

如何在运动感应项目中使用SparkFun 9DoF ICM-20948转接板。此突破是可穿戴传感器和物联网应用的理想选择。