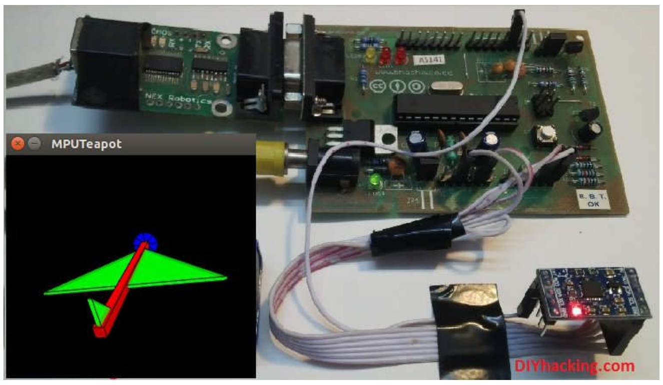
IMU Interfacing Tutorial: Get started with Arduino and the MPU 6050 Sensor!

By Arvind Sanjeev, Founder of DIY Hacking



Arduino MPU 6050 Setup

In this post, I will be reviewing a few basic IMU (Inertia Measurement Unit) sensors that are compatible Arduino. I will also give a short tutorial for interfacing an Arduino with the best IMU sensor available. IMU sensors like the MPU 6050 are used in self-balancing robots, UAVs, smartphones, and more.

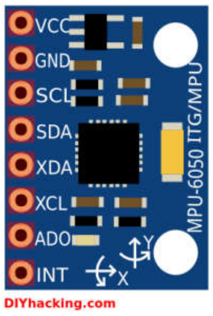
IMU sensors are one of the most common types of sensors used today in all kinds of electronic gadgets. IMU sensors help us in getting the attitude of an object, attached to the sensor in three-dimensional space. These values are usually in angles to help us to determine its attitude. They are used in smartphones to detect their orientation or in wearable gadgets like the Fit Bit, which use IMU sensors to track movement.

IMU sensors have a prolific number of applications. It is even considered to be an inexorable component in quadcopters. Some of the sensors I was able to get my hands on were:

1. ADXL 345 accelerometer.
2. ITG 3200 gyroscope.
3. Sparkfun 6 DOF IMU sensor board.
4. MPU 6050.

I was able to work with both accelerometers and gyroscopes separately. However, they are not as accurate alone as when they are combined. Among the lot, I found the Invensense MPU 6050 to be the most reliable and accurate IMU sensor. Apart from being significantly cheaper than the other sensors, the MPU 6050 performs better too.

In this tutorial, I will give you a basic introduction to the MPU 6050, demonstrate how it can be interfaced to an Arduino, and show you how to make a 3D model using the data from your Arduino MPU 6050.



Arduino MPU 6050 Pin out

Required Materials

Hardware:

1. Arduino or an Arduino clone board (freeduino). Or make your own custom Arduino board with this tutorial.
2. MPU 6050 sensor.
3. Interconnecting wires.

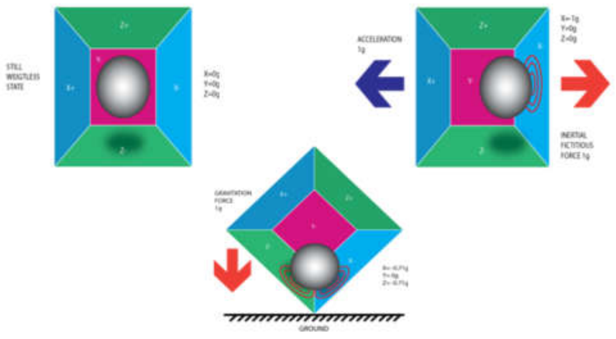
Software:

1. Arduino IDE: Arduino
2. Processing IDE: Processing (optional)

How Does it Work?

IMU sensors usually consist of two or more parts. Listing them by priority, they are the accelerometer, gyroscope, magnetometer, and altimeter. The MPU 6050 is a 6 DOF (Degrees of Freedom) or a six-axis IMU sensor, which means that it gives six values as output. Three values from the accelerometer and three from the gyroscope. The MPU 6050 is a sensor based on MEMS (Micro Electro Mechanical Systems) technology. Both the accelerometer and the gyroscope are embedded inside a single chip. This chip uses I2C (Inter Integrated Circuit) protocol for communication.

