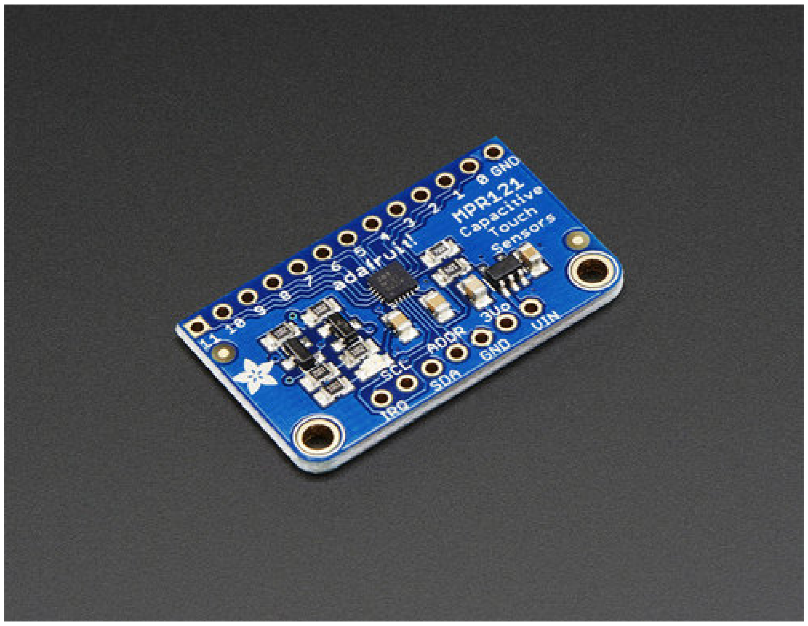
adafruit learning system

Adafruit MPR121 12-Key Capacitive Touch Sensor Breakout Tutorial

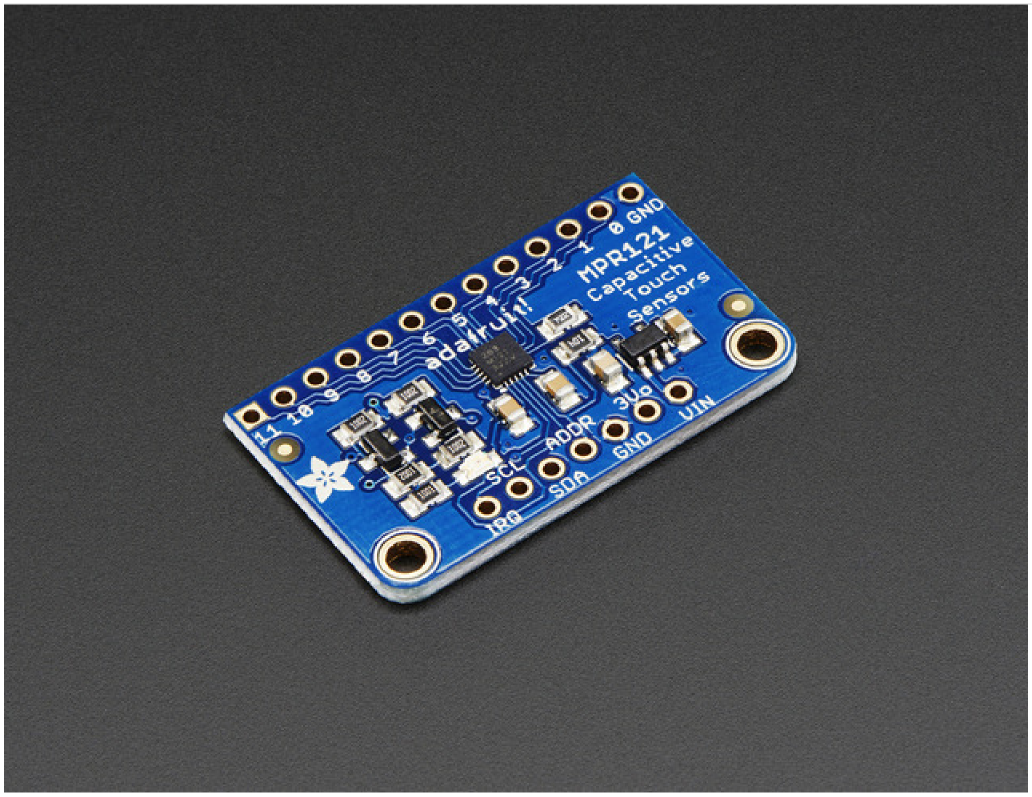
Created by lady ada



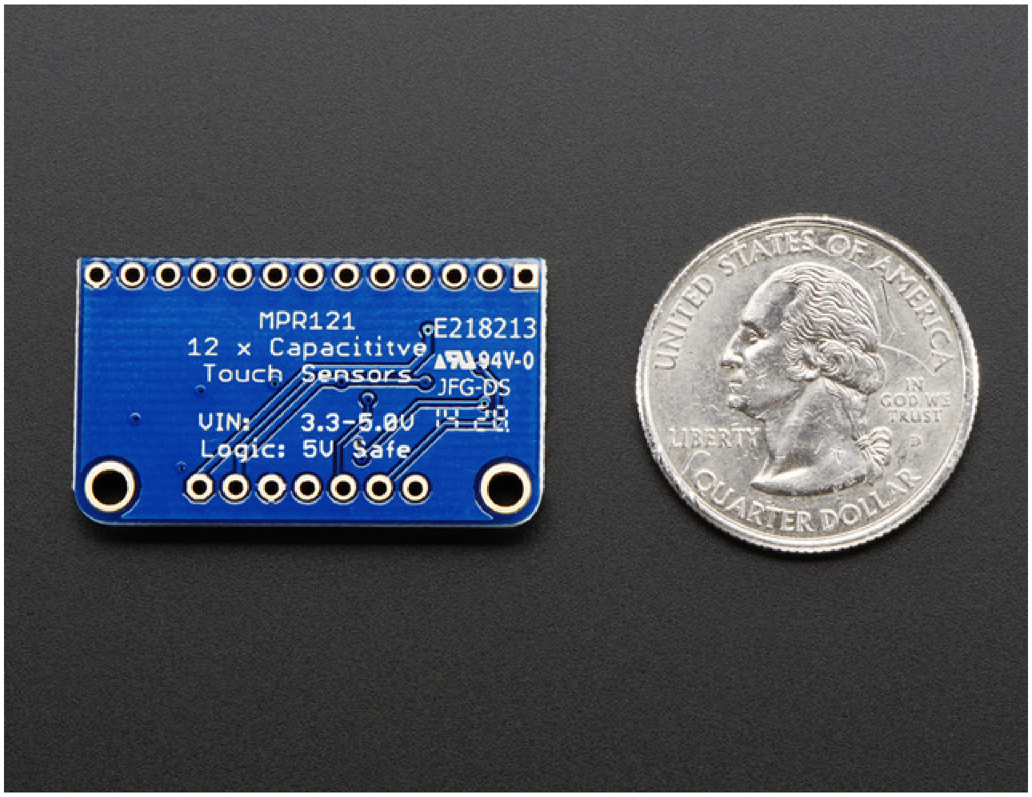
Last updated on 2017-11-29 12:46:45 AM UTC

