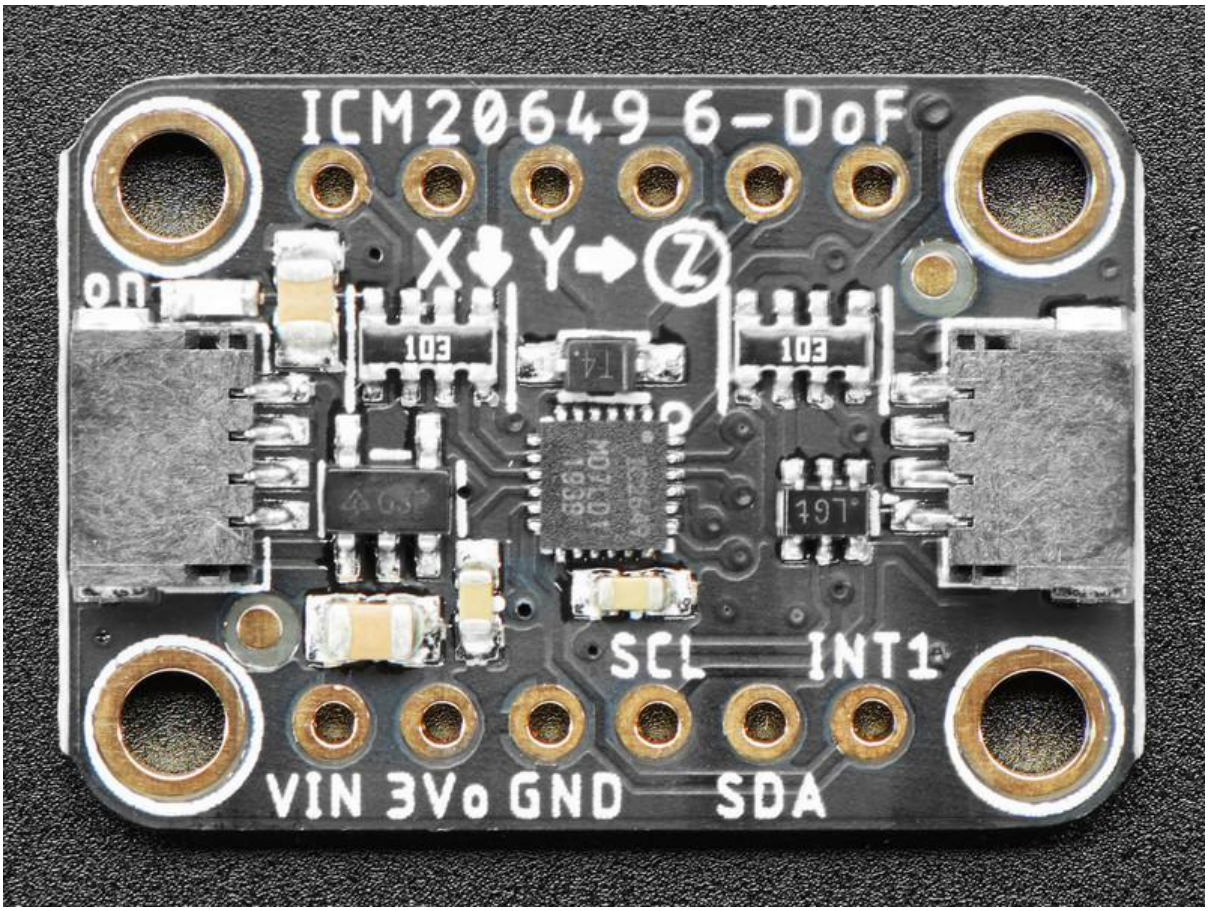
Adafruit ICM20649 Wide-Range 6-DoF IMU Accelerometer and Gyro