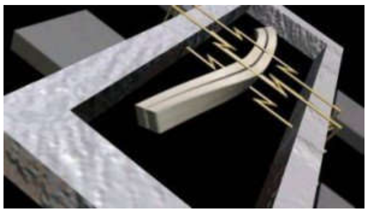
How Does an Accelerometer Work?



Piezo Electric Accelerometer

An accelerometer works on the principle of the piezo electric effect. Here, imagine a cuboidal box with a small ball inside it, like in the picture above. The walls of this box are made with piezo electric crystals. Whenever you tilt the box, the ball is forced to move in the direction of the inclination, due to gravity. The wall that the ball collides with creates tiny piezo electric currents. There are three pairs of opposite walls in a cuboid. Each pair corresponds to an axis in 3D space: X, Y, and Z axes. Depending on the current produced from the piezo electric walls, we can determine the direction of inclination and its magnitude. For more information check this.

How Does a Gyroscope Work?

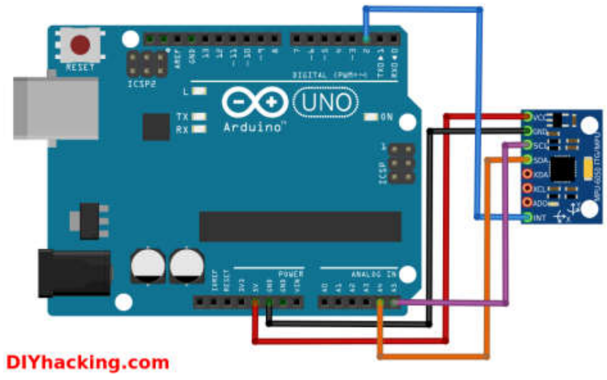


Piezo Electric Gyroscope

Gyroscopes work on the principle of Coriolis acceleration. Imagine that there is a fork like structure that is in a constant back and forth motion. It is held in place using piezo electric crystals. Whenever you try to tilt this arrangement, the crystals experience a force in the direction of inclination. This is caused as a result of the inertia of the moving fork. The crystals thus produce a current in consensus with the piezo electric effect, and this current is amplified. The values are then refined by the host microcontroller. Check this short video that explains how a MEMS gyroscope works.

Interfacing the Arduino MPU 6050

The MPU 6050 communicates with the Arduino through the I2C protocol. The MPU 6050 is connected to Arduino as shown in the following diagram. If your MPU 6050 module has a 5V pin, then you can connect it to your Arduino's 5V pin. If not, you will have to connect it to the 3.3V pin. Next, the GND of the Arduino is connected to the GND of the MPU 6050.



Arduino MPU 6050 connections

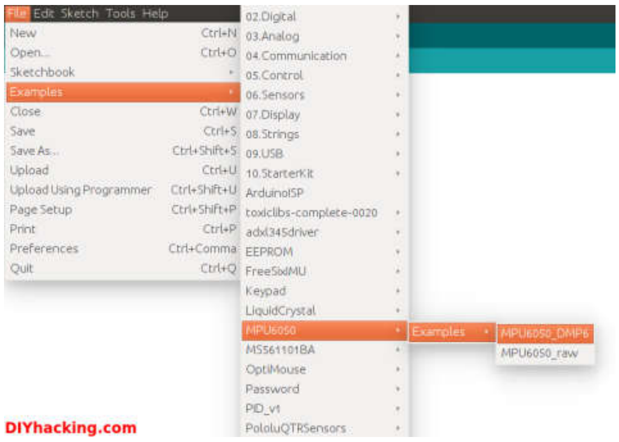
The program we will be running here, also takes advantage of the Arduino's interrupt pin. Connect your Arduino's digital pin 2 (interrupt pin 0) to the pin labeled as INT on the MPU 6050. Next, we need to set up the I2C lines. To do this, connect the pin labeled SDA on the MPU 6050 to the Arduino's analog pin 4 (SDA) and the pin labeled as SCL on the MPU 6050 to the Arduino's analog pin 5 (SCL). That's it, you have finished wiring up the Arduino

MPU 6050!

