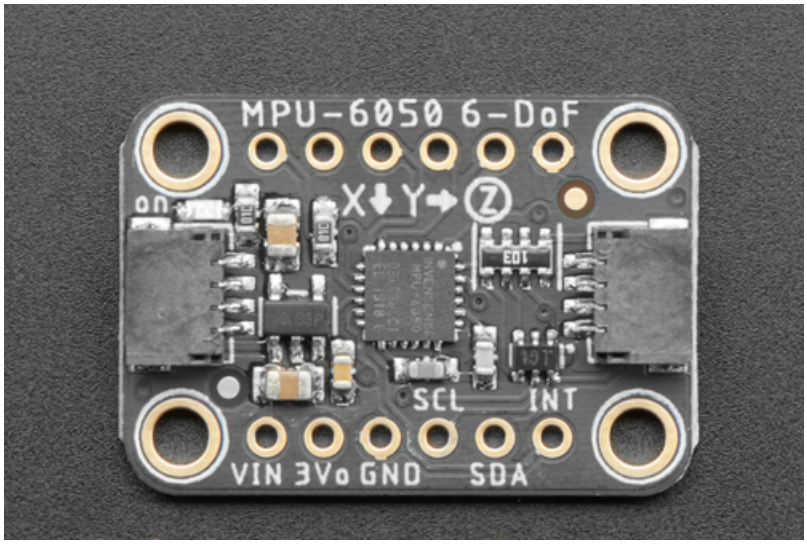
adafruit learning system

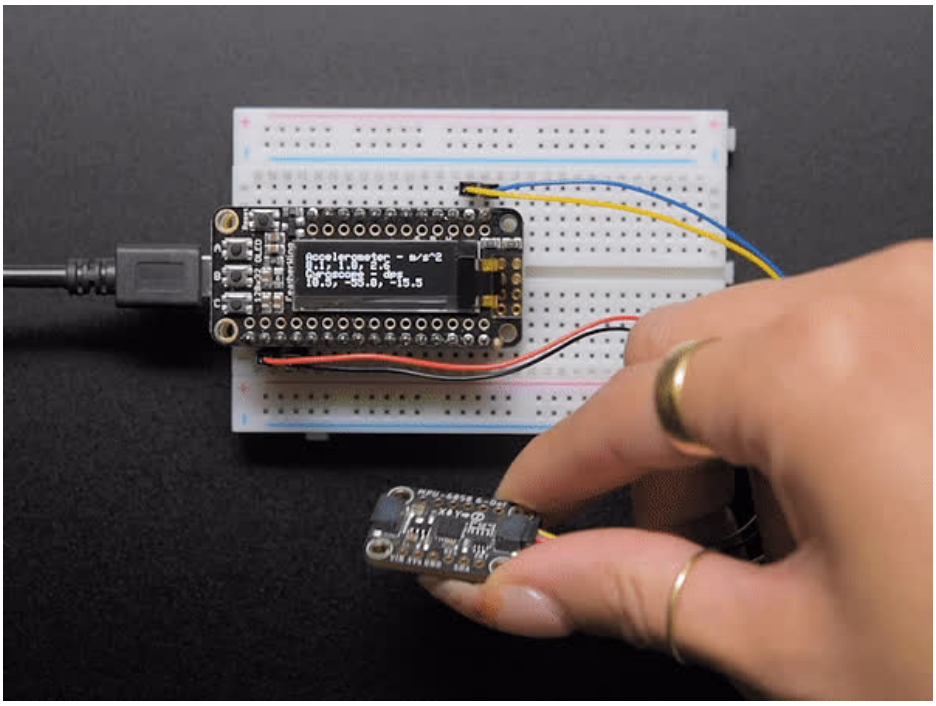
# MPU6050 6-DoF Accelerometer and Gyro

Created by Bryan Siepert



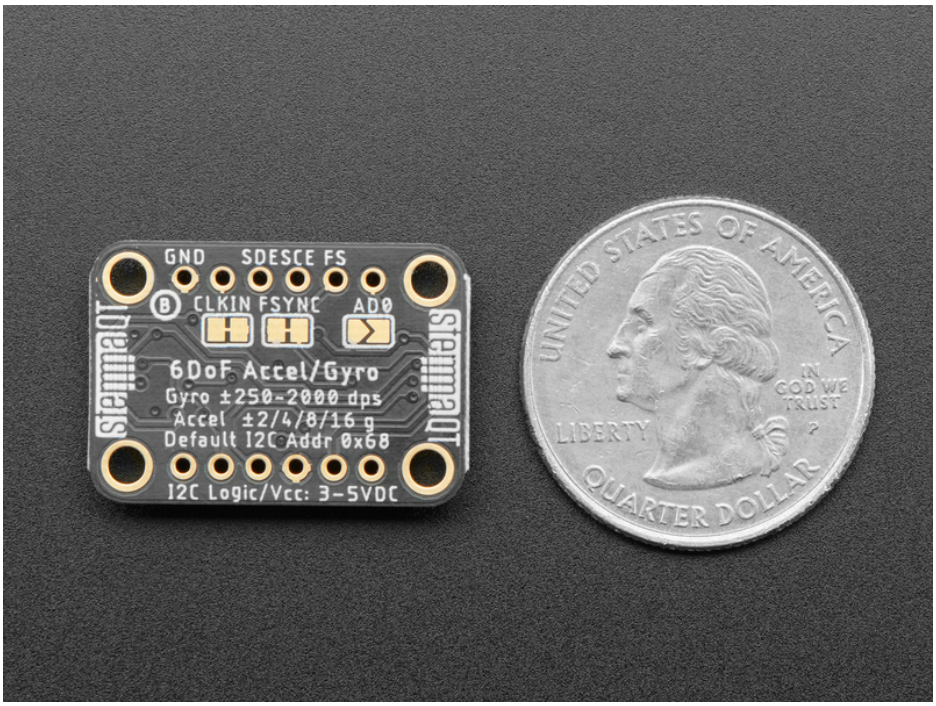

Last updated on 2020-01-24 12:03:28 AM UTC

# Overview

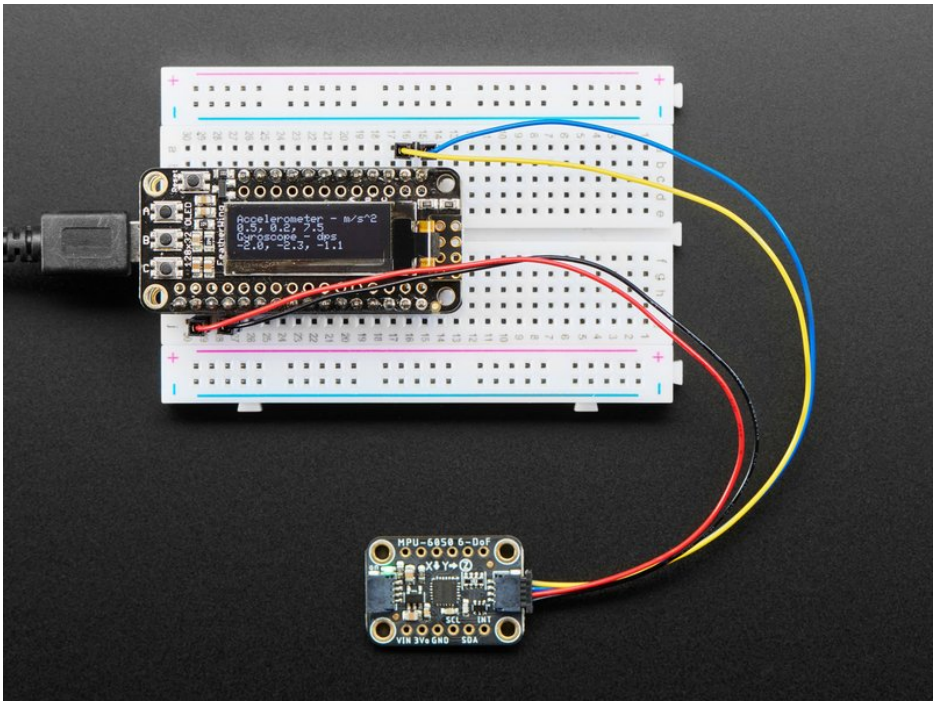


The MPU-6050 is a popular six DoF accelerometer and gyroscope (gyro) that has all the info you need on how things are shakin' and spinnin' . With six axes of sensing and 16-bit measurements, you'll have everything you need to give your robot friend a sense of balance, using the MPU-6050 as its inner ear.