Created by Bryan Siepert

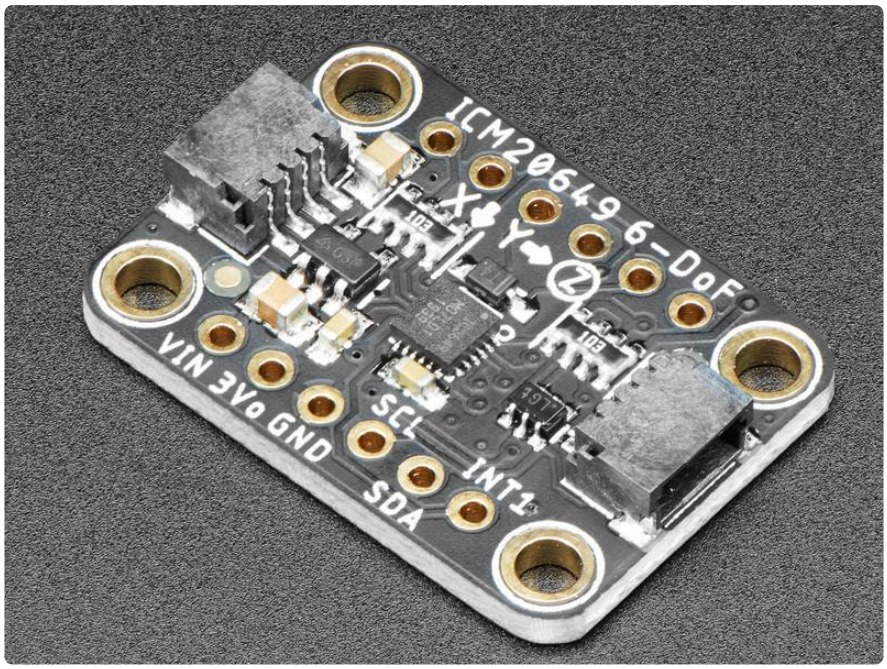


https://learn.adafruit.com/adafruit-icm20649-wide-range-6-dof-imu-accelerometer-and-gyro

Last updated on 2022-12-01 03:49:22 PM EST

Table of Contents

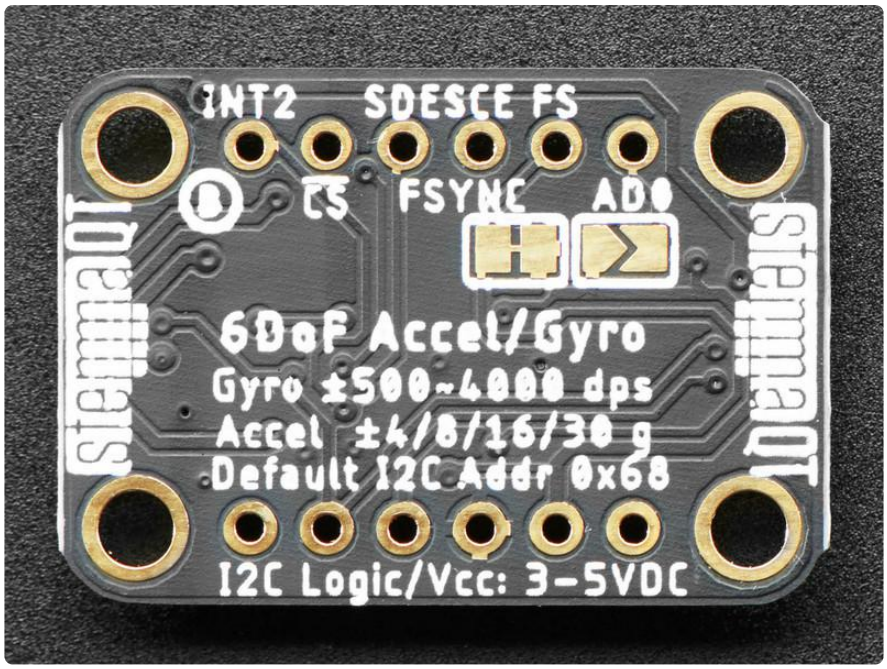
Overview



Most accelerometers have a similar range of measurements that they can make: often around 2G - 16G. Similarly most gyros can measure in the range of 250 degree/s to 2000 degrees/s. That's enough for many situations, however there are many situations where it's not quite enough. When measuring things like a golf swing, soccer ball kick or perhaps a fancy racing car, you need a bit more performance out of your motion sensor.

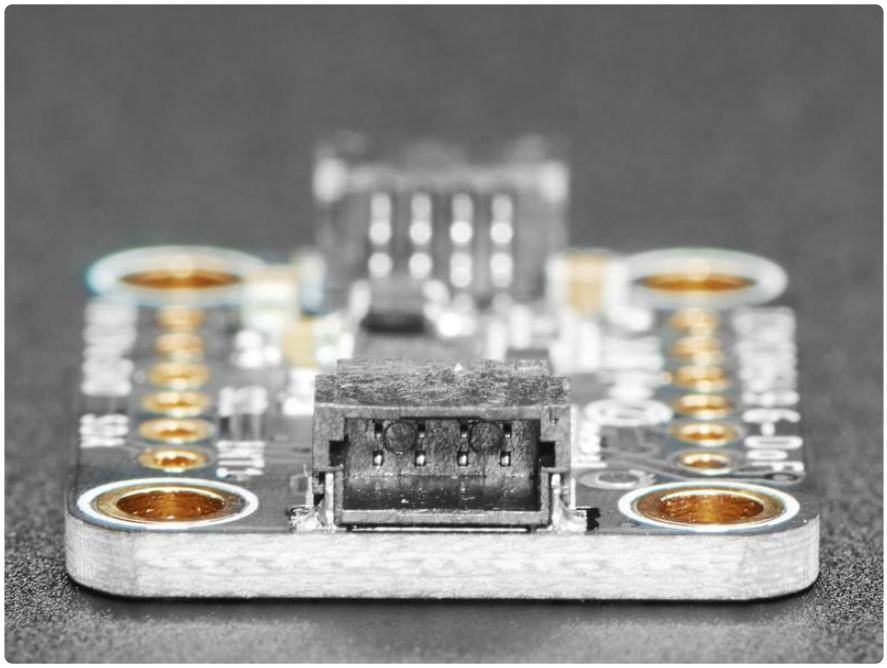
The ICM20649 from InvenSense is a Wide-Range 6-DoF accelerometer and Gyro, capable of measuring up to ±30g and ±4000 dps! That's an impressive increase from the norm, about twice the range.

