在github上面有几个BMI088加速度传感器的Arduino库，其中有一个库做的很棒 - bolderflight/BMI088。该库支持SPI和I2C连接方式。非常感谢Bolder Flight Systems做出的贡献。本文对github中该库的使用方法进行了翻译，如有错漏，敬请指正。

****

BMI088

本Arduino库用于与BMI088六轴惯性测量单元（IMU）通信。

简介

博世Sensortec BMI088是一款高性能六轴惯性测量单元（IMU），具有高振动稳定性，专为无人机和机器人应用而设计。 BMI088专门设计用于有效抑制由于PCB上的共振或整个系统的结构而可能发生的振动。除了高振动稳健性外，BMI088卓越的温度稳定性有助于提高估算滤波器性能，IMU具有极宽的24G加速度计范围。

BMI088是一种系统级封装（SiP），将加速度计和陀螺仪组合到一个封装中。 BMI088支持高达400 kHz的I2C和高达10 MHz的SPI通信。加速度计和陀螺仪可以单独访问，也可以同步输出数据。 BMI088还具有可编程满量程范围、输出数据速率和中断。

用法

该库支持与BMI088进行I2C和SPI通信。

安装

只需将此库克隆或下载到Arduino / libraries文件夹中即可。

功能说明

该库支持与BMI088进行I2C和SPI通信。类可用于加速度计、陀螺仪和两者的同步输出。

Bmi088Accel类

该类用于使用BMI088加速度计。 Bmi088Accel对象声明重载了I2C和SPI通信的不同声明。所有其他函数保持不变。

I2C对象声明

Bmi088Accel(TwoWire &bus,uint8_t address)：应声明Bmi088Accel对象，指定I2C总线和Bmi088Accel I2C地址。如果SDO1引脚接地，则I2C地址为0x18；如果SDO1引脚拉高，则I2C地址为0x19。例如，以下代码声明一个名为accel的Bmi088Accel对象，其中BMI088传感器位于I2C总线0上，传感器地址为0x18（SDO1接地）。

```
01.  Bmi088Accel accel(Wire,0x18);
```
复制代码

SPI对象声明

Bmi088Accel(SPIClass &bus,uint8_t csPin)：应声明Bmi088Accel对象，指定SPI总线和使用的芯片选择引脚。可以在同一SPI总线上使用多个BMI088或其他SPI对象，每个都有自己的片选引脚。片选引脚可以是任何可用的数字引脚。例如，以下代码声明了一个名为accel的Bmi088Accel对象，其中BMI088传感器位于SPI总线0上，芯片选择引脚10。

```
01.  Bmi088Accel accel(SPI,10);
```
复制代码

常用设置函数

以下函数用于设置BMI088传感器。这些应该在数据收集之前调用一次，通常这是在Arduino ***void setup()***函数中完成的。应始终使用begin函数。以下函数可选地，setOdr可用于设置输出数据速率和数字低通滤波，setRange可用于设置传感器的满量程范围。 pinModeInt1和pinModeInt2可用于设置引脚1和引脚2中断引脚（推挽与开漏，高电平有效与低电平有效）。 mapDrdyInt1和mapDrdyInt2分别将数据就绪中断映射到引脚1和引脚2。如果不使用这些可选函数，则使用输出数据速率和数字低通滤波的默认值，而不配置数据就绪中断引脚。

int begin()：这应该在你的setup函数中调用。它初始化与BMI088加速度计的通信，并设置传感器以读取数据。此函数在成功初始化时返回正值，并在不成功初始化时返回负值。如果不成功，请检查您的接线或尝试复位传感器的电源。以下是设置Bmi088Accel的示例。

```
01.  int status;
02.  status = accel.begin();
```
复制代码

（可选）***bool setOdr(Odr odr)*** ：BMI088具有可编程输出数据速率和数字低通滤波功能。支持以下枚举设置：

输出数据速率	**DLPF**带宽	**ODR**枚举名称
1600 Hz	280 Hz	ODR_1600HZ_BW_280HZ
1600 Hz	234 Hz	ODR_1600HZ_BW_234HZ
1600 Hz	145 Hz	ODR_1600HZ_BW_145HZ

