EARTH PEOPLE TECHNOLOGY

EPT-M6050-BD-S8 Gyro/Accelerometer Breakout Board User Manual

This breakout board is based on the MPU-6050 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.

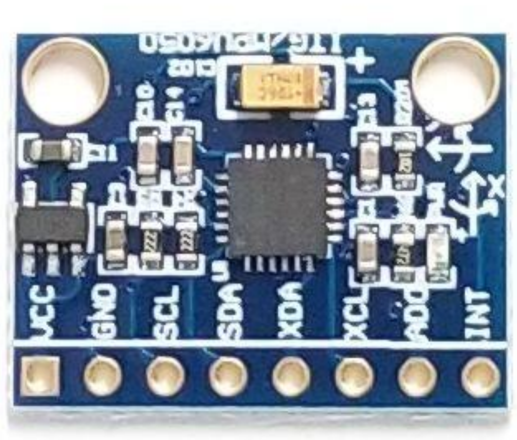
he MPU6050 has both a gyroscope and an accelerometer, using which we can measure rotation along all three axes, static acceleration due to gravity, as well as motion, shock, or dynamic acceleration due to vibration..

MPU-6050 Sensor

The mpu6050 sensor platform allows you to use your MPU6050 Accelerometer/Gyroscope sensors with any MCU including the Arduino platform. It is compatible with the +5V Arduino's and can also connect to +3.3V boards. Just wire the EPT-M6050-BD-S8 into the Arduino and load the code. The I²C Bus is required to be set up in your configuration for this sensor to work.

EARTHPEOPLE
Technology

MPU-6050 Breakout Board User Manual

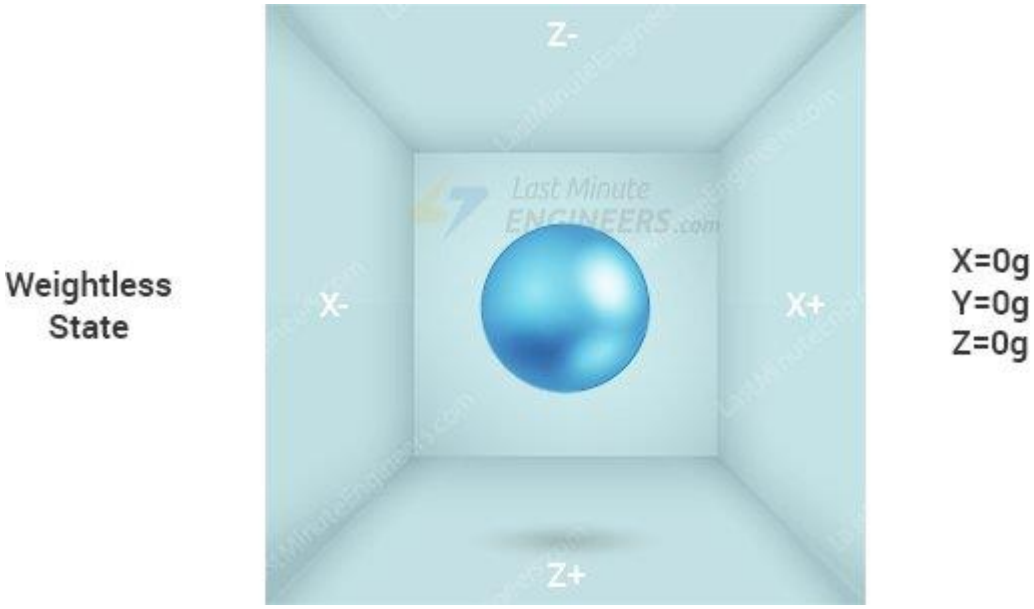


Hardware Features:

- Small form factor, 14 x 21mm(0.55x0.83")
- Compatible with +3.3V or +5V interface
- On board +3.3V regulator
- Power LED
- I2C address selection
- Simple common pinout with other Arduino style sensors

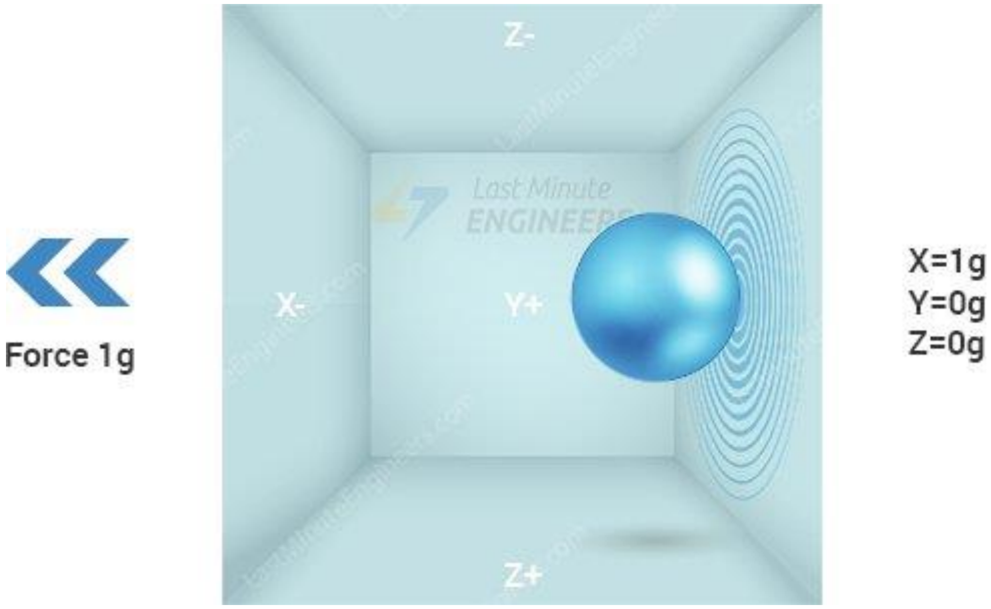
How Accelerometer Works?

To know how accelerometers work, it is often useful to imagine a ball inside a 3D cube.

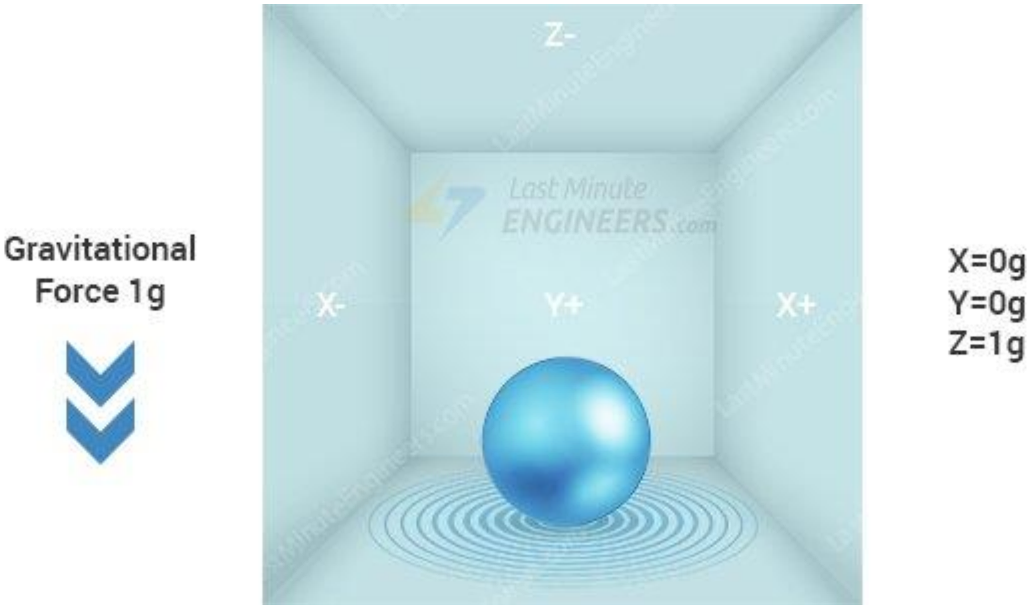


Suppose, the cube is in outer-space where everything is in weightless state, the ball will simply float in the middle of the cube.

Now let’s imagine each wall represents particular axis. If we suddenly move the box to the left with acceleration 1g(A single G-force 1g is equivalent to gravitational acceleration 9.8 m/s2), no doubt the ball will hit the wall X. If we measure the force that the ball applies to the wall X, we can get an output value of 1g on the X axis.