If you've found yourself limited by the standard ranges available, or if you just want to make sure the measurement range of your IMU is never an issue, pick up one of our handy ICM20649 breakouts.



Each of our breakouts start with the sensor of interest, and in this case we begin with a hot-off-the-reel ICM20649 Wide-Range IMU. These chips are only a few mm across and have 0.4mm pitch contacts. To make it easier for folks to work with this chip, we've taken the sensor and mounted it on a breadboard-compatible breakout board.

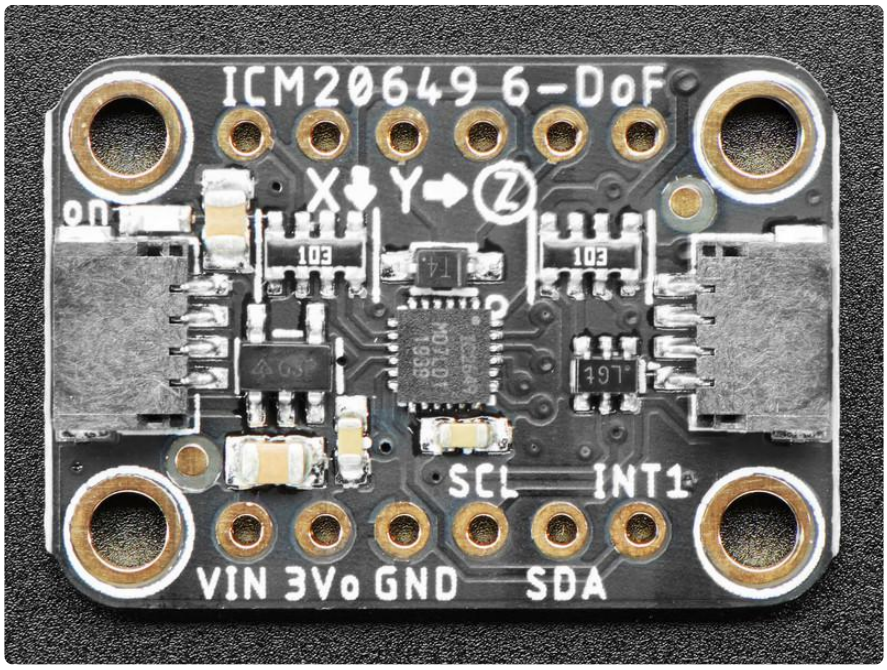
Along with the sensor, the breakout board includes a voltage regulator and level shifting circuitry to allow you to use it with a range of devices nearly as large as its measurement ranges. Works great with the 3.3V logic level of a Feather or Raspberry Pi, or the 5v level of a Metro 328 or Arduino Uno, this breakout is ready to work with most common microcontrollers or SBCs. and since it speaks I2C, you can easily connect it up with two data wires plus power and ground.

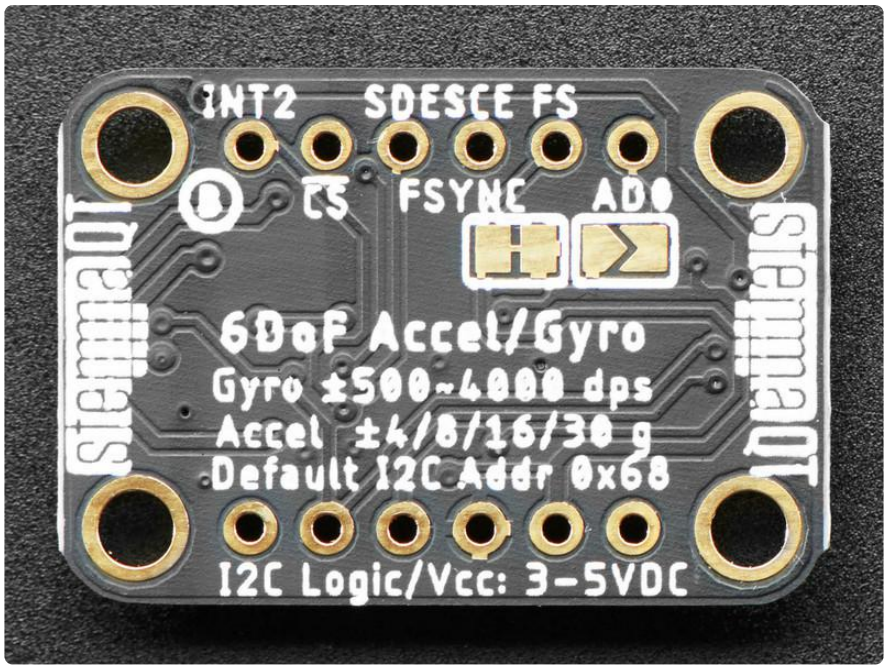


As if that weren't enough, we've also added SparkFun qwiic () compatible STEMMA QT () connectors for the I2C bus so you don't even need to solder. Just wire up to your favorite micro with a plug-and-play cable to get 6 DoF data ASAP. For a no-solder experience, just wire up to your favorite micro, like the STM32F405 Feather () using a STEMMA QT adapter cable. () The Stemma QT connectors also mean the ICM20649 can be used with our various associated accessories ()

