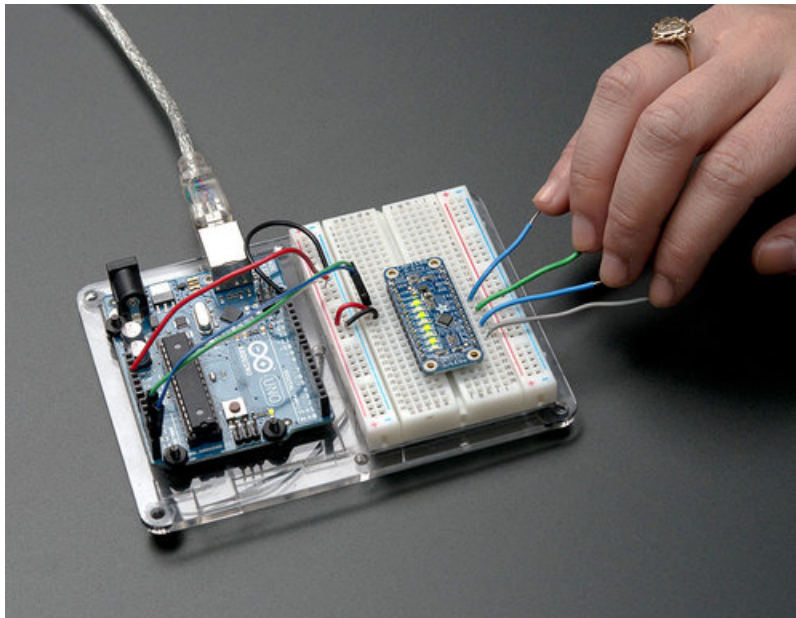


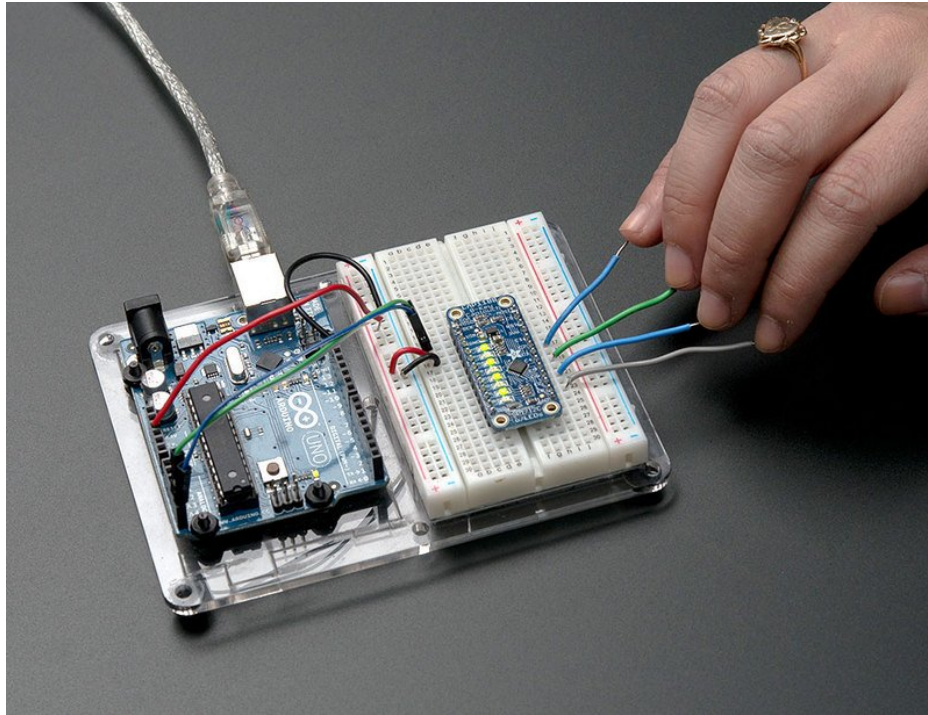
Adafruit CAP1188 Breakout

Created by lady ada



Last updated on 2019-11-15 06:56:15 PM UTC

Overview



Add lots of touch sensors to your next microcontroller project with this easy-to-use 8-channel capacitive touch sensor breakout board, starring the CAP1188. This chip can handle up to 8 individual touch pads, and has a very nice feature that makes it stand out for us: it will light up the 8 onboard LEDs when the matching touch sensor fires to help you debug your sensor setup.

