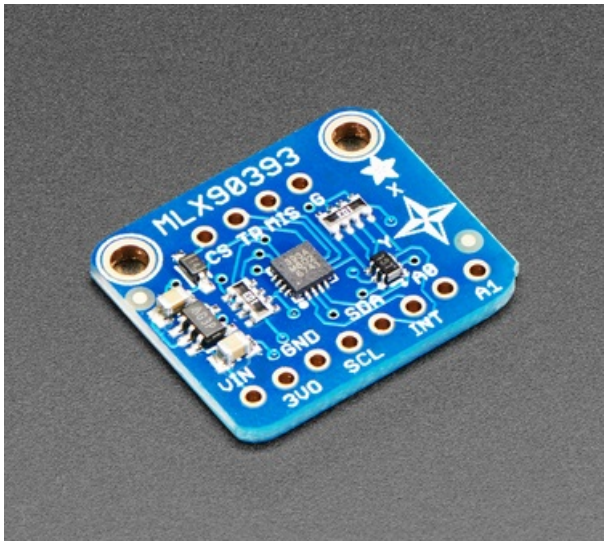
adafruit learning system

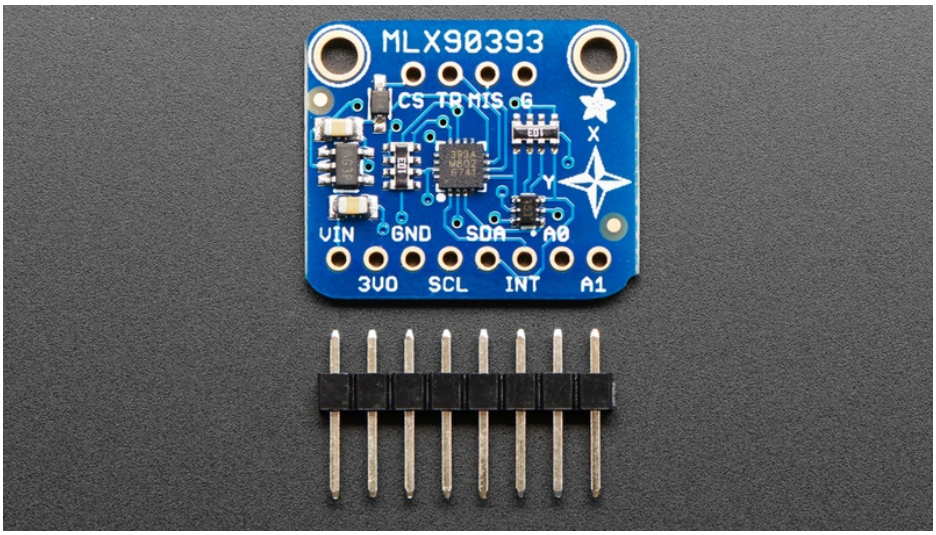
# MLX90393 Wide-Range 3-Axis Magnetometer