Let’s see what happens if we put that cube on Earth. The ball will simply fall on the wall Z and will apply a force of 1g, as shown in the picture below:

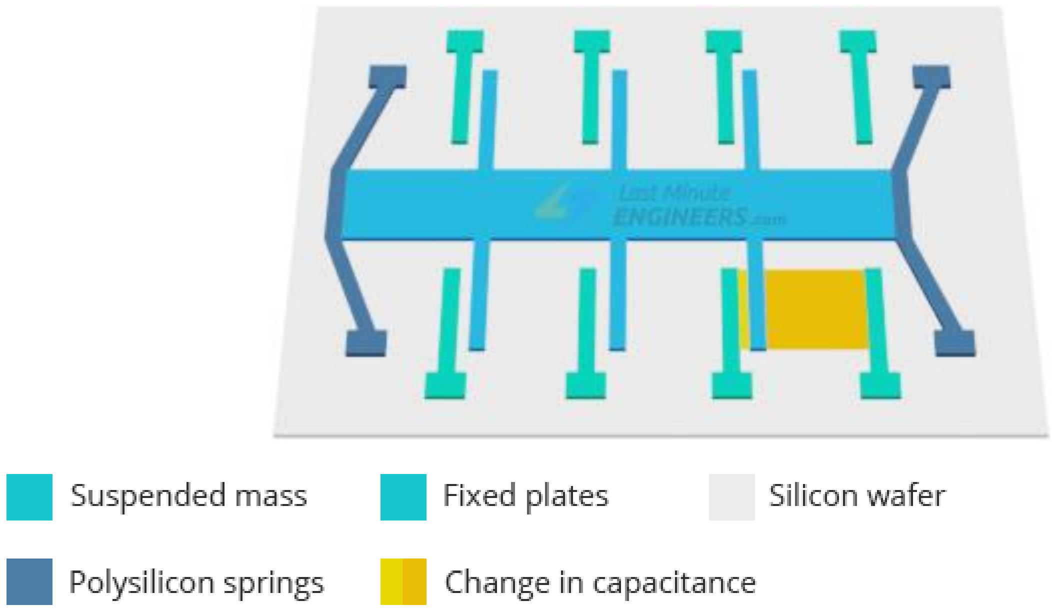


In this case the box isn’t moving but we still get a reading of 1g on the Z axis. This is because the gravitational force is pulling the ball down with force 1g.

The accelerometer measures the static acceleration of gravity in tilt-sensing applications as well as dynamic acceleration resulting from motion, shock, or vibration.

How MEMS Accelerometer Works?

MEMS(Micro Electro Mechanical Systems) accelerometer consists of a micro-machined structure built on top of a silicon wafer.



This structure is suspended by polysilicon springs. It allows the structure to deflect at the time when the acceleration is applied on the particular axis.

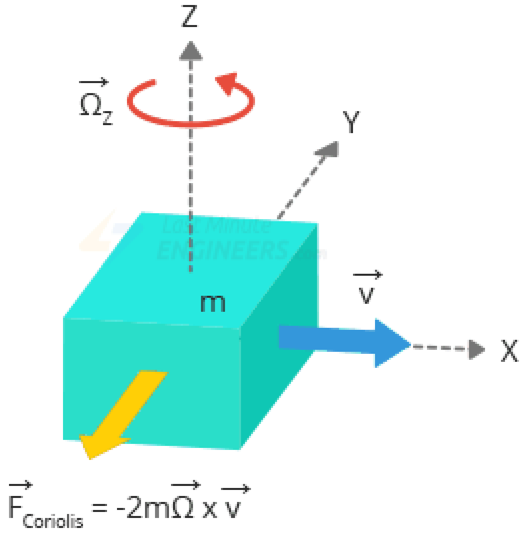
Due to deflection the capacitance between fixed plates and plates attached to the suspended structure is changed. This change in capacitance is proportional to the acceleration on that axis. The sensor processes this change in capacitance and converts it into an analog output voltage.

How Gyroscope Works?

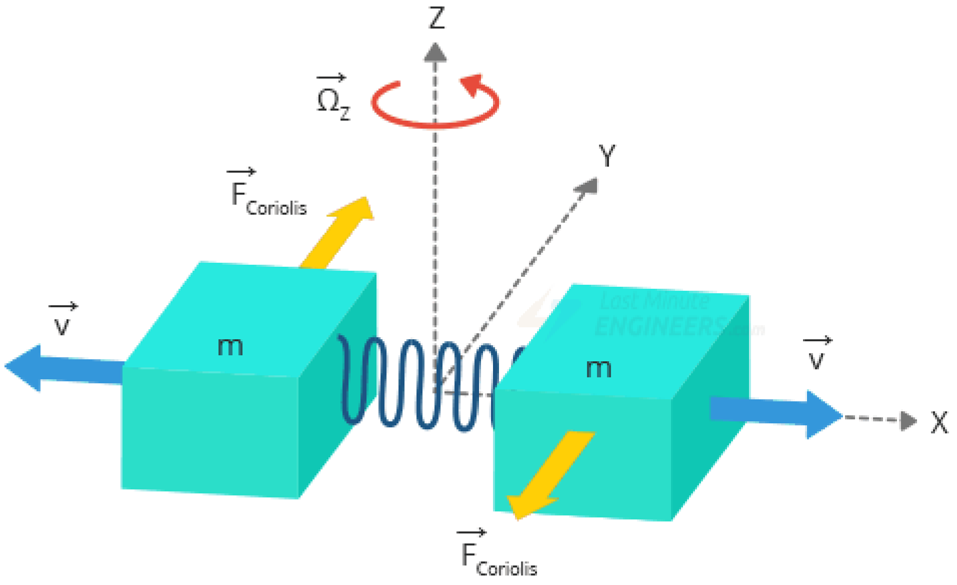
While accelerometers measure linear acceleration, MEMS gyroscopes measure angular rotation. To do this they measure the force generated by what is known as The Coriolis Effect.

Coriolis Effect

Coriolis Effect tells us that when a mass (m) moves in a particular direction with a velocity (v) and an external angular rate (Ω) is applied (Red arrow); the Coriolis Effect generates a force (Yellow arrow) that causes a perpendicular displacement of the mass. The value of this displacement is directly related to the angular rate applied.

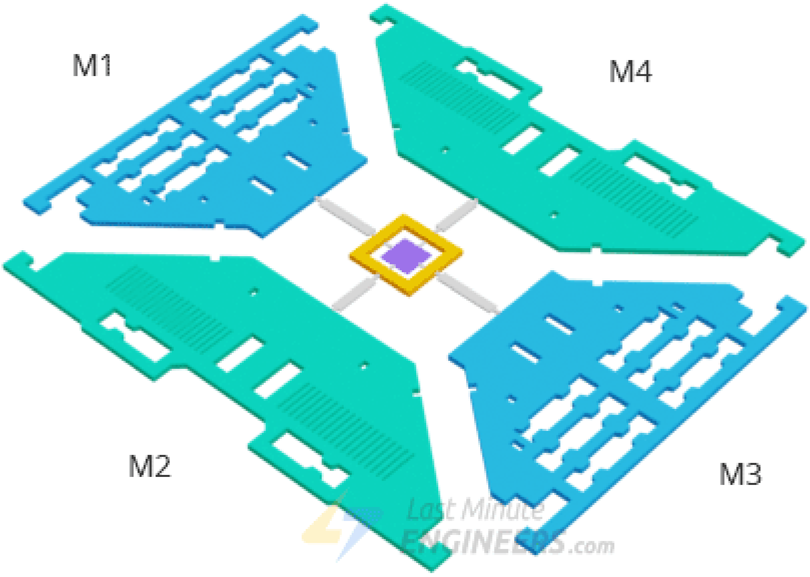


Now suppose that there are two masses that are kept in constant oscillating motion so that they move continuously in opposite directions. When angular rate is applied, the Coriolis effect on each mass is also in opposite directions, which results in a change in the capacitance between them; this change is sensed.



How MEMS Gyroscope Works?

The MEMS sensor is composed of a proof mass (containing 4 parts M1, M2, M3 and M4) which is kept in a continuously oscillating movement so that is reacts to the coriolis effect. They move inward and outward simultaneously in the horizontal plane.