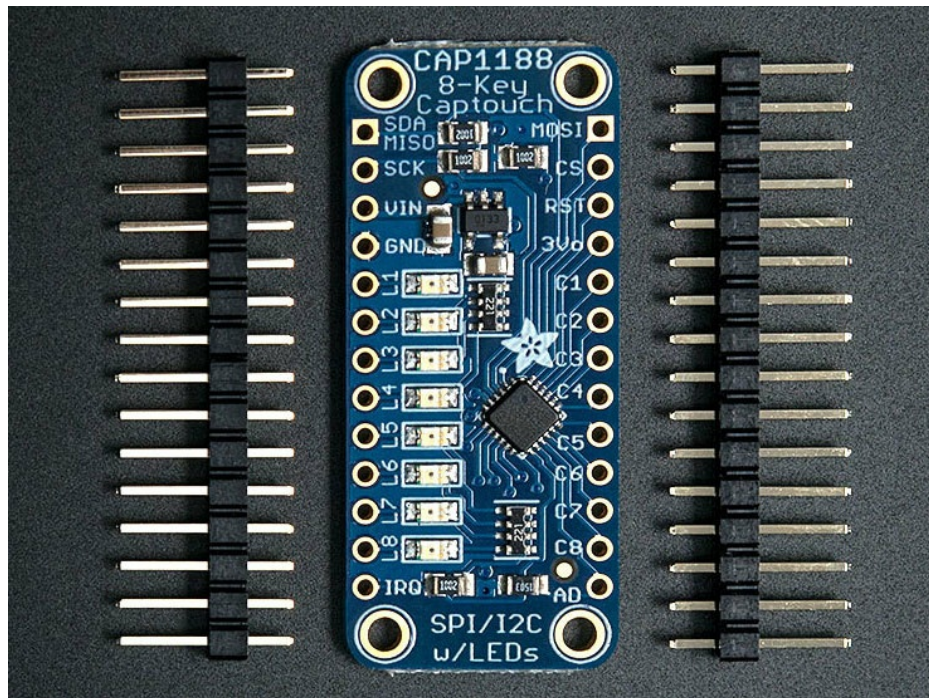
The CAP1188 has support for both I2C and SPI, so it's easy to use with any microcontroller. If you are using I2C, you can select one of 5 addresses, for a total of 40 capacitive touch pads on one I2C 2-wire bus. Using this chip is a lot easier than doing the capacitive sensing with analog inputs: it handles all the filtering for you and can be configured for more/less sensitivity.

Comes with a fully assembled board, and a stick of 0.1" header so you can plug it into a breadboard. For contacts, we suggest using copper foil, then solder a wire that connects from the foil pad to the breakout.

Getting started is a breeze with our Arduino library and tutorial. You'll be up and running in a few minutes, and if you are using another microcontroller, it's easy to port our code.

Pinouts

The CAP1188 has a lot going on, so much so that we had to make the breakout double-sided! It fits nicely into a breadboard and has the sensors all in a row on one side and if you're using plain i2c you can connect to to the left side only



Here's all the pins and what they do!

Power pins

VIN and **GND** are power in pins, you can use 3-5VDC so its great for any kind of microcontrollers. There's an on-board 3V regulator as well, the output is available on the **3Vo** pin (you can snag up to 150mA)

I2C interface pins

For I2C, connect to the **SCK** (i2c clock a.k.a SCL) and **SDA** (i2c data) pins. These are 5V safe so you can use them with 3V or 5V logic

SPI inteface pins

If you want to use SPI instead, you'll be using the SCK, MOSI, MISO and CS pins

Other interfacing pins

The **AD** pin is used to select SPI or I2C interface, and if I2C what address to use. See the wiring page for more details
The **IRQ** pin goes *low* when a pin is touched. We don't use it on our code examples, but if you want to have an interrupt pin used, connect it to this IRQ pin and use active-low triggering.

The **RST** pin is used to reset the chip, either in I2C or SPI mode. It's optional but using it will make the system more reliable so we suggest it.

Sensor input pins

This is the part you touch - there are 8 individual capacitive sensor pins, called **C1** thru **C8**. On restart the system

recalibrates them so don't touch these when powering up!

Sensor output pins

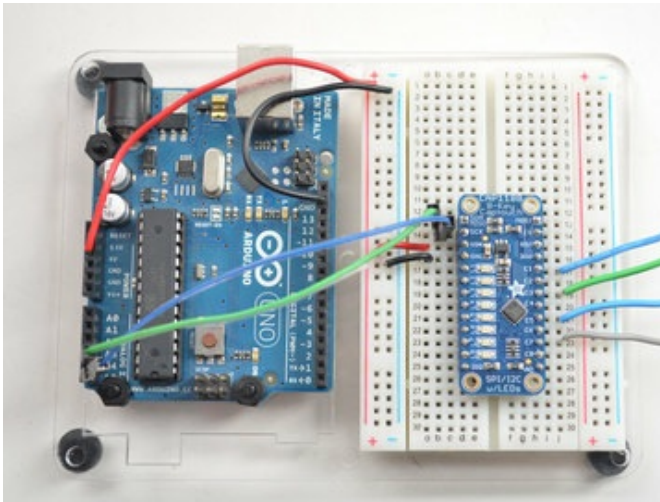
The **L1** thru **L8** pins are the LED driver/sensor output pins. The indicators are really useful for debugging your touch sensor system, but you can also use the indicator output pins for triggering some other electronics. Each **L** pin corresponds to the matching **C** sensor input. These pins are 3V normally, and drop to 0V when triggered.