Created by Kevin Townsend



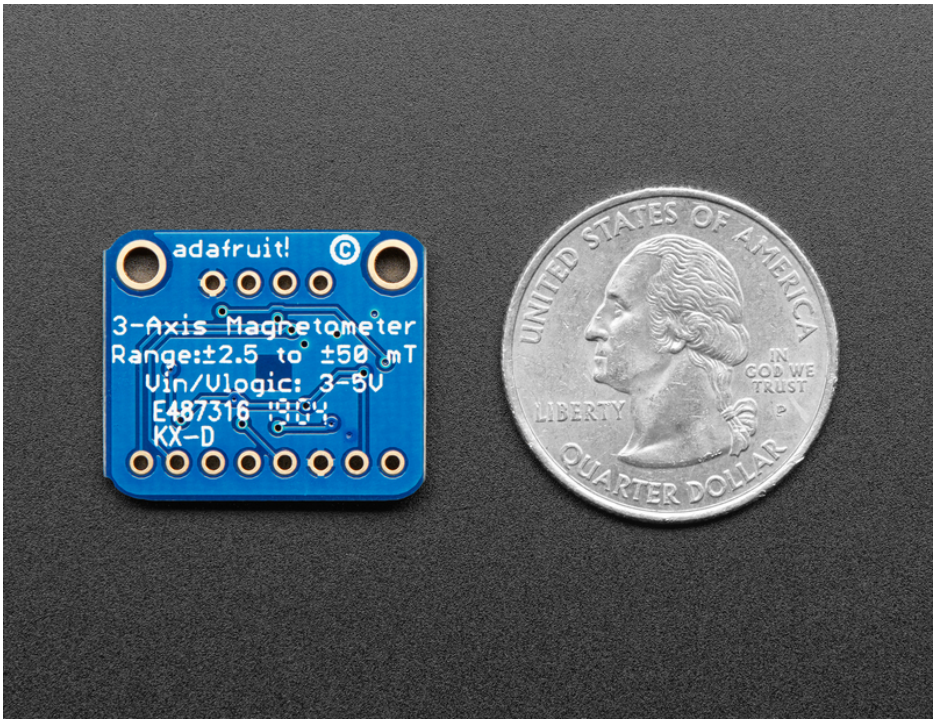

Last updated on 2019-03-29 06:37:59 PM UTC

# Overview



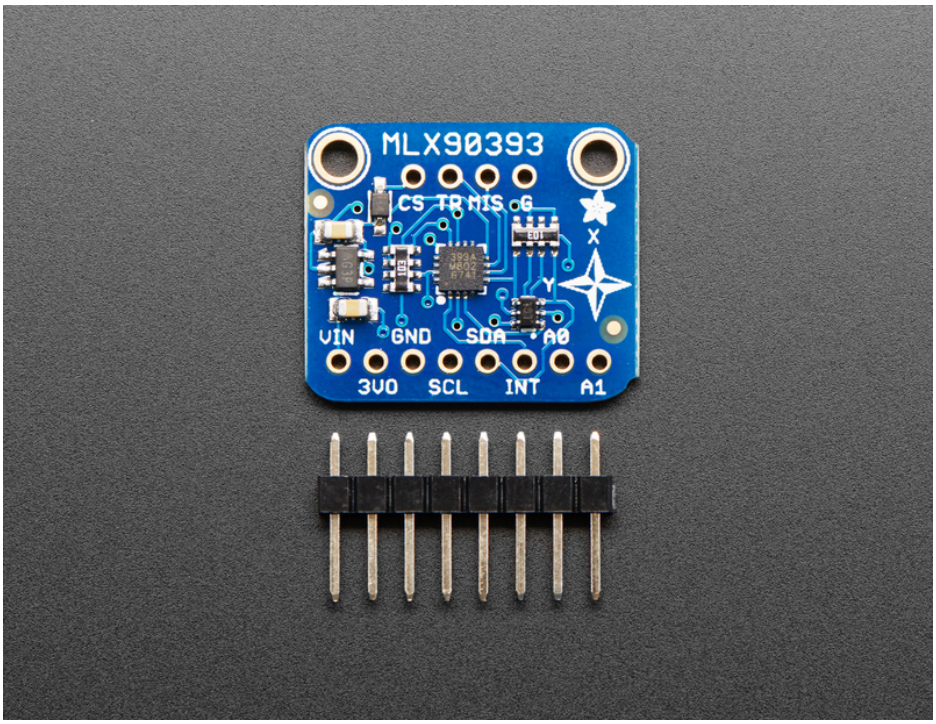

Measure the invisible magnetic fields that surround us, with this wide-range magnetometer. The MLX90393 is a wide range magnetic field sensor, that can measure 16-bits in ranges from +/-5mT up to +/-50mT in all 3 axes.

Compared to most magenetometers, this gives a huge range, which makes it excellent for detecting *magnets* and magnetic orientation, rather than the Earths magnetic field. (Magnets have a much stronger field that overwhelms most magnetometers that would normally be used for orientation with respect to the North Pole)




To make it easy to use, we've placed this tiny little sensor onto a breakout board, with a 3.3V power supply and level shifter. This makes it easy to use with any 3 or 5V microcontroller. Our Arduino and CircuitPython code will get you started in a jiffy, with I2C communication to the sensor. You will be readin' out those Gauss's in minutes! With the address select pins you can have up to 4 sensors on one I2C bus.

Comes as a fully assembled and tested breakout board with a small piece of header for use with a breadboard.




## Specifications

The MLX90393 has the following key technical specifications:

- 16-bit output on all three (XYZ) magnetic field sensors
- An unusually large dynamic range of **5-50 mT** (1 mT or millitesla = 10 G or Gauss).
  By comparison, the LSM303DLHC saturates at +/-8.1 G (0.81 mT) at maximum range setting.
- Up to ~500 Hz sample rate [1]
- User-adjustable I2C address to allow multiple sensors in your project (two I2C ADDR pins for four possible I2C addresses).

[1] Based on `OSR=0, DIG_FILT=2, HALLCONF=0xC` for **1.84ms** conversion time. See 15.1.5 HALLCONF [3:0} in the datasheet for details.

## Pinout

The MLX90393 3-Axis magnetometer breakout has the following pins:


## Power Pins

This breakout board can be run on **3.3V** and **5V** systems, although only the **SCL** and **SDA** lines are 5V safe (other pins like INT will need to be manually level-shifted by you).

- **VIN** - This is the input to the 3.3V voltage regulator, which makes it possible to use the 3.3V sensor on 5V systems. It also determines the logic level of the SCL and SDA pins. Connect this to **3.3V** on the MCU for 3.3V boards (Adafruit Feathers), or **5.0V** for 5V Arduinos (Arduino Uno, etc.).
- **3VO** - This is the **OUTPUT** of the 3.3V regulator, and can be used to provide 3.3V power to other parts of your project if required (<100mA).
- **GND** - Connect this to the **GND** pin on your development board to make sure they are sharing a common GND connection, or the electrons won't have anywhere to flow!

NOTE: Only SCL and SDA are 5V safe on this board. Using any other pins on a 5V system will require manual level shifting of the pins used (INT, etc.)