This combination of gyroscopes and accelerometers is commonly referred to as an **I**nertial **M**easurement **U**nit or **IMU.** Not so long ago IMUs were the size of a breadbox (https://adafru.it/GEp)and cost upwards of **$50,000!** While you're not going to be using it to guide your mars rocket, the MPU-6050 is several orders of magnitude smaller and a bargain at a price three orders of magnitude less! Now you can add telemetry to your water rocket (https://adafru.it/GEq)(with some waterproofing).

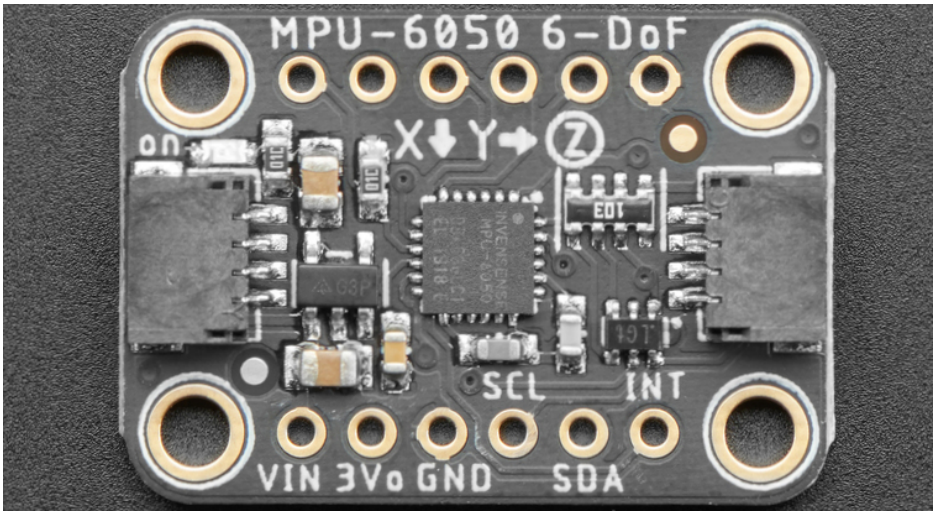



As with all Adafruit breakouts, we've done the work to make the MPU-6050 super easy to use. We've put it on a breakout board with the required support circuitry and connectors to make it easy to work with. And of course we've added SparkFun Qwiic (https://adafru.it/Fpw) compatible **STEMMA QT** (https://adafru.it/Ft4) JST SH connectors that allow you to get going **without needing to solder.** Just use a STEMMA QT adapter cable (https://adafru.it/FA-), plug it into your favorite micro or Blinka supported SBC and you're ready to rock!




"What is a sensor without a driver?" you might say. To that I would reply "Who cares, we got some for you right here". Be it Arduino, CircuitPython or Python on a computer (single board or even multi-board! (https://adafru.it/FWD)), we've got you covered.

# Pinouts




## Power Pins

- **Vin** - this is the power pin. Since the sensor chip uses 3 VDC, we have included a voltage regulator on board that will take 3-5VDC and safely convert it down. To power the board, give it the same power as the logic level of your microcontroller - e.g. for a 5V microcontroller like Arduino, use 5V
- **3Vo** - this is the 3.3V output from the voltage regulator, you can grab up to 100mA from this if you like
- **GND** - common ground for power and logic

## I2C Logic Pins

- **SCL** - I2C clock pin, connect to your microcontroller's I2C clock line. This pin is level shifted so you can use 3-5V logic, and there's a **10K pullup** on this pin.
- **SDA** -I2C data pin, connect to your microcontroller's I2C data line. This pin is level shifted so you can use 3-5V logic, and there's a **10K pullup** on this pin.
- **STEMMA QT (https://adafru.it/Ft4) -** These connectors allow you to connectors to dev boards with **STEMMA QT** connectors or to other things with various associated accessories (https://adafru.it/Ft6)

