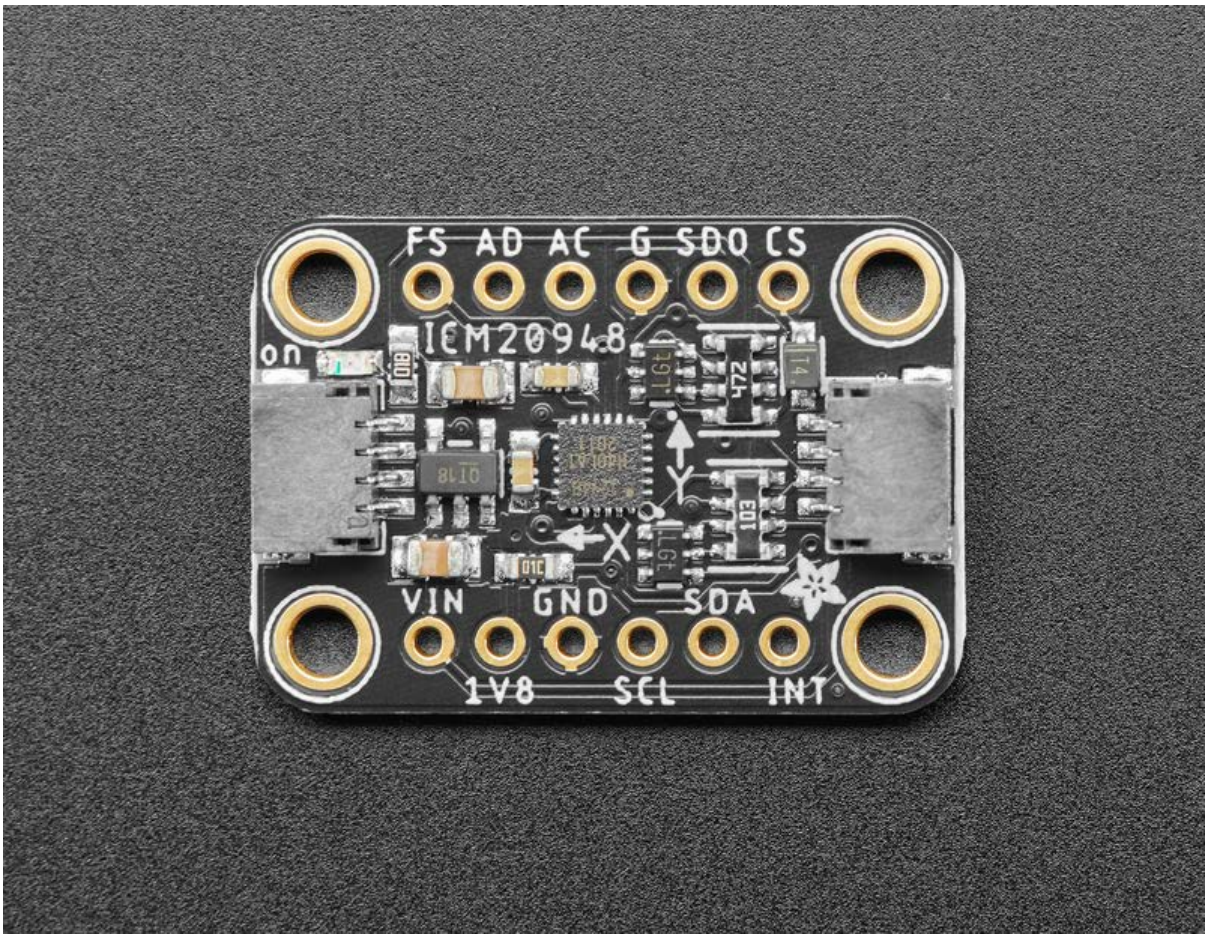
Adafruit TDK InvenSense ICM-20948 9-DoF IMU