## Digital Pins

- **SCL** - The clock line on the I2C bus. This pin has an internal pullup resistor on the PCB, which is required as part of the I2C spec, meaning you don't need to add one externally yourself.
- **SDA** - The data line on the I2C bus. This pin has an internal pullup resistor on the PCB, which is required as part of the I2C spec, meaning you don't need to add one externally yourself.
- **INT** - This INT pin can be configured to 'fire' whenever a new data sample is read in single measurement mode, which is what the device boots up to by default in our driver.
- **A0** and **A1**: These two pins are LOW (0) by default, but if you need to connect multiple MLX90393s to your MCU or resolve an address conflict, you can adjust the logic on these two pins which will change the last two bits of the I2C address that this breakout responds to. Most of the time, you will simply **leave these disconnected**.

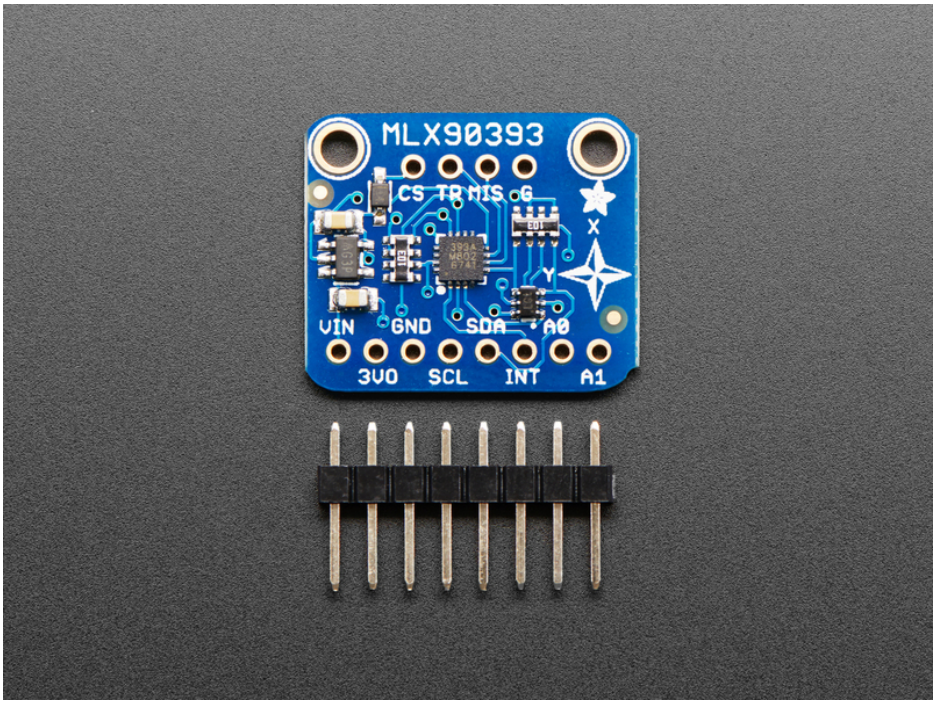
On the top row, you have a few more rarely used specialty pins, though normally you will simply leave these unsoldered and unconnected:

- **CS (AKA SENB)** - This is used to set the sensor in I2C or SPI mode, but **this breakout only supports I2C** so the pin is provided purely for testing purposes unless you want to dig into writing your own SPI driver and making the required changes to the PCB.
- **TR** (AKA **INT/TRG**) - This pin can optionally be setup to fire an interrupt in addition to the **INT** pin, and is basically

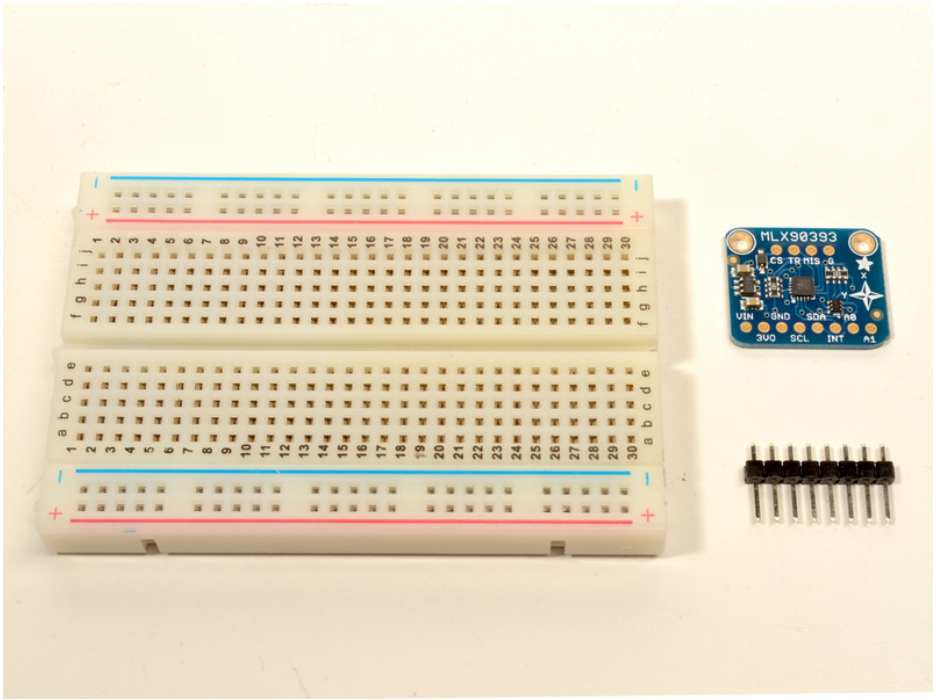
a mirror on INT when configured as such.