Guide Contents

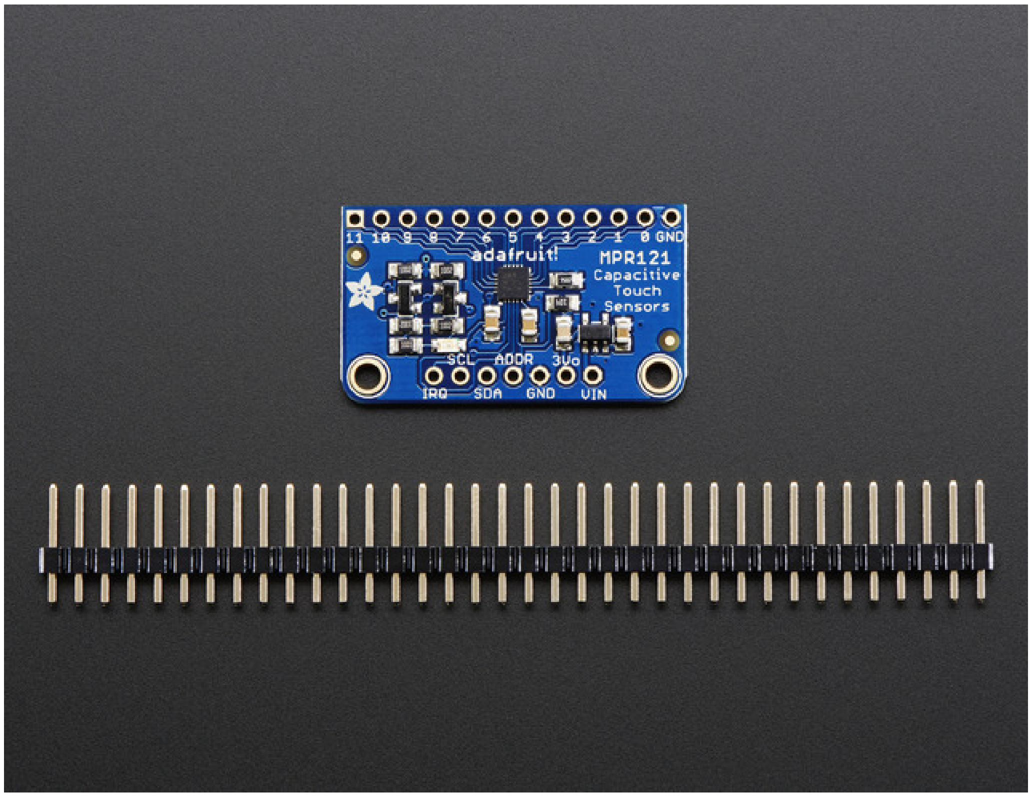
Overview



Add lots of touch sensors to your next microcontroller project with this easy-to-use 12-channel capacitive touch sensor breakout board, starring the MPR121. This chip can handle up to 12 individual touch pads.

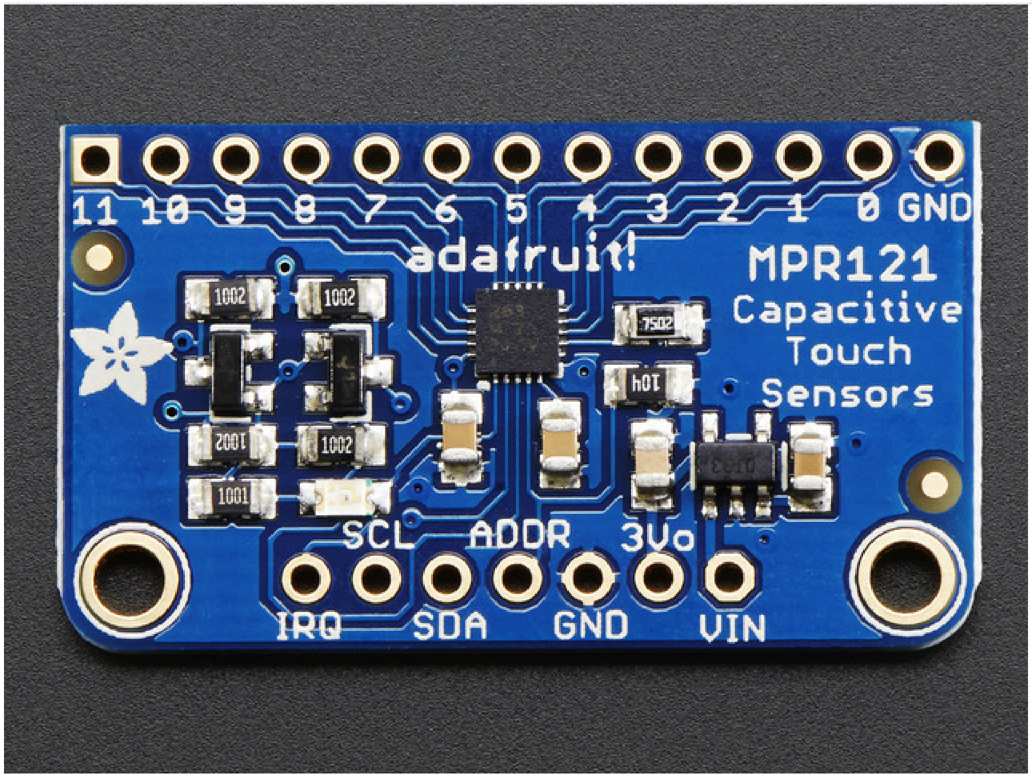


The MPR121 has support for only I2C, which can be implemented with nearly any microcontroller. You can select one of 4 addresses with the ADDR pin, for a total of 48 capacitive touch pads on one I2C 2-wire bus. Using this chip is a lot easier than doing the capacitive sensing with analog inputs: it handles all the filtering for you and can be configured for more/less sensitivity.



This sensor comes as a tiny hard-to-solder chip so we put it onto a breakout board for you. Since it's a 3V-only chip, we added a 3V regulator and I2C level shifting so its safe to use with any 3V or 5V microcontroller/processor like Arduino. We even added an LED onto the IRQ line so it will blink when touches are detected, making debugging by sight a bit easier on you. 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 or pyralux, then solder a wire that connects from the foil pad to the breakout.

Pinouts


11 10 9 8 7 6 5 4 3 2 1 0 GND
adafruit!
MPR121 Capacitive Touch Sensors
SCL ADDR 3Vo
IRQ SDA GND VIN

The little chip in the middle of the PCB is the actual MPR121 sensor that does all the capacitive sensing and filtering. We add all the extra components you need to get started, and 'break out' all the other pins you may want to connect to onto the PCB. For more details you can check out the schematics in the Downloads page.

Power Pins

The sensor on the breakout requires 3V power. Since many customers have 5V microcontrollers like Arduino, we tossed a 3.3V regulator on the board. Its ultra-low dropout so you can power it from 3.3V-5V just fine.

- **Vin -** this is the power pin. Since the 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 micro 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

