Introduction

The InvenSense MPU-6050 sensor contains a MEMS accelerometer and a MEMS gyro in a single chip. It is very accurate, as it contains 16-bits analog to digital conversion hardware for each channel. Therefor it captures the x, y, and z channel at the same time. The sensor uses the I2C-bus to interface with the Arduino.

The MPU-6050 is not expensive, especially given the fact that it combines both an accelerometer and a gyro.

Photo: GY-521 breakout board

Also note that Invensense has combined the MPU-6050 with a magnetometer (compass) in a single chip called MPU-9150.

Reading raw values is easy, the rest is not

Reading the raw values for the accelerometer and gyro is easy. The sleep mode has to be disabled, and then the registers for the accelerometer and gyro can be read.

But the sensor also contains a 1024 byte FIFO buffer. The sensor values can be programmed to be placed in the FIFO buffer. And the buffer can be read by the Arduino.

The FIFO buffer is used together with the interrupt signal. If the MPU-6050 places data in the FIFO buffer, it signals the Arduino with the interrupt signal so the Arduino knows that there is data in the FIFO buffer waiting to be read.

A little more complicated is the ability to control a second I2C-device.
The MPU-6050 always acts as a slave to the Arduino with the SDA and SCL pins connected to the I2C-bus.
But beside the normal I2C-bus, it has it's own I2C controller to be a master on a second (sub)-I2C-bus. It uses the pins AUX_DA and AUX_CL for that second (sub)-I2C-bus.

It can control, for example, a magnetometer. The values of the magnetometer can be passed on to the Arduino.

Things get really complex with the "DMP".
The sensor has a "Digital Motion Processor" (DMP), also called a "Digital Motion Processing Unit". This DMP can be programmed with firmware and is able to do complex calculations with the sensor values.
For this DMP, InvenSense has a discouragement policy, by not supplying enough information how to program the DMP. However, some have used reverse engineering to capture firmware.

The DMP ("Digital Motion Processor") can do fast calculations directly on the chip. This reduces the load for the microcontroller (like the Arduino). The DMP is even able to do calculations with the sensor values of another chip, for example a magnetometer connected to the second (sub)-I2C-bus.

Code

The accelerometer and gyro values are called the "raw" values. This is just as with other accelerometer and gyro sensors. A more sophisticated application is using the DMP to retrieve specific computed values from the sensor.

The Short example sketch on this page is a very short sketch that shows all the raw values. Click "Get code" at right, below the sketch, and copy it into a sketch.

The Example sketch (base code) on this page is also just showing the raw values, but it is an attempt to be a complete base for more functions.

For serious use of the MPU-6050, Jeff Rowberg has done an excellent job.
See his I2C lib: http://www.i2cdevlib.com/devices/mpu6050
Or the latest code on GitHub:
https://github.com/jrowberg/i2cdevlib/tree/master/Arduino/MPU6050

The FreeIMU library includes the MPU-6050 code from Jeff Rowberg.
The FreeIMU library: http://www.varesano.net/projects/hardware/FreeIMU

To start with the MPU-6050, see the page of InvenSense:
http://www.invensense.com/mems/gyro/mpu6050.html

For other programs and sensors, see the Degrees Of Freedom, 6DOF, 9DOF, 10DOF, 11DOF-section in the Playground index.

Multiple sensors

The pin "AD0" selects between I2C address 0x68 and 0x69. That makes it possible to have two of these sensors in a project. Most breakout boards have a pullup or pulldown resistor to make AD0 default low or high. Connect AD0 to GND or 3.3V for the other I2C address.

When more MPU-6050 sensors are needed in a project, the I2C-bus can be extended with multiplexers. However, someone in the forum mentioned a nice trick:

Trick
Using more than two MPU-6050 sensors can be achieved by connecting each of the AD0 pins to a separate output of the Arduino. If the number of pins is a problem, then a shift register or a port expander can be used.

The output of a 5V Arduino can not be used. In that case a voltage divider or level shifter for 3.3 volts on each of the outputs is needed. The 5V output pins can also be converted in 3.3V open collector outputs by using transistors or an open-collector driver. Pullup resistors to 3.3V should be added for a high level of 3.3V.

Suppose all AD0 lines are default high (3.3V), so every MPU-6050 is I2C address 0x69. That I2C address is however never used ! The Arduino makes one of the AD0 lines low, and uses that sensor at I2C address 0x68. After that is finished, the Arduino selects another AD0 line, and can use that sensor.
So every sensor is used at I2C address 0x68 (one by one) and 0x69 is never used.