Wiring

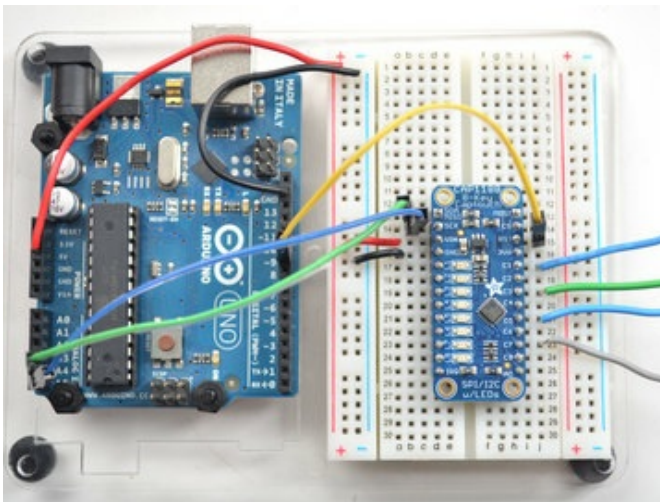
Wiring for use with I2C

Chances are you'll use this board with the I2C interface pins. I2C is handy because you can have multiple sensors all connected on two I2C pins and share them nicely as long as each one has a unique address. I2C isn't particularly fast but that's fine for a sensor like this one (it's not good for video or audio type data)

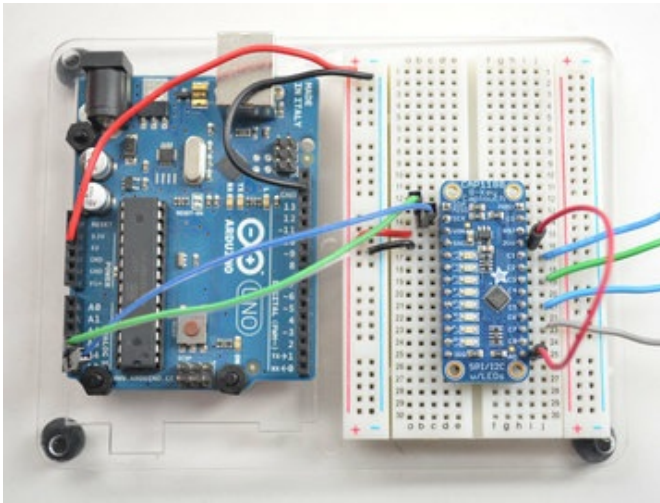
I2C is really fast to get started, so we suggest that. Connect the **VIN** pin to 5V and **GND** pin to ground. Then connect the **SDA** pin to your I2C SDA/data line and **SCL** pin to your I2C SCL/clock line



- On UNO/Duemilanove/etc, SDA == Analog 4, SCL == Analog 5
- On Leonardo/Micro, SDA == Digital 2, SCL == Digital 3
- On Mega/ADK/Due, SDA == Digital 20, SCL == Digital 21



The reset pin is not required for use, but if you can spare a pin, it will make the system a little more rugged - the Arduino will do a hard reset of the CAP1188 on startup, connect **RST** to any digital I/O pin. In the library example code you can set up that pin in the CAP1188 object creation.



If you're using multiple sensors, or you just want to change the I2C address to something else, you can choose from 5 different options - **0x28**, **0x29** (default), **0x2A**, **0x2B**, and **0x2C**

The I2C address are selected by connecting a resistor to the **AD** pin in the lower right: different resistors set a different address. The easiest address to set is **0x28** which is just a wire from **AD** to the **3Vo** pin.

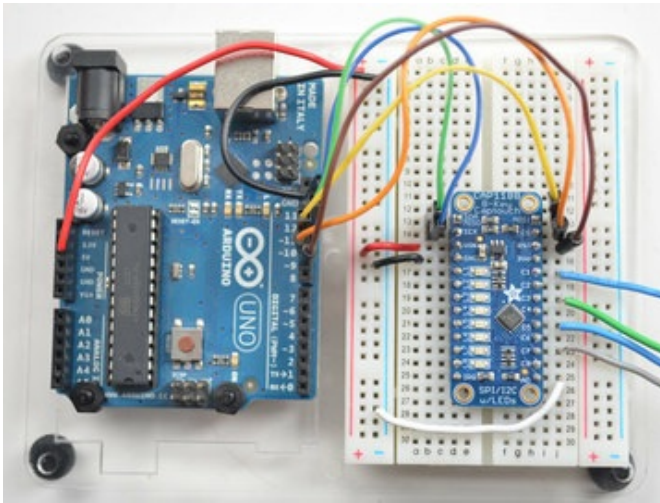
If you want to set the I2C address to a different value you'll need to connect a resistor from **AD** to ground instead of a wire to 3V. The datasheet talks about different resistor values in table 3.1 - but since the breakout board already has a 150K resistor on it, you'll need to use different values so that the parallel resistance comes out right. Here's the values you should use:

- Wire connecting **AD** to **3V** -> I2C address **0x28**
- No resistor or wire attached to **AD** -> I2C address **0x29**
- 600K resistor from **AD** to ground -> I2C address **0x2A**
- 300K resistor from **AD** to ground -> I2C address **0x2B**
- 180K resistor from **AD** to ground -> I2C address **0x2C**

Wiring for use with SPI

You can put the CAP1188 in SPI mode by powering it up with **AD** connected to ground. Then the 4 SPI pins are used to communicate instead of I2C. SPI may be preferable for your project if you have an I2C address collision (which is unlikely given you can choose 5 addresses!) or say if you have an UNO and you want to use the I2C pins for analog input instead of I2C, or if you are porting to a microcontroller that does not have hardware I2C.

Either way, SPI is there for you!



