Adafruit MPR121 12-Key Capacitive Touch Sensor Breakout Tutorial

Created by lady ada
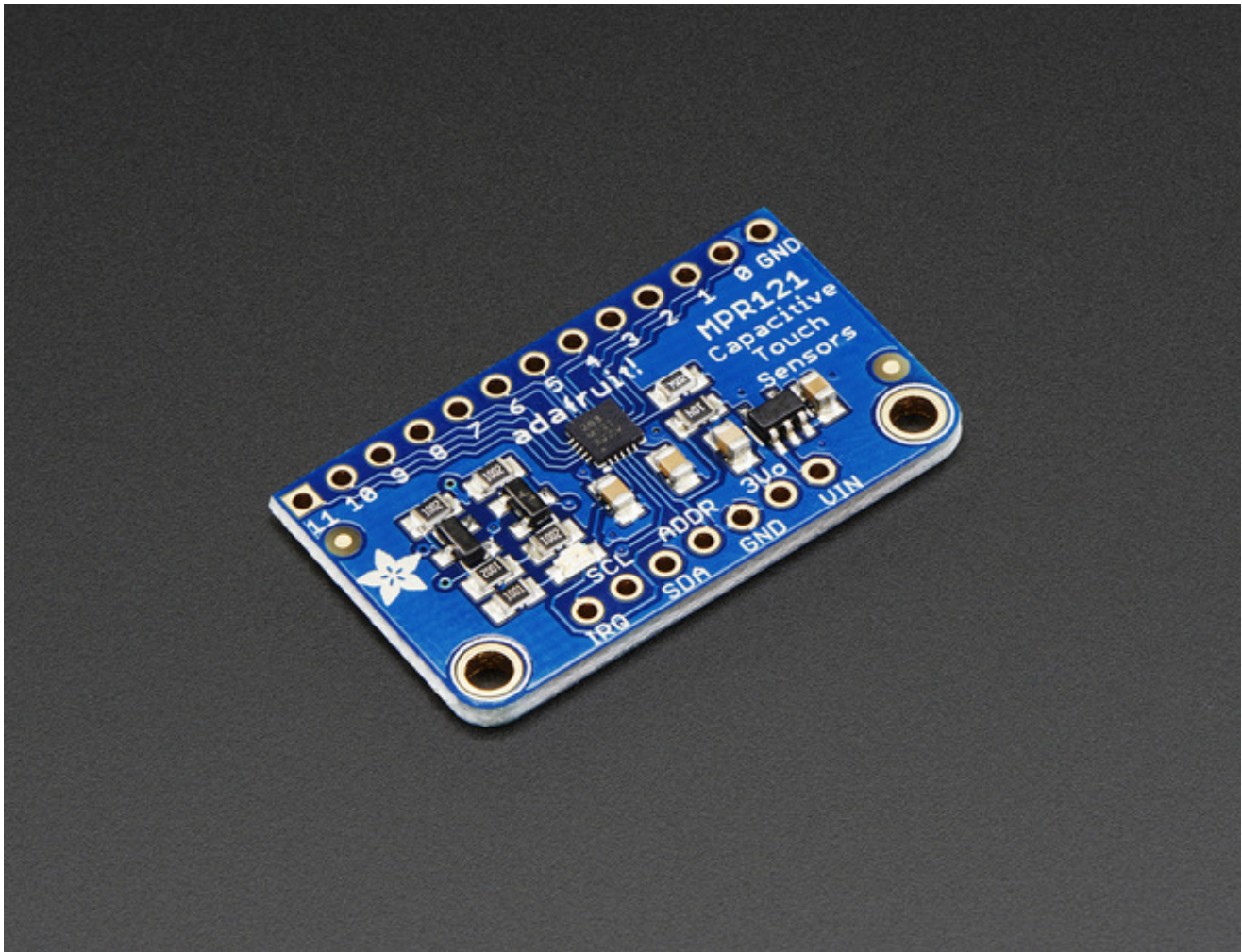
Guide Contents

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.