This should make it possible to have many MPU-6050 sensors in a project. Even more than 10 sensor should be possible.
Note that requesting data from many MPU-6050 sensors is slow, because the I2C-bus is slow. A sensor with SPI interface is faster.

At this moment (15 July 2014) it is not known if this trick works for the MPU-6050.

Update March 25 2016: This trick is confirmed working nicely using three GY-521 breakouts along with an ESP8266 wifi/microcontroller (NodeMCU).

Breakout boards

There are a number of "breakout boards" or "sensor boards" with the MPU-6050. The price dropped fast, only 2.60 dollars in August 2014 on Ebay.

Sparkfun SEN-11028
http://www.sparkfun.com/products/11028
With schematic and full information. This breakout board must be used with 3.3V. There is no voltage regulator and no I2C-level shifter on the board. The pull-up resistors for the I2C-bus are 10k.

Drotek IMU 10DOF - MPU6050 + HMC5883 + MS5611

http://www.drotek.fr/shop/en/62-imu-10dof-mpu6050-hmc5883-ms5611.html

This sensor board contains three sensors. A schematic is not provided. The interrupt ('INT') of the MPU-6050 is not made available. Therefor the FIFO and the Jeff Rowberg library can not be used.

Drotek MPU-6050 Invensense

http://www.drotek.fr/shop/en/42-mpu6050-gyro-accelerometer.html

This breakout board contains a voltage regulator. It can be used with 3.3V and with 5V. A schematic is not provided.

GY-521

This sensor board has a voltage regulator. When using 3.3V to the VCC the resulting voltage (after the onboard voltage regulator) might be too low for a good working I2C bus. It is preferred to apply 5V to the VCC pin of the sensor board. The board has pull-up resistors on the I2C-bus. The value of those pull-up resistors are sometimes 10k and sometimes 2k2. The 2k2 is rather low. If it is combined with other sensor board which have also pull-up resistors, the total pull-up impedance might be too low.
Some GY-521 modules have the wrong capacitor (or a bad capacitor) and that results into a high noise level : http://forum.arduino.cc/index.php?topic=394691.0
This schematic is hard to find, so here is a copy: http://playground.arduino.cc/uploads/Main/MPU6050-V1-SCH.jpg
This part is designed in Fritzing: http://fritzing.org/projects/mpu-6050-board-gy-521-acelerometro-y-giroscopio

A data visualiser that makes life easier when starting out. Also includes an extended version (By http://www.geekmomprojects.com/) of Kordal's code. On GitHub https://github.com/janaka/Gy521-Dev-Kit

GY-52

This sensor board has a voltage regulator. So it can be used with 3.3V and with 5V. The board was originally designed for the MPU-3050, therefor the text "MPU-3050" or "MPU-X050" is sometimes printed on the board. The pull-up resistors are sometimes 10k and sometimes 4k7. If they are 10k, two extra pull-up resistors of 10k to the 3.3V could be added (from the SDA and SCL to the 3.3V of the sensor board or the 3.3V of the Arduino).

Flyduino MPU6050 Break Out onboard 3.3V reg

http://flyduino.net/MPU6050-Break-Out-onboard-33V-reg_1

This sensor board contains a voltage regulator, so it can also be used with 5V. The pull-up resistors of the I2C-bus are 4k7. It is actually a GY-52 breakout board.

Flyduino 10DOF IMU GY-86 MPU6050+HMC5883l+MS5611

A sensor board with the MPU-6050 and a magnetometer and barometer. The sensor board contains a voltage regulator, so it can be used with 5V. There seems to be also a level shifter on the board for the I2C-bus. The pull-up resistors for the I2C-bus seems to be 2k2, which is rather low.

no name breakout board

In august 2012 the cheapest breakout board with the MPU-6050 was a breakout board (for about 12 dollars/ 10euros) without any name or code.

The header is on the right with the pins in this order: "5V", "3V3", "GND", "SCL", "SDA", "INT", "SYNC", "CLK", "ASCL", "ADSA".
There are two pull-up resistors for the SCL and SDA, but the value is unknown. On the back are three solder jumpers, one of them for AD0.

no name breakout board 2

In 2014 a new MPU-6050 appeared without any name or code.
http://www.i2cdevlib.com/forums/topic/8-mpu6050-connection-failed/?p=347
The header is on the left with the pins in this order: "VCC", "GND", "SCL", "SDA", "XDA", "XCL", "AD0", "INT".
It is almost equal to the GY-521 board. There is a voltage regulator on the board for 3.3V. There are two 10k pull-up resistors for the SCL and SDA, and also 330 ohm resistors in the SCL and SDA signal. Because of the voltage regulator, connect 5V to the VCC of this sensor board.