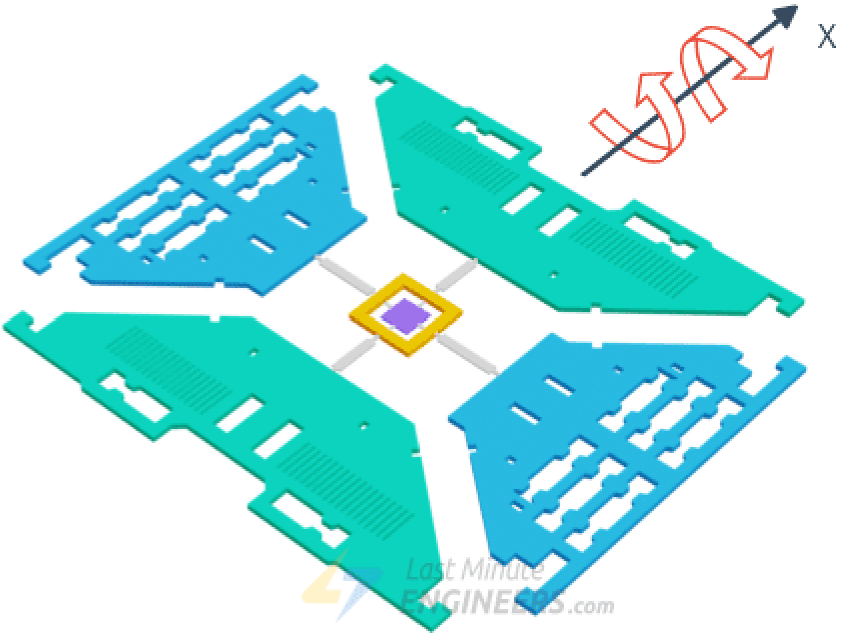


When we start rotating the structure, the Coriolis force acting on the moving proof mass changes the direction of the vibration from horizontal to vertical.

There are three modes depending on which axis the angular rotation is applied.

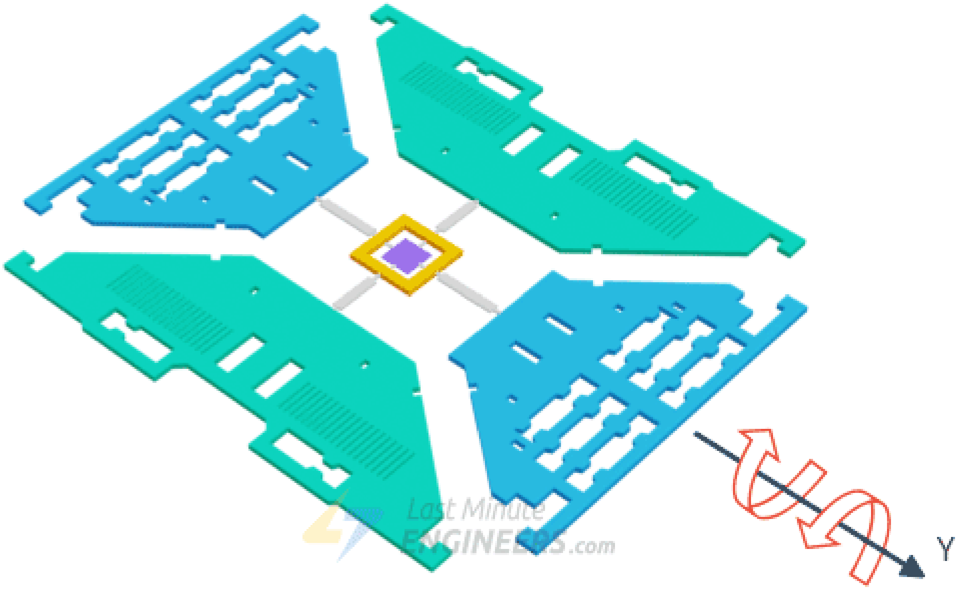
Roll Mode:

When an angular rate is applied along the X-axis, due to the coriolis effect, then M1 and M3 will move up and down out of the plane. This causes the roll angle to change hence it is called Roll Mode.



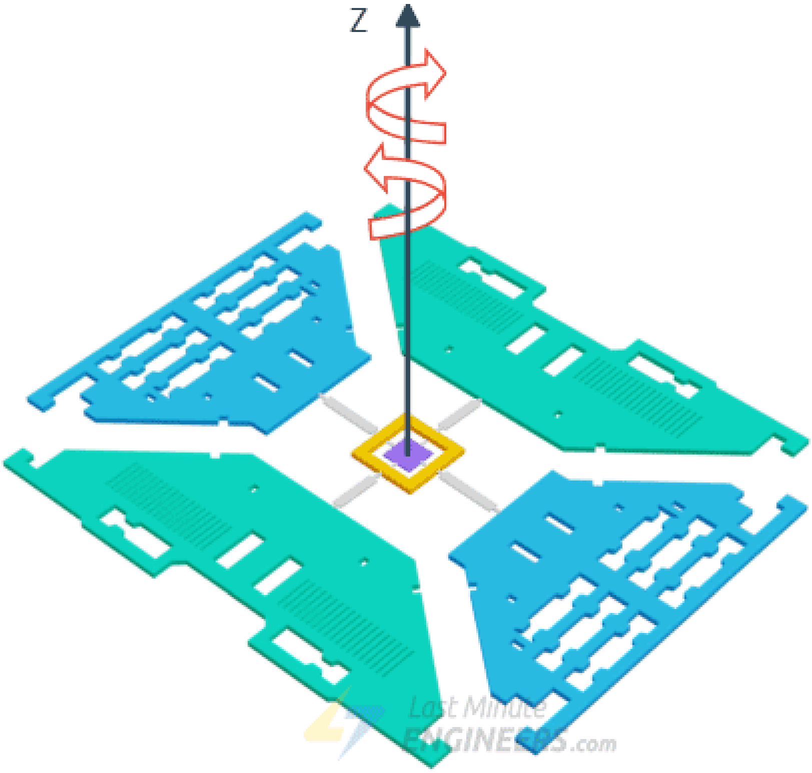
Pitch Mode:

When an angular rate is applied along the Y-axis, then M2 and M4 will move up and down. This causes the pitch angle to change hence it is called Pitch Mode.



Yaw Mode:

When an angular rate is applied along the Z-axis, M2 and M4 will move in the same horizontal plane in opposite directions. This causes the yaw angle to change hence it is called Yaw Mode.

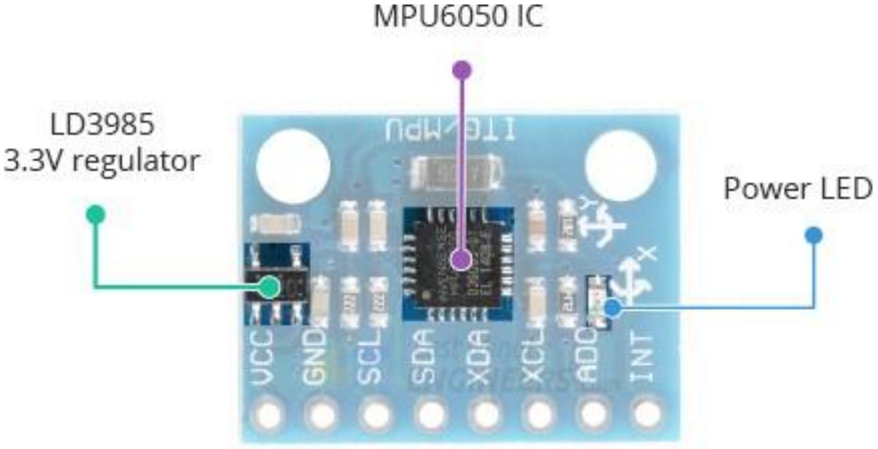


Whenever the coriolis effect is detected, the continuous movement of the driving mass will cause a capacitance change (ΔC) which is picked up by the sensing structure and then converted to a voltage signal.

MPU6050 Module Hardware Overview

At the heart of the module is a low power, inexpensive 6-axis MotionTracking chip that combines a 3-axis gyroscope, 3-axis accelerometer, and a Digital Motion Processor (DMP) all in a small 4mm x 4mm package.