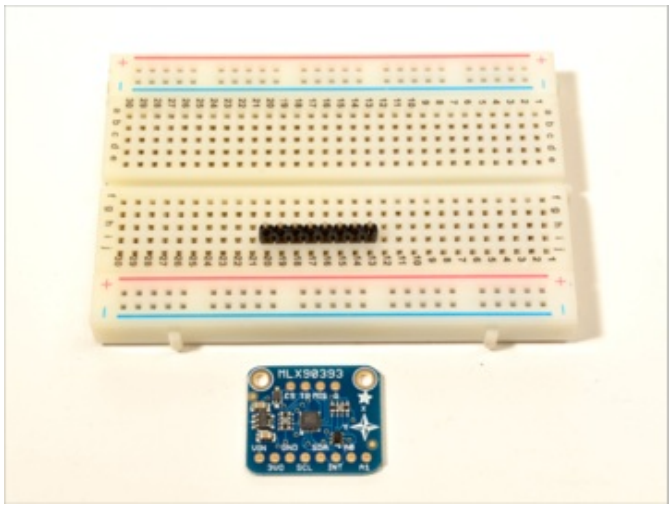
- **MIS** (AKA **TRIG**) - This is used as MISO on the SPI bus, but SPI isn't supported on this breakout, so it is only provided to testing purposes.
- **G** - This is simply an additional GND pin, and can be left unconnected if not required.

## Assembly



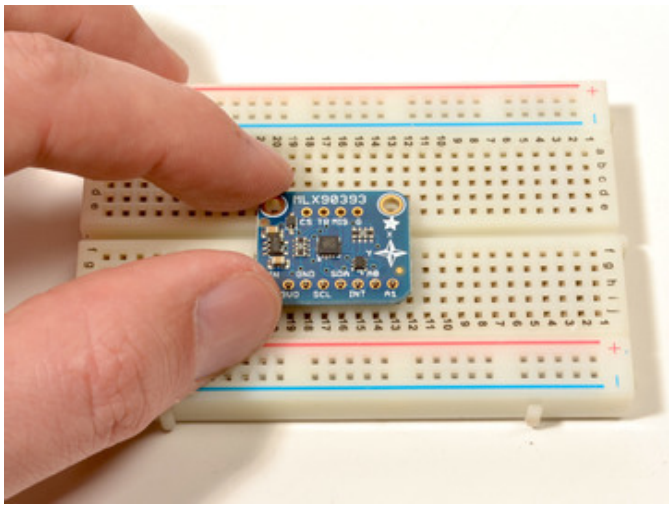

## Assembly





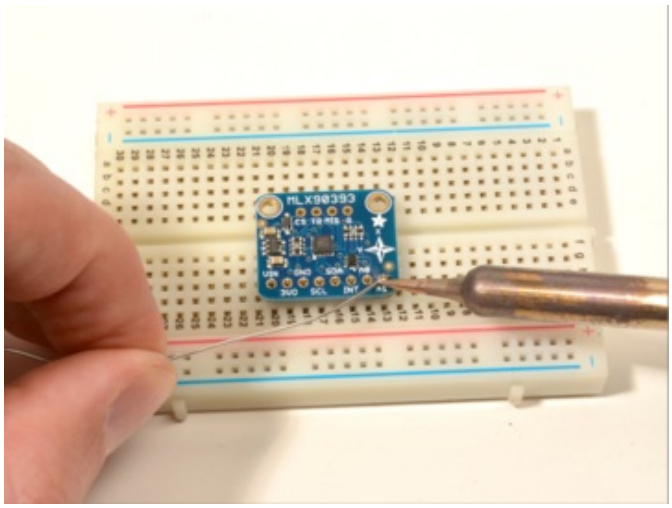
## Prepare the header strip:

Cut the strip to length if necessary. It will be easier to solder if you insert it into a breadboard - **long pins down**.



## Add the MLX90393:

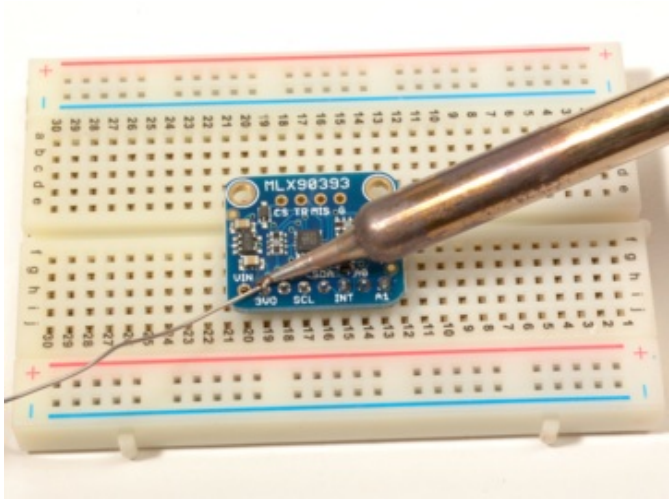
Place the board over the pins so that the short pins poke through the top of the breakout pads.