Created by Bryan Siepert



https://learn.adafruit.com/adafruit-tdk-invensense-icm-20948-9-dof-imu

Last updated on 2024-06-03 03:10:54 PM EDT

Table of Contents

Overview 3

Pinouts 6

- I2C Logic Pins
- SPI Logic pins

Arduino 8

- I2C Wiring
- Library Installation
- Load Example
- Example Code

Arduino Docs 13

Python & CircuitPython 13

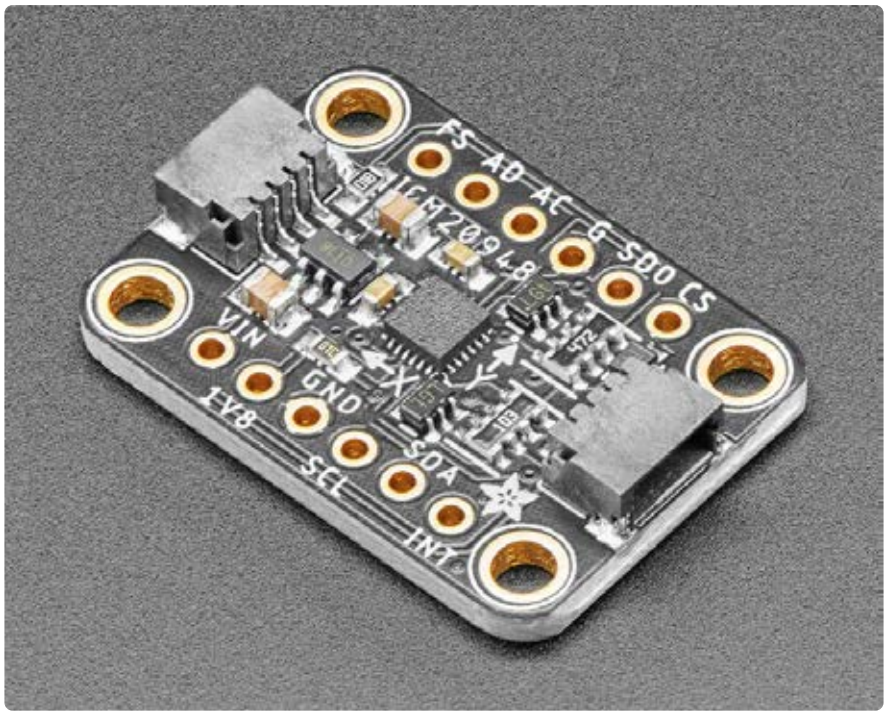
- CircuitPython Microcontroller Wiring
- Python Computer Wiring
- Python Installation of ICM20X Library
- CircuitPython & Python Usage
- Example Code

Python Docs 17

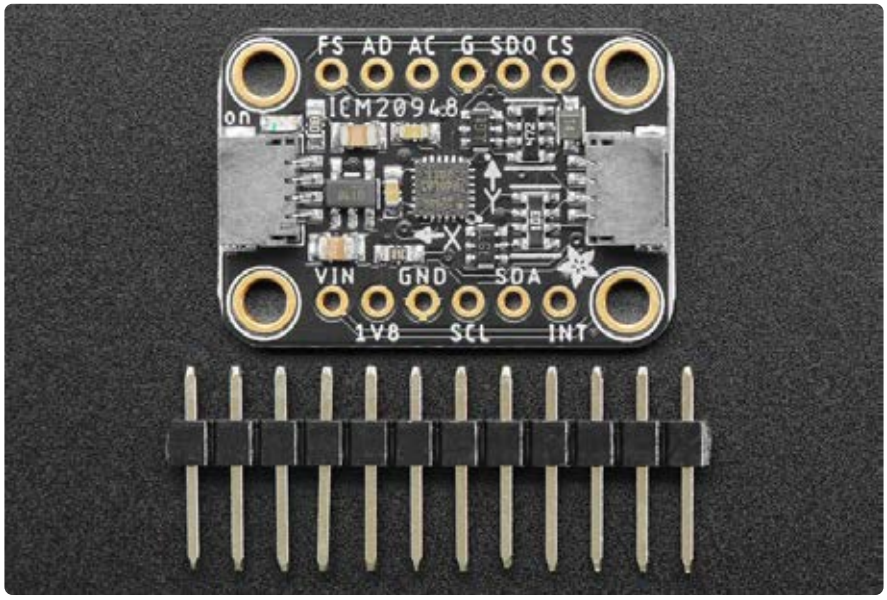
Downloads 17

- Files
- Schematic
-
- Fab Print

Overview



When you want to sense orientation using inertial measurements, you need an Inertial Measurement Unit, and when it comes to IMUs, the more DoFs, the better! The ICM20948 from Invensense packs 9 Degrees of freedom into a teeny package, making it a one stop shop for all the DOFs you need! Within it's svelte 3x3mm package there are not just one MEMS sensor die like your common sensors, but two sensor dies! The ICM20948 pair's Invensense's MEMS 3-axis accelerometer and gyro with the AK09916 3-axis magnetometer from Asahi Kasei Microdevices Corporation.