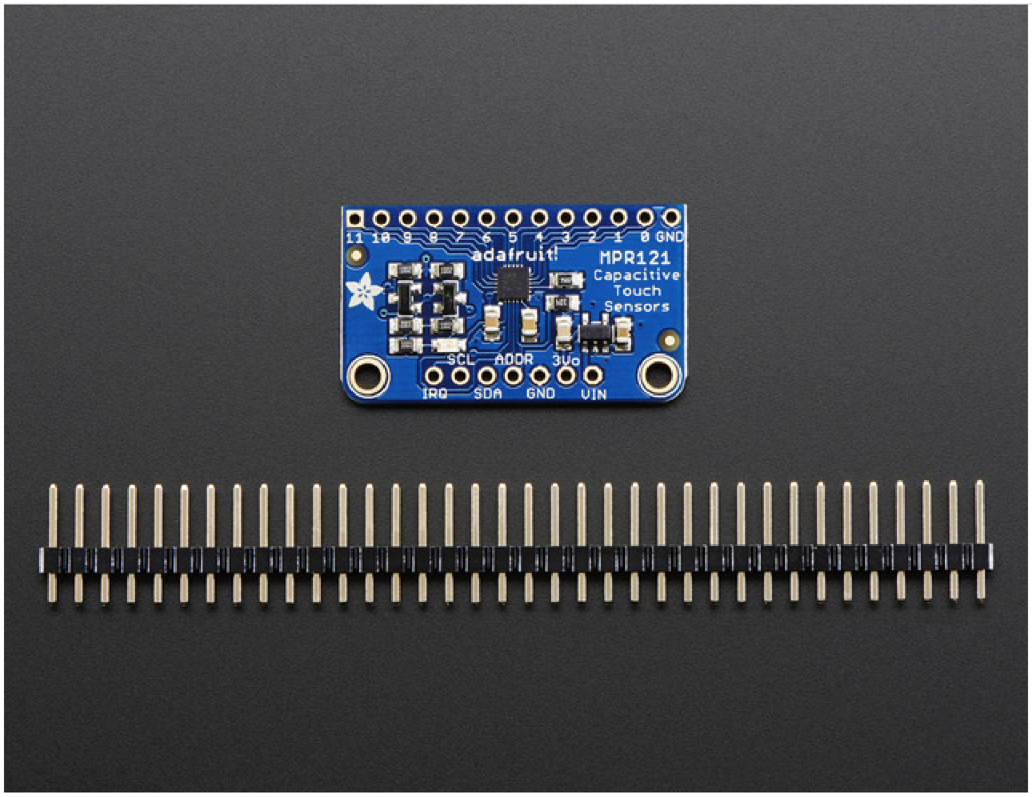
I2C Pins

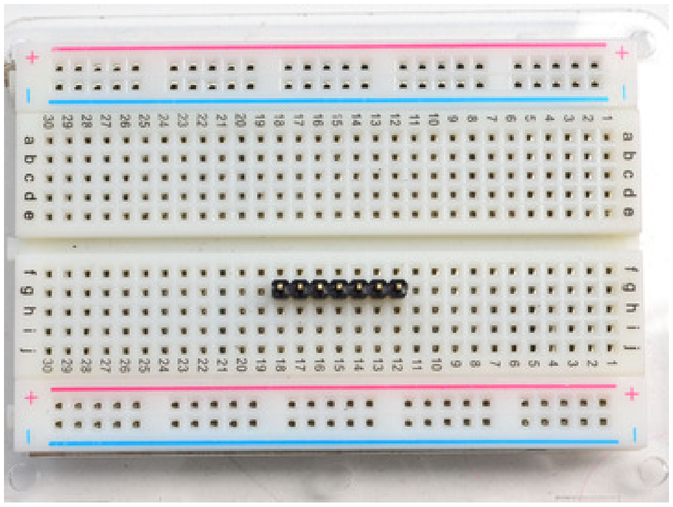
- **SCL** - I2C clock pin, connect to your microcontrollers I2C clock line.
- **SDA** - I2C data pin, connect to your microcontrollers I2C data line.

IRQ and ADDR Pins

- **ADDR** is the I2C address select pin. By default this is pulled down to ground with a 100K resistor, for an I2C address of 0x5A. You can also connect it to the 3Vo pin for an address of 0x5B, the SDA pin for 0x5C or SCL for address 0x5D
- **IRQ** is the Interrupt Request signal pin. It is pulled up to 3.3V on the breakout and when the sensor chip detects a change in the touch sense switches, the pin goes to 0V until the data is read over i2c

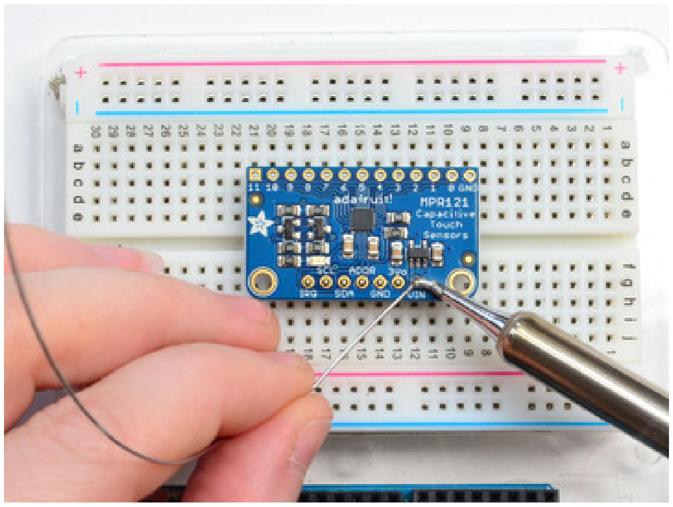
Assembly





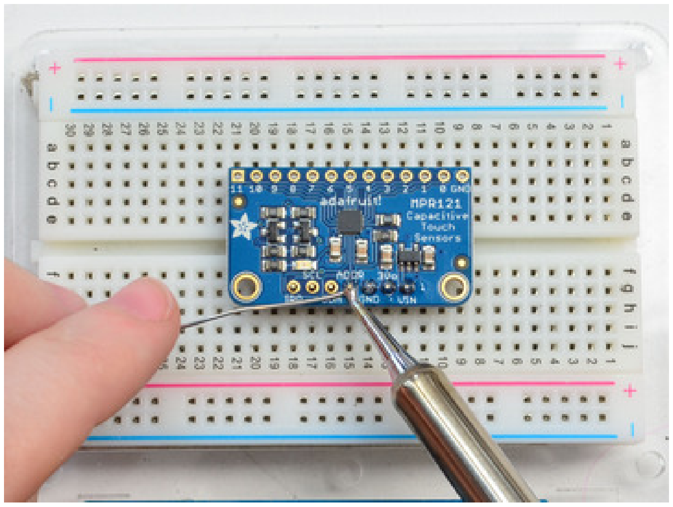
Prepare the header strip:

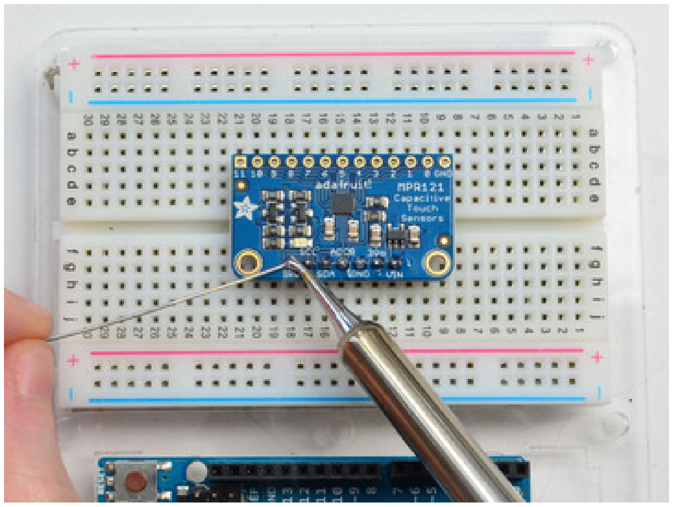
Cut the strip to length if necessary. It will be easier to solder if you insert it into a breadboard - **long pins down**



Add the breakout board:

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





And Solder!

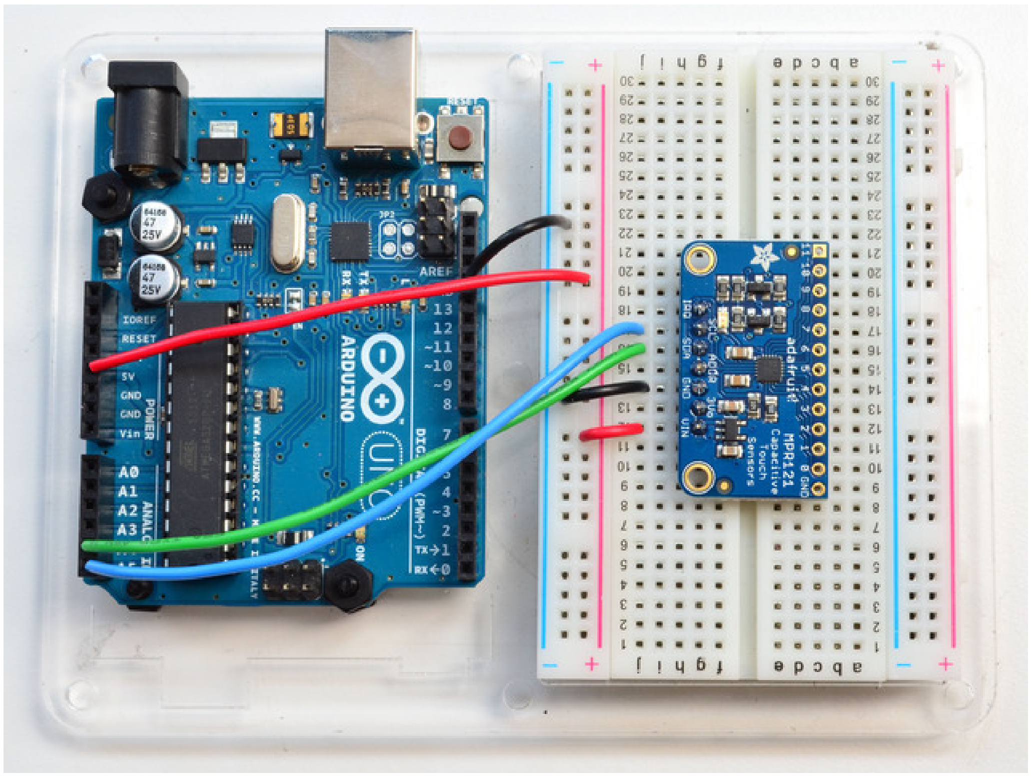
Be sure to solder all pins for reliable electrical contact.