## And Solder!

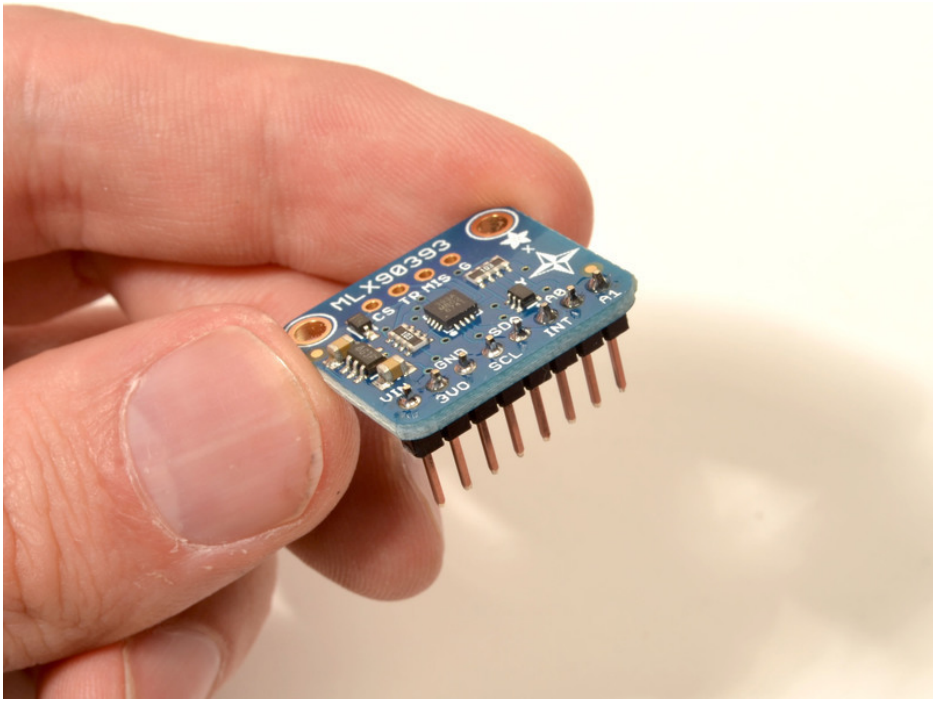
Be sure to solder all pins for reliable electrical contact.

*(For tips on soldering, be sure to check out the Guide to Excellent Soldering (https://adafru.it/aTk).*




OK, you're done!

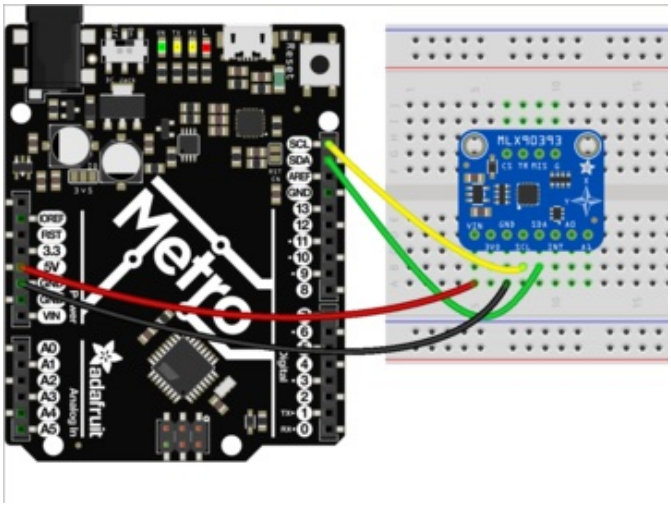
You can now plug in your Magnetometer and start measuring some magnetic fields!


MLX90393
CS
SDA
SCL
INT
A1
3VO

# Arduino

## Wiring

Hooking up the MLX90393 to your Feather or Arduino is easy:



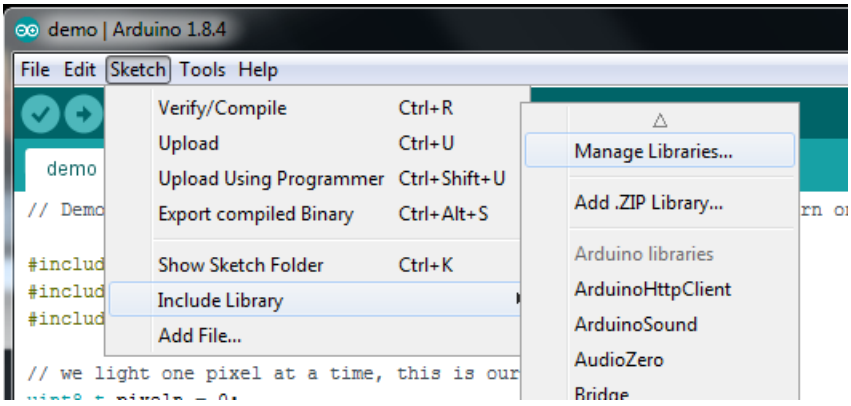
- If you are running a Feather (3.3V), connect the **3V** pin to **VIN** on the MLX90393
- If you are running a 5V Arduino (Uno, etc.), connect **5V** to **VIN** on the MLX90393
- Connect **GND** on the MCU to **GND** on the MLX90393
- Connect the **SCL** pins together ...
- ... and finally connect the **SDA** pins together