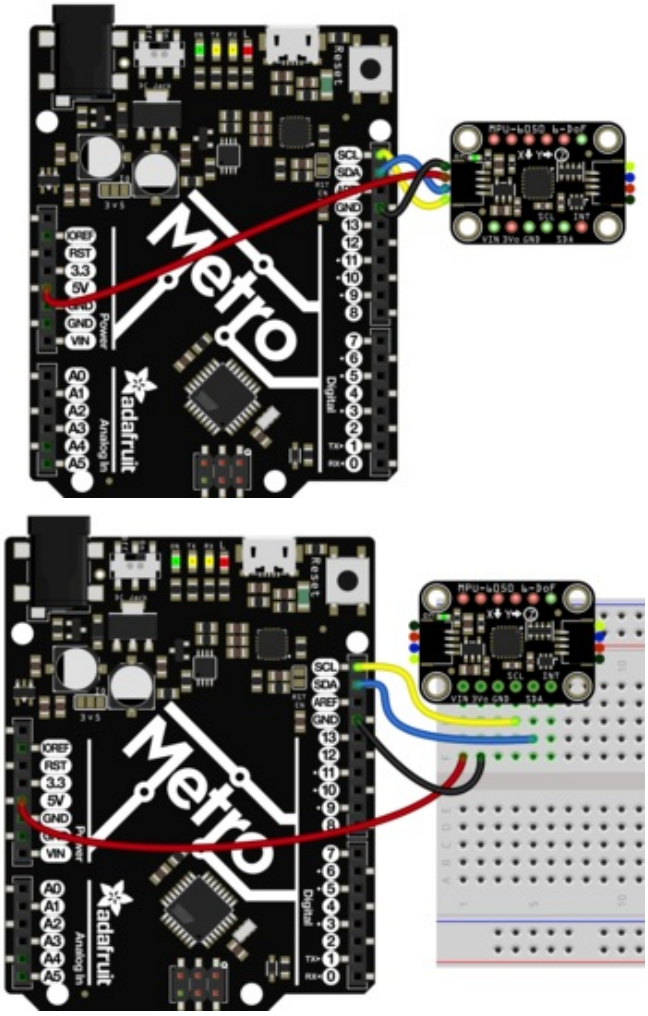
## Other Pins

- **INT** -This is the interrupt pin. You can setup the MPU-6050 to pull this low when certain conditions are met such as new measurement data being available. Consult the datasheet and register map (https://adafru.it/GEr) for usage
- **AD0** - I2C Address pin. Pulling this pin high or bridging the solder jumper on the back will change the I2C address from **0x68** to **0x69**
- **FS, SCE, SDE, CLKIN** - Pins for advanced users to connect the MPU-6050 to another sensor. Consult the datasheet and register map (https://adafru.it/GEr) for usage

# Arduino

## Wiring

Wiring the MPU-6050 to communicate with your microcontroller is straight forward thanks to the I2C interface. For these examples we can use the Metro or Arduino to take measurements. The instructions below show a Metro (https://adafru.it/METROXMETR), but the same applies to an Arduino

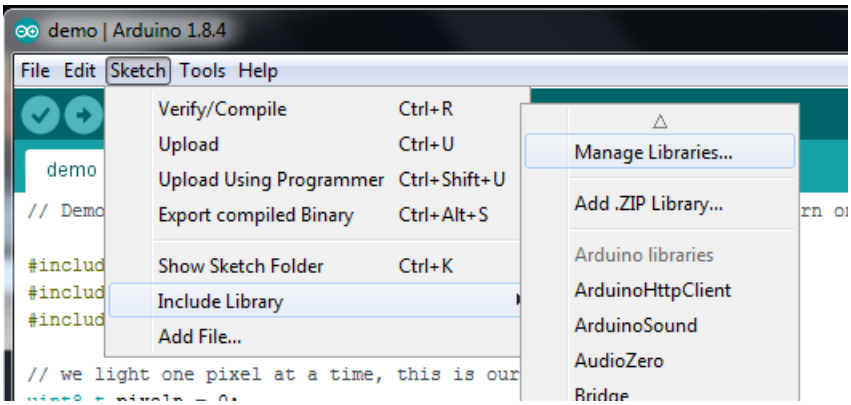


- Connect **board VCC (red wire)** to **Arduino 5V** if you are running a **5V** board Arduino (Uno, etc.). If your board is **3V,** connect to that instead.
- Connect **board GND (black wire)** to **Arduino GND**
- Connect **board SCL (yellow wire)** to **Arduino SCL**
- Connect **board SDA (blue wire)** to **Arduino SDA**

