MPU9150Lib

MPU-9150 9-Axis Data Fusion

Introduction

MPU9150Lib demonstrates how 9-axis fused data can be obtained from an InvenSense MPU-9150 when used with an Arduino. MPU9150Lib was tested on a LilyPad with ATMega 328 and an Arduino Due with a Sparkfun MPU-9150 breakout board (Sparkfun SEN-11486). However, the code should work on any Arduino with at least 32k bytes of flash memory (code size is around 29k bytes). Note however that the example sketches will not fit in boards that have 32k bytes of flash memory and also have a USB port (Leonardo and USB LilyPad for example) as the USB boot loader takes a significant amount of space.

This is in no way intended to be a highly optimized implementation and should be regarded as a demonstration of how to use the MPU-9150. Code size could be reduced by simplifying the code to get rid of unused functionality. The algorithm steps have been kept simple and (hopefully) understandable. Quite possibly some of the steps could be combined in more efficient ways.

There are several libraries included:

- MPU9150Lib. This is responsible for implementing the 9-axis fusion.
- CalLib. This contains code related to magnetometer and accelerometer calibration.
- Motion Driver. This contains selected (and modified) files from the InvenSense Embedded SDK v5.1.
- I2CDev. This contains helpful I2C utilities and is by Jeff Rowberg. It has been slightly modified for use with the Due.
- DueFlash. This emulates the EEPROM library for AVR Arduinos.

There are also six example sketches:

- Arduino9150. This is a simple sketch that demonstrates how to use MPU9150Lib. By default, it just prints out the fused Euler angles (roll, pitch and yaw).
- ArduinoDual9150. Shows how to use two MPU-9150s in a single system.
- Due9150. Demonstrates how to use the MPU-9150 with the Due and also provides mag and accel calibration code.
- Accel9150. This sketch uses the output of the data fusion to derive residual accelerations in the frame of reference of the IMU (the body frame). Note that accelerometer calibration must have been performed for this to work.
- MagCal9150. This is a utility that provides simple calibration data for the magnetometers.
- AccelCal9150. This is a utility that allows accelerometer offsets to be calibrated.

Licenses

The Pansenti components (sketches, MPU9150Lib and CalLib) are released under the MIT License. Please see the I2CDev and Motion Driver libraries for information on their respective licenses.

Quickstart

Hardware

Only four wires are needed between the LilyPad (or other Arduino) and the MPU-9150: +3.3V, GND, SDA and SCL. Note that the MPU-9150 is not 5V I/O tolerant. The LilyPad can be operated at 3.3V making it ideal for this purpose. The interrupt signal from the MPU-9150 is not used by this library. The Due is a 3.3V device so connection to the Due should be straightforward.

Software

The code was developed using Arduino 1.0.3 for 8 bit Arduinos and Arduino 1.5.2 for the Due.

For 8 bit Arduinos, compile and upload the MagCal9150 sketch. When it starts running, the serial monitor (running at 115200 bits per second by default) will display the current max and min values for each axis every time one of them changes. The trick is to move the MPU-9150 so that each axis sees the maximum and minimum value at least once. This requires putting each side, top and bottom of the MPU-9150 towards the ground and waggling it a little to ensure that the maximum or minimum is obtained. When the display stops updating, this means that the maximum or minimum has been obtained. When this has been done for all three axes, enter “s” followed by return in the serial monitor to save the calibration data.

It is very important to calibrate the magnetometers as otherwise the yaw readings will be unusable.

The accelerometers can also be calibrated using AccelCal9150. This is not as important as the magnetometer calibration but still useful. The code works in a very similar way and is used to adjust the offset from zero of the accelerometer axes. A word of caution about accelerometer calibration – move the MPU-9150 very slowly and carefully. Normal values are in the range 14000 to 18000 for maxima and the opposite for minima. Numbers outside of this mean that the MPU-9150 has been jerked and the accelerometer calibration needs to be restarted.

Magnetometer and accelerometer data is stored in EEPROM on the Arduino so the calibration process should only be needed once unless some part of the system is changed.

The Due sketch Due9150 includes code to perform accel and mag calibration. Check the comments in Due9150.ino for instructions.

MPU9150Lib

MPU9150Lib.cpp and MPU9150Lib.h implement most of the high level functionality. The MPU-9150 itself performs data fusion of the gyro and accelerometer data. The library performs the final step of fusing the magnetometer data. The fusing algorithm is pretty basic and could certainly be improved but it does work quite well.

Where Euler angles are used, the convention (described in terms of an aircraft as seen by the pilot inside it) has the x (roll) axis pointing forward from the nose, the y (pitch) axis pointing out along the right wing and the z (yaw) axis pointing straight down. Using this convention, a roll that puts the right wing down is a positive roll angle change, a pitch that puts the nose up is a positive pitch angle change and a rotation of the aircraft about the yaw axis in the clockwise direction increases the heading angle. Note that the raw data from the MPU-9150 does not fit with this convention but the processed variables do use this convention.

The Sparkfun breakout board has the gyro axis orientation marked, along with the magnetometer. The gyro x axis does correspond with the computed x axis but the others do not – the y and z axes are reversed in the computed versions.

Please see MPU9150Lib.h for more details of available functions and computed variables.