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

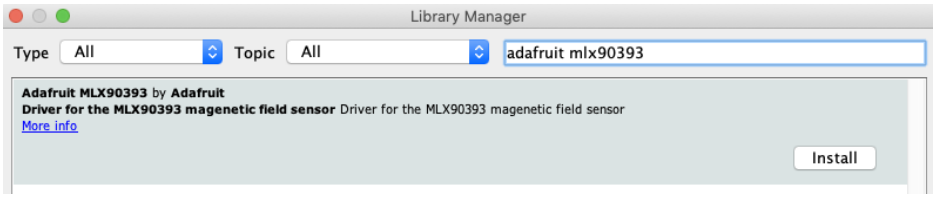
Only the SCL and SDA pins on the MLX90393 are level shifted and safe to use on 5V systems like the Arduino Uno. If you are using other pins on the breakout (INT, etc.) on a 5V system, you will need to level shift these yourself. We have some tutorials on how to do this in the learning system, simply search for 'level shifting'!

## Installation

You can install the **Adafruit MLX90393 Library** for Arduino using the Library Manager in the Arduino IDE:




Click the **Manage Libraries ...** menu item, search for **Adafruit MLX90393**, and select the **Adafruit MLX90393** library:




## Load Example

Open up **File -> Examples -> Adafruit MLX90393 -> basicdemo** and upload to your Arduino wired up to the sensor

Upload the sketch to your board and open up the Serial Monitor (**Tools->Serial Monitor**). You should see the temperature in magnetic field values for X/Y/Z.

## Example Code

The following example code is part of the standard library, but illustrates how you can retrieve sensor data from the MLX90393 for the X, Y and Z axis:

```
#include <Wire.h>

#include "Adafruit_MLX90393.h"

Adafruit_MLX90393 sensor = Adafruit_MLX90393();

void setup(void)
{
  Serial.begin(9600);

  /* Wait for serial on USB platforms. */
  while(!Serial) {
      delay(10);
  }

  Serial.println("Starting Adafruit MLX90393 Demo");

  if (sensor.begin())
  {
    Serial.println("Found a MLX90393 sensor");
  }
  else
  {
    Serial.println("No sensor found ... check your wiring?");
    while (1);
  }
}

void loop(void)
{
    float x, y, z;

    if(sensor.readData(&x, &y, &z)) {
        Serial.print("X: "); Serial.print(x, 4); Serial.println(" uT");
        Serial.print("Y: "); Serial.print(y, 4); Serial.println(" uT");
        Serial.print("Z: "); Serial.print(z, 4); Serial.println(" uT");
    } else {
        Serial.println("Unable to read XYZ data from the sensor.");
    }

    delay(500);
}
```

You should get something resembling the following output when you open the Serial Monitor at 9600 baud:


## Setting the Gain

The driver will default to 1x gain, but if you wish to adjust the gain you can do so using the `setGain(enum

`mlx90393_gain gain)` function, passing in one of the following values:

```
enum  mlx90393_gain {
    MLX90393_GAIN_5X          = (0x00),
    MLX90393_GAIN_4X,
    MLX90393_GAIN_3X,
    MLX90393_GAIN_2_5X,
    MLX90393_GAIN_2X,
    MLX90393_GAIN_1_67X,
    MLX90393_GAIN_1_33X,
    MLX90393_GAIN_1X
};
```

For example, to set the gain to 2x you would call the function as follows:

```
sensor.setGain(MLX90393_GAIN_2X);
```

Resolution is managed internally in the driver itself, and defaults to the maximum value of +/- 2^15 LSBs.

## Arduino API

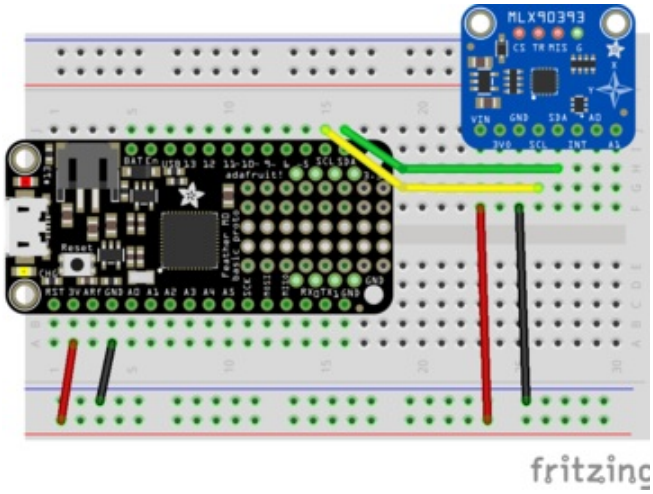
Arduino API (https://adafru.it/DQK)

## Python and CircuitPython

Using the MLX90393 with CircuitPython is easy. The Adafruit_CircuitPython_MLX90393 (https://adafru.it/DQL) repo on Github always contains the latest public code, and allows you to quickly and easily get started with the Adafruit MLX90393 breakout board.

