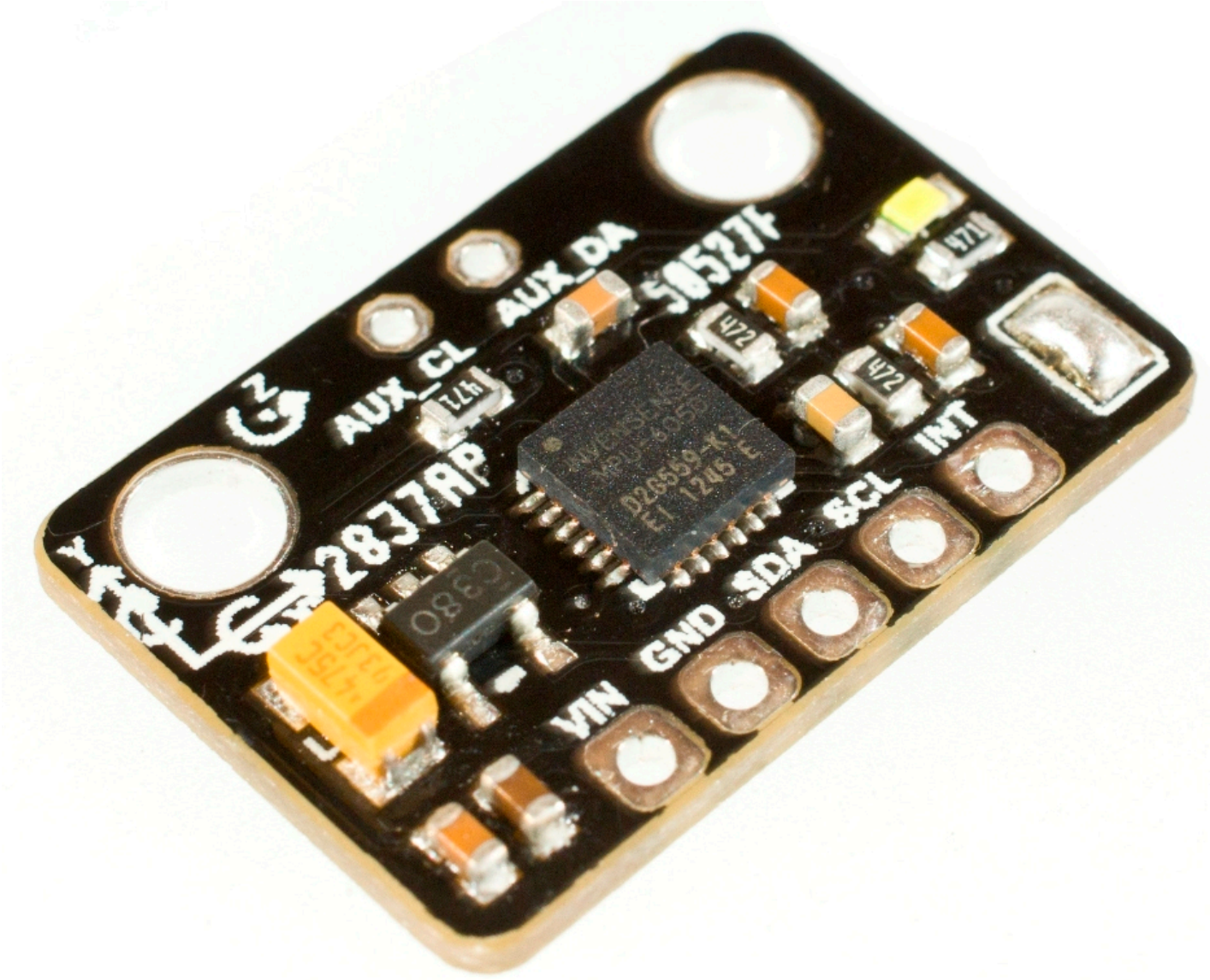
SKU:SEN0142 (https://www.dfrobot.com/product-880.html)



(https://www.dfrobot.com/product-880.html)

Introduction

At the beginning,the inertial measurement unit is an electronic device that measures and reports on a craft's velocity, orientation, and gravitational forces, using a combination of accelerometers, gyroscopes, and magnetometers. Now IMUs are

combination of accelerometers, gyroscopes, and magnetometers. Now IMUs are commonly used in the Human-computer interaction (HCI), navigational purposes and balancing technology used in the Segway Personal Transporter as we all known.

The MPU-6000/MPU-6050 family of parts are the world's first and only 6-axis MotionTracking devices designed for the low power, low cost, and high performance requirements of smartphones, tablets and wearable sensors.