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

You're done! Check your solder joints visually and continue onto the next steps

Wiring

You can easily wire this breakout to any microcontroller, we'll be using an Arduino. For another kind of microcontroller, just make sure it has I2C, then port the code - its pretty simple stuff!



- Connect **Vin** to the power supply, 3-5V is fine. Use the same voltage that the microcontroller logic is based off of. For most Arduinos, that is 5V
- Connect **GND** to common power/data ground
- Connect the **SCL** pin to the I2C clock **SCL** pin on your Arduino. On an UNO & '328 based Arduino, this is also known as **A5**, on a Mega it is also known as **digital 21** and on a Leonardo/Micro, **digital 3**
- Connect the **SDA** pin to the I2C data **SDA** pin on your Arduino. On an UNO & '328 based Arduino, this is also known as **A4**, on a Mega it is also known as **digital 20** and on a Leonardo/Micro, **digital 2**

The MPR121 **ADDR** pin is pulled to ground and has a default I2C address of **0x5A**
You can adjust the I2C address by connecting **ADDR** to other pins:

- ADDR not connected: 0x5A
- ADDR tied to 3V: 0x5B
- ADDR tied to SDA: 0x5C
- ADDR tied to SCL: 0x5D

We suggest sticking with the default for the test demo, you can always change it later.

Download Adafruit_MPR121

To begin reading sensor data, you will need to download Adafruit_MPR121_Library from our github repository. You can do that by visiting the github repo and manually downloading or, easier, just click this button to download the zip

Download Adafruit_MPR121

https://adafru.it/dKF

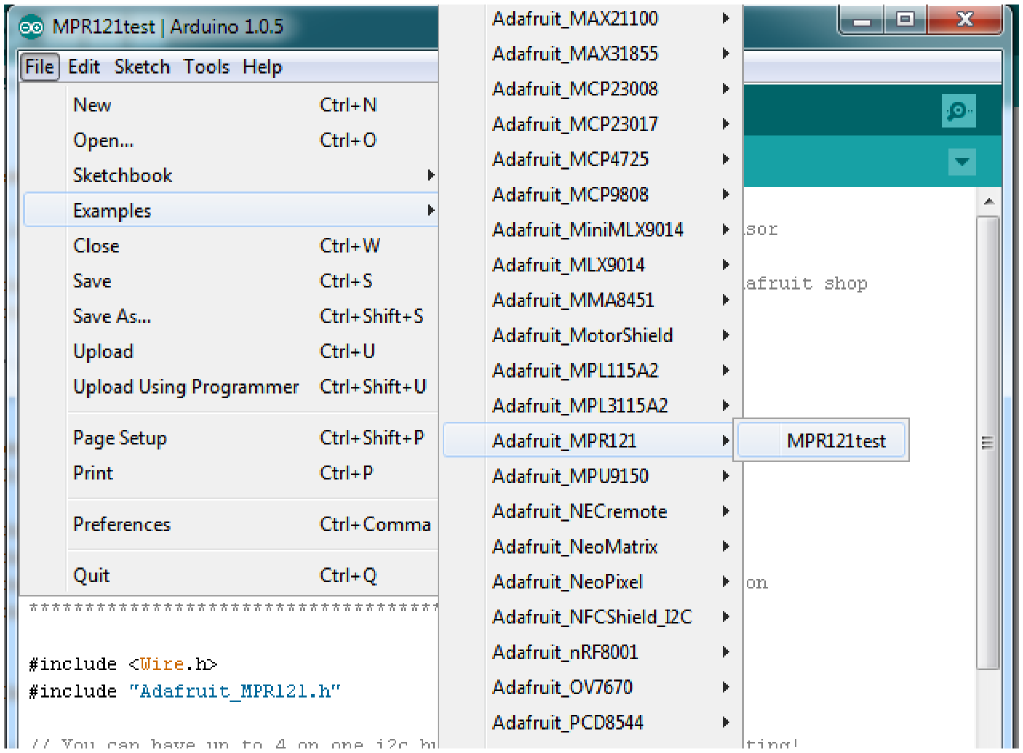
Rename the uncompressed folder **Adafruit_MPR121** and check that the **Adafruit_MPR121** folder contains **Adafruit_MPR121.cpp** and **Adafruit_MPR121.h**

Place the **Adafruit_MPR121** library folder your **arduinosketchfolder/libraries/** folder.
You may need to create the **libraries** subfolder if its your first library. Restart the IDE.

We also have a great tutorial on Arduino library installation at:
http://learn.adafruit.com/adafruit-all-about-arduino-libraries-install-use

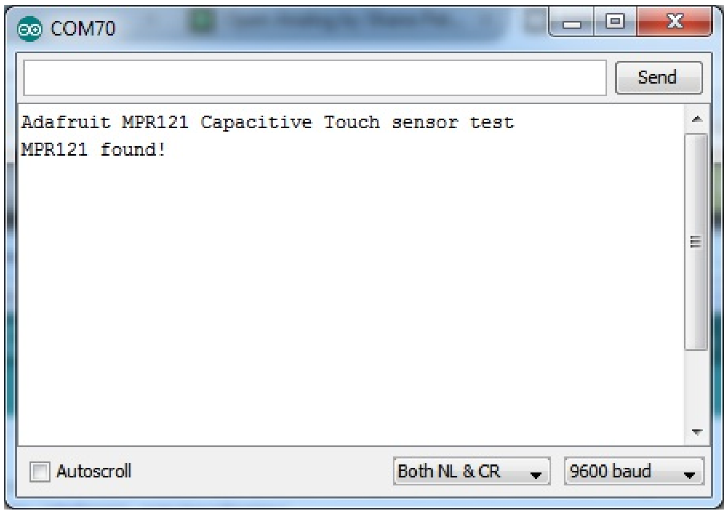
Load Demo

Open up **File->Examples->Adafruit_MPR121->MPR121test** and upload to your Arduino wired up to the sensor



```
#include <Wire.h>
#include "Adafruit_MPR121.h"
```