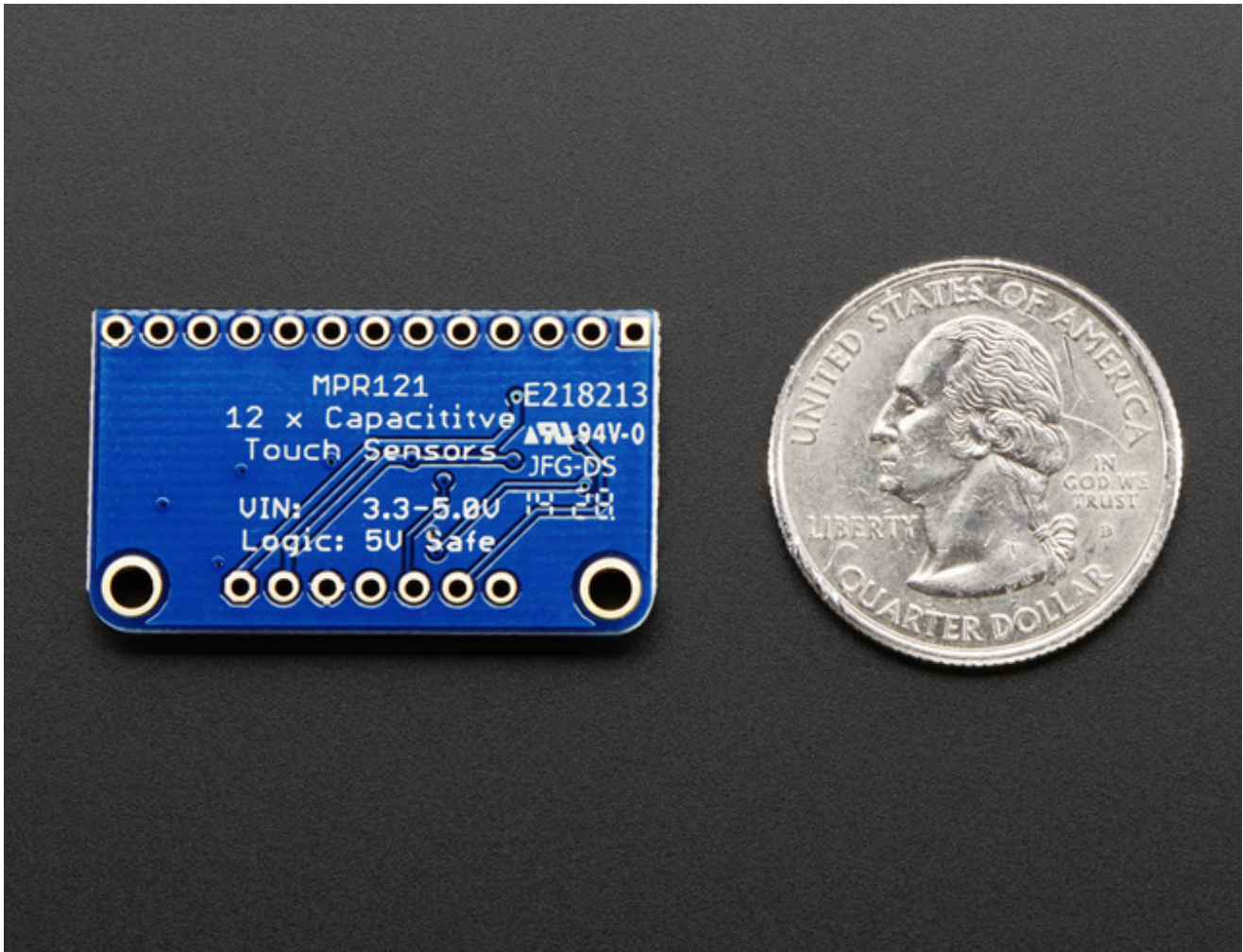
The MPR121 has support for only I2C, which can be implemented with nearly any microcontroller. You can select one of 4 addresses with the ADDR pin, for a total of 48 capacitive touch pads on one I2C 2-wire bus. Using this chip is a lot easier than doing the capacitive sensing with analog inputs: it handles all the filtering for you and can be configured for more/less sensitivity.