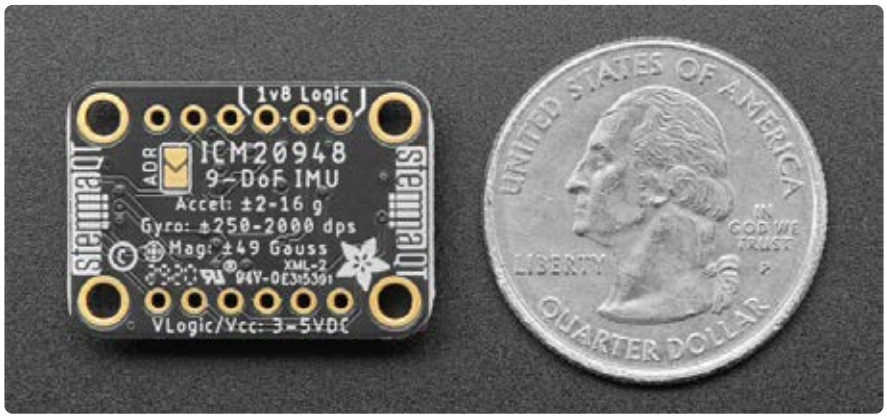
All 9 axes of measurement are made available thanks to a crew of 16-bit Analog to Digital Converters, diligently converting the raw analog signals from the MEMs sensors to digital readings that are accessed via I2C or SPI. Each of the sensors have the quality specs you would expect from such a sensor. Just see what the datasheet has to say:

- **3-Axis** Gyroscope with Programmable FSR of **±250 dps, ±500 dps, ±1000 dps, and ±2000 dps**

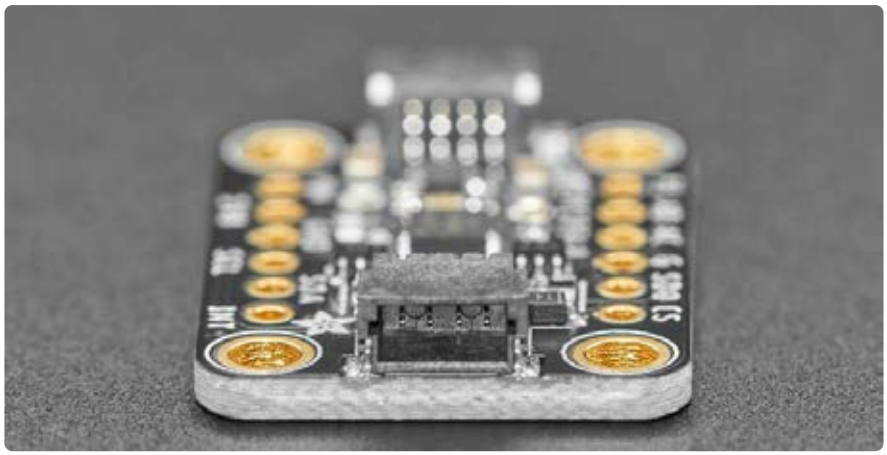
- **3-Axis Accelerometer** with Programmable FSR of **±2g, ±4g, ±8g, and ±16g**

- **3-Axis Compass** with a wide range to **±4900 µT**

Now that's a handy and capable team of sensors, ready to help orient your project in the right direction!