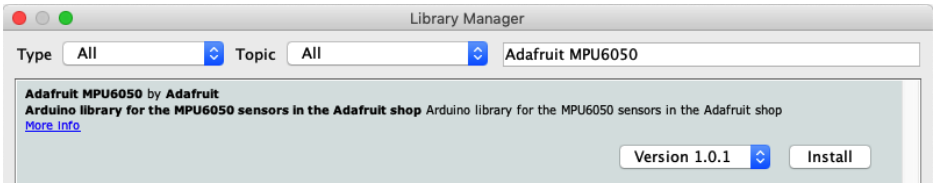
## Library Installation

Once wired up, to start using the MPU-6050 you'll need to install the Adafruit_MPU6050 library (https://adafru.it/GEs). The library is available through the Arduino library manager so we recommend taking that approach.

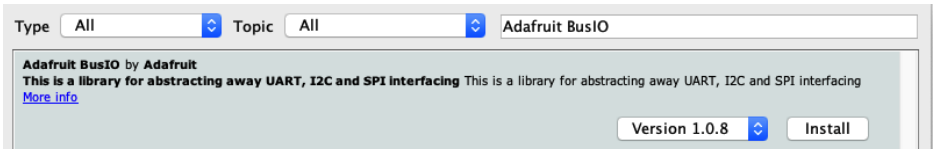
From the Arduino IDE, open up the Library Manager:



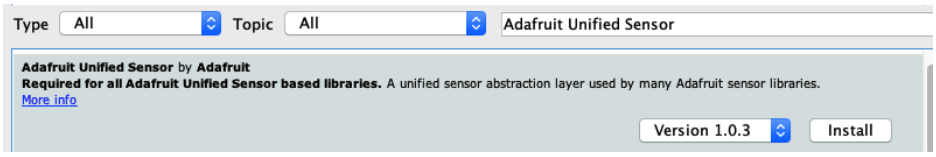

Click the **Manage Libraries ...** menu item, search for **Adafruit MPU6050**, and select the **Adafruit MPU6050** library and click **Install**:




Then follow the same process for the **Adafruit BusIO** library.



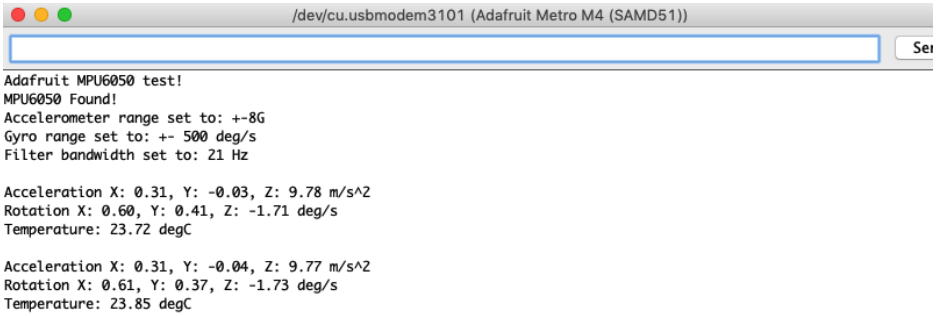

Lastly find and install the **Adafruit Unified Sensor** library




## Basic Reading Example

Open up **File -> Examples -> Adafruit MPU6050 -> basic_readings** and upload to your Arduino wired up to the sensor.

One you've uploaded the sketch to your board open up the Serial Monitor (**Tools->Serial Monitor**) at **115200 baud**. You should see the acceleration, rotation measurements, and temperature being printed like so:



Give the sensor a wiggle or a spin and watch how the measurements change!