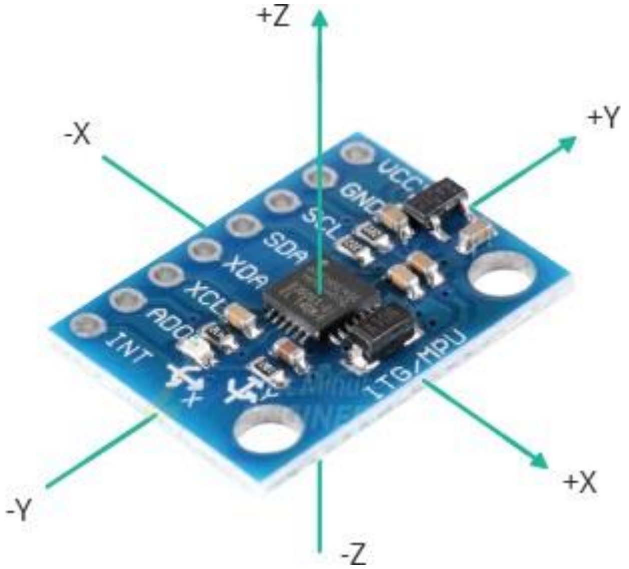
It can measure angular momentum or rotation along all the three axis, the static acceleration due to gravity, as well as dynamic acceleration resulting from motion, shock, or vibration.



The module comes with an on-board LD3985 3.3V regulator, so you can use it with a 5V logic microcontroller like Arduino. The MPU6050 consumes less than 3.6mA during measurements and only 5μA during idle. This low power consumption allows the implementation in battery driven devices. In addition, the module has a power LED that lights up when the module is powered.

Measuring Acceleration

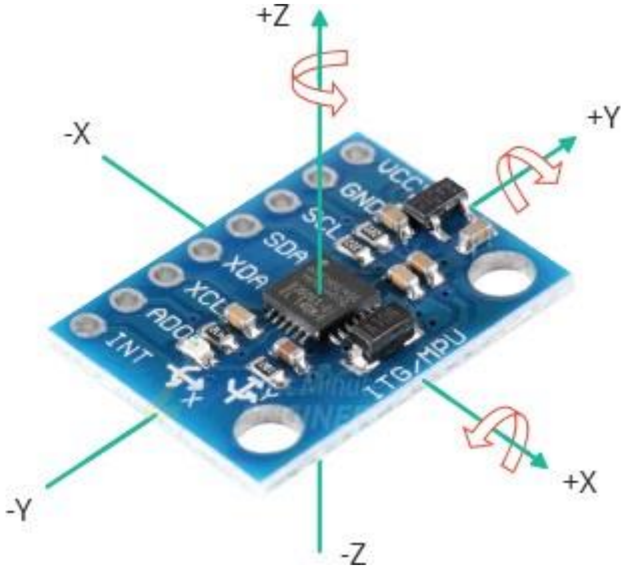
The MPU6050 can measure acceleration using its on-chip accelerometer with four programmable full scale ranges of ±2g, ±4g, ±8g and ±16g.



The MPU6050 has three 16-bit analog-to-digital converters that simultaneously sample the 3 axis of movement (along X, Y and Z axis).

Measuring Rotation

The MPU6050 can measure angular rotation using its on-chip gyroscope with four programmable full scale ranges of ±250°/s, ±500°/s, ±1000°/s and ±2000°/s.



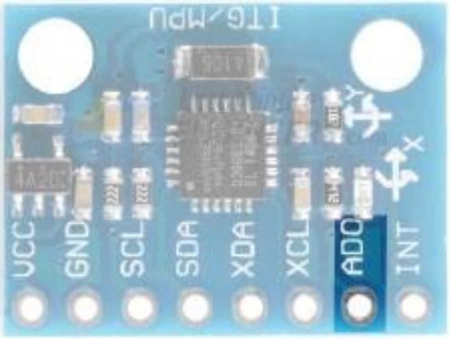
The MPU6050 has another three 16-bit analog-to-digital converters that simultaneously samples 3 axes of rotation (around X, Y and Z axis). The sampling rate can be adjusted from 3.9 to 8000 samples per second.

Measuring Temperature

The MPU6050 includes an embedded temperature sensor that can measure temperature over the range of -40 to 85°C with accuracy of ±1°C.
Note that this temperature measurement is of the silicon die itself and not the ambient temperature. Such measurements are commonly used to offset the calibration of accelerometer and gyroscope or to detect temperature changes rather than measuring absolute temperatures.

The I2C Interface

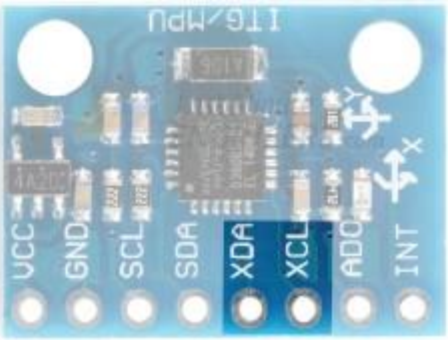
The module uses the I2C interface for communication with the Arduino. It supports two separate I2C addresses: 0x68HEX and 0x69HEX. This allows two MPU6050s to be used on the same bus or to avoid address conflicts with another device on the bus.



The ADO pin determines the I2C address of the module. This pin has a built-in 4.7K pull-down resistor. Therefore, when you leave the ADO pin unconnected, the default I2C address is 0x68HEX and when you connect it to 3.3V, the line is pulled HIGH and the I2C address becomes 0x69HEX.

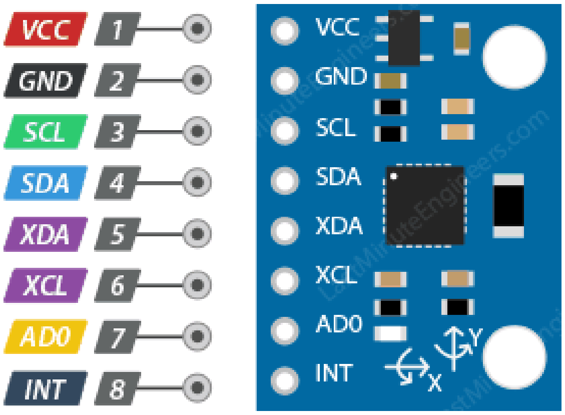
Adding External Sensors

To increase the level of accuracy even further, the MPU6050 module provides a feature for connecting external sensors. These external sensors are connected to the MPU6050 via a second I2C bus (XDA and XCL), which is completely independent of the main I2C bus.



MPU6050 Module Pinout

The pin descriptions of the MPU6050 module are as follows:



ACC is the power supply for the module. Connect it to the 5V output of the Arduino.
GND should be connected to the ground of Arduino.
SCL is a I2C Clock pin. This is a timing signal supplied by the Bus Master device. Connect to the SCL pin on the Arduino.

SDA is a I2C Data pin. This line is used for both transmit and receive. Connect to the SDA pin on the Arduino.
XDA is the external I2C data line. The external I2C bus is for connecting external sensors.
XCL is the external I2C clock line.
AD0 allows you to change the internal I2C address of the MPU6050 module. It can be used if the module is conflicting with another I2C device, or if you wish to use two MPU6050s on the same I2C bus. When you leave the ADO pin unconnected, the default I2C address is 0x68$_{\text{HEX}}$ and when you connect it to 3.3V, the I2C address becomes 0x69$_{\text{HEX}}$.
INT is the Interrupt Output. MPU6050 can be programmed to raise interrupt on gesture detection, panning, zooming, scrolling, tap detection, and shake detection.

Wiring MPU6050 Module with Arduino

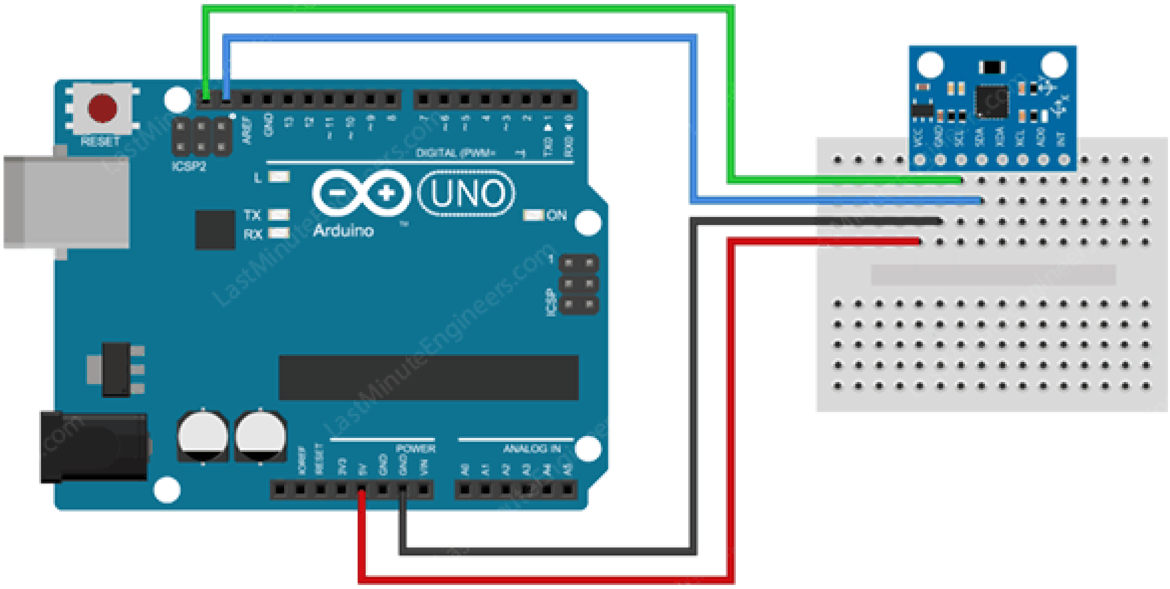
Connections are fairly simple. Start by connecting VCC pin to the 5V output on the Arduino and connect GND to ground.