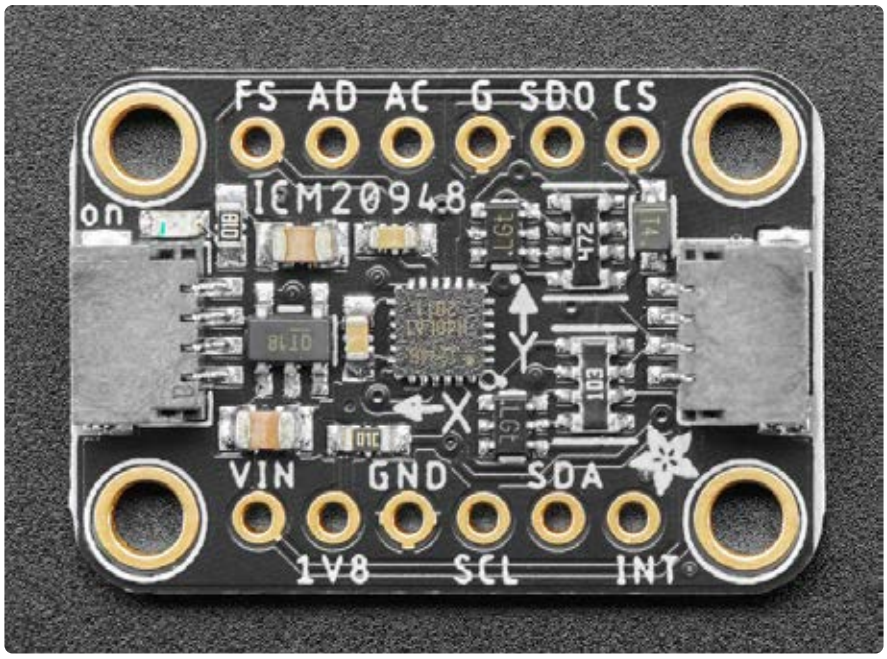
Like any high performance device the ICM20948 can a bit particular about how it needs to be worked with. Unlike the pick and place machines that normally handle these sensors, most of us humans can't readily take a little guy like the ICM20948 and plop it into our circuit; it's small! What's more, the ICM20948 runs on 1.8V which is increasingly common for device manufacturers but isn't hardly common for makers, learners, prototypers or the like. With that in mind, we've put the ICM20948 on a breakout with a 1.8V voltage regulator as well as level shifting circuitry to allow your 3.3V device such as a Feather M4 or Raspberry Pi, or a 5V device such as the Arduino Uno.



To make connections easy, our breakout puts makes the pins of the ICM20948 available on standard 0.100”/ 2.54mm pitch headers for use with a breadboard. Should you wish to avoid soldering, the Stemma QT form factor breakout also includes our Stemma QT connectors which just like the **SparkFun Qwiic** (https://adafru.it/Fpw) connectors they’re inspired by (and compatible with). Using these handy connectors you can simply plug in the sensor and get rolling with your project. You can even use them to daisy chain multiple sensors together! All of this is explained with wiring diagrams for use with Arduino or CircuitPython on the pages to come.

Lastly, all the wiring in the world wouldn’t do you much good if you didn’t know how to use those wires to talk to your sensor, so we’ve done the work of writing libraries for Arduino and CircuitPython that will allow you to use the ICM20948 with your favorite development board, be it an Arduino, Feather, Raspberry Pi, or one of the many other Arduino and CircuitPython compatible boards.

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
- **1V8** - this is the 1.8V 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 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 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)
- **SDO/ADR Jumper** - **I2C Address pin**. Pulling this pin high or bridging the solder jumper on the back will change the I2C address from **0x69** to **0x68**

SPI Logic pins

- **SCL** - This is also the **S**PI **C**lock pin / **SCK**, it's an input to the chip
- **SDA** - this is also the **S**erial **D**ata **I**n / **M**icrocontroller **O**ut **S**ensor **I**n / **MOSI** pin, for data sent from your processor to the ICM20948
- **SDO** - this is the **Serial D**ata **O**ut / **M**icrocontroller **I**n **S**ensor **O**ut / **MISO** pin, for data sent from the ICM20948 to your processor.
- **CS** - this is the **C**hip **S**elect pin, drop it low to start an SPI transaction. It's an input to the chip

If you want to connect multiple ICM20948s to one microcontroller, have them share the **SCL, SDA**, and **DO** pins. Then assign each one a unique **CS** pin.

Other Pins

- **INT** -This is the primary interrupt pin. You can setup the ICM20948 to pull this low or high when certain conditions are met such as new measurement data being available. Consult the datasheet (https://adafru.it/Iqf) for usage
- **AC -** Auxillary I2C bus clock. Advanced users can consult the datasheet (https://adafru.it/MC5) to learn how to use this pin to communicate with sensors on the ICM20948's auxillary bus. **1.8V Logic Only!**
- **AD -** Auxillary I2C bus data. Advanced users can consult the datasheet (https://adafru.it/MC5) to learn how to use this pin to communicate with sensors on the ICM20948's auxillary bus. **1.8V Logic Only!**

- **FS** - External frame sync. Advanced users can consult the datasheet (https://adafru.it/MC5) to learn how to use this pin to synchronize measurements with additional sensors. **1.8V Logic Only!**