## Basic Readings Example Code

```
// Basic demo for accelerometer readings from Adafruit MPU6050

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

Adafruit_MPU6050 mpu;

void setup(void) {
  Serial.begin(115200);
  while (!Serial)
    delay(10); // will pause Zero, Leonardo, etc until serial console opens

  Serial.println("Adafruit MPU6050 test!");

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

  mpu.setAccelerometerRange(MPU6050_RANGE_8_G);
  Serial.print("Accelerometer range set to: ");
  switch (mpu.getAccelerometerRange()) {
  case MPU6050_RANGE_2_G:
    Serial.println("+-2G");
    break;
  case MPU6050_RANGE_4_G:
    Serial.println("+-4G");
    break;
  case MPU6050_RANGE_8_G:
    Serial.println("+-8G");
    break;
  case MPU6050_RANGE_16_G:
    Serial.println("+-16G");
    break;
  }
  mpu.setGyroRange(MPU6050_RANGE_500_DEG);
  Serial.print("Gyro range set to: ");
  switch (mpu.getGyroRange()) {
  case MPU6050_RANGE_250_DEG:
    Serial.println("+- 250 deg/s");
    break;
  case MPU6050_RANGE_500_DEG:
    Serial.println("+- 500 deg/s");
    break;
  case MPU6050_RANGE_1000_DEG:
    Serial.println("+- 1000 deg/s");
    break;
  case MPU6050_RANGE_2000_DEG:
    Serial.println("+- 2000 deg/s");
```

```
    Serial.println("+-2000 deg/s");
    break;
  }

  mpu.setFilterBandwidth(MPU6050_BAND_21_HZ);
  Serial.print("Filter bandwidth set to: ");
  switch (mpu.getFilterBandwidth()) {
  case MPU6050_BAND_260_HZ:
    Serial.println("260 Hz");
    break;
  case MPU6050_BAND_184_HZ:
    Serial.println("184 Hz");
    break;
  case MPU6050_BAND_94_HZ:
    Serial.println("94 Hz");
    break;
  case MPU6050_BAND_44_HZ:
    Serial.println("44 Hz");
    break;
  case MPU6050_BAND_21_HZ:
    Serial.println("21 Hz");
    break;
  case MPU6050_BAND_10_HZ:
    Serial.println("10 Hz");
    break;
  case MPU6050_BAND_5_HZ:
    Serial.println("5 Hz");
    break;
  }

  Serial.println("");
  delay(100);
}

void loop() {

  /* Take a new reading */
  mpu.read();

  /* 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(" deg/s");
```

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

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

# Arduino Docs

Arduino Docs (https://adafru.it/GtD)

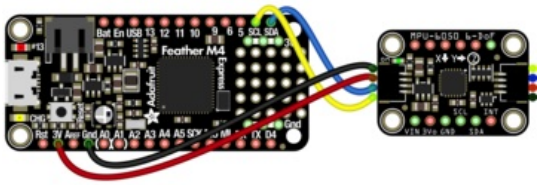
# Python and CircuitPython

It's easy to use the MPU6050 sensor with CircuitPython and the Adafruit CircuitPython MPU6050 (https://adafru.it/GEt) library. This library allows you to easily write Python code that reads the acceleration and adjust the measurement settings.

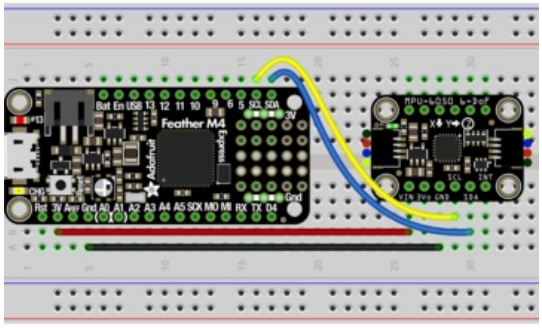
You can use this sensor with any CircuitPython microcontroller board or with a Linux single board computer that has GPIO and Python thanks to Adafruit_Blinka, our CircuitPython-for-Python compatibility library (https://adafru.it/BSN).