Now we are remaining with the pins that are used for I2C communication. Note that each Arduino Board has different I2C pins which should be connected accordingly. On the Arduino boards with the R3 layout, the SDA (data line) and SCL (clock line) are on the pin headers close to the AREF pin. They are also known as A5 (SCL) and A4 (SDA).

If you are using a different Arduino board, please refer below table.

	SCL	SDA
Arduino Uno	A5	A4
Arduino Nano	A5	A4
Arduino Mega	21	20
Leonardo/Micro	3	2

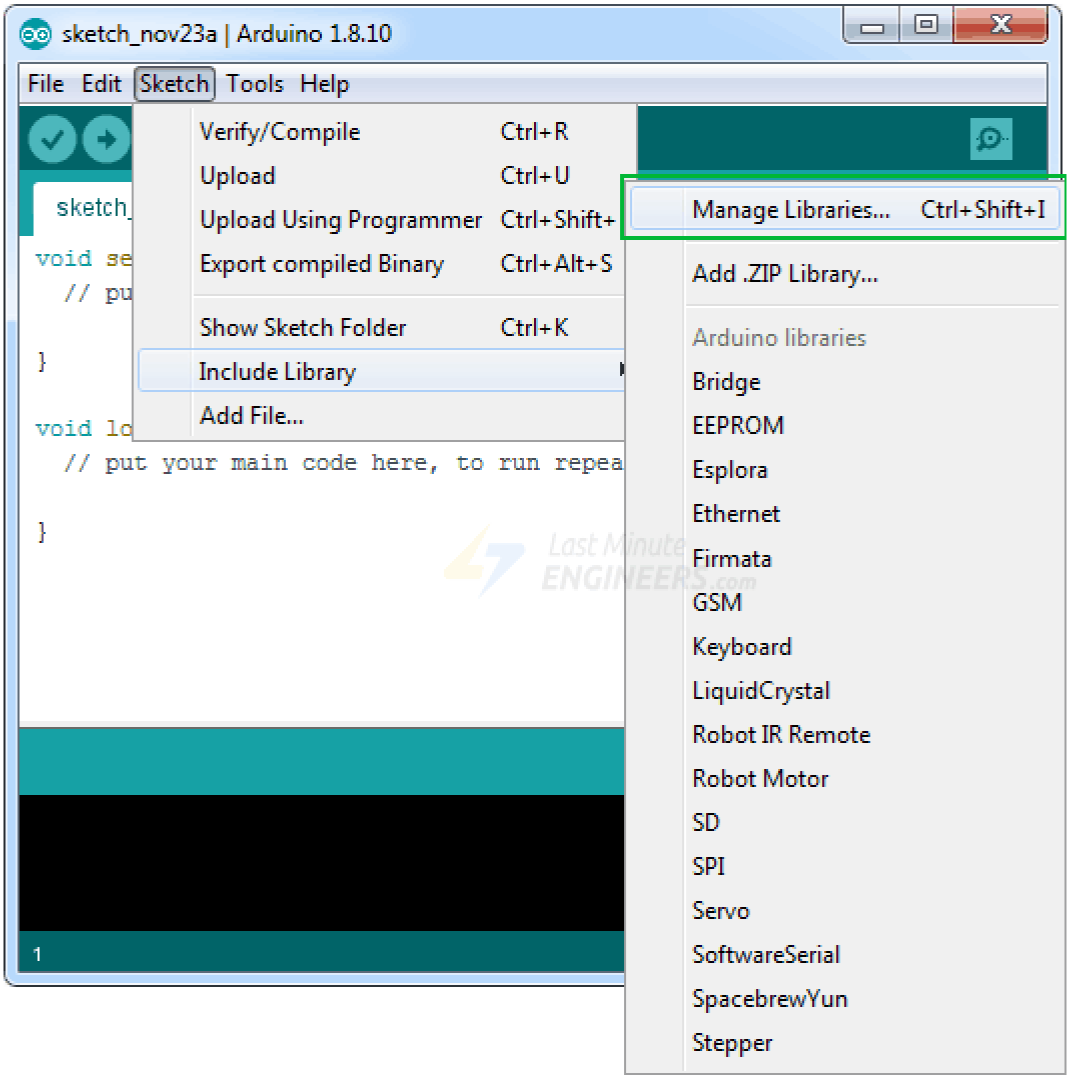
The following diagram shows you how to wire everything.



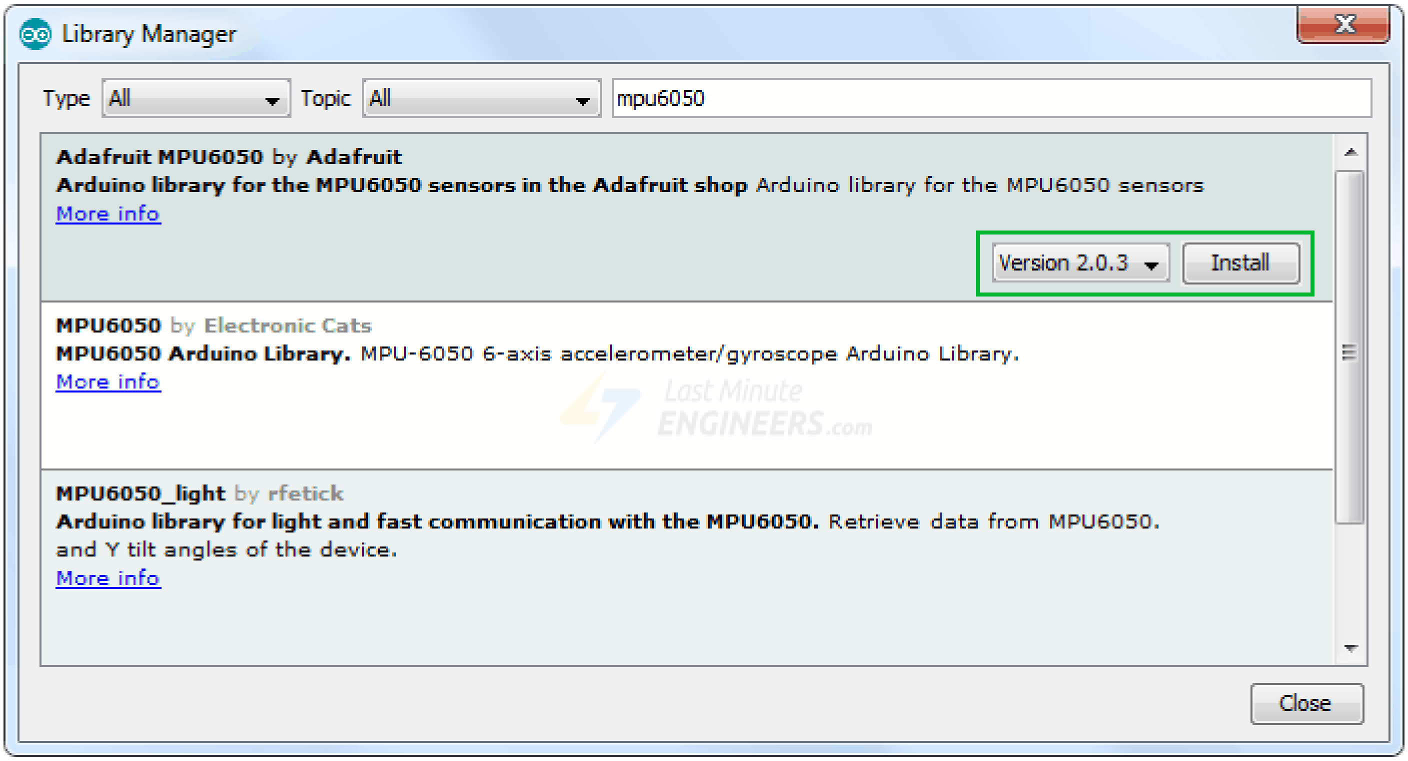
Library Installation

The MPU6050 module is relatively easy to get up and running and capturing the raw data output of the device. Manipulating the data into something meaningful, however, is more of a challenge, but there are some libraries available for using the device.