Easy wiring is nice, but even nicer is when it's combined with drivers and examples that are ready to go. We wrote libraries and examples for Python and Arduino to make easy interfacing of the ICM20649 breakout. Together they'll get you measuring an astoundingly wide range of measurements in no time!

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 () - These connectors allow you to connectors to dev boards with S TEMMA QT connectors or to other things with various associated accessories ()
- AD0 - I2C Address pin and jumper. Pulling this pin high or bridging the solder jumper on the back will change the I2C address from 0x68 to 0x69

SPI Logic pins:

All pins going into the breakout have level shifting circuitry to make them 3-5V logic level safe. Use whatever logic level is on Vin!

- SCL (SCK)- This is also the SPI Clock pin, it's an input to the chip
- SDA (MOSI) - this is also the Serial Data In / Microcontroller Out Sensor In pin, for data sent from your processor to the sensor
- AD0 (MISO) - this is the Serial Data Out / Microcontroller In Sensor Out pin, for data sent from the sensor to your processor.
- CS - this is the Chip Select pin, drop it low to start an SPI transaction. Its an input to the chip

If you want to connect multiple ICM20649's to one microcontroller, have them share the SDA, SDO and SCL pins. Then assign each one a unique CS pin.

Other Pins

- INT1 -The first interrupt pin. You can setup the ICM20649 to pull this low when certain conditions are met such as a value exceeding a threshold. Consult the datasheet () for usage
- INT2 -The second interrupt pin. You can setup the ICM20649 to pull this low when certain conditions are met such as a value exceeding a threshold. Consult the datasheet () for usage
- FSYNC_IN - Used to sync the sensor with an external source. Consult the datasheet () for usage. Connected to GND by default. Cut the jumper on the bottom of the breakout to use the external pin
- SDE, SCE Pins for advanced users to connect the ICM20649 to another, external I2C sensor. Consult the datasheet () for usage

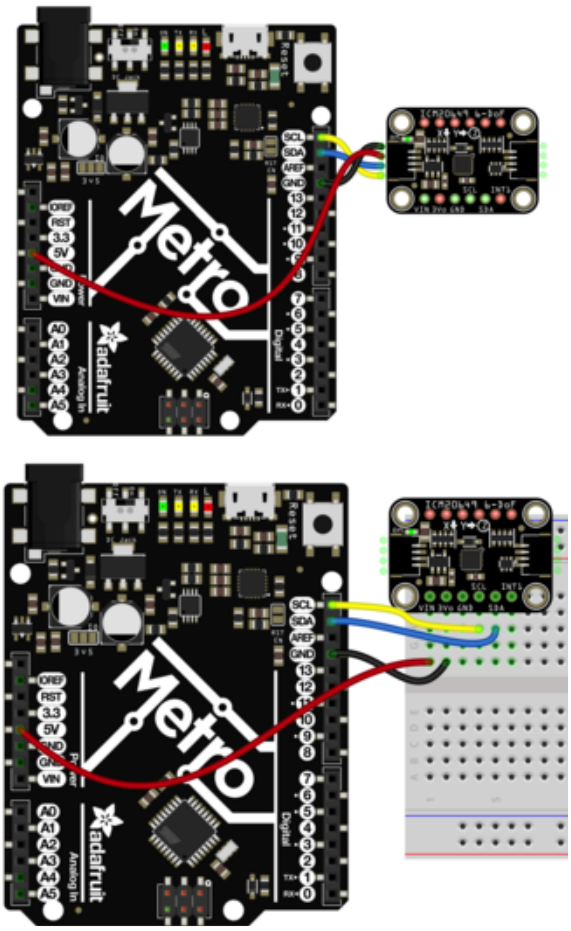
---

Arduino

I2C Wiring

Use this wiring if you want to connect via I2C interface

By default, the I2C address is 0x68. If you add a jumper from DO to 3.3V the address will change to 0x69