Connect **Vin** to 5V and **GND** to ground, then tie a wire from **AD** to ground as well. Now connect the **SCK**, **MISO**, **MOSI**, **CS** and **RST** pins to your microcontroller. If using an Arduino you can use either the hardware SPI pins which are fixed for each Arduino, or go with software SPI where you select any 5 digital I/O pins

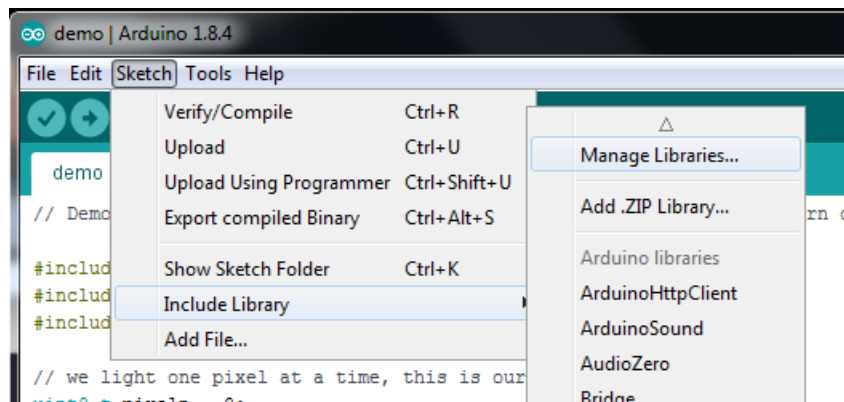
Using with Arduino

Its super easy to use this sensor board with an Arduino thanks to the great Adafruit library. Once you've installed the library you can connect the sensor board via I2C or SPI to your Arduino, it will work with any kind or flavor. If you're using a different kind of microcontroller, the library is a good reference to help you port the code over.

Download the library

First up, we'll download the Arduino library from the Arduino library manager.

Open up the Arduino library manager:



Search for the **Adafruit CAP1188** library and install it

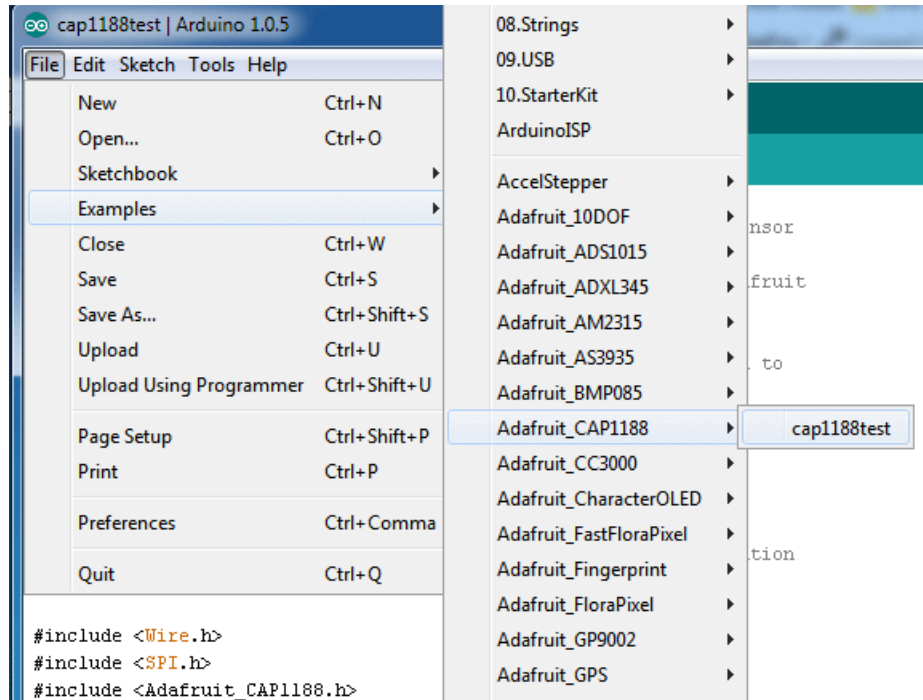


We also have a great tutorial on Arduino library installation at:

<http://learn.adafruit.com/adafruit-all-about-arduino-libraries-install-use> (<https://adafru.it/aYM>)