To install the library navigate to the Sketch > Include Library > Manage Libraries… Wait for Library Manager to download libraries index and update list of installed libraries.



Filter your search by typing ‘mpu6050’. There should be a couple entries. Look for Adafruit MPU6050 Library by Adafruit. Click on that entry, and then select Install.



Library Manager

Type: All | Topic: All | mpu6050

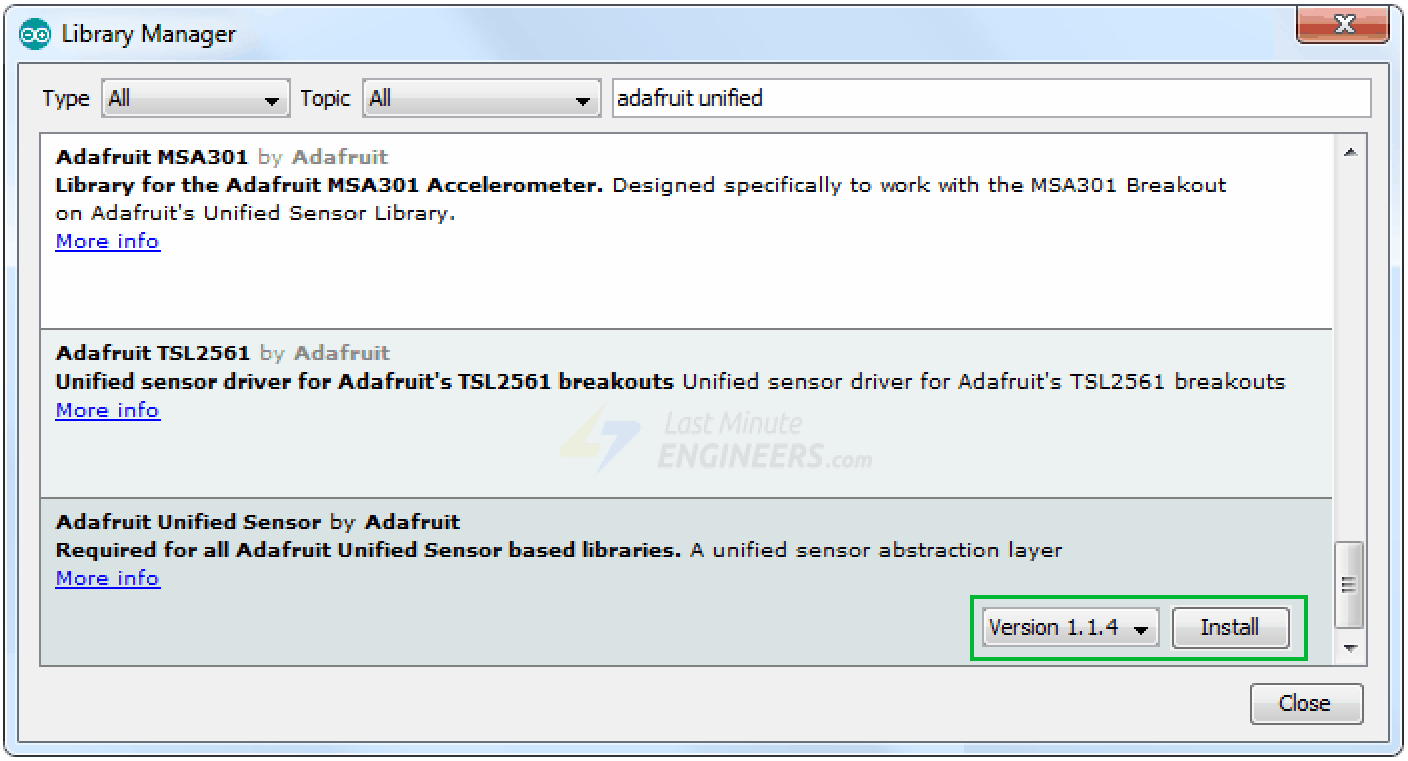
Adafruit MPU6050 by **Adafruit**
Arduino library for the MPU6050 sensors in the Adafruit shop Arduino library for the MPU6050 sensors
More info

Version 2.0.3 | Install

MPU6050 by Electronic Cats
MPU6050 Arduino Library. MPU-6050 6-axis accelerometer/gyroscope Arduino Library.
More info

MPU6050_light by rfetick
Arduino library for light and fast communication with the MPU6050. Retrieve data from MPU6050.
and Y tilt angles of the device.
More info

The Adafruit MPU6050 library uses Adafruit Unified Sensor Driver and Adafruit Bus IO Library internally. Therefore, search the library manager for Adafruit Unified Sensor and BusIO and install them as well.

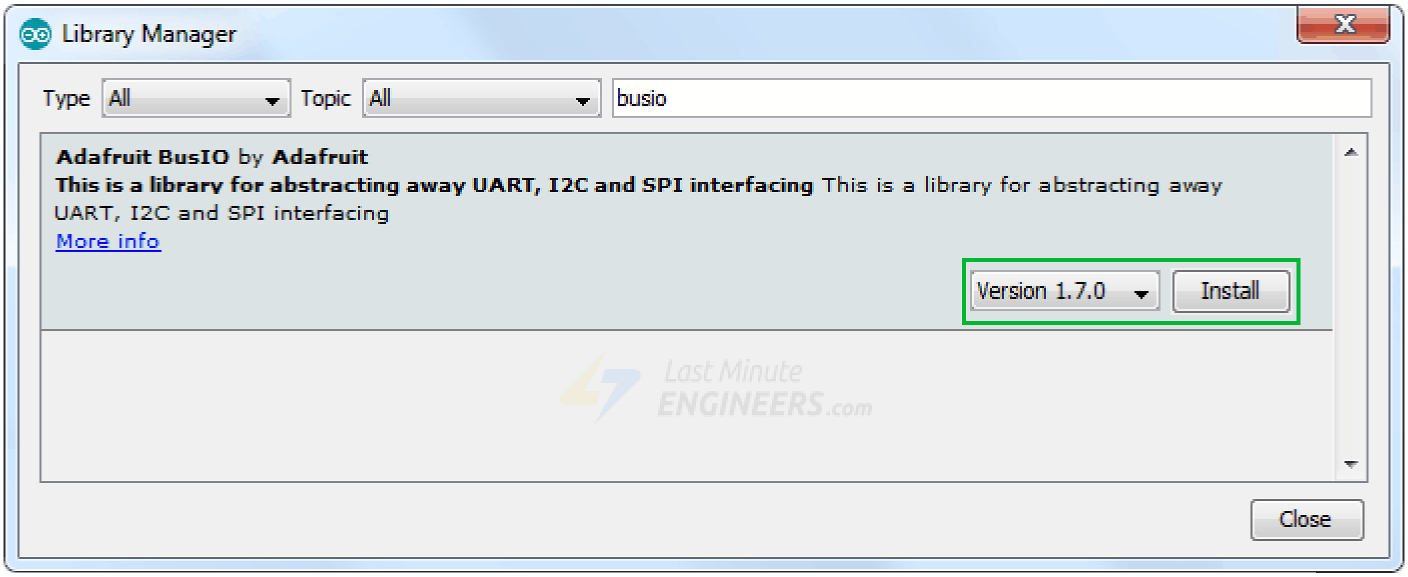


adafruit unified

Adafruit MSA301 by Adafruit
Library for the Adafruit MSA301 Accelerometer. Designed specifically to work with the MSA301 Breakout on Adafruit's Unified Sensor Library.
More info

Adafruit TSL2561 by Adafruit
Unified sensor driver for Adafruit's TSL2561 breakouts Unified sensor driver for Adafruit's TSL2561 breakouts
More info

Adafruit Unified Sensor by **Adafruit**
Required for all Adafruit Unified Sensor based libraries. A unified sensor abstraction layer
More info