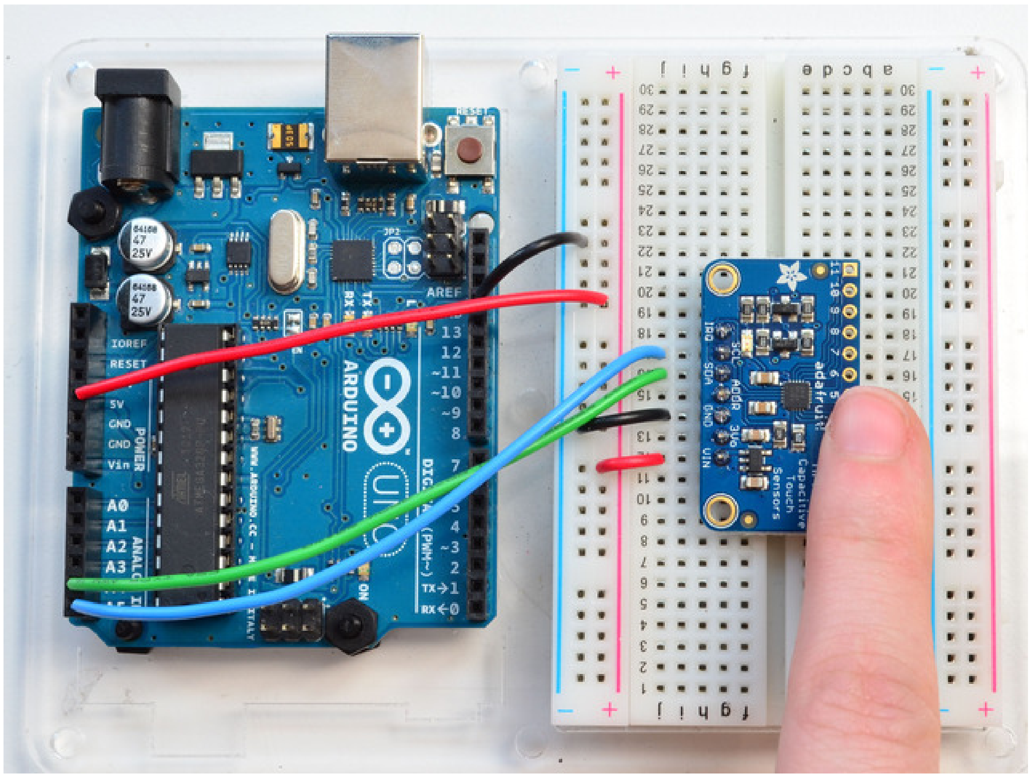
Thats it! Now open up the serial terminal window at 9600 speed to begin the test.

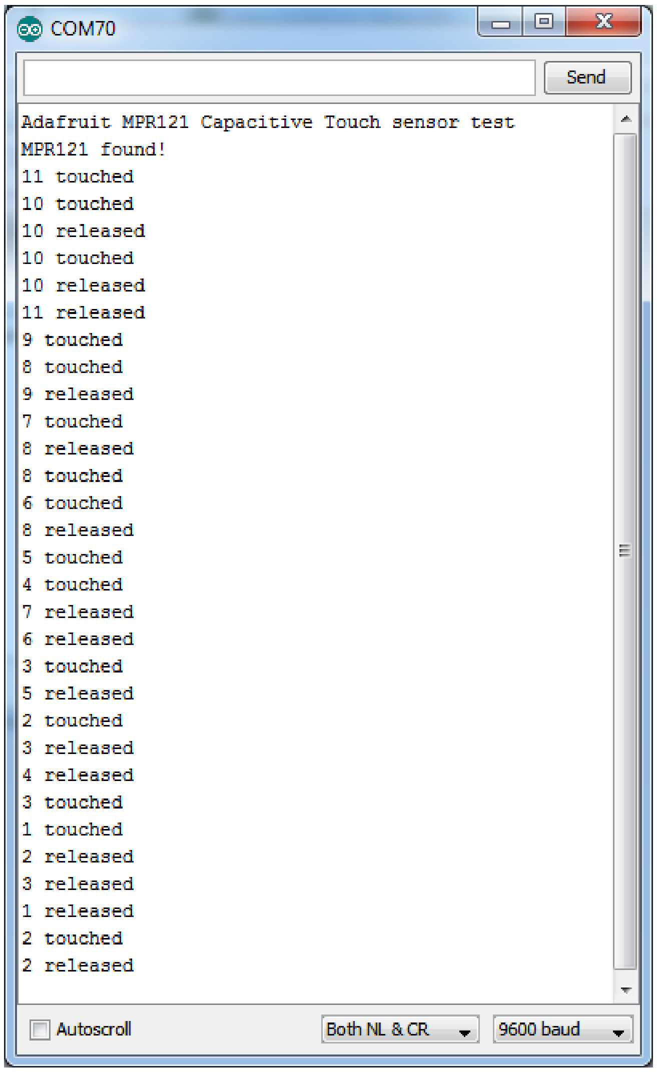


```
Adafruit MPR121 Capacitive Touch sensor test
MPR121 found!
```

Make sure you see the "MPR121 found!" text which lets you know that the sensor is wired correctly.

Now touch the 12 pads with your fingertip to activate the touch-detection





```
Adafruit MPR121 Capacitive Touch sensor test
MPR121 found!
11 touched
10 touched
10 released
10 touched
10 released
11 released
9 touched
8 touched
9 released
7 touched
8 released
8 touched
6 touched
8 released
5 touched
4 touched
7 released
6 released
3 touched
5 released
2 touched
3 released
4 released
3 touched
1 touched
2 released
3 released
1 released
2 touched
2 released
```

For most people, that's all you'll need! Our code keeps track of the 12 'bits' for each touch and has logic to let you know when a contect is touched or released.

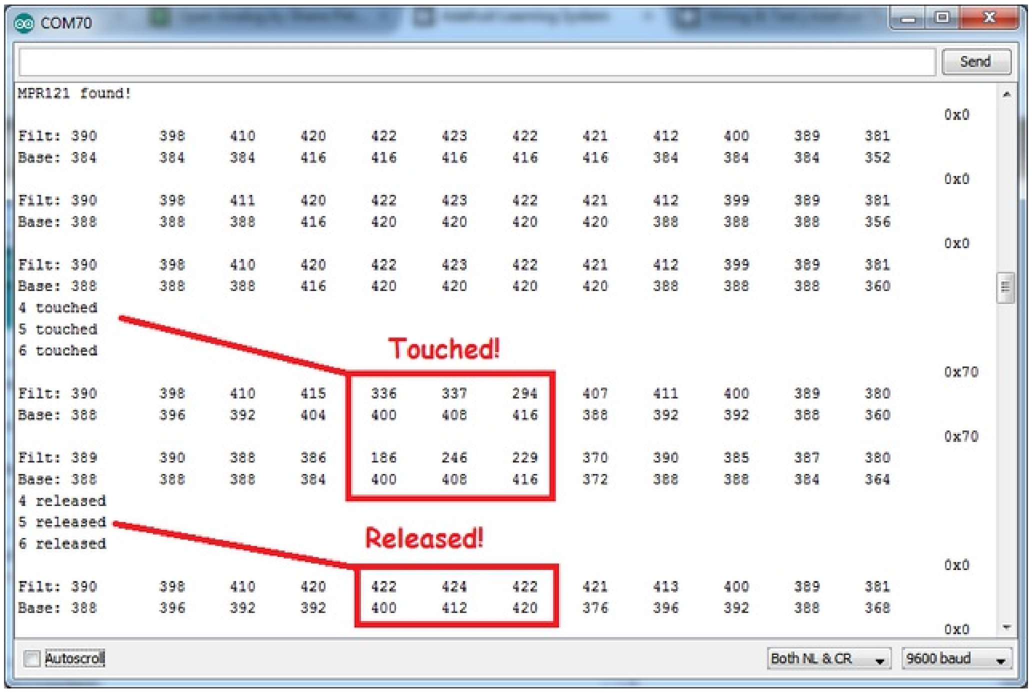
If you're feeling more advanced, you can see the 'raw' data from the chip. Basically, what it does it keep track of the capacitance it sees with "counts". There's some baseline count number that depends on the temperature, humidity, PCB, wire length etc. Where's a dramatic change in number, its considered that a person touched or released the wire.