Measurements

The raw values raises questions in the forums, since the raw values might seem unstable. Below are the raw values of the sensor that I measured, so you can compare them with your own raw values.

The raw values changes a lot due to a number of reasons. The default sensitivity is high, and the sensor returns 16 bits, but the actual valid number of bits is less than 16 bits. Since they are 16 bits, a variation of 50 is just a very small variation.

The next measurement were done in these conditions:

- The sensor was placed as horizontal as possible.
- It was placed on concrete, not a wooden table.
- During the measurements, there was no traffic in the street.
- An battery of 12V was used, not the less stable voltage from the USB bus. I used a battery instead of an adapter to avoid any mains noise.
- The circuit was on for 15 minutes, to stabilize any temperature influence.
- The room temperature was 25 degrees Celsius.

```
MPU-6050
Read accel, temp and gyro, error = 0
accel x,y,z: 184, -484, 14992
temperature: 29.635 degrees Celsius
gyro x,y,z : 367, 220, -812,

MPU-6050
Read accel, temp and gyro, error = 0
accel x,y,z: 116, -364, 15056
temperature: 29.635 degrees Celsius
gyro x,y,z : 373, 226, -766,

MPU-6050
Read accel, temp and gyro, error = 0
```

```
accel x,y,z: 232, -432, 15100
temperature: 29.682 degrees Celsius
gyro x,y,z : 382, 232, -790,

MPU-6050
Read accel, temp and gyro, error = 0
accel x,y,z: 280, -468, 15136
temperature: 29.635 degrees Celsius
gyro x,y,z : 368, 211, -820,

MPU-6050
Read accel, temp and gyro, error = 0
accel x,y,z: 140, -432, 15108
temperature: 29.588 degrees Celsius
gyro x,y,z : 388, 203, -806,

MPU-6050
Read accel, temp and gyro, error = 0
accel x,y,z: 220, -464, 14920
temperature: 29.541 degrees Celsius
gyro x,y,z : 374, 196, -774,

MPU-6050
Read accel, temp and gyro, error = 0
accel x,y,z: 172, -440, 15100
temperature: 29.588 degrees Celsius
gyro x,y,z : 363, 200, -769,
```

Short example sketch

The short example sketch is a very short sketch and it shows all the raw values (accelerometer, gyro and temperature). It should work on Arduino Uno, Nano, Leonardo, and also Due.

```
1.  // MPU-6050 Short Example Sketch
2.  // By Arduino User JohnChi
3.  // August 17, 2014
4.  // Public Domain
5.  #include<Wire.h>
6.  const int MPU_addr=0x68;  // I2C address of the MPU-6050
7.  int16_t AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ;
8.  void setup(){
9.    Wire.begin();
10.   Wire.beginTransmission(MPU_addr);
11.   Wire.write(0x6B);  // PWR_MGMT_1 register
12.   Wire.write(0);     // set to zero (wakes up the MPU-6050)
13.   Wire.endTransmission(true);
14.   Serial.begin(9600);
15. }
16. void loop(){
17.   Wire.beginTransmission(MPU_addr);
18.   Wire.write(0x3B);  // starting with register 0x3B (ACCEL_XOUT_H)
19.   Wire.endTransmission(false);
```

```
  Wire.requestFrom(MPU_addr,14,true);  // request a total of 14 registers
  AcX=Wire.read()<<8|Wire.read();  // 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
  AcY=Wire.read()<<8|Wire.read();  // 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
  AcZ=Wire.read()<<8|Wire.read();  // 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
  Tmp=Wire.read()<<8|Wire.read();  // 0x41 (TEMP_OUT_H) & 0x42 (TEMP_OUT_L)
  GyX=Wire.read()<<8|Wire.read();  // 0x43 (GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L)
  GyY=Wire.read()<<8|Wire.read();  // 0x45 (GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)
  GyZ=Wire.read()<<8|Wire.read();  // 0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)
  Serial.print("AcX = "); Serial.print(AcX);
  Serial.print(" | AcY = "); Serial.print(AcY);
  Serial.print(" | AcZ = "); Serial.print(AcZ);
  Serial.print(" | Tmp = "); Serial.print(Tmp/340.00+36.53);  //equation for temperature in
    degrees C from datasheet
  Serial.print(" | GyX = "); Serial.print(GyX);
  Serial.print(" | GyY = "); Serial.print(GyY);
  Serial.print(" | GyZ = "); Serial.println(GyZ);
  delay(333);
}
```