Run Test Sketch

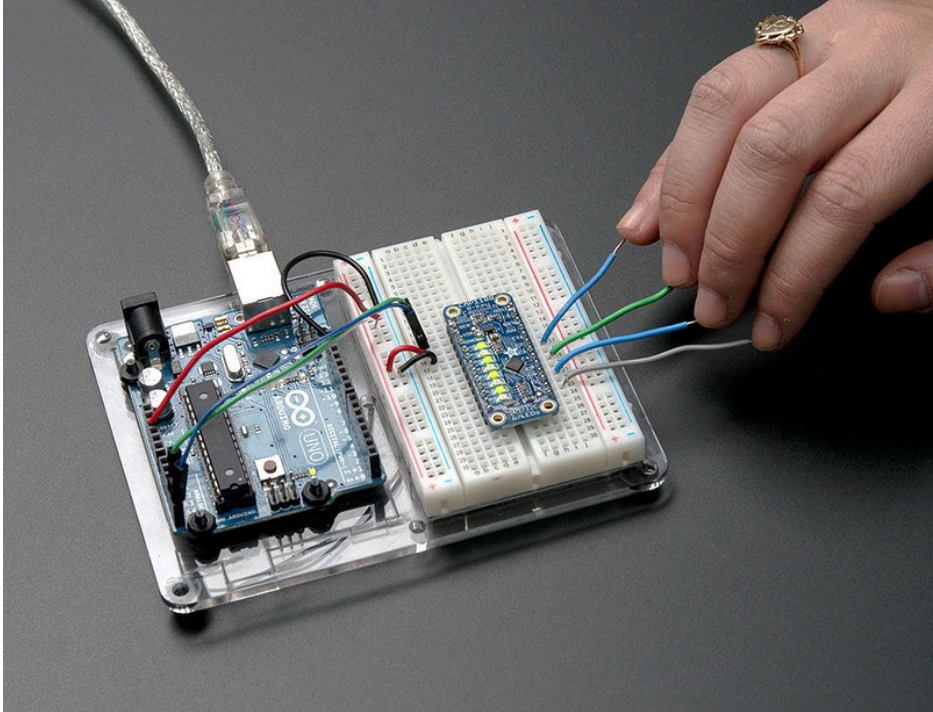
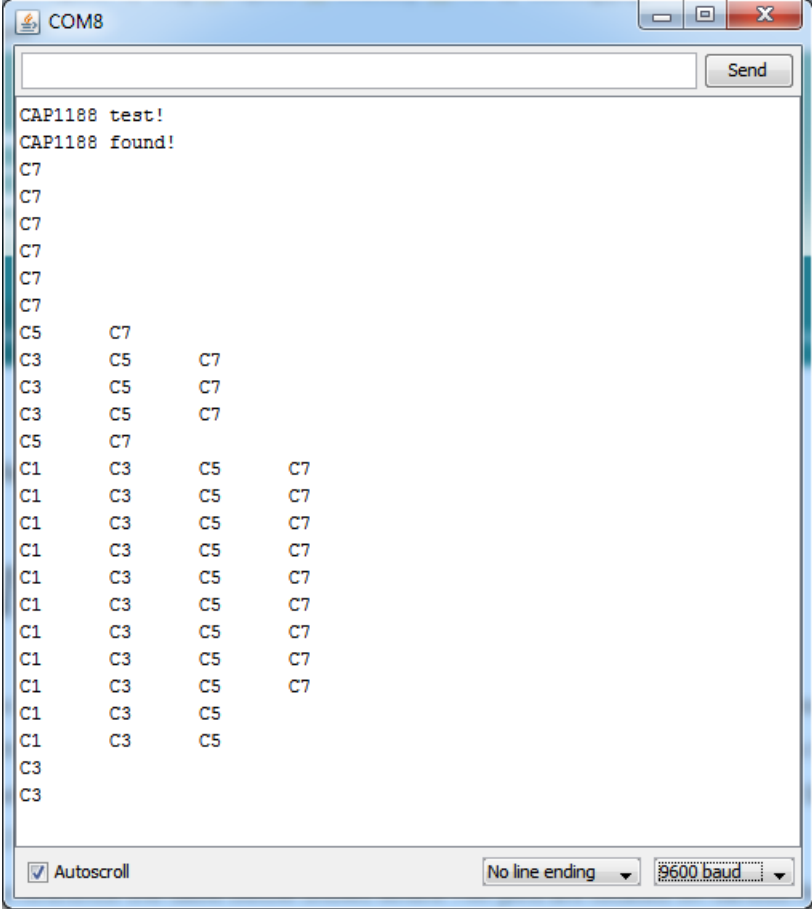
After you've restarted, you should be able to load up the **File->Examples->Adafruit_CAP1188->cap1188test** sketch



Wire up the sensor as shown in the Wiring section, for I2C. Connect the **GND** pin to ground, **VIN** pin to 5V and connect **SDA** to your Arduino's **SDA** pin, **SCL** to **SCL** pin

- On UNO/Duemilanove/etc, SDA == Analog 4, SCL == Analog 5
- On Leonardo/Micro, SDA == Digital 2, SCL == Digital 3
- On Mega/ADK/Due, SDA == Digital 20, SCL == Digital 21

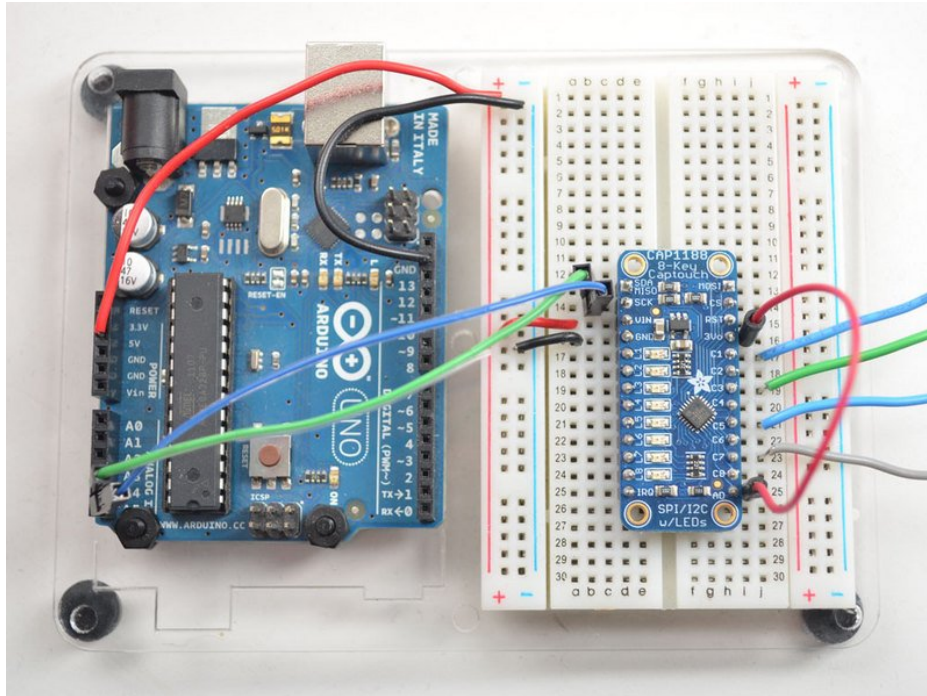
Upload the sketch and open up the serial console at 9600 baud. You should see that the CAP1188 is found (good!) and then you can touch the C1 thru C8 pads with your fingers to see the touch sensor go off. When you touch a pin, you'll also see the matching LED light up.