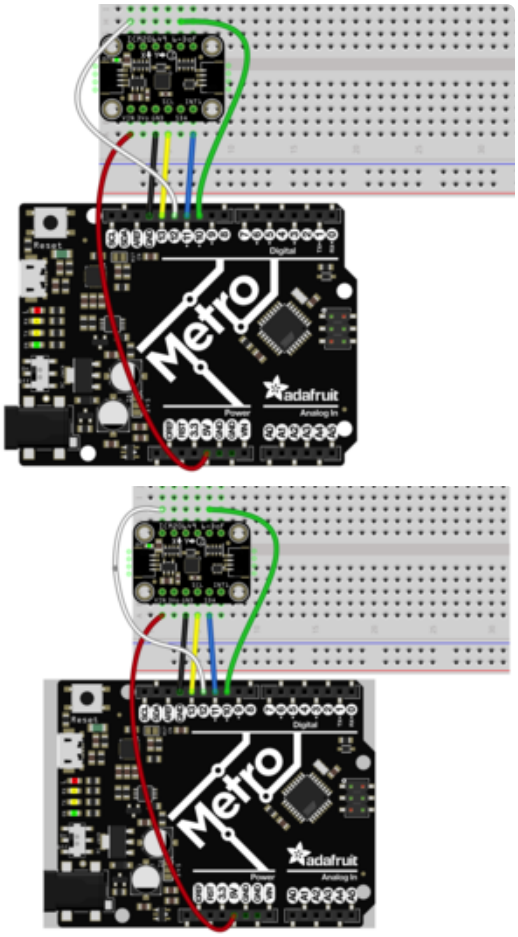


Connect board VIN (red wire) to Arduino 5V if you are running a 5V board Arduino (Uno, etc.). If your board is 3V, connect to that instead.
Connect board GND (black wire) to Arduino GND
Connect board SCL (yellow wire) to Arduino SCL
Connect board SDA (blue wire) to Arduino SDA

The final results should resemble the illustration above, showing an Adafruit Metro development board.

SPI Wiring

Since this is a SPI-capable sensor, we can use hardware or 'software' SPI. To make wiring identical on all microcontrollers, we'll begin with 'software' SPI. The following pins should be used:

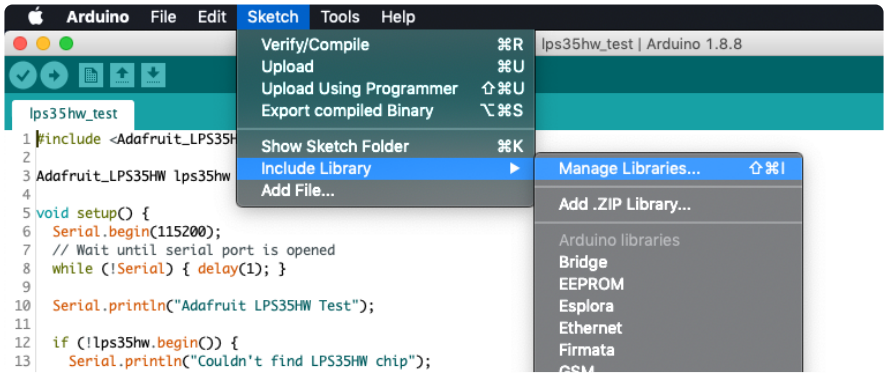


Connect Vin to the power supply, 3V or 5V is fine. Use the same voltage that the microcontroller uses
Connect GND to common power/data ground
Connect the SCL pin to Digital #13 (Yellow wire) but any pin can be used later
Connect the AD0 pin to Digital #12 (White wire) but any pin can be used later
Connect the SDA pin to Digital #11 (Blue wire) but any pin can be used later
Connect the CS pin Digital #10 (Green wire) but any pin can be used later

Later on, once we get it working, we can adjust the library to use hardware SPI if you desire, or change the pins to others.

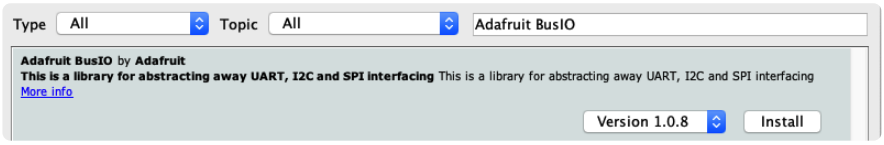
Library Installation

You can install the Adafruit ICM20649 Library for Arduino using the Library Manager in the Arduino IDE.

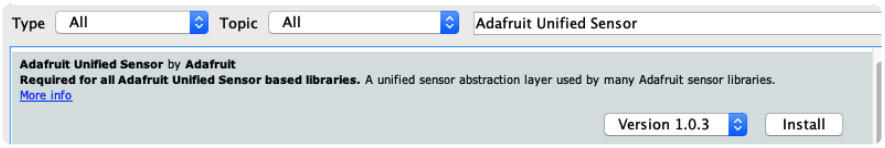


Click the Manage Libraries ... menu item, search for Adafruit ICM20649, and select the Adafruit ICM20649 library:

Then follow the same process for the Adafruit BusIO library.



Finally follow the same process for the Adafruit Unified Sensor library:



Load Example

Open up File -> Examples -> Adafruit ICM20649 -> adafruit_icm20649_test and upload to your Arduino wired up to the sensor.

