9 Degrees of Freedom MPU-9150

https://www.sparkfun.com/products/11486
https://github.com/Pansenti/MPU9150Lib

The MPU-9150 is the world's first 9-axis MotionTracking device designed for the low power, low cost, and high performance requirements of consumer electronics equipment including smartphones, tablets and wearable sensors. And guess what? You get to play with it.

This breakout board makes it easy to prototype with the InvenSense MPU-9150 by breaking out all the pins you need to standard 0.1" spaced headers. The board also provides I2C pullup resistors and a solder jumper to switch the I2C address of the device. The MPU-9150 is a System in Package (SiP) that combines two chips: the MPU-6050, which contains a 3-axis gyroscope, 3-axis accelerometer, and an onboard Digital Motion Processor (DMP) capable of processing complex 9-axis MotionFusion algorithms; and the AK8975, a 3-axis digital compass. The part's integrated 9-axis MotionFusion algorithms access all internal sensors to gather a full set of sensor data. The part is offered in a 4x4x1mm LGA package and is upgrade-compatible with the MPU-6050 integrated 6-axis MotionTracking device, providing a simple upgrade path and making it easy to fit on space constrained boards.

HARDWARE REQUIRED

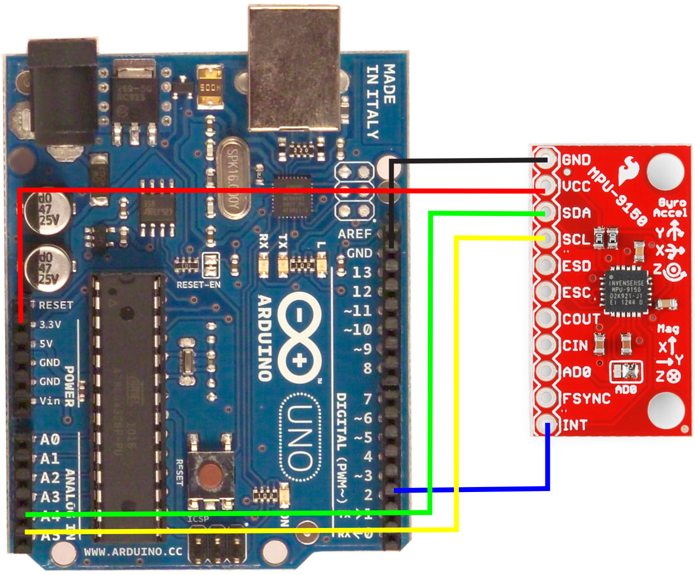
Arduino Board
(1) MPU-9150 Motion Tracking Device

CIRCUIT

MPU-9150 GRD > Arduino Ground
MPU-9150 VCC > Arduino 3.3 VOLT
MPU-9150 SDA > Arduino pin A4
MPU-9150 SCL > Arduino pin A5
MPU-9150 INT > Arduino pin 2

Futures

- Digital-output 9-axis MotionFusion data in rotation matrix, quaternion, Euler Angle, or raw data format
- Tri-Axis angular rate sensor (gyro) with a sensitivity up to 131 LSBs/dps and a full-scale range of ±250, ±500, ±1000, and ±2000dps
- Tri-Axis accelerometer with a programmable full scale range of ±2g, ±4g, ±8g and ±16g
- Tri-axis compass with a full scale range of ±1200µT
- Reduced settling effects and sensor drift by elimination of board-level cross-axis alignment errors between accelerometer, gyroscope, and compass
- VDD Supply voltage range of 2.4V–3.46V; VLOGIC of 1.8V±5% or VDD
- Gyro operating current: 3.6mA (full power, gyro at all rates)
- Gyro + Accel operating current: 3.8mA (full power, gyro at all rates, accel at 1kHz sample rate)
- Gyro + Accel + Compass + DMP operating current: 4.25mA (full power, gyro at all rates, accel at 1kHz sample rate, compass at 8Hz rate)
- Accel low power mode operating current: 10uA at 1Hz, 20uA at 5Hz, 70uA at 20Hz, 140uA at 40Hz
- Full Chip Idle Mode Supply Current: 8µA
- 400kHz Fast Mode I^2C serial host interface
- On-chip timing generator with ±1% frequency variation over full temperature range
- 10,000g shock tolerant
- I2C Pullup Resistors populated on board.
- All Pins Broken Out to Standard 0.1» Spaced Headers
- Solder Jumper for Switching LSB of I2C Address