I2C with a different address

If you're using multiple sensors, or you just want to change the I2C address to something else, you can choose from 5 different options - **0x28**, **0x29** (default), **0x2A**, **0x2B**, **0x2C** and **0x2D**

The I2C address are selected by connecting a resistor to the **AD** pin in the lower right: different resistors set a different address. The easiest address to set is 0x28 which is just a wire from **AD** to the **3Vo** pin.



Now look in the test sketch for the lines

```
if (!cap.begin()) {  
  Serial.println("CAP1188 not found");  
  while (1);  
}
```

And change the **cap.begin()** statement to **cap.begin(0x28)** to initialize it to use the **0x28** address. Don't forget to do a full power off of the Arduino & sensor, the sensor only detects the i2c address on power up so you can't change it on the fly!

Using with SPI

You can also use SPI if you wish. The library supports both hardware SPI (using the 'hardware SPI' port on your arduino) or software/bit-bang SPI, where you can define the pins. In general, these sensors are not very fast so I2C is a good way to interface them but if you wish, SPI is there for you too! For example, if you want to have more than 5 of these connected to one board, that is possible to do with SPI, but the I2C interface on this chip doesn't support that many shared I2C addresses.

To enable SPI, be sure to wire for SPI as shown in the Wiring section, and do a power reset of your board so that the chip 'wakes up' in SPI mode

If you're using hardware SPI, check the [SPI page](https://adafruit.it/d5h) for what pins you need to use, sometimes they are only on the ICSP header which makes them trickier to use. (<https://adafruit.it/d5h>)

To enable the hardware SPI interface, create the Adafruit_CAP1188 object with

```
Adafruit_CAP1188 cap = Adafruit_CAP1188(CAP1188_CS, CAP1188_RESET);
```

If you are using software SPI, you can use *any* pins that are not already used for some purpose. In that case, create the object with

```
Adafruit_CAP1188 cap = Adafruit_CAP1188(CAP1188_CLK, CAP1188_MISO, CAP1188_MOSI, CAP1188_CS, CAP1188_RESET);
```

Using the external IRQ Interrupt

Arduino has some basic ability to attach to pin interrupts, here's an example from Nobody123 of connecting the IRQ pin from the CAP1188 to digital pin #3 on an Uno (Interrupt #1) for tracking touches asynchronously

```
/*
*****
This is a library for the CAP1188 I2C/SPI 8-chan Capacitive Sensor

Designed specifically to work with the CAP1188 sensor from Adafruit
----> https://www.adafruit.com/products/1602

These sensors use I2C/SPI to communicate, 2+ pins are required to
interface
Adafruit invests time and resources providing this open source code,
please support Adafruit and open-source hardware by purchasing
products from Adafruit!

Written by Limor Fried/Ladyada for Adafruit Industries.
BSD license, all text above must be included in any redistribution
*****/

#include <Wire.h>
#include <SPI.h>
#include <Adafruit_CAP1188.h>

// Reset Pin is used for I2C or SPI
#define CAP1188_RESET 4

// CS pin is used for software or hardware SPI
#define CAP1188_CS 10

// These are defined for software SPI, for hardware SPI, check your
// board's SPI pins in the Arduino documentation
#define CAP1188_MOSI 11
#define CAP1188_MISO 12
#define CAP1188_CLK 13

volatile byte interrupt = 0;

// For I2C, connect SDA to your Arduino's SDA pin, SCL to SCL pin
// On UNO/Duemilanove/etc, SDA == Analog 4, SCL == Analog 5
// On Leonardo/Micro, SDA == Digital 2, SCL == Digital 3
// On Mega/ADK/Due, SDA == Digital 20, SCL == Digital 21

// Use I2C, no reset pin!
// Adafruit_CAP1188 cap = Adafruit_CAP1188();

// Or...Use I2C, with reset pin
//Adafruit_CAP1188 cap = Adafruit_CAP1188(CAP1188_RESET);
```

```

// Or... Hardware SPI, CS pin & reset pin
Adafruit_CAP1188 cap = Adafruit_CAP1188(CAP1188_CS, CAP1188_RESET);

// Or.. Software SPI: clock, miso, mosi, cs, reset
//Adafruit_CAP1188 cap = Adafruit_CAP1188(CAP1188_CLK, CAP1188_MISO, CAP1188_MOSI, CAP1188_CS,
CAP1188_RESET);

void setup() {
  Serial.begin(9600);
  Serial.println("CAP1188 test!");
  pinMode(3,INPUT);
  // Raise SPI slave select (SS) pins
  // Communication begins when you drop the individual select signals to LOW
  digitalWrite(10,HIGH);

  // Initialize the sensor, if using i2c you can pass in the i2c address
  // if (!cap.begin(0x28)) {
  if (!cap.begin()) {
    Serial.println("CAP1188 not found");
    while (1);
  }
  Serial.println("CAP1188 found!");
  pinMode(3, INPUT);
  // Turn off multitouch so only one button pressed at a time
  cap.writeRegister(0x2A, 0x80); // 0x2A default 0x80 use 0x41 - Set multiple touches back to off
  cap.writeRegister(0x41, 0x39); // 0x41 default 0x39 use 0x41 - Set "speed up" setting back to off
  cap.writeRegister(0x72, 0x00); // 0x72 default 0x00 - Sets LED links back to off (default)
  cap.writeRegister(0x44, 0x41); // 0x44 default 0x40 use 0x41 - Set interrupt on press but not
release
  cap.writeRegister(0x28, 0x00); // 0x28 default 0xFF use 0x00 - Turn off interrupt repeat on button
hold
  EIFR = 1; // clear flag for interrupt 1
  attachInterrupt(1, routine_Interrupt_CAP1188, FALLING);
}

void loop() {

  // Serial.println(digitalRead(3));
  uint8_t touched = cap.touched();

  if (touched == 0) {
    // No touch detected
    // return;
  }

  for (uint8_t i=0; i<8; i++) {
    if (touched & (1 << i)) {
      Serial.print("C"); Serial.print(i+1); Serial.print("\t");
    }
  }
  Serial.println();
  delay(50);
  Serial.print("Interrupt: "); Serial.println(interrupt);
}

void routine_Interrupt_CAP1188() {
  ++interrupt;
,