Depending on whether you are using I2C or SPI, change the pin names and comment or uncomment the following lines.

```
// Try to initialize!
if (!icm.begin_I2C()) {
  // if (!icm.begin_SPI(ICM_CS)) {
  // if (!icm.begin_SPI(ICM_CS, ICM_SCK, ICM_MISO, ICM_MOSI)) {
```

Once you upload the code, you will see the accelerometer, gyro, and temperature measurements being printed when you open the Serial Monitor (Tools->Serial Monitor) at 115200 baud, similar to this:

```
15:01:00.126 -> Adafruit ICM20649 test!
15:01:00.194 -> ICM20649 Found!
15:01:00.194 -> Accelerometer range set to: +-4G
15:01:00.194 -> Gyro range set to: 500 degrees/s
15:01:00.194 -> Accelerometer data rate divisor set to: 20
15:01:00.194 -> Accelerometer data rate (Hz) is approximately: 53.57
15:01:00.194 -> Gyro data rate divisor set to: 10
15:01:00.194 -> Gyro data rate (Hz) is approximately: 100.00
15:01:00.194 ->
15:01:00.231 ->                Temperature 24.98 deg C
15:01:00.231 ->                Accel X: -9.29  Y: -0.20        Z: 6.26 m/s^2
15:01:00.231 ->                Gyro X: -2.55   Y: 2.73         Z: -2.62 radians
15:01:00.231 ->
15:01:00.338 ->                Temperature 24.74 deg C
15:01:00.338 ->                Accel X: -9.29  Y: -0.20        Z: 6.26 m/s^2
15:01:00.338 ->                Gyro X: -0.78   Y: -7.89        Z: -3.29 radians
```

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

Example Code

```
// Basic demo for accelerometer readings from Adafruit ICM20649

#include <Adafruit_ICM20X.h>
#include <Adafruit_ICM20649.h>
#include <Adafruit_Sensor.h>
#include <Wire.h>

Adafruit_ICM20649 icm;
uint16_t measurement_delay_us = 65535; // Delay between measurements for testing
// For SPI mode, we need a CS pin
#define ICM_CS 10
// For software-SPI mode we need SCK/MOSI/MISO pins
#define ICM_SCK 13
#define ICM_MISO 12
#define ICM_MOSI 11

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

  Serial.println("Adafruit ICM20649 test!");

  // Try to initialize!
  if (!icm.begin_I2C()) {
    // if (!icm.begin_SPI(ICM_CS)) {
    // if (!icm.begin_SPI(ICM_CS, ICM_SCK, ICM_MISO, ICM_MOSI)) {

    Serial.println("Failed to find ICM20649 chip");
    while (1) {
      delay(10);
    }
  }
  Serial.println("ICM20649 Found!");
  // icm.setAccelRange(ICM20649_ACCEL_RANGE_4_G);
  Serial.print("Accelerometer range set to: ");
  switch (icm.getAccelRange()) {
  case ICM20649_ACCEL_RANGE_4_G:
    Serial.println("+-4G");
    break;
  case ICM20649_ACCEL_RANGE_8_G:
    Serial.println("+-8G");
    break;
  case ICM20649_ACCEL_RANGE_16_G:
    Serial.println("+-16G");
    break;
  case ICM20649_ACCEL_RANGE_30_G:
    Serial.println("+-30G");
    break;
  }

  // icm.setGyroRange(ICM20649_GYRO_RANGE_500_DPS);
  Serial.print("Gyro range set to: ");
  switch (icm.getGyroRange()) {
  case ICM20649_GYRO_RANGE_500_DPS:
    Serial.println("500 degrees/s");
    break;
  case ICM20649_GYRO_RANGE_1000_DPS:
    Serial.println("1000 degrees/s");
    break;
  case ICM20649_GYRO_RANGE_2000_DPS:
```

```
    Serial.println("2000 degrees/s");
    break;
  case ICM20649_GYRO_RANGE_4000_DPS:
    Serial.println("4000 degrees/s");
    break;
  }

  //  icm.setAccelRateDivisor(4095);
  uint16_t accel_divisor = icm.getAccelRateDivisor();
  float accel_rate = 1125 / (1.0 + accel_divisor);

  Serial.print("Accelerometer data rate divisor set to: ");
  Serial.println(accel_divisor);
  Serial.print("Accelerometer data rate (Hz) is approximately: ");
  Serial.println(accel_rate);

  //  icm.setGyroRateDivisor(255);
  uint8_t gyro_divisor = icm.getGyroRateDivisor();
  float gyro_rate = 1100 / (1.0 + gyro_divisor);

  Serial.print("Gyro data rate divisor set to: ");
  Serial.println(gyro_divisor);
  Serial.print("Gyro data rate (Hz) is approximately: ");
  Serial.println(gyro_rate);
  Serial.println();
}

void loop() {

  //  /* Get a new normalized sensor event */
  sensors_event_t accel;
  sensors_event_t gyro;
  sensors_event_t temp;
  icm.getEvent(&accel, &gyro, &temp);

  Serial.print("\t\tTemperature ");
  Serial.print(temp.temperature);
  Serial.println(" deg C");

  /* Display the results (acceleration is measured in m/s^2) */
  Serial.print("\t\tAccel X: ");
  Serial.print(accel.acceleration.x);
  Serial.print(" \tY: ");
  Serial.print(accel.acceleration.y);
  Serial.print(" \tZ: ");
  Serial.print(accel.acceleration.z);
  Serial.println(" m/s^2 ");

  /* Display the results (acceleration is measured in m/s^2) */
  Serial.print("\t\tGyro X: ");
  Serial.print(gyro.gyro.x);
  Serial.print(" \tY: ");
  Serial.print(gyro.gyro.y);
  Serial.print(" \tZ: ");
  Serial.print(gyro.gyro.z);
  Serial.println(" radians/s ");
  Serial.println();

  delay(100);

  //  Serial.print(temp.temperature);
  //
  //  Serial.print(",");
  //
  //  Serial.print(accel.acceleration.x);
  //  Serial.print(","); Serial.print(accel.acceleration.y);
  //  Serial.print(","); Serial.print(accel.acceleration.z);
  //
  //  Serial.print(",");
```

```
  //  Serial.print(gyro.gyro.x);
  //  Serial.print(","); Serial.print(gyro.gyro.y);
  //  Serial.print(","); Serial.print(gyro.gyro.z);

  //  Serial.println();
  //
  //  delayMicroseconds(measurement_delay_us);
}
```