Comment this "return" line to activate that mode:

```
// comment out this line for detailed data from the sensor!
return;
```

Then reupload. Open up the serial console again - you'll see way more text

Each reading has 12 columns. One for each sensor, #0 to #11. There's two rows, one for the 'baseline' and one for the current filtered data reading. When the current reading is within about 12 counts of the baseline, that's considered untouched. When the reading is more than 12 counts smaller than the baseline, the chip reports a touch.



```
MPR121 found!
                                                                                0x0
Filt: 390    398    410    420    422    423    422    421    412    400    389    381
Base: 384    384    384    416    416    416    416    416    384    384    384    352
                                                                                0x0
Filt: 390    398    411    420    422    423    422    421    412    399    389    381
Base: 388    388    388    416    420    420    420    420    388    388    388    356
                                                                                0x0
Filt: 390    398    410    420    422    423    422    421    412    399    389    381
Base: 388    388    388    416    420    420    420    420    388    388    388    360
4 touched
5 touched
6 touched
                                                                                0x70
Filt: 390    398    410    415    336    337    294    407    411    400    389    380
Base: 388    396    392    404    400    408    416    388    392    392    388    360
                                                                                0x70
Filt: 389    390    388    386    186    246    229    370    390    385    387    380
Base: 388    388    388    384    400    408    416    372    388    388    384    364
4 released
5 released
6 released
                                                                                0x0
Filt: 390    398    410    420    422    424    422    421    413    400    389    381
Base: 388    396    392    392    400    412    420    376    396    392    388    368
                                                                                0x0
```

Both NL & CR   9600 baud

Most people don't need raw data too much, but it can be handy if doing intense debugging. It can be helpful if you are tweaking your sensors to get good responsivity.

Library Reference

Since the sensors use I2C, there's no pins to be defined during instantiation. You can just use:

> Adafruit_MPR121 cap = Adafruit_MPR121();

When you initialize the sensor, pass in the I2C address. It can range from 0x5A (default) to 0x5D

> **cap.begin(0x5A)**

begin() returns true if the sensor was found on the I2C bus, and false if not.

Touch detection

99% of users will be perfectly happy just querying what sensors are currentlt touched. You can read all at once with cap.touched()
Which returns a 16 bit value. Each of the bottom 12 bits refers to one sensor. So if you want to test if the #4 is touched, you can use

> **if (cap.touched() & (1 << 4)) { do something }**

You can check its not touched with:

> **if (! (cap.touched() & (1 << 4))) { do something }**

Raw Data

You can grab the current baseline and filtered data for each sensor with

> **filteredData(*sensornumber*);**
> **baselineData(*sensornumber*);**

It returns a 16-bit number which is the number of counts, there's no unit like "mg" or "capacitance". The baseline is initialized to the current ambient readings when the sensor begin() is called - you can always reinitialize by re-calling begin()! The baseline will drift a bit, that's normal! It is trying to compensate for humidity and other environmental changes.

If you need to change the threshholds for touch detection, you can do that with

> **setThreshholds(uint8_t touch, uint8_t release)**

By default, the touch threshhold is 12 counts, and the release is 6 counts. It's reset to these values whenever you call begin() by the way.

CircuitPython Wiring

Parts

You'll need the following hardware to follow this guide:

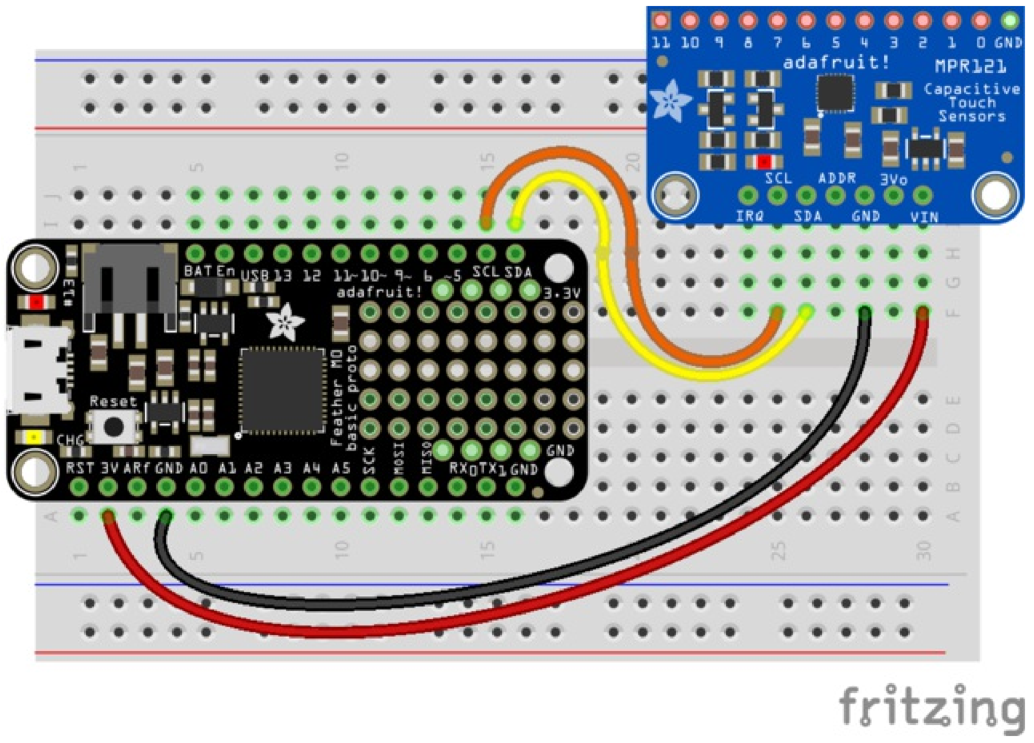
- **Board running CircuitPython.** Currently ESP8266 or SAMD21-based boards are supported by CircuitPython. The Feather HUZZAH ESP8266 or Feather M0 boards are great options that can easily be connected to an accelerometer. See the guide on loading MicroPython & CircuitPython firmware on a board for details on loading CircuitPython.
- **MPR121 capacitive touch breakout board** or **MPR121 capacitive touch Arduino shield**.
- **Breadboard**, **hookup wires**, **and soldering tools**. You'll need to solder headers to the accelerometer breakout, be sure to see the guide to excellent soldering if you're new to it.
- **Alligator clip wires** or **alligator clip to male jumper wires**, these are useful for connecting conductive objects to the MPR121 breakout inputs. For example you can connect the inputs to pieces of fruit to make a funky, fruity musical instrument!

Start by following the MPR121 breakout guide to assemble and test the board. Then continue on below to learn how to wire it to a Feather for use with CircuitPython.

For the MPR121 Arduino shield just solder headers to the shield (don't forget to solder the 2x3 SPI header in the center of the board too) and connect it to a compatible Arduino (like the Arduino Zero flashed with CircuitPython firmware).

Wiring

Connect the MPR121 to your board using its I2C interface as follows:



- **MPR121 VIN** to **board 3V (or 5V) output** - red wire.
- **MPR121 GND** to **board GND/ground** - black wire.
- **MPR121 SCL** to **board SCL (I2C clock)** - orange wire.

- **MPR121 SDA** to **board SDA (I2C data)** - yellow wire.

Continue on to learn how to install a CircuitPython module to control the MPR121 board.

CircuitPython Software

Adafruit CircuitPython Module Install

To use the MPR121 with your Adafruit CircuitPython board you'll need to install the Adafruit_CircuitPython_MPR121 module on your 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. For example the Circuit Playground Express guide has a great page on how to install the library bundle for both express and non-express boards.

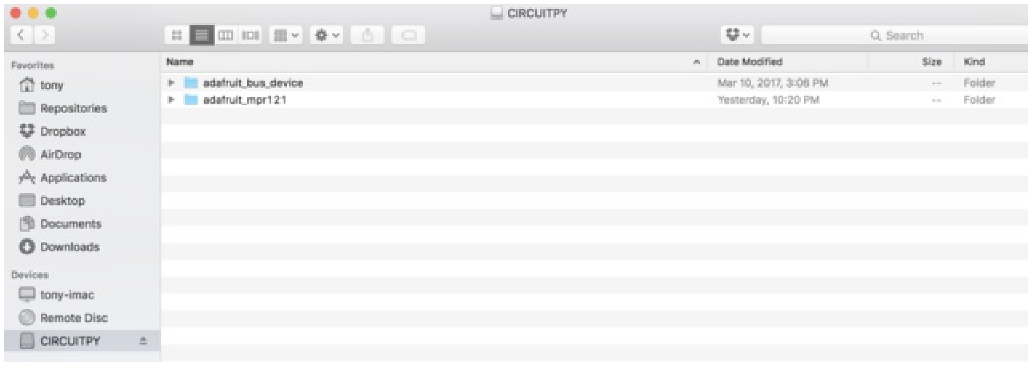
Remember for non-express boards like the Trinket M0, Gemma M0, and Feather/Metro M0 basic you'll need to manually install the necessary libraries from the bundle:

- **adafruit_mpr121**
- **adafruit_bus_device**

You can also download the **adafruit_mpr121.zip** which contains the **adafruit_mpr121** folder from its releases page on Github.