## CircuitPython Microcontroller Wiring

First wire up a MPU6050 to your board exactly as follows. Here is an example of the MPU6050 wired to a Feather (https://adafru.it/Cmy) using I2C:



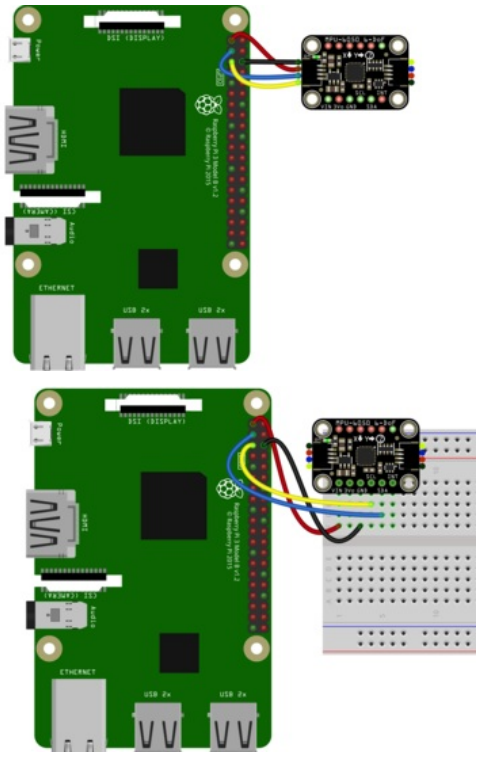
- **Board 3V** to **sensor VIN (red wire)**
- **Board GND** to **sensor GND (black wire)**
- **Board SCL** to **sensor SCL (yellow wire)**
- **Board SDA** to **sensor SDA (blue wire)**



## Python Computer Wiring

Since there's *dozens* of Linux computers/boards you can use we will show wiring for Raspberry Pi (https://adafru.it/scY). For other platforms, please visit the guide for CircuitPython on Linux to see whether your platform is supported (https://adafru.it/BSN).

Here's the Raspberry Pi wired with I2C:



- **Pi 3V** to **sensor VCC (red wire)**
- **Pi GND** to **sensor GND (black wire)**
- **Pi SCL** to **sensor SCL (yellow wire)**
- **Pi SDA** to **sensor SDA (blue wire)**

## CircuitPython Installation of MPU6050 Library

You'll need to install the Adafruit CircuitPython MPU6050 (https://adafru.it/GEt) library on your CircuitPython board.

First make sure you are running the latest version of Adafruit CircuitPython (https://adafru.it/Amd) for your board.

Next you'll need to install the necessary libraries to use the hardware--carefully follow the steps to find and install these libraries from Adafruit's CircuitPython library bundle (https://adafru.it/ENC). Our CircuitPython starter guide has a great page on how to install the library bundle (https://adafru.it/ABU).

For non-express boards like the Trinket M0 or Gemma M0, you'll need to manually install the necessary libraries from the bundle:

- **adafruit_mpu6050.mpy**
- **adafruit_bus_device**
- **adafruit_register**

Before continuing make sure your board's **lib** folder or root filesystem has the **adafruit_mpu6050.mpy**, **adafruit_bus_device**, and **adafruit_register** files and folders copied over.

Next connect to the board's serial REPL (https://adafru.it/Awz)so you are at the CircuitPython `>>>` prompt.

## Python Installation of MPU6050 Library

You'll need to install the **Adafruit_Blinka** library that provides the CircuitPython support in Python. This may also

require enabling I2C on your platform and verifying you are running Python 3. Since each platform is a little different, and Linux changes often, please visit the CircuitPython on Linux guide to get your computer ready (https://adafru.it/BSN)!

Once that's done, from your command line run the following command:

- `sudo pip3 install adafruit-circuitpython-mpu6050`

If your default Python is version 3 you may need to run 'pip' instead. Just make sure you aren't trying to use CircuitPython on Python 2.x, it isn't supported!

### CircuitPython & Python Usage

To demonstrate the usage of the sensor we'll initialize it and read the acceleration, rotation, and temperature measurements from the board's Python REPL.