IMAGE

CODE

```
// I2C device class (I2Cdev) demonstration
Arduino sketch for MPU9150
// 1/4/2013 original by Jeff Rowberg <jeff@
rowberg.net> at https://github.com/jrowberg/
i2cdevlib
//  modified by Aaron Weiss <aaron@sparkfun.com>
//
// Changelog:
//     2011-10-07 - initial release
//     2013-1-4 - added raw magnetometer output

/* =================================
I2Cdev device library code is placed under the
MIT license

Permission is hereby granted, free of charge, to
any person obtaining a copy
of this software and associated documentation
files (the «Software»), to deal
in the Software without restriction, including
without limitation the rights
to use, copy, modify, merge, publish, distribute,
sublicense, and/or sell
copies of the Software, and to permit persons to
whom the Software is
furnished to do so, subject to the following
conditions:

The above copyright notice and this permission
notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED «AS IS», WITHOUT
WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE
WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE
FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF
CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
=============================== */
```

```
// Arduino Wire library is required if I2Cdev I2CDEV_ARDUINO_WIRE
implementation is used in I2Cdev.h
#include “Wire.h”

// I2Cdev and MPU6050 must be installed as libraries, or else the
.cpp/.h files
// for both classes must be in the include path of your project
#include “I2Cdev.h”
#include “MPU6050.h”

// class default I2C address is 0x68
// specific I2C addresses may be passed as a parameter here
// AD0 low = 0x68 (default for InvenSense evaluation board)
// AD0 high = 0x69
MPU6050 accelgyro;

int16_t ax, ay, az;
int16_t gx, gy, gz;
int16_t mx, my, mz;

#define LED_PIN 13
bool blinkState = false;

void setup() {
    // join I2C bus (I2Cdev library doesn’t do this automatically)
    Wire.begin();

    // initialize serial communication
    // (38400 chosen because it works as well at 8MHz as it does at
16MHz, but
    // it’s really up to you depending on your project)
    Serial.begin(38400);

    // initialize device
    Serial.println(“Initializing I2C devices...”);
    accelgyro.initialize();

    // verify connection
    Serial.println(“Testing device connections...”);
    Serial.println(accelgyro.testConnection() ? “MPU6050 con-
nection successful” : “MPU6050 connection failed”);

    // configure Arduino LED for
    pinMode(LED_PIN, OUTPUT);
}

void loop() {
    // read raw accel/gyro measurements from device
    accelgyro.getMotion9(&ax, &ay, &az, &gx, &gy, &gz, &mx, &my,
&mz);

    // these methods (and a few others) are also available
    //accelgyro.getAcceleration(&ax, &ay, &az);
    //accelgyro.getRotation(&gx, &gy, &gz);

    // display tab-separated accel/gyro x/y/z values
    Serial.print(“a/g/m:\t”);
    Serial.print(ax); Serial.print(“\t”); // \t is used to print TAB
    Serial.print(ay); Serial.print(“\t”);
    Serial.print(az); Serial.print(“\t”);
    Serial.print(gx); Serial.print(“\t”);
    Serial.print(gy); Serial.print(“\t”);
    Serial.print(gz); Serial.print(“\t”);
    Serial.print(mx); Serial.print(“\t”);
    Serial.print(my); Serial.print(“\t”);
    Serial.println(mz);

    // blink LED to indicate activity
    blinkState = !blinkState;
    digitalWrite(LED_PIN, blinkState);
}
```