If your board doesn't support USB mass storage, like the ESP8266, then use a tool like ampy to copy the file to the board.

Before continuing make sure your board's lib folder or root filesystem has the **adafruit_mpr121,** and **adafruit_bus_device** folders copied over.



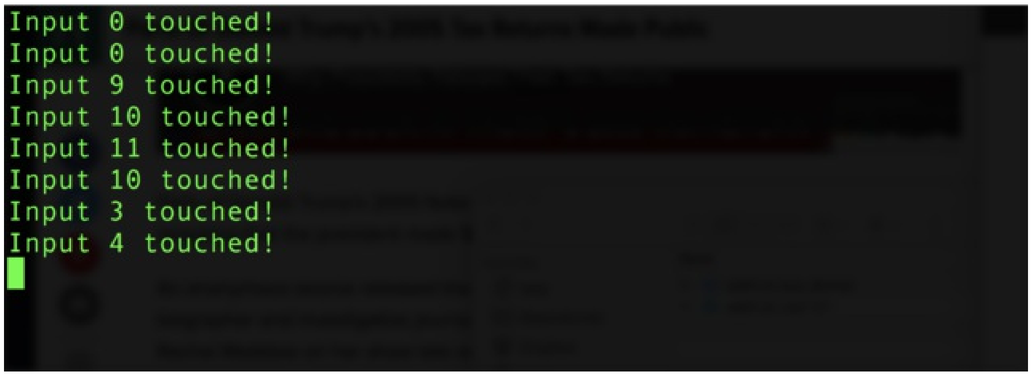
Name	Date Modified	Size	Kind
adafruit_bus_device	Mar 10, 2017, 3:06 PM	--	Folder
adafruit_mpr121	Yesterday, 10:20 PM	--	Folder

Example

To learn how to use the MPR121 module code you can look at the **simpletest.py example** included in the library. Save this as a **main.py** file on your board:

```
# Simple test of the MPR121 capacitive touch sensor library.
# Will print out a message when any of the 12 capacitive touch inputs of the
# board are touched.  Open the serial REPL after running to see the output.
# Author: Tony DiCola
import time

# Import MPR121 module.
import adafruit_mpr121

import busio

# Create I2C bus.
import board
i2c = busio.I2C(board.SCL, board.SDA)

# Create MPR121 class.
mpr121 = adafruit_mpr121.MPR121(i2c)
# Note you can optionally change the address of the device:
#mpr121 = adafruit_mpr121.MPR121(i2c, address=0x91)

# Loop forever testing each input and printing when they're touched.
while True:
    # Loop through all 12 inputs (0-11).
    for i in range(12):
        # Call is_touched and pass it then number of the input.  If it's touched
        # it will return True, otherwise it will return False.
        if mpr121.is_touched(i):
            print('Input {} touched!'.format(i))
    time.sleep(0.25)  # Small delay to keep from spamming output messages.
```

Once the example is running open the serial REPL for your board. With your finger try pressing any of the 0-11 input pins on the MPR121. You should see a message printed every time an input is touched:



```
Input 0 touched!
Input 0 touched!
Input 9 touched!
Input 10 touched!
Input 11 touched!
Input 10 touched!
Input 3 touched!
Input 4 touched!
```

If you don't see any messages when you touch the inputs you might need to ground yourself to the board by touching the GND pin on the board with one finger and then touching the input pads with another finger.

Also make sure nothing is touching the pins when you first run the script or else it might confuse the MPR121's touch detection (unmount the board's file system from your operating system, then press the board's reset button to reset the script and run it again with nothing touching the pins).

Usage

Examine the simpletest.py code to see how to use the MPR121 module. First the module is imported with code like:

```
# Import MPR121 module.
import adafruit_mpr121
```

This code initializes the I2C bus:

```
# Create I2C bus.
import board
import busio
i2c = busio.I2C(board.SCL, board.SDA)
```

Remember for boards without hardware I2C like the ESP8266 you might need to use the **bitbangio** module to create the I2C bus:

```
# Create I2C bus for software I2C boards (ESP8266)
import board
import bitbangio
i2c = bitbangio.I2C(board.SCL, board.SDA)
```

Once the I2C bus is intialized the MPR121 class can be created by passing it an instance of the I2C bus:

```
# Create MPR121 class.
mpr121 = adafruit_mpr121.MPR121(i2c)
# Note you can optionally change the address of the device:
#mpr121 = adafruit_mpr121.MPR121(i2c, address=0x91)
```

Notice the MPR121 class initializer takes an optional **address** parameter if you changed the board's I2C address.

Now you're ready to start checking if inputs are touched. Notice the main loop for the example calls the MPR121 class **is_touched** function for every input:

```
# Loop forever testing each input and printing when they're touched.
while True:
    # Loop through all 12 inputs (0-11).
    for i in range(12):
        # Call is_touched and pass it then number of the input.  If it's touched
        # it will return True, otherwise it will return False.
        if mpr121.is_touched(i):
            print('Input {} touched!'.format(i))
    time.sleep(0.25)  # Small delay to keep from spamming output messages.
```

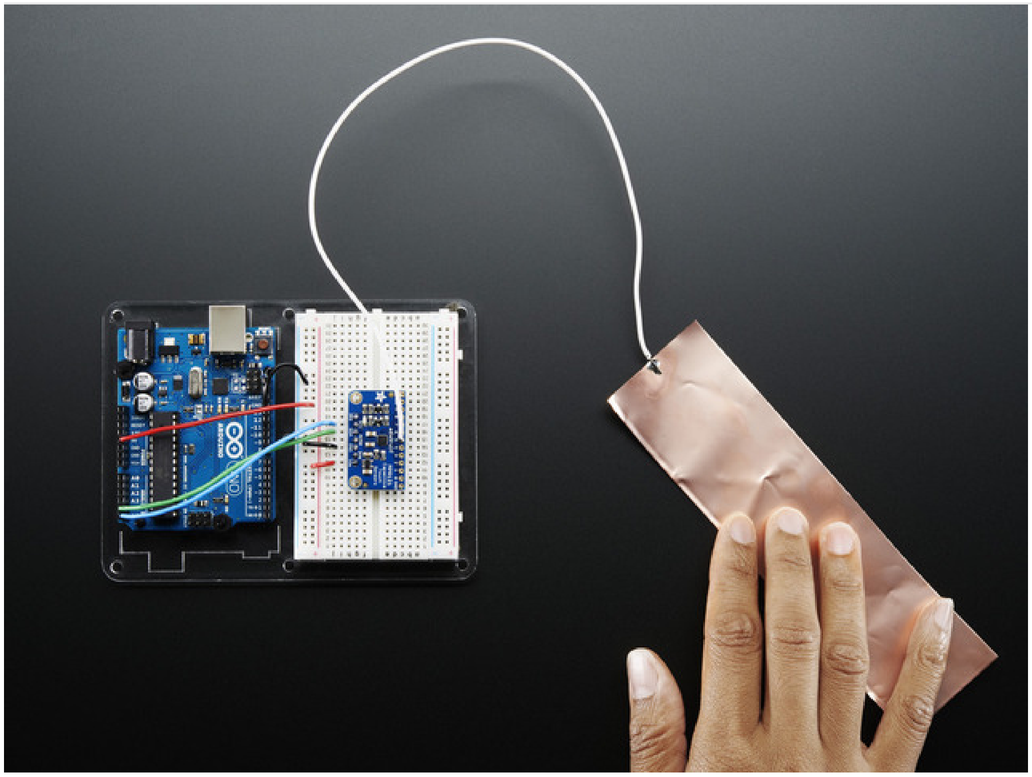
Just pass a value 0 to 11 to the **is_touched** function and it will return a boolean **True** or **False** value depending on if the input is currently being touched or not. In the example it prints a small message when an input is touched.