Adafruit BusIO by **Adafruit**
This is a library for abstracting away UART, I2C and SPI interfacing This is a library for abstracting away UART, I2C and SPI interfacing

Arduino Code – Reading Accelerometer, Gyroscope and Temperature Data

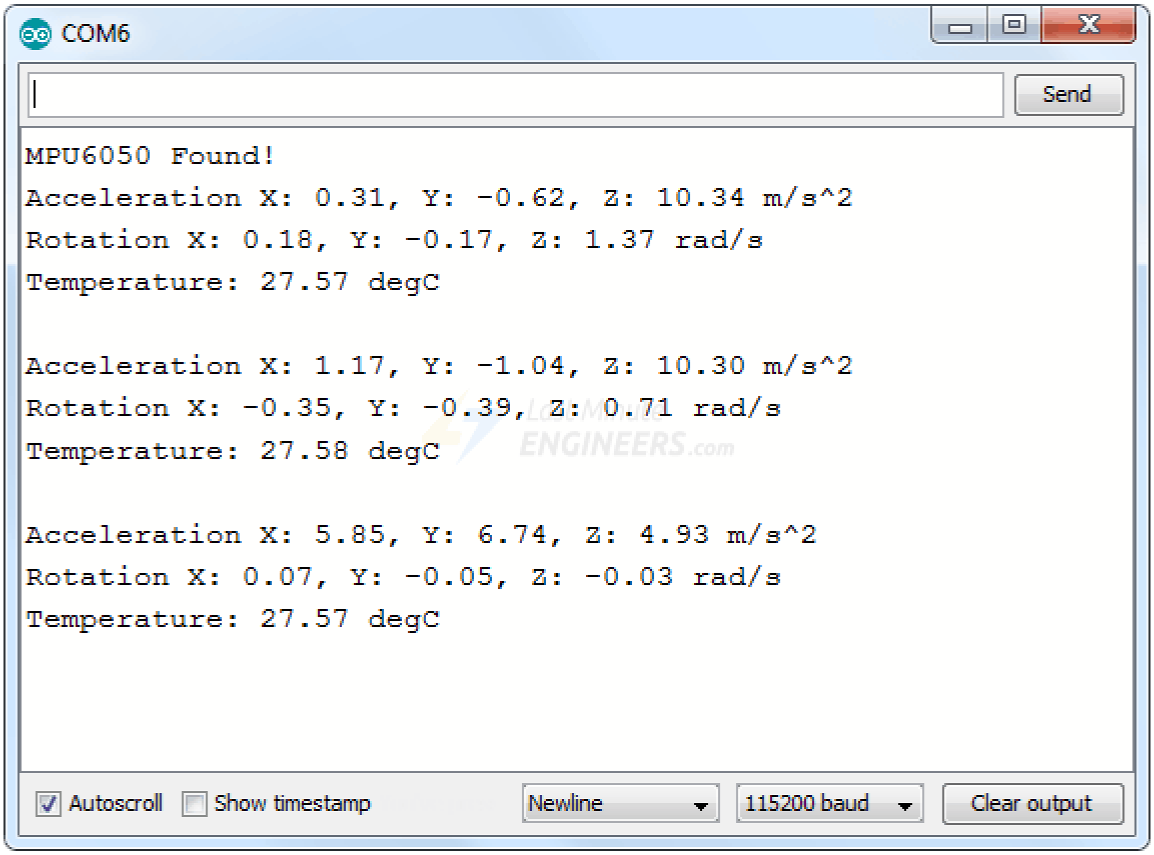
When you have everything hooked up try running the below sketch. It will give you a complete understanding on how to read linear acceleration, angular rotation & temperature from the MPU6050 module and can serve as the basis for more practical experiments and projects.

```
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Wire.h>

Adafruit_MPU6050 mpu;
```

```
void setup(void) {
        Serial.begin(115200);

        // Try to initialize!
        if (!mpu.begin()) {
                Serial.println("Failed to find MPU6050 chip");
                while (1) {
                  delay(10);
                }
        }
        Serial.println("MPU6050 Found!");

        // set accelerometer range to +-8G
        mpu.setAccelerometerRange(MPU6050_RANGE_8_G);

        // set gyro range to +- 500 deg/s
        mpu.setGyroRange(MPU6050_RANGE_500_DEG);

        // set filter bandwidth to 21 Hz
        mpu.setFilterBandwidth(MPU6050_BAND_21_HZ);

        delay(100);
}

void loop() {
        /* Get new sensor events with the readings */
        sensors_event_t a, g, temp;
        mpu.getEvent(&a, &g, &temp);
        /* Print out the values */
        Serial.print("Acceleration X: ");
        Serial.print(a.acceleration.x);
        Serial.print(", Y: ");
        Serial.print(a.acceleration.y);
        Serial.print(", Z: ");
        Serial.print(a.acceleration.z);
        Serial.println(" m/s^2");

        Serial.print("Rotation X: ");
        Serial.print(g.gyro.x);
        Serial.print(", Y: ");
        Serial.print(g.gyro.y);
        Serial.print(", Z: ");
        Serial.print(g.gyro.z);
        Serial.println(" rad/s");

        Serial.print("Temperature: ");
        Serial.print(temp.temperature);
        Serial.println(" degC");

        Serial.println("");
        delay(500);
}
```

Note that you must set your serial monitor to a speed of 115200 baud to try out the sketch. Because too much data is sent back from the MPU6050, it requires this higher speed to display it.

You will see myriad of data displaying linear acceleration, angular rotation and temperature values. Try moving your sensor around and notice how the data changes.



```
MPU6050 Found!
Acceleration X: 0.31, Y: -0.62, Z: 10.34 m/s^2
Rotation X: 0.18, Y: -0.17, Z: 1.37 rad/s
Temperature: 27.57 degC

Acceleration X: 1.17, Y: -1.04, Z: 10.30 m/s^2
Rotation X: -0.35, Y: -0.39, Z: 0.71 rad/s
Temperature: 27.58 degC

Acceleration X: 5.85, Y: 6.74, Z: 4.93 m/s^2
Rotation X: 0.07, Y: -0.05, Z: -0.03 rad/s
Temperature: 27.57 degC
```

Code Explanation:

The first step is to include all the required Arduino libraries. As mentioned before, the Adafruit_MPU6050 library implements the hardware functions of the MPU6050 and the Adafruit_Sensor library the unified sensor abstraction layer. You will also need to include the Wire library, which comes pre-installed in the Arduino IDE. This library allows us to communicate with I2C devices.

```
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Wire.h>
```

Next, a new instance of the Adafruit_MPU6050 class is created so that we can access related functions.

```
Adafruit_MPU6050 mpu;
```

In the setup section of the code, we first initialize the serial communication with PC and call the begin() function. The begin() function initializes I2C interface and checks if the chip ID is correct. It then resets the chip using soft-reset & waits for the sensor for calibration after wake-up.

```
Serial.begin(115200);

// Try to initialize!
if (!mpu.begin()) {
       Serial.println("Failed to find MPU6050 chip");
       while (1) {
         delay(10);
       }
}
```

Before using the device object you constructed, you must initialize it with the sensitivity range you want to use. Below are three functions that set the measurement range of the MPU6050.

setAccelerometerRange(mpu6050_accel_range_t)

The setAccelerometerRange() function sets the accelerometer range. Allowed values for ‘setAccelerometerRange’ are:

MPU6050_RANGE_2_G – for ±2g range (default)

MPU6050_RANGE_4_G – for ±4g range

MPU6050_RANGE_8_G – for ±8g range

MPU6050_RANGE_16_G – for ±16g range

Note that, the smaller the range, the more sensitive the readings will be from the accelerometer.

setGyroRange(mpu6050_gyro_range_t)

The setGyroRange() function sets the accelerometer range. Allowed values for ‘setGyroRange’ are:

MPU6050_RANGE_250_DEG – for 250 degrees-per-second range (default)

MPU6050_RANGE_500_DEG – for 500 degrees-per-second range

MPU6050_RANGE_1000_DEG – for 1000 degrees-per-second range

MPU6050_RANGE_2000_DEG – for 2000 degrees-per-second range

Note that, a smaller degrees-per-second range means a more sensitive output.

setFilterBandwidth(mpu6050_bandwidth_t)

The setFilterBandwidth() function sets the digital low pass filter bandwidth options. Allowed values for ‘setFilterBandwidth’ are:

MPU6050_BAND_260_HZ, – for 260 Hz bandthwidth (Docs imply this disables the filter)

MPU6050_BAND_184_HZ, – for 184 Hz bandthwidth

MPU6050_BAND_94_HZ, – for 94 Hz bandthwidth

MPU6050_BAND_44_HZ, – for 44 Hz bandthwidth

MPU6050_BAND_21_HZ, – for 21 Hz bandthwidth

MPU6050_BAND_10_HZ, – for 10 Hz bandthwidth

MPU6050_BAND_5_HZ, – for 5 Hz bandthwidth

The output data of the gyroscope are filtered by a low pass filter. The bandwidth selection allows you to change the cutoff frequency of this filter. All the bandwidth setting does is simply smooth out the signal a little by removing high frequency noise.