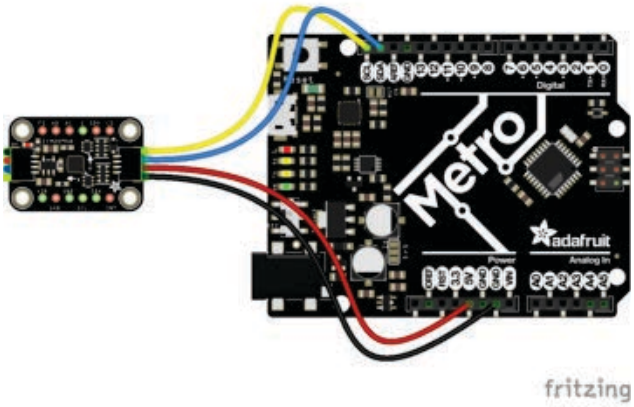
Arduino

Using the ICM20948 with Arduino is a simple matter of wiring up the sensor to your Arduino-compatible microcontroller, installing the Adafruit ICM20X (https://adafru.it/MC6) library we've written, and running the provided example code.

I2C Wiring

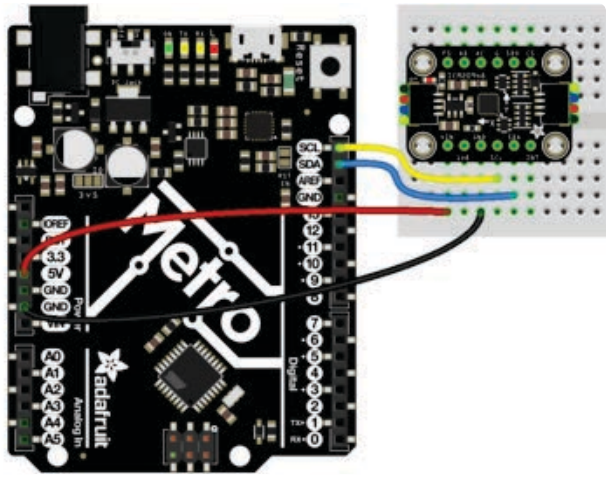
Use this wiring if you want to connect via I2C interface. The default I2C address for the ICM20948 is **0x69** but it can be switched to **0x68** by pulling the address pin low to GND.

Here is how to wire up the sensor using one of the **STEMMA QT** (https://adafru.it/Ft4) connectors. The examples show a Metro but wiring will work the same for an Arduino or other compatible board.



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**

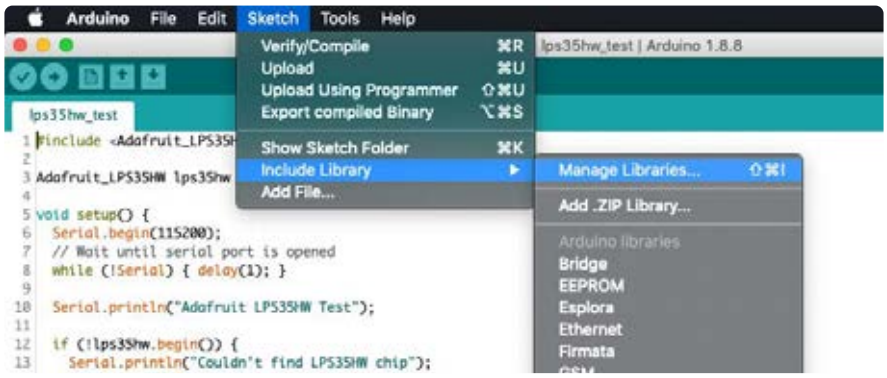
Here is how to wire the sensor to a board using a solderless breadboard:



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**

Library Installation

You can install the **Adafruit ICM20X l**ibrary for Arduino using the Library Manager in the Arduino IDE. This library is compatible with both the ICM20948 and it's sister sensor, the ICM20649



Click the **Manage Libraries ...** menu item, search for **Adafruit ICM20X**, and select the **Adafruit ICM20X** library:

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 ICM20X -> adafruit_ICM20948_test**