The MPU-6000/6050 devices combine a 3-axis gyroscope and a 3-axis accelerometer on the same silicon die together with an onboard Digital Motion Processor (DMP) capable of processing complex 9-axis MotionFusion algorithms. The parts' integrated 9-axis MotionFusion algorithms access external magnetometers or other sensors through an auxiliary master I2C bus, allowing the devices to gather a full set of sensor data without intervention from the system processor.

The **6 Dof sensor** (https://www.dfrobot.com/product-880.html) breakout integrate with the MPU6050 sensor and the low noise 3.3v regulator and pull-up resistors for the I2C bus. So it's available to directly hook up the sensor with the Arduino processors for your robotics,HCI and wearable projects. With the Arduino library from i2cdevlib it's easy for you to drive this sensor and get the pitch, roll, yaw, quaternion, euler data.

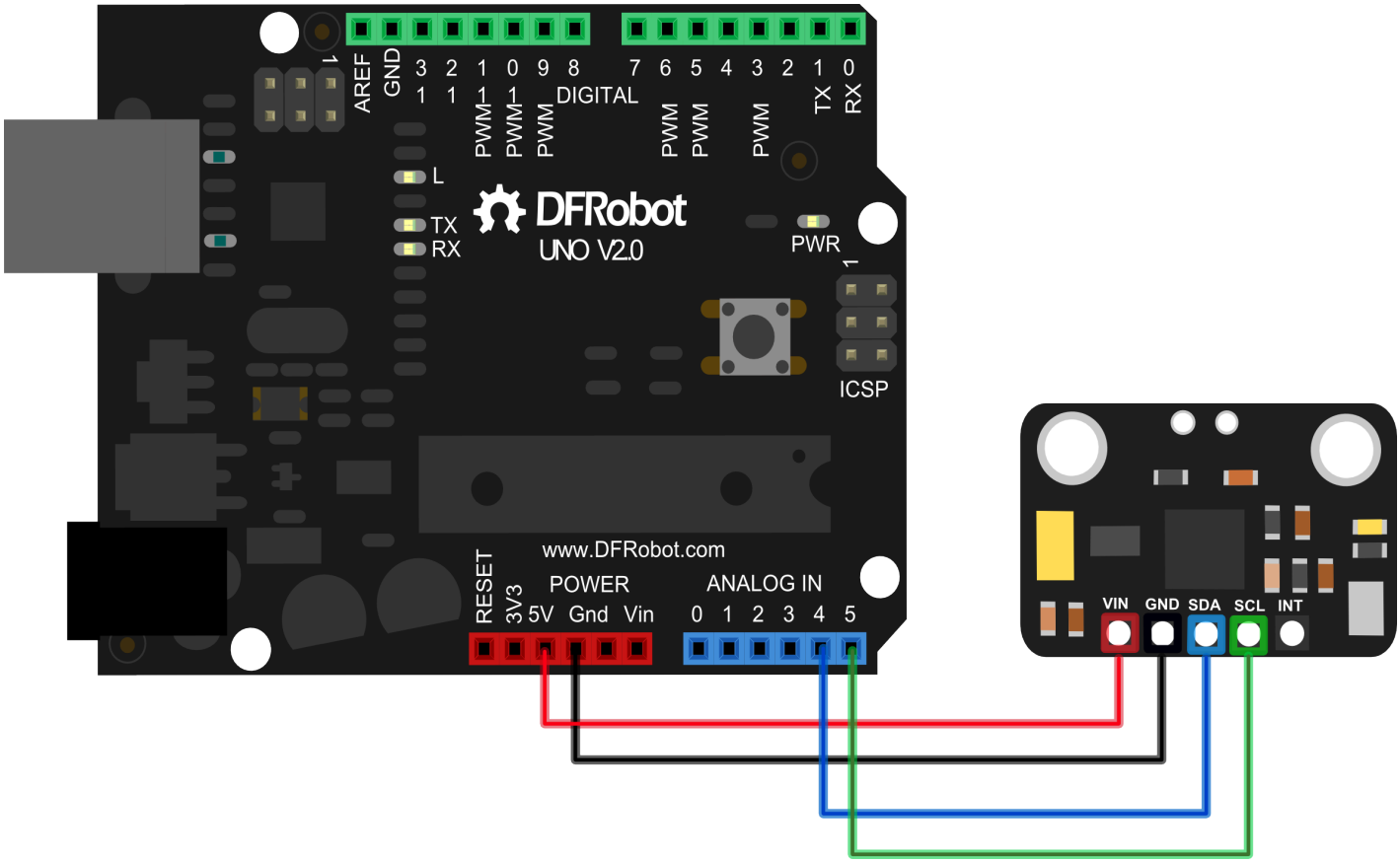
Specification

- Working voltage: 3~5v
- I2C Digital-output of 6 or 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
- Digital Motion Processing (DMP) engine offloads complex MotionFusion, sensor timing synchronization and gesture detection
- Embedded algorithms for run-time bias and compass calibration. No user