## CircuitPython Wiring

The diagram below shows how you can wire up your CircuitPython board (in Feather form below) to the MLX90393:



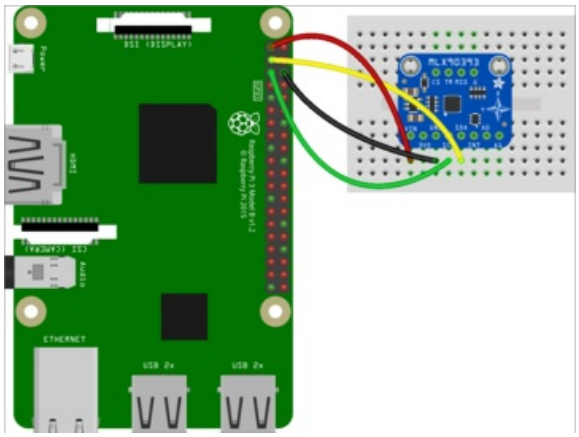

Simple connect:

- **3V** on the dev board to **VIN** on the MLX90393
- **GND** on the dev board to **GND** on the MLX90393
- **SCL** on the dev board to **SCL** on the MLX90393
- **SDA** on the dev board to **SDA** on the MLX90393

## Python 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).

Make the following connections between the Pi and the MLX90393:



Simple connect:

- **3.3V** on the RPi to **VIN** on the MLX90393
- **GND** on the RPi to **GND** on the MLX90393
- **SCL** on the RPi to **SCL** on the MLX90393
- **SDA** on the RPi to **SDA** on the MLX90393

## Library Installation

You'll need to install the Adafruit CircuitPython MLX90393 (https://adafru.it/DQL) 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/uap). 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_mlx90393.mpy**
- **adafruit_bus_device**

Before continuing make sure your board's lib folder or root filesystem has the **adafruit_mlx90393.mpy,** and **adafruit_bus_device** files and folders copied over.

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

## Python Installation of the MLX90393 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-mlx90393`

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!

## Example

The following example shows a minimal python script to make use of the MLX90393.

Running this example, a three axis magnetic field measurement will be read every second, and displayed via the serial output. A timestamp will precede each sample, and if any error condition was encountered during the read attempt, the details of the error code will be displayed:

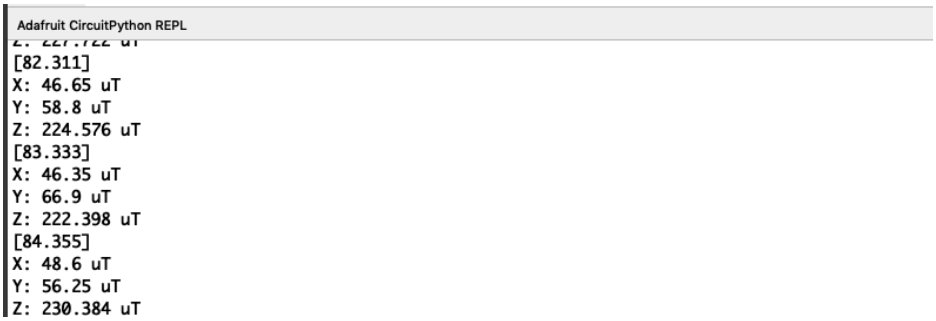
```
import time
import busio
import board

import adafruit_mlx90393

I2C_BUS = busio.I2C(board.SCL, board.SDA)
SENSOR = adafruit_mlx90393.MLX90393(I2C_BUS, gain=adafruit_mlx90393.GAIN_1X)

while True:
    MX, MY, MZ = SENSOR.magnetic
    print("[{}]".format(time.monotonic()))
    print("X: {} uT".format(MX))
    print("Y: {} uT".format(MY))
    print("Z: {} uT".format(MZ))
    # Display the status field if an error occured, etc.
    if SENSOR.last_status > adafruit_mlx90393.STATUS_OK:
        SENSOR.display_status()
    time.sleep(1.0)
```

If you open the **Serial** tab in mu-editor with the sample code running, you should get output resembling the following if everything was setup correctly:




## Adjusting Gain

The gain on the sensor can be adjusted via the `.gain` property, as shown below:

```
SENSOR.gain = adafruit_mlx90393.GAIN_2X
```

**Alternatively**, you can set the gain property via the constructor, as shown below:

```
SENSOR = adafruit_mlx90393.MLX90393(I2C_BUS, gain=adafruit_mlx90393.GAIN_1X)
```

The value assigned to the `gain` property can be one of the following entries:

- GAIN_5X
- GAIN_4X
- GAIN_3X
- GAIN_2_5X
- GAIN_2X
- GAIN_1_67X

- GAIN_1_33X
- GAIN_1X

Resolution is managed internally in the driver itself, and defaults to the maximum value of +/- 2^15 LSBs.