After opening the demo file, upload to your Arduino wired up to the sensor. Once you upload the code, you will see the **Temperature** as well as **X, Y, and Z values** for the **Gyro Accelerometer, and Magnetometer** being printed when you open the Serial Monitor (**Tools->Serial Monitor**) at **115200 baud**, similar to this:

```
Adafruit ICM20948 test!
ICM20948 Found!
Accelerometer range set to: +-16G
OK
Gyro range set to: 2000 degrees/s
Accelerometer data rate divisor set to: 20
Accelerometer data rate (Hz) is approximately: 53.57
Gyro data rate divisor set to: 10
Gyro data rate (Hz) is approximately: 100.00
Magnetometer data rate set to: 100 Hz

Temperature 28.62 deg C
Accel X: 0.05   Y: -1.00        Z: 9.66 m/s^2
Mag X: -7.50    Y: -54.60       Z: 15.30 uT
Gyro X: 0.00    Y: 0.02         Z: -0.00 radians/s

Temperature 28.67 deg C
Accel X: 0.10   Y: -0.93        Z: 9.63 m/s^2
Mag X: -7.80    Y: -54.60       Z: 15.60 uT
Gyro X: 0.00    Y: 0.02         Z: 0.00 radians/s

Temperature 29.00 deg C
Accel X: 0.08   Y: -0.96        Z: 9.63 m/s^2
Mag X: -8.40    Y: -55.20       Z: 16.20 uT
Gyro X: 0.00    Y: 0.02         Z: -0.00 radians/s
```