```


Python & CircuitPython

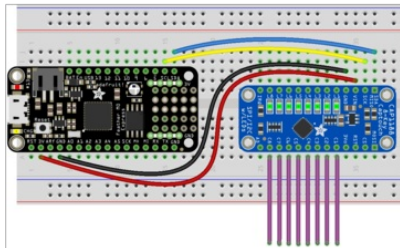
It's easy to use the CAP1188 sensor with Python or CircuitPython and the [Adafruit CircuitPython CAP1188 \(https://adafru.it/CLo\)](https://adafru.it/CLo) module. This module allows you to easily write Python code that reads capacitive touch from the sensor.

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

CircuitPython Microcontroller Wiring

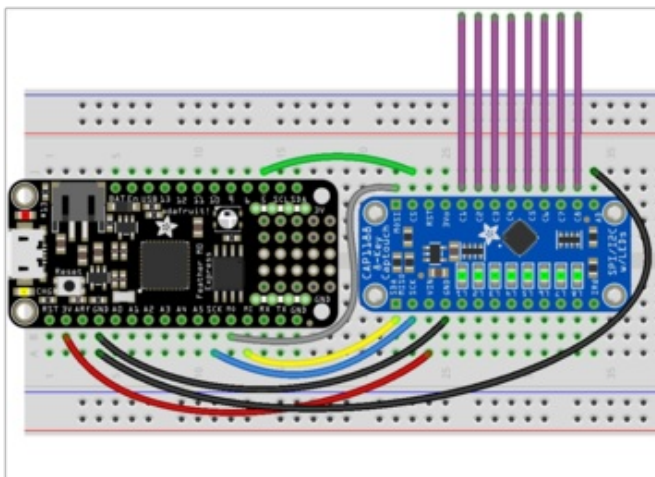
First wire up a CAP1188 to your board exactly as shown on the previous pages for Arduino. You can use either I2C or SPI wiring, although it's recommended to use I2C for simplicity.

Here's an example of wiring a Feather M0 to the sensor with I2C:



- Board 3V to sensor VIN
- Board GND to sensor GND
- Board SCK to sensor SCK
- Board SDA to sensor SDA

And an example of a Feather M0 wired with hardware SPI:

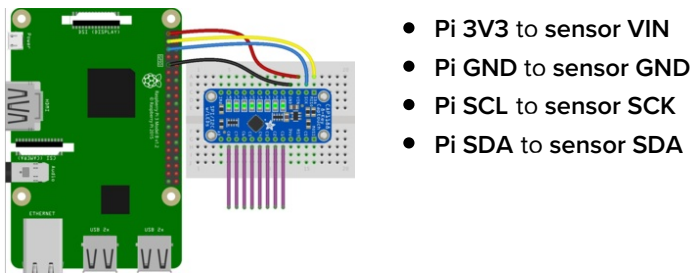


- Board 3V to sensor VIN
- Board GND to sensor GND
- Board GND to sensor AD
- Board SCK to sensor SCK
- Board MOSI to sensor MOSI
- Board MISO to sensor MISO
- Board D5 to sensor CS (or use any other free digital I/O pin)