## Python Docs

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

# Downloads

## Downloads and Design Files

- Board Files on Github (https://adafru.it/DQN)
- Manufacturer's Product Page (https://adafru.it/DQO)
- Arduino Library on Github (https://adafru.it/DQP) (for PRs and driver development!)
- CircuitPython Library on Github (https://adafru.it/DQL)
- Fritzing Part (https://adafru.it/DQQ)

## Datasheet

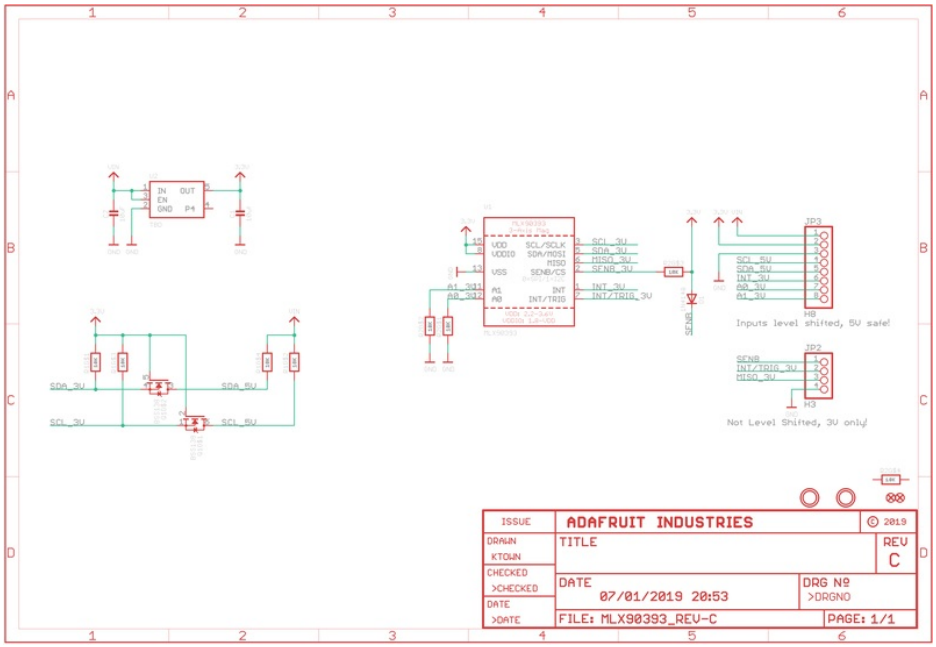
You can download the datasheet for this chip using the following link:

https://adafru.it/DQR

https://adafru.it/DQR

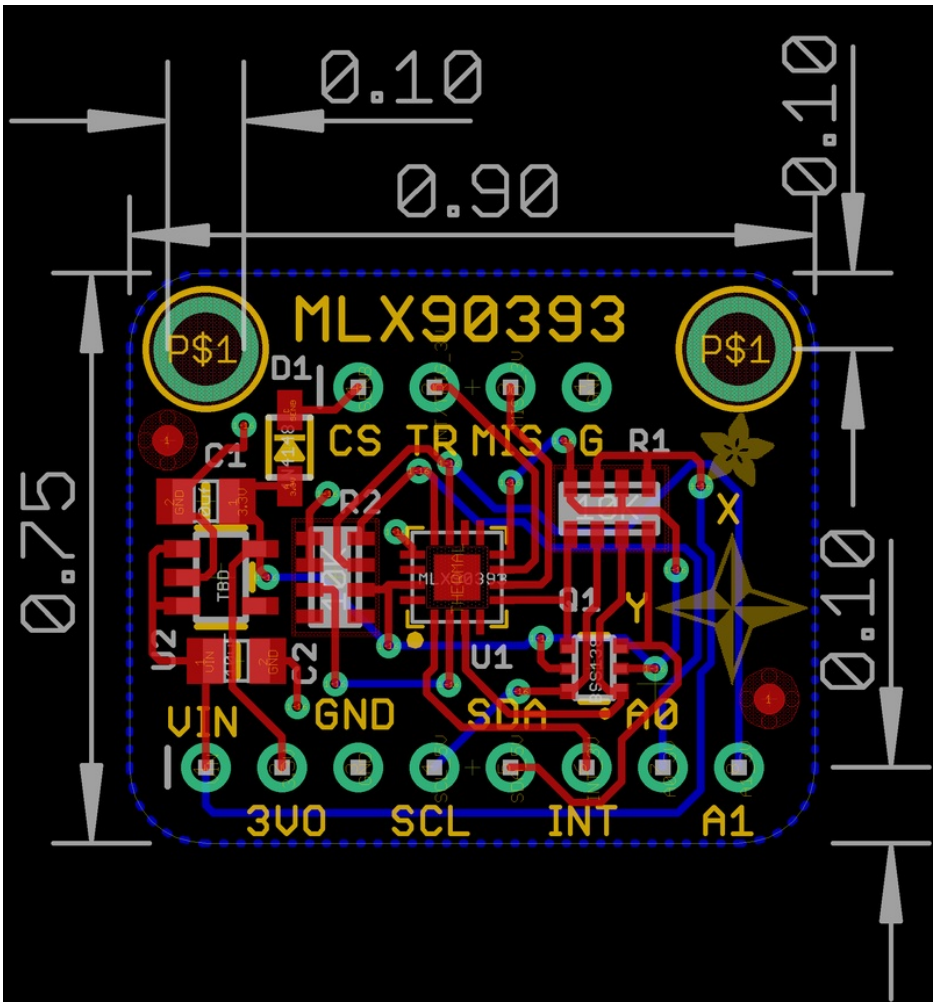
## Schematic

The schematic for this breakout is available below:




## Board Layout

The breakout has the following physical dimensions:


0.10
0.10
0.90
0.75
0.10
MLX90393
P$1
P$1
D1
CS
TRIG
R1
C1
R2
X
Y
Q1
U1
C2
U2
TBD
VIN
GND
SDA
A0
3VO
SCL
INT
A1