Example Code

```
// Basic demo for accelerometer readings from Adafruit ICM20948

#include <Adafruit_ICM20X.h>
#include <Adafruit_ICM20948.h>
#include <Adafruit_Sensor.h>
#include <Wire.h>

Adafruit_ICM20948 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 ICM20948 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 ICM20948 chip");
    while (1) {
      delay(10);
    }
  }
  Serial.println("ICM20948 Found!");
  // icm.setAccelRange(ICM20948_ACCEL_RANGE_16_G);
  Serial.print("Accelerometer range set to: ");
  switch (icm.getAccelRange()) {
  case ICM20948_ACCEL_RANGE_2_G:
    Serial.println("+-2G");
    break;
  case ICM20948_ACCEL_RANGE_4_G:
    Serial.println("+-4G");
    break;
  case ICM20948_ACCEL_RANGE_8_G:
    Serial.println("+-8G");
    break;
  case ICM20948_ACCEL_RANGE_16_G:
    Serial.println("+-16G");
    break;
  }
  Serial.println("OK");

  // icm.setGyroRange(ICM20948_GYRO_RANGE_2000_DPS);
  Serial.print("Gyro range set to: ");
  switch (icm.getGyroRange()) {
  case ICM20948_GYRO_RANGE_250_DPS:
    Serial.println("250 degrees/s");
    break;
  case ICM20948_GYRO_RANGE_500_DPS:
    Serial.println("500 degrees/s");
    break;
  case ICM20948_GYRO_RANGE_1000_DPS:
    Serial.println("1000 degrees/s");
    break;
  case ICM20948_GYRO_RANGE_2000_DPS:
    Serial.println("2000 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);

  // icm.setMagDataRate(AK09916_MAG_DATARATE_10_HZ);
  Serial.print("Magnetometer data rate set to: ");
  switch (icm.getMagDataRate()) {
```

```
  case AK09916_MAG_DATARATE_SHUTDOWN:
    Serial.println("Shutdown");
    break;
  case AK09916_MAG_DATARATE_SINGLE:
    Serial.println("Single/One shot");
    break;
  case AK09916_MAG_DATARATE_10_HZ:
    Serial.println("10 Hz");
    break;
  case AK09916_MAG_DATARATE_20_HZ:
    Serial.println("20 Hz");
    break;
  case AK09916_MAG_DATARATE_50_HZ:
    Serial.println("50 Hz");
    break;
  case AK09916_MAG_DATARATE_100_HZ:
    Serial.println("100 Hz");
    break;
  }
  Serial.println();

}

void loop() {

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

  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 ");

  Serial.print("\t\tMag X: ");
  Serial.print(mag.magnetic.x);
  Serial.print(" \tY: ");
  Serial.print(mag.magnetic.y);
  Serial.print(" \tZ: ");
  Serial.print(mag.magnetic.z);
  Serial.println(" uT");

  /* 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.print(",");
  //  Serial.print(mag.magnetic.x);
  //  Serial.print(","); Serial.print(mag.magnetic.y);
  //  Serial.print(","); Serial.print(mag.magnetic.z);

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

Arduino Docs

Arduino Docs (https://adafru.it/MC0)

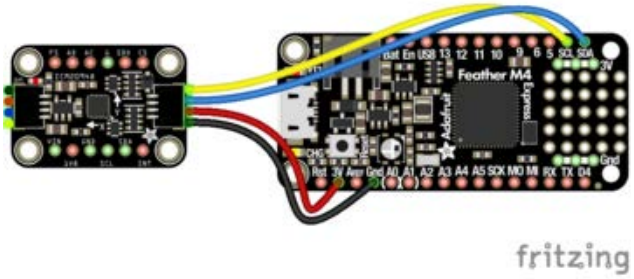
Python & CircuitPython

It's easy to use the **ICM20948** with Python and CircuitPython, and the Adafruit CircuitPython ICM20X (https://adafru.it/LAp) module. This module allows you to easily write Python code that reads measurements from the accelerometer and gyro, and will work with either sensor.

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 (https://adafru.it/BSN).

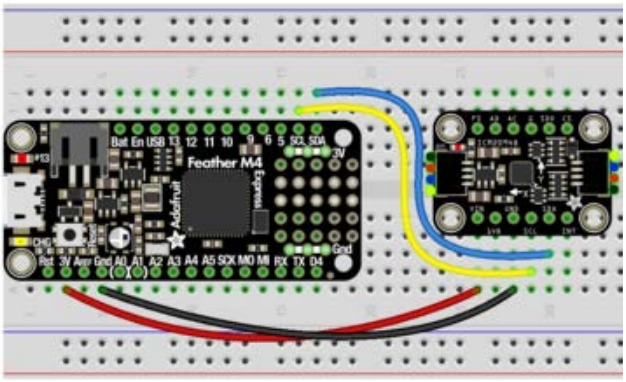
CircuitPython Microcontroller Wiring

First wire up a ICM20948 to your board exactly as shown below. Here's an example of wiring a Feather M4 to the sensor with I2C using one of the handy **STEMMA QT** (https://adafru.it/Ft4) connectors:



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)**

You can also use the standard **0.100" pitch** headers to wire it up on a breadboard:

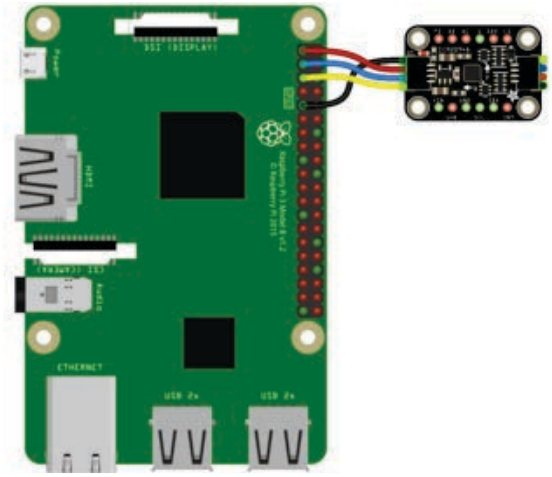


Board 3V to **sensor VIN (red wire)**
Board GND to **sensor GND (black wire)**
Board SCL to **sensor SCL (yellow wire)**
Board SDA to **sensor SDA (blue wire)**

Python Computer Wiring

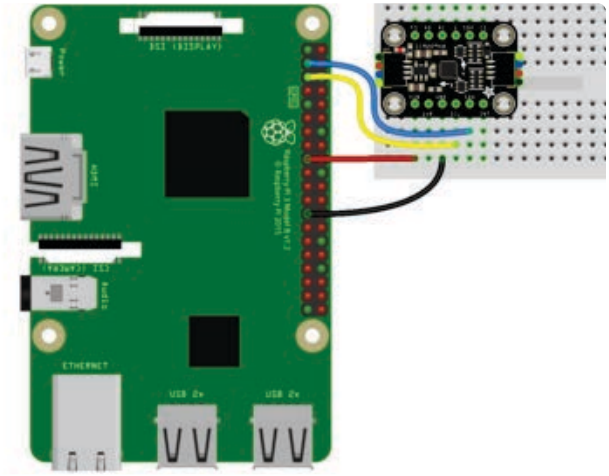
Since there's dozens of Linux computers/boards you can use, we will show wiring for Raspberry Pi. For other platforms, please visit the guide for CircuitPython on Linux to see whether your platform is supported (https://adafru.it/BSN).

Here's the Raspberry Pi wired to the sensor using I2C and a **STEMMA QT** (https://adafru.it/Ft4) connector:



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)**

Finally here is an example of how to wire up a Raspberry Pi to the sensor using a solderless breadboard



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 (https://adafru.it/LAp) 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_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 (https://adafru.it/Awz)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 (https://adafru.it/BSN)!

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 temperature and humidity measurements from the board's Python REPL.

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

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
- **magnetic** - The magnetic forces on the X, Y, and Z axes in micro-Teslas (uT)

For example, to print out the acceleration, gyro and magnetic measurements use this code:

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.ICM20948(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("Magnetometer X:%.2f, Y: %.2f, Z: %.2f uT" % (icm.magnetic))
    print("")
    time.sleep(0.5)
```