intervention required

- Dimensions: 14 x 21mm

Connection Diagram



Sample Code

Please download the libraries (https://github.com/jrowberg/i2cdevlib/tree/master/Arduino/MPU6050) for the all the IMU sensors first!

```
// I2C device class (I2Cdev) demonstration Arduino sketch for MPU6050 class
// 10/7/2011 by Jeff Rowberg <jeff@rowberg.net>
// Updates should (hopefully) always be available at https://github.com/jrowbe
//
// Changelog:
//      2013-05-08 - added multiple output formats
//                 - added seamless Fastwire support
//      2011-10-07 - initial release

/* ============================================
I2Cdev device library code is placed under the MIT license
Copyright (c) 2011 Jeff Rowberg
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.
===============================================
*/

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

// Arduino Wire library is required if I2Cdev I2CDEV_ARDUINO_WIRE implementati
```

```
// is used in I2Cdev.h
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
    #include "Wire.h"
#endif

// 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;
//MPU6050 accelgyro(0x69); // <-- use for AD0 high
//MPU6050 accelgyro(0x68, &Wire1); // <-- use for AD0 low, but 2nd Wire (TWI/I

int16_t ax, ay, az;
int16_t gx, gy, gz;



// uncomment "OUTPUT_READABLE_ACCELGYRO" if you want to see a tab-separated
// list of the accel X/Y/Z and then gyro X/Y/Z values in decimal. Easy to read
// not so easy to parse, and slow(er) over UART.
#define OUTPUT_READABLE_ACCELGYRO

// uncomment "OUTPUT_BINARY_ACCELGYRO" to send all 6 axes of data as 16-bit
// binary, one right after the other. This is very fast (as fast as possible
// without compression or data loss), and easy to parse, but impossible to rea
// for a human.
//#define OUTPUT_BINARY_ACCELGYRO


#define LED_PIN 13
bool blinkState = false;

void setup() {
    // join I2C bus (I2Cdev library doesn't do this automatically)
    #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
        Wire.begin();
    #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
        Fastwire::setup(400, true);
    #endif

    // 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 connection successful

    // use the code below to change accel/gyro offset values
    /*
    Serial.println("Updating internal sensor offsets...");
    // -76    -2359    1688    0    0    0
    Serial.print(accelgyro.getXAccelOffset()); Serial.print("\t"); // -76
    Serial.print(accelgyro.getYAccelOffset()); Serial.print("\t"); // -2359
    Serial.print(accelgyro.getZAccelOffset()); Serial.print("\t"); // 1688
    Serial.print(accelgyro.getXGyroOffset()); Serial.print("\t"); // 0
    Serial.print(accelgyro.getYGyroOffset()); Serial.print("\t"); // 0
    Serial.print(accelgyro.getZGyroOffset()); Serial.print("\t"); // 0
    Serial.print("\n");
    accelgyro.setXGyroOffset(220);
    accelgyro.setYGyroOffset(76);
    accelgyro.setZGyroOffset(-85);
    Serial.print(accelgyro.getXAccelOffset()); Serial.print("\t"); // -76
    Serial.print(accelgyro.getYAccelOffset()); Serial.print("\t"); // -2359
    Serial.print(accelgyro.getZAccelOffset()); Serial.print("\t"); // 1688
    Serial.print(accelgyro.getXGyroOffset()); Serial.print("\t"); // 0
    Serial.print(accelgyro.getYGyroOffset()); Serial.print("\t"); // 0
    Serial.print(accelgyro.getZGyroOffset()); Serial.print("\t"); // 0
    Serial.print("\n");
    */

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

void loop() {
    // read raw accel/gyro measurements from device
    accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
```

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

    #ifdef OUTPUT_READABLE_ACCELGYRO
        // display tab-separated accel/gyro x/y/z values
        Serial.print("a/g:\t");
        Serial.print(ax); Serial.print("\t");
        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.println(gz);
    #endif

    #ifdef OUTPUT_BINARY_ACCELGYRO
        Serial.write((uint8_t)(ax >> 8)); Serial.write((uint8_t)(ax & 0xFF));
        Serial.write((uint8_t)(ay >> 8)); Serial.write((uint8_t)(ay & 0xFF));
        Serial.write((uint8_t)(az >> 8)); Serial.write((uint8_t)(az & 0xFF));
        Serial.write((uint8_t)(gx >> 8)); Serial.write((uint8_t)(gx & 0xFF));
        Serial.write((uint8_t)(gy >> 8)); Serial.write((uint8_t)(gy & 0xFF));
        Serial.write((uint8_t)(gz >> 8)); Serial.write((uint8_t)(gz & 0xFF));
    #endif

    // blink LED to indicate activity
    blinkState = !blinkState;
    digitalWrite(LED_PIN, blinkState);
    delay(100);
}
```

More Documents