Arduino Docs

Arduino Docs ()

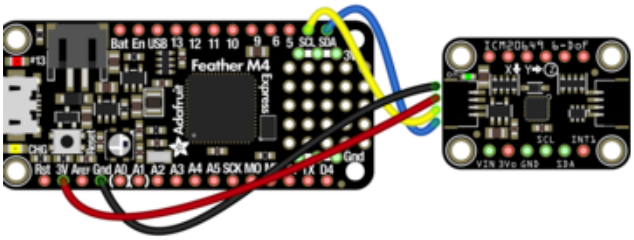
Python & CircuitPython

It's easy to use the ICM20649 sensor with Python and CircuitPython, and the Adafruit CircuitPython ICM20X () module. This module allows you to easily write Python code that reads measurements from the accelerometer and gyro, and will work with the ICM20649 as well as the ICM20948.

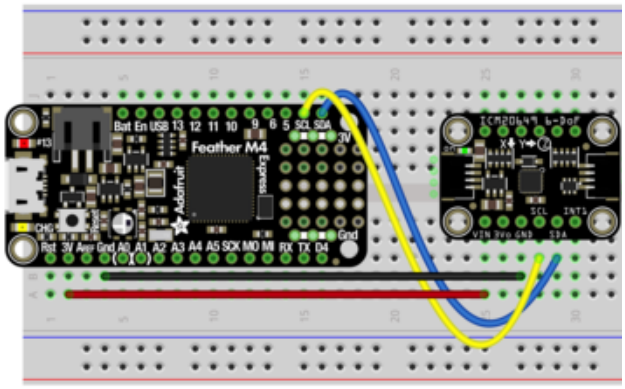
You can use this sensor with any CircuitPython microcontroller board or with a computer that has GPIO and Python thanks to Adafruit_Blinka, our CircuitPython-for-Python compatibility library ().

CircuitPython Microcontroller Wiring

First wire up a ICM20649 to your board for an I2C connection, exactly as shown below. Here's an example of wiring a Feather M4 to the sensor with I2C:



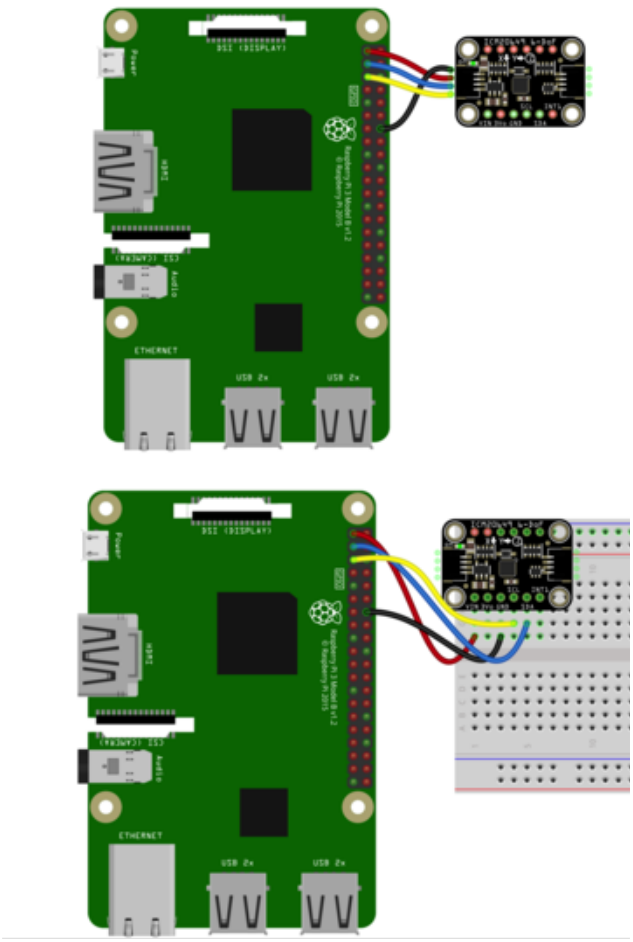
Board 3V to sensor VIN (red wire)
Board GND to sensor GND (black wire)
Board SCL to sensor SCL (yellow wire)
Board SDA to sensor SDA (blue wire)



Python Computer Wiring

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

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 ICM20X Library

You'll need to install the Adafruit CircuitPython ICM20X () library on your CircuitPython board.