Uploading the Code and Testing the Arduino MPU 6050

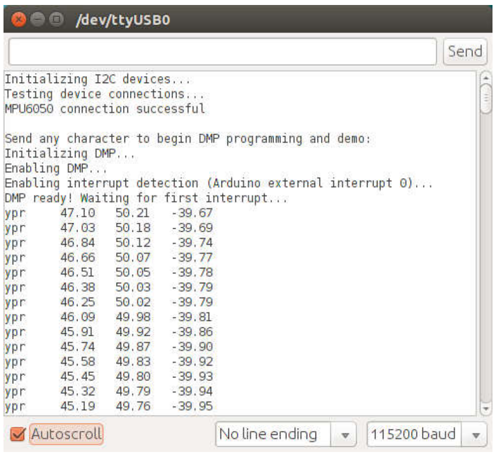
To test the Arduino MPU 6050, first download the Arduino library for MPU 6050, developed by Jeff Rowberg. You can find the library here. Next, you have to unzip/extract this library and take the folder named "MPU6050" and paste it inside the Arduino's "library" folder. To do this, go to the location where you have installed Arduino (Arduino –> libraries) and paste it inside the libraries folder. You might also have to do the same thing to install the I2Cdev library if you don't already have it for your Arduino. Do the same procedure as above to install it, you can find the file here: I2Cdev library.

If you have done this correctly, when you open the Arduino IDE, you can see "MPU6050" in File –> Examples. Next, open the example program from: File –> Examples –> MPU6050 –> Examples –> MPU6050_DMP6.



Arduino MPU 6050 DMP code

Next, you have to upload this code to your Arduino. After uploading the code, open up the serial monitor and set the baud rate as 115200. Next, check if you see stuff like "Initializing I2C devices…" on the serial monitor. If you don't, just press the reset button. Now, you'll see a line saying "Send any character to begin DMP programming and demo." Just type in any character on the serial monitor and send it and you should start seeing the yaw, pitch, and roll values coming in from the MPU 6050. Like so:



```
Initializing I2C devices...
Testing device connections...
MPU6050 connection successful

Send any character to begin DMP programming and demo:
Initializing DMP...
Enabling DMP...
Enabling interrupt detection (Arduino external interrupt 0)...
DMP ready! Waiting for first interrupt...
ypr     47.10   50.21   -39.67
ypr     47.03   50.18   -39.69
ypr     46.84   50.12   -39.74
ypr     46.66   50.07   -39.77
ypr     46.51   50.05   -39.78
ypr     46.38   50.03   -39.79
ypr     46.25   50.02   -39.79
ypr     46.09   49.98   -39.81
ypr     45.91   49.92   -39.86
ypr     45.74   49.87   -39.90
ypr     45.58   49.83   -39.92
ypr     45.45   49.80   -39.93
ypr     45.32   49.79   -39.94
ypr     45.19   49.76   -39.95
```

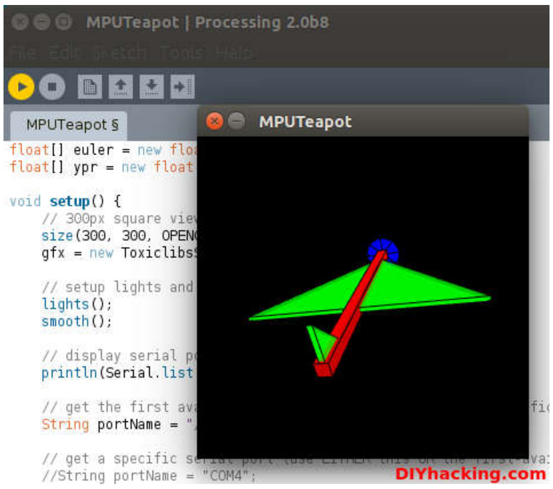
Arduino MPU 6050 Serial Monitor

DMP stands for Digital Motion Processing. The MPU 6050 has a built-in motion processor. It processes the values from the accelerometer and gyroscope to give us accurate 3D values.

Also, you will need to wait about 10 secs before you get accurate values from the Arduino MPU 6050. After which, the values will begin to stabilize. Just check out the video below to see if yours is working correctly.