Python Docs

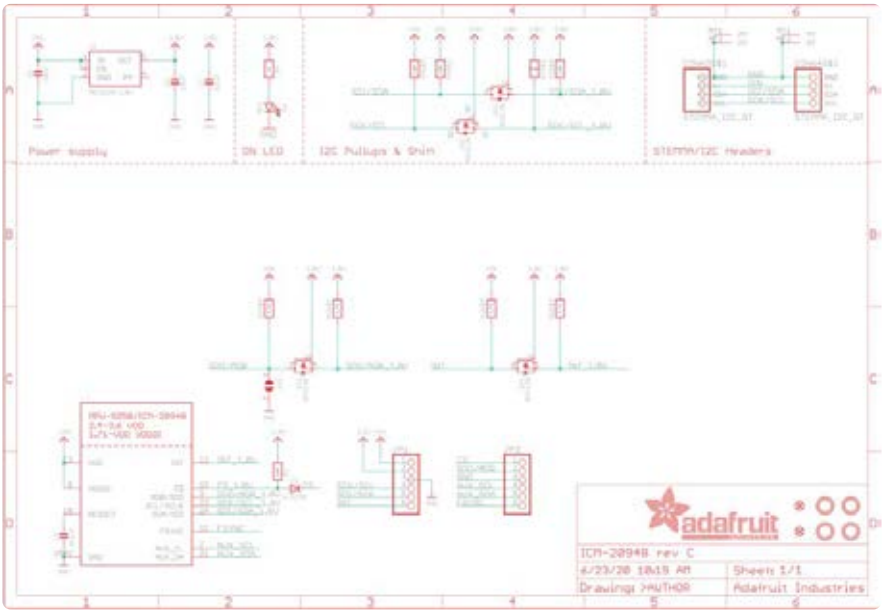
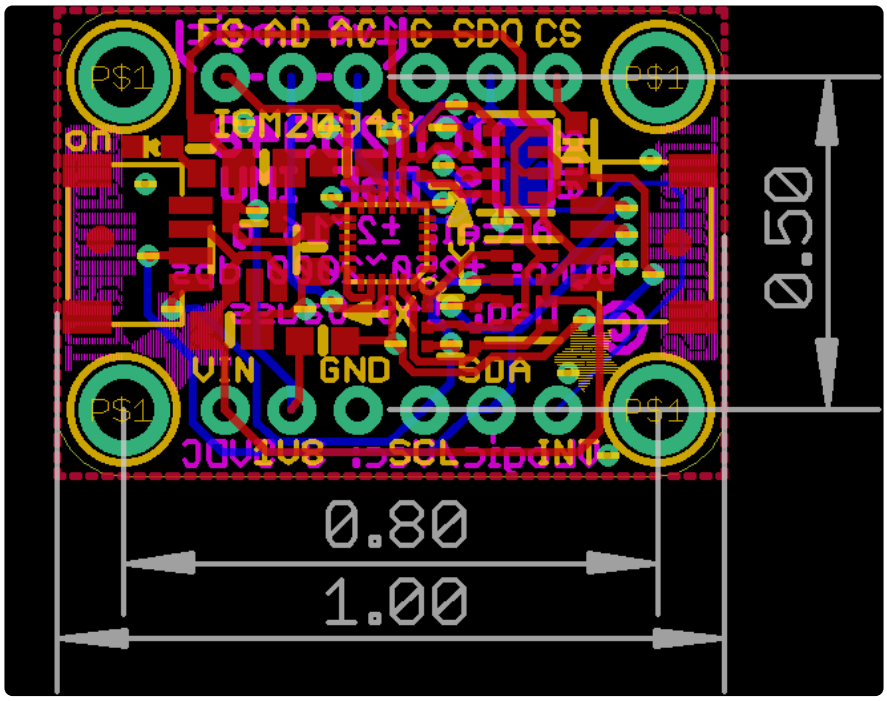
Python Docs (https://adafru.it/MC1)

Downloads

Files

- ICM20948 Datasheet (https://adafru.it/MC5)
- EagleCAD files on GitHub (https://adafru.it/MC7)
- Fritzing object in the Adafruit Fritzing Library (https://adafru.it/MC8)

Schematic



Fab Print