The example doesn't show its usage but if you want to check all of the inputs at once you can call the **touched** function. This function returns a 12-bit value where each bit represents the touch state of an input. So bit 0 is input 0 and will be 1 if it's touched and 0 if it's not touched, bit 1 would be input 1, etc. For example to test if input 0 and 11 are being touched with one call you could run code like:

```
# Call touched to get a 12-bit value with the state of each MPR121 input.
touch = mpr121.touched()
# Test if exactly bit 0 and 11 are set to 1, i.e. input 0 and 11 are touched.
if touch == 0b100000000001:
    print('Input 0 and 11 touched!')
# Or test if either bit 0 or 11 are set to 1, i.e. input 0 or 11 are touched:
if touch & 0b100000000000 > 0 or touch & 0b000000000001 > 0:
    print('Input 0 or 11 touched!')
```

That's all there is to using the MPR121 module with CircuitPython!

Electrodes



Once you have the MPR121 breakout working you'll want to construct electrodes. These are large conductive piece of copper, foil, paint, etc that will act as the "thing you touch"

Remember that electrodes must be electrically conductive! We suggest copper foil tape, conductive fabrics, ITO, pyralux flex PCB, etc. We have tons of great conductive materials in our Materials category. Some can be soldered to, others can be clipped to with alligator chips.

Remember, it doesn't have to be metal to be electrically conductive. Other things that work are tap or salt water, many kinda of food, even fruit!

We suggest soldering a wire to the electrode pad on the breakout and then soldering or clipping it to whatever you want your electrode to be.

The wires and electrodes themselves have a certain amount of 'inherent capacitcance'!

This means that whenever you attach an alligator clip, or a large piece of copper, or whatever your electrode is, the capacitive sense chip will detect it and may think you're touching it. What you have to do is *recalibrate* the sensor. The easiest way to do that is to restart the python sketch since calibration is done when the chip is initialized. So, basically...

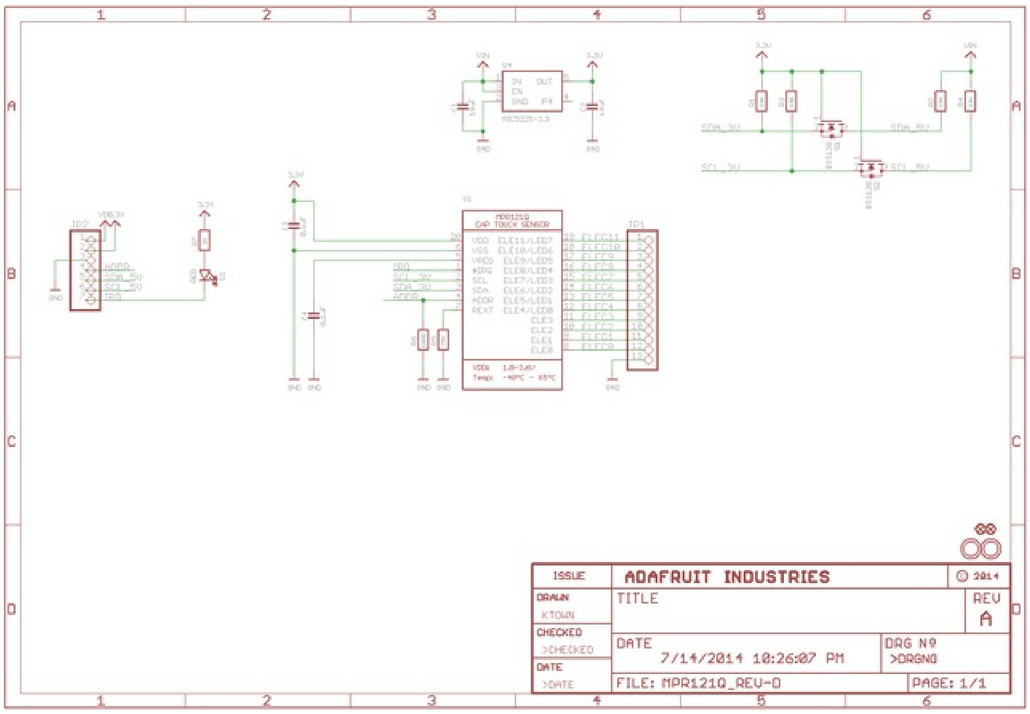
connect all your wires, electrodes, fruit, etc...*then* start up the capacitive touch program!

Downloads

Datasheets & Files

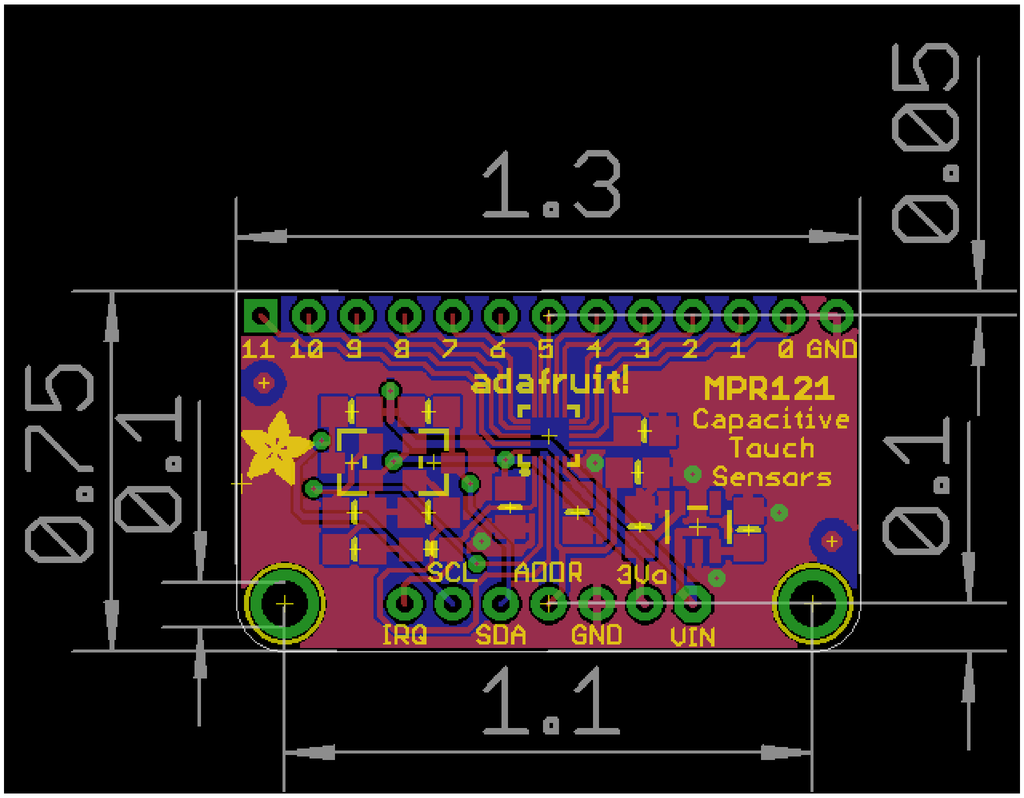
- MPR121 Datasheet
- EagleCAD PCB files on GitHub
- Fritzing object in Adafruit Fritzing library

Breakout Board Schematic



Fabrication Print

Dimensions in Inches


1.3
0.05
0.75
0.1
0.1
1.1
11 10 9 8 7 6 5 4 3 2 1 0 GND
adafruit!
MPR121
Capacitive
Touch
Sensors
SCL
ADDR
3Vo
IRQ
SDA
GND
VIN