Modeling the Values from the Arduino MPU 6050 in 3D Using Processing (Optional)

If you want to see the 3D model of the sensor, continue reading. To view the 3D representation of the data from the MPU 6050, you need to install the "processing" software, you can find it here: processing IDE. Processing is similar to Arduino, except for a couple of functions. Processing is mainly used for visualizing data and rendering it in 2D/3D models.



```
float[] euler = new flo
float[] ypr = new float

void setup() {
    // 300px square vie
    size(300, 300, OPEN
    gfx = new Toxiclibs

    // setup lights and
    lights();
    smooth();

    // display serial p
    println(Serial.list

    // get the first av                                    fi
    String portName = "

    // get a specific serial port (use EITHER this OR the first ava
    //String portName = "COM4";
```

Arduino MPU 6050 with Processing

After installing the processing IDE, next, you will need to download a library called "Toxi". This library is necessary for our Arduino MPU 6050 processing example. You can download the latest zip file for the library from here. Next, you need to extract this file and paste the folder named "toxiclibs-complete-0020" in the "libraries" folder's directory of processing. You can find the "libraries" folder inside the Sketchbook folder for processing. If you don't, then you will have to create a new folder called "libraries" there and paste the toxiclibs inside it.

To visualize the 3D model in processing, first you have to upload the Arduino code for MPU 6050 (MPU6050_DMP6). Before doing that, you need to comment the line in the Arduino MPU6050_DMP6 code which says:

#define OUTPUT_READABLE_YAWPITCHROLL by //#define OUTPUT_READABLE_YAWPITCHROLL.

And uncomment the line which says:

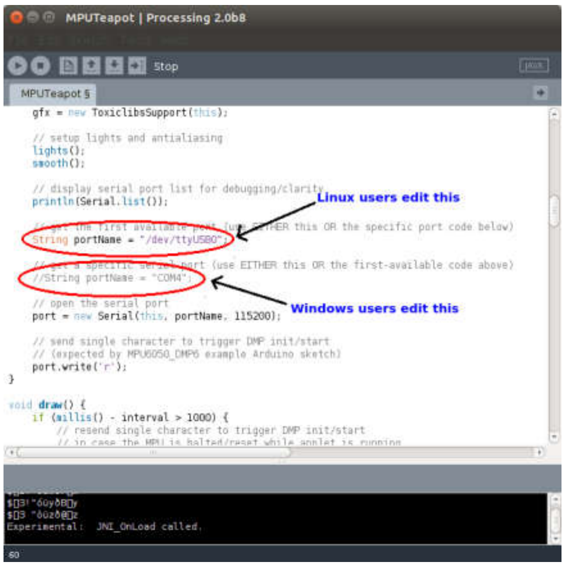
//#define OUTPUT_TEAPOT by #define OUTPUT_TEAPOT.

Next, you have to open the processing example for the MPU 6050. Open processing, then: File –> Open then navigate to the folder where you installed the MPU6050 library for Arduino. You can find the processing example in: MPU6050 –> Examples –> MPU6050_DMP6 –> Processing –> MPUTeapot.

In this code, you have to then check the serial port which is defined in it. By default the line that defines it for LinuxMmac users is:

String portName = "/dev/ttyUSB1";.

You need to change ttyUSB1 to the port where your Arduino is connected.



```
    gfx = new ToxiclibsSupport(this);

    // setup lights and antialiasing
    lights();
    smooth();

    // display serial port list for debugging/clarity
    println(Serial.list());

    // get the first available port (use EITHER this OR the specific port code below)
    String portName = "/dev/ttyUSB0";

    // get a specific serial port (use EITHER this OR the first-available code above)
    //String portName = "COM4";

    // open the serial port
    port = new Serial(this, portName, 115200);

    // send single character to trigger DMP init/start
    // (expected by MPU6050_DMP6 example Arduino sketch)
    port.write('r');
}

void draw() {
    if (millis() - interval > 1000) {
        // resend single character to trigger DMP init/start
        // in case the MPU is halted/reset while applet is running
```

```
$▯3!"ôûyðB▯y
$▯3 "ôûzð@▯z
Experimental:  JNI_OnLoad called.
```

Editing the Processing file for the MPU 6050

For windows users, you need to comment the line that says:

String portName = "/dev/ttyUSB1"; by //String portName = "/dev/ttyUSB1";.

And uncomment the line that says:

//String portName = "COM4"; by String portName = "COM4";.

And replace "COM4" with the COM port where your Arduino is connected (check this by going intoArduinoo and Tools –> Serial Port).

Now we can test the whole setup. First, upload the Arduino code (MPU6050_DMP6) through Arduino and remember NOT to open the serial monitor. Next, run the processing code (MPUTeapot) by pressing the button with the "play" symbol. You will see a small plane like object. Wait for about 10 seconds for the MPU 6050 values to stabilize. After which, you can see the 3D model of your MPU 6050, which moves in accordance with the sensor. Now, check out the demo for the Arduino MPU 6050 3D model!