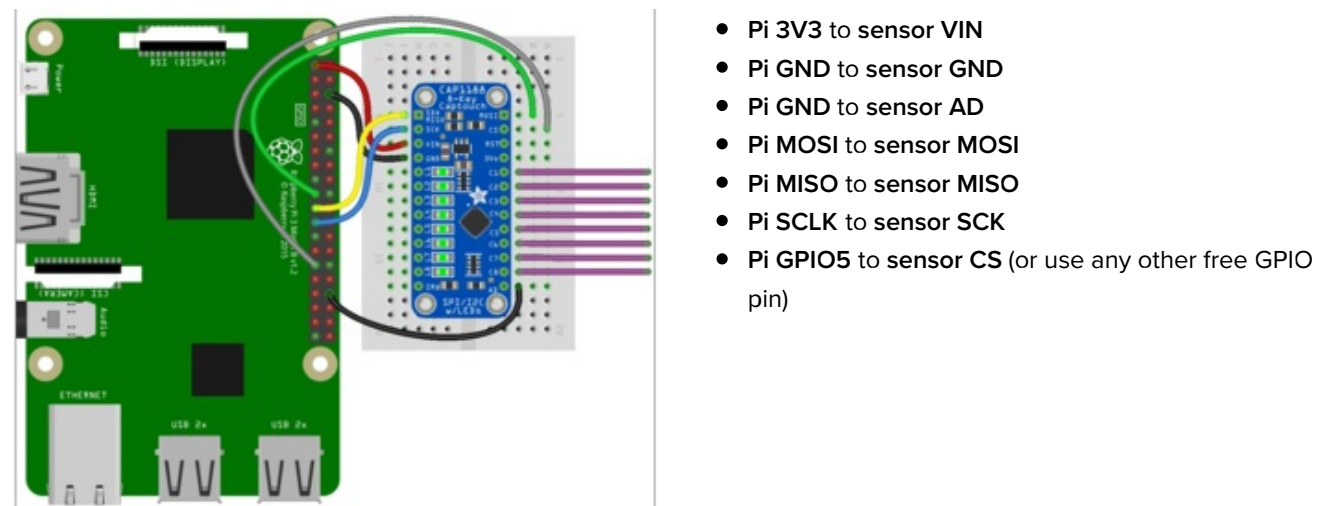
Python Computer Wiring

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

Here's the Raspberry Pi wired with I2C:



And an example on the Raspberry Pi wired with SPI:



CircuitPython Installation of CAP1188 Library

You'll need to install the [Adafruit CircuitPython CAP1188 \(https://adafru.it/CLo\)](https://adafru.it/CLo) library on your CircuitPython board.

First make sure you are running the [latest version of Adafruit CircuitPython \(https://adafru.it/Amd\)](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 \(https://adafru.it/uap\)](https://adafru.it/uap). Our CircuitPython starter guide has [a great page on how to install the library bundle \(https://adafru.it/ABU\)](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_cap1188`
- `adafruit_bus_device`

Before continuing make sure your board's lib folder or root filesystem has the `adafruit_cap1188`, and `adafruit_bus_device` files and folders copied over.

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

Python Installation of CAP1188 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 \(https://adafru.it/BSN\)!](https://adafru.it/BSN)

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

- `sudo pip3 install adafruit-circuitpython-cap1188`

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 capacitive touch from the board's Python REPL.

If you're using an I2C connection run the following code to import the necessary modules and initialize the I2C connection with the sensor:

```
import board
import busio
from adafruit_cap1188.i2c import CAP1188_I2C
i2c = busio.I2C(board.SCL, board.SDA)
cap = CAP1188_I2C(i2c)
```

Or if you're using a SPI connection run this code instead to setup the SPI connection and sensor:

```
import board
import busio
import digitalio
from adafruit_cap1188.spi import CAP1188_SPI
spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
cs = digitalio.DigitalInOut(board.D5)
cap = CAP1188_SPI(spi, cs)
```

Now you're ready to read capacitive touch from the sensor using any of these properties:

- **value** - Whether the pin is being touched or not.
- **raw_value** - The raw touch measurement.

For example, to print when pin 1 is touched:

```
while True:
    if cap[1].value:
        print("Pin 1 touched!")
```

```
>>> while True:
...     if cap[1].value:
...         print("Pin 1 touched!")
...
Pin 1 touched!
Pin 1 touched!
Pin 1 touched!
Pin 1 touched!
Pin 1 touched!
Pin 1 touched!
```

Or, to print when any pin is touched:

```
while True:
    for i in range(1, 9):
        if cap[i].value:
            print("Pin {} touched!".format(i))
```

```
>>> while True:
...     for i in range(1, 9):
...         if cap[i].value:
...             print("Pin {} touched!".format(i))
...
Pin 1 touched!
Pin 2 touched!
Pin 3 touched!
Pin 4 touched!
```

That's all there is to using CAP1188 with CircuitPython!

Full Example Code

```
import board
import busio

# I2C setup
from adafruit_cap1188.i2c import CAP1188_I2C
i2c = busio.I2C(board.SCL, board.SDA)
cap = CAP1188_I2C(i2c)

# SPI setup
# from digitalio import DigitalInOut, Direction
# from adafruit_cap1188.spi import CAP1188_SPI
# spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
# cs = DigitalInOut(board.D5)
# cap = CAP1188_SPI(spi, cs)

while True:
    for i in range(1, 9):
        if cap[i].value:
            print("Pin {} touched!".format(i))
```


File

- The [CAP1188 Datasheet](https://adafru.it/d5i) has lots of details about this chip (<https://adafru.it/d5i>)
- [EagleCAD PCB files on GitHub](https://adafru.it/ruE) (<https://adafru.it/ruE>)
- [Fritzing object in Adafruit Fritzing library](https://adafru.it/aP3) (<https://adafru.it/aP3>)

Schematic