800 Hz	230 Hz	ODR_800HZ_BW_230HZ
800 Hz	140 Hz	ODR_800HZ_BW_140HZ
800 Hz	80 Hz	ODR_800HZ_BW_80HZ
400 Hz	145 Hz	ODR_400HZ_BW_145HZ
400 Hz	75 Hz	ODR_400HZ_BW_75HZ
400 Hz	40 Hz	ODR_400HZ_BW_40HZ
200 Hz	80 Hz	ODR_200HZ_BW_80HZ
200 Hz	38 Hz	ODR_200HZ_BW_38HZ
200 Hz	20 Hz	ODR_200HZ_BW_20HZ
100 Hz	40 Hz	ODR_100HZ_BW_40HZ
100 Hz	19 Hz	ODR_100HZ_BW_19HZ
100 Hz	10 Hz	ODR_100HZ_BW_10HZ
50 Hz	20 Hz	ODR_50HZ_BW_20HZ
50 Hz	9 Hz	ODR_50HZ_BW_9HZ
50 Hz	5 Hz	ODR_50HZ_BW_5HZ
25 Hz	10 Hz	ODR_25HZ_BW_10HZ
25 Hz	5 Hz	ODR_25HZ_BW_5HZ
25 Hz	3 Hz	ODR_25HZ_BW_3HZ
12.5 Hz	5 Hz	ODR_12_5HZ_BW_5HZ
12.5 Hz	2 Hz	ODR_12_5HZ_BW_2HZ
12.5 Hz	1 Hz	ODR_12_5HZ_BW_1HZ

以下示例是将输出数据速率设置为100 Hz，带宽设置为19 Hz。

```
01.  bool status;
02.  status = accel.setOdr(Bmi088Accel::ODR_100HZ_BW_19HZ);
```
复制代码

（可选）***bool setRange(Range range)***：BMI088可以编程满量程范围。支持以下枚举设置：

满量程范围	枚举变量名称
+/- 24 G	RANGE_24G
+/- 12 G	RANGE_12G
+/- 6 G	RANGE_6G
+/- 3 G	RANGE_3G

以下是将量程范围设置为+/- 6 G的示例。

```
01.  bool status;
02.  status = accel.setRange(Bmi088Accel::RANGE_6G);
```
复制代码

（可选）***pinModeInt1(PinMode mode, PinLevel level)*** 、（可选）***pinModeInt2(PinMode mode, PinLevel level)*** ：BMI088加速度计具有两个中断引脚，数据就绪中断可以映射两个。这两个中断引脚的每一个都可以设置成输出是推挽还是漏极开路，以及引脚是高电平有效还是低电平有效。

引脚模式	名称
推拉（Push-Pull）	PUSH_PULL
漏极开路	OPEN_DRAIN

有效电平	名称
高电平	ACTIVE_HIGH
低电平	ACTIVE_LOW

下面是将中断引脚1设置为推挽和高电平有效的示例。

```
01.  bool status;
02.  status = accel.pinModeInt1(Bmi088Accel::PUSH_PULL,Bmi088Accel::ACTIVE_HIGH);
     复制代码
```

（可选）***mapDrdyInt1(bool enable)*** 、（可选）***mapDrdyInt2(bool enable)*** ：这些函数设置BMI088加速度计是否将其数据就绪中断映射到引脚1和/或引脚2。设置为true将数据就绪中断映射到引脚，或false表示禁用映射。

```
01.  bool status;
02.  status = accel.mapDrdyInt1(true);
     复制代码
```

常用数据收集函数

以下函数用于从BMI088加速度计收集数据。将数据缩放到工程单位并转换为右手坐标系，Z轴正向下。请参阅"坐标系"部分。除加速度外，BMI088加速度计还提供BMI088芯片温度和传感器时间值。加速度计数据以m / s / s为单位返回，温度数据以摄氏度为单位返回，时间数据以皮秒为单位返回。 readSensor用于读取传感器并将最新数据存储在缓冲区中，每次要从传感器检索数据时都应调用它。 getAccelX_mss、getAccelY_mss、getAccelZ_mss、getTemperature_C和getTime_ps返回该缓冲区的加速度计值、温度和时间值。

bool readSensor() ：读取传感器并将最新数据存储在缓冲区中，每次要从传感器检索数据时都应调用它。

```
01.  bool status;
02.  status = accel.readSensor();
     复制代码
```

float getAccelX_mss() ：从数据缓冲区中获取X方向的加速度计值，并以m / s / s为单位返回。

```
01.  float ax;
02.  ax = accel.getAccelX_mss();
     复制代码
```

float getAccelY_mss() ：从数据缓冲区中获取Y方向的加速度计值，并以m / s / s为单位返回。

```
01.  float ay;
02.  ay = accel.getAccelY_mss();
     复制代码
```

float getAccelZ_mss()：从数据缓冲区中获取Z方向的加速度计值，并以m / s / s为单位返回。

```
01.  float az;
02.  az = accel.getAccelZ_mss();
     复制代码
```

float getTemperature_C()：从数据缓冲区中获取die温度值，并以C为单位返回。

```
01.  float t;
02.  t = accel.getTemperature_C();
     复制代码
```

uint64_t getTime_ps()：从数据缓冲区获取传感器时间值，并以ps为单位返回。

```
01.  uint_64t time_ps;
02.  time_ps = accel.getTime_ps();
     复制代码
```