First make sure you are running the latest version of Adafruit CircuitPython () 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 (). Our CircuitPython starter guide has a great page on how to install the library bundle ().

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

- adafruit_icm20X.mpy
- adafruit_bus_device
- adafruit_register

Before continuing make sure your board's lib folder has the adafruit_icm20X.mpy, adafruit_bus_device, and adafruit_register files and folders copied over.

Next connect to the board's serial REPL ()so you are at the CircuitPython `>>>` prompt .

Python Installation of ICM20X 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 ()!

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

- `sudo pip3 install adafruit-circuitpython-icm20x`

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 and rotation measurements from the board's Python REPL.

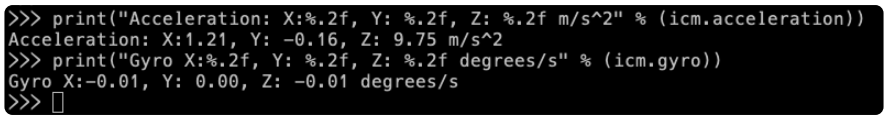
Run the following code to import the necessary modules and initialize the I2C connection with the sensor:

```
import time
import board
import adafruit_icm20x

i2c = board.I2C()
icm =  adafruit_icm20x.ICM20649(i2c)
```

```
>>> import time
>>> import board
>>> import adafruit_icm20x
>>> i2c = board.I2C()
>>> icm =  adafruit_icm20x.ICM20649(i2c)
```

```
print("Acceleration: X:%.2f, Y: %.2f, Z: %.2f m/s^2" % (icm.acceleration))
print("Gyro X:%.2f, Y: %.2f, Z: %.2f degrees/s" % (icm.gyro))
```



That's all it takes to get going with the ICM20649 Wide-Range 6-DoF IMU. You're now prepared to measure more extreme events that other sensors can't handle

Example Code

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

import time
import board
import adafruit_icm20x

i2c = board.I2C()  # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C()  # For using the built-in STEMMA QT connector on a
microcontroller
icm = adafruit_icm20x.ICM20649(i2c)

while True:
    print("Acceleration: X:%.2f, Y: %.2f, Z: %.2f m/s^2" % (icm.acceleration))
    print("Gyro X:%.2f, Y: %.2f, Z: %.2f rads/s" % (icm.gyro))
    print("")
    time.sleep(0.5)
```

Python Docs

Python Docs ()

Downloads

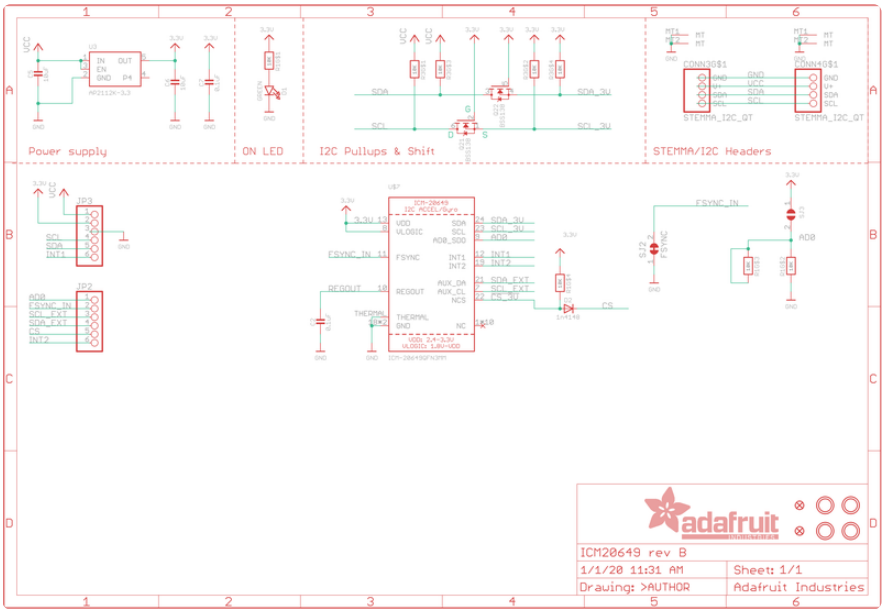
Files

- EagleCAD files on GitHub ()
- Fritzing object in Adafruit Fritzing Library ()

Datasheet

datasheet.pdf

Schematic



Fab Print