For our experiment we are setting the accelerometer range to ±8G, gyro range to ±500°/s and filter bandwidth to 21 Hz.

```
mpu.setAccelerometerRange(MPU6050_RANGE_8_G);
mpu.setGyroRange(MPU6050_RANGE_500_DEG);
mpu.setFilterBandwidth(MPU6050_BAND_21_HZ);
```

The measurement range, or full-scale range, is the maximum acceleration or angular velocity that your MPU6050 can read. Think about what you are measuring and set limits accordingly. Do you need to measure the spin of a record player (which is very slow) or a spinning wheel (which can be very fast)?

In the loop section of the code, we first create a sensors_event_t object in memory to hold our results. sensors_event_t is just a user defined datatype (Structures in C) that holds many types of sensor data such as acceleration, gyro, temperature, light, pressure and many more. You can read more about it on github.

```
sensors_event_t a, g, temp;
```

Next, we call getEvent() function. This function reads a new set of values from you sensor (a sensor ‘event’), convert them to the appropriate SI units and scale, and then assign the results to our mpu object. This is the function you call to ‘read’ your sensor!

```
mpu.getEvent(&a, &g, &temp);
```

Finally, we output the values on the serial monitor.

```
Serial.print("Acceleration X: ");
Serial.print(a.acceleration.x);
Serial.print(", Y: ");
Serial.print(a.acceleration.y);
Serial.print(", Z: ");
Serial.print(a.acceleration.z);
Serial.println(" m/s^2");

Serial.print("Rotation X: ");
Serial.print(g.gyro.x);
Serial.print(", Y: ");
Serial.print(g.gyro.y);
Serial.print(", Z: ");
Serial.print(g.gyro.z);
Serial.println(" rad/s");

Serial.print("Temperature: ");
Serial.print(temp.temperature);
Serial.println(" degC");
```

Arduino Code – Plotting MPU6050 data

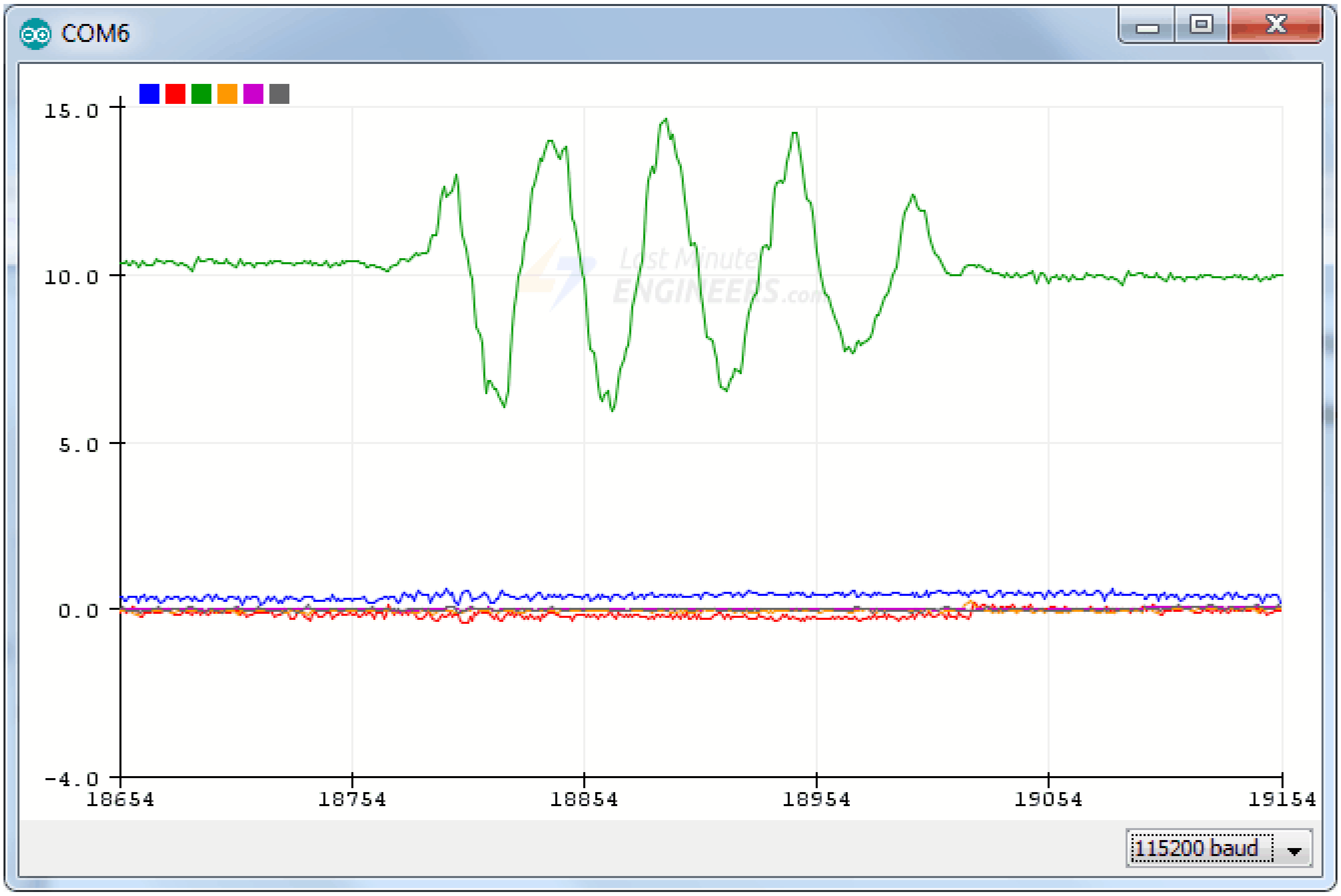
Just looking at the raw data coming from the MPU6050 will do nothing good. If you really want to see how your MPU6050 reacts when you move it around, use Serial Plotter.

The Arduino IDE comes with a cool tool called the serial plotter. It can give you visualizations of variables in real-time. This is super useful for visualizing data, troubleshooting your code, and visualizing your variables as waveforms.

Let’s try it out with the new code below. Compile and upload the program below, then navigate to Tools > Serial Plotter (Ctrl+Shift+L). The code uses a baud rate of 115200, make sure it’s set in the serial plotter as 115200 too.

```
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Wire.h>

Adafruit_MPU6050 mpu;

void setup(void) {
        Serial.begin(115200);

        // Try to initialize!
        if (!mpu.begin()) {
                Serial.println("Failed to find MPU6050 chip");
                while (1) {
                  delay(10);
                }
        }
        // set accelerometer range to +-8G
        mpu.setAccelerometerRange(MPU6050_RANGE_8_G);

        // set gyro range to +- 500 deg/s
        mpu.setGyroRange(MPU6050_RANGE_500_DEG);

        // set filter bandwidth to 21 Hz
        mpu.setFilterBandwidth(MPU6050_BAND_21_HZ);

        delay(100);
}

void loop() {
        /* Get new sensor events with the readings */
        sensors_event_t a, g, temp;
        mpu.getEvent(&a, &g, &temp);

        /* Print out the values */
        Serial.print(a.acceleration.x);
        Serial.print(",");
        Serial.print(a.acceleration.y);
        Serial.print(",");
        Serial.print(a.acceleration.z);
        Serial.print(", ");
        Serial.print(g.gyro.x);
        Serial.print(",");
        Serial.print(g.gyro.y);
        Serial.print(",");
        Serial.print(g.gyro.z);
        Serial.println("");
        delay(10);
}
```

You should see something like this when moving the module up and down along the Z axis.



Code Explanation:

You will notice that the majority of this sketch is exactly the same as the previous sketch, except:

Temperature readings are not printed

All other readings are printed such that it forms a comma-separated list of values

The readings are taken every 10 milliseconds.