Run the following code to import the necessary modules and initialize the I2C connection with the sensor:

```
import time
import board
import busio
import adafruit_mpu6050

i2c = busio.I2C(board.SCL, board.SDA)
mpu = adafruit_mpu6050.MPU6050(i2c)
```

```
>>> import time
>>> import board
>>> import busio
>>> import adafruit_mpu6050
>>> i2c = busio.I2C(board.SCL, board.SDA)
>>> mpu = adafruit_mpu6050.MPU6050(i2c)
>>>
```

Now you're ready to read values from the sensor using these properties:

- **acceleration** - The acceleration forces in the X, Y, and Z axes in $m/s^2$
- **gyro** - The rotation measurement on the X, Y, and Z axes in degrees/sec
- **temperature** - The temperature of the sensor in degrees C

For example, to print out the acceleration, gyro, and temperature values:

```
>>> print("Acceleration: X:%.2f, Y: %.2f, Z: %.2f m/s^2"%(mpu.acceleration))
Acceleration: X:0.39, Y: 0.00, Z: 9.87 m/s^2
>>> print("Gyro X:%.2f, Y: %.2f, Z: %.2f degrees/s"%(mpu.gyro))
Gyro X:0.58, Y: 0.33, Z: -1.85 degrees/s
>>> print("Temperature: %.2f C"%mpu.temperature)
Temperature: 24.98 C
>>>
```

That's all there is to it! Go forth and imbue your robot friends with the gift of being able to maybe not fall down all the time. Maybe.

## Example Code

```
import time
import board
import busio
import adafruit_mpu6050

i2c = busio.I2C(board.SCL, board.SDA)
mpu = adafruit_mpu6050.MPU6050(i2c)

while True:
    print("Acceleration: X:%.2f, Y: %.2f, Z: %.2f m/s^2"%(mpu.acceleration))
    print("Gyro X:%.2f, Y: %.2f, Z: %.2f degrees/s"%(mpu.gyro))
    print("Temperature: %.2f C"%mpu.temperature)
    print("")
    time.sleep(1)
```

# Python Docs

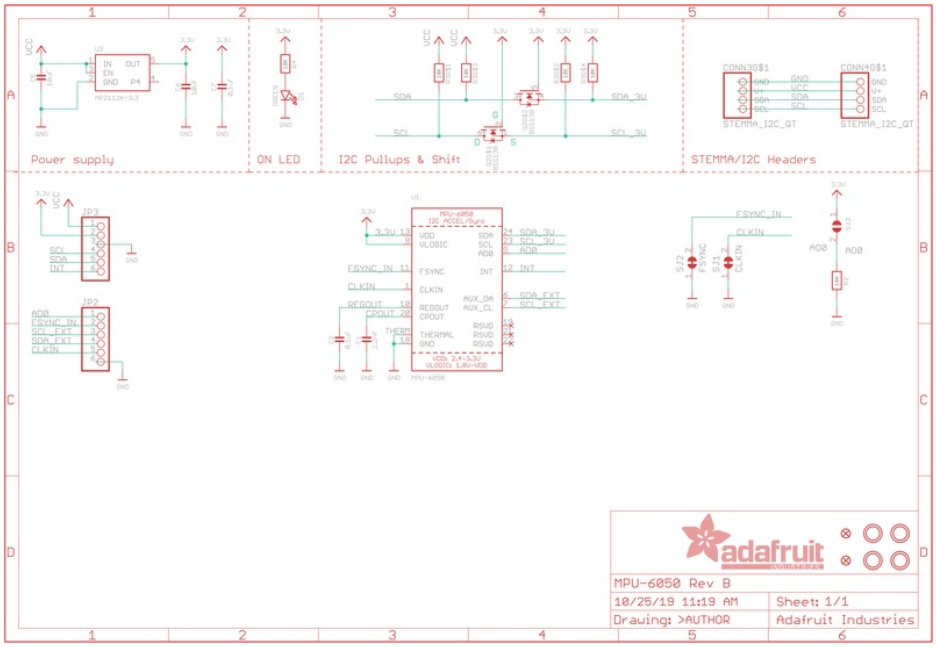
Python Docs (https://adafru.it/GtE)

# Downloads

## Files

- MPU6050 Datasheet (https://adafru.it/GEu)
- MPU6050 Register Map (https://adafru.it/GEv)
- EagleCAD files on GitHub (https://adafru.it/GEw)
- Fritzing object in Adafruit Fritzing Library (https://adafru.it/Gud)

## Schematic




## Fab Print