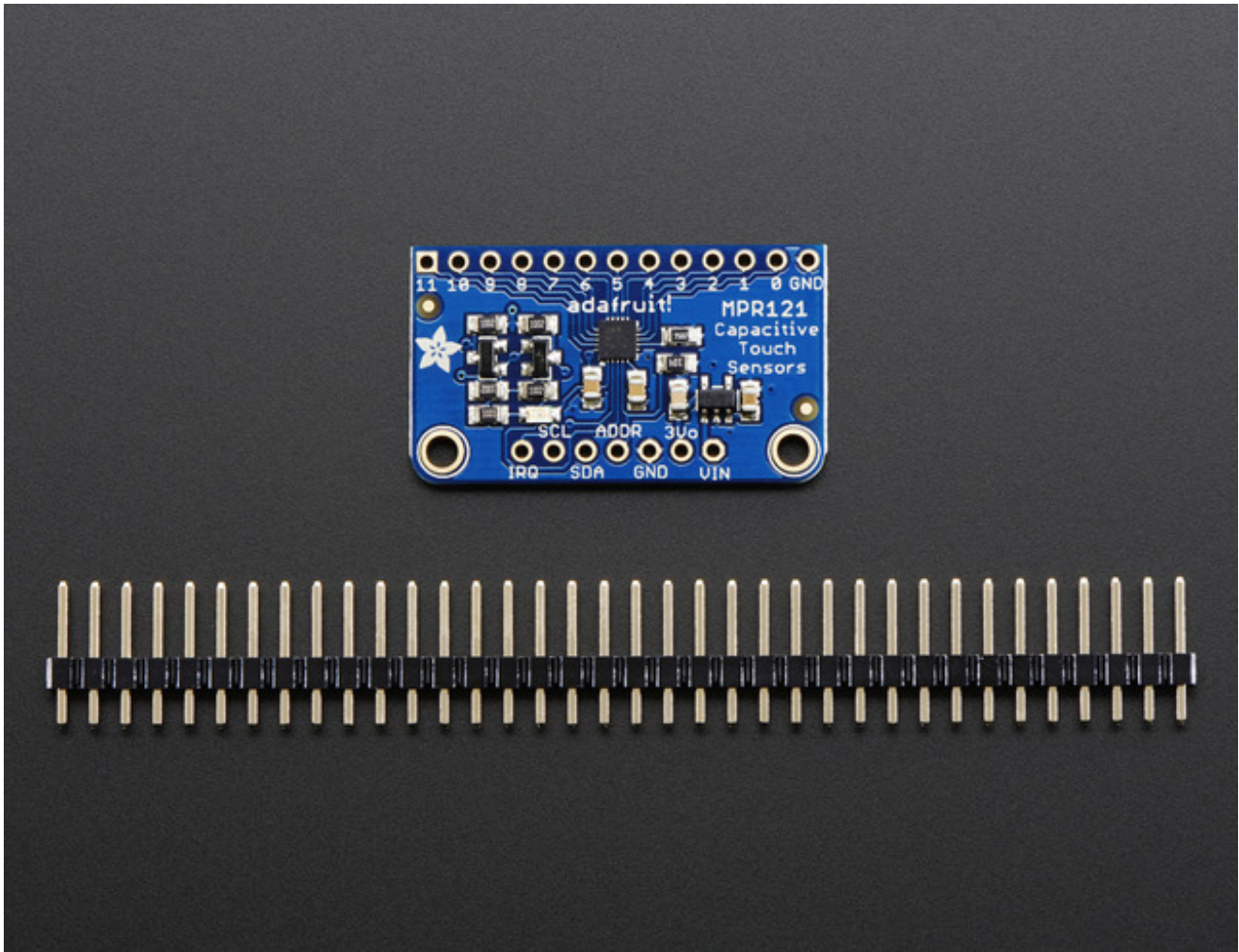
This sensor comes as a tiny hard-to-solder chip so we put it onto a breakout board for you. Since it's a 3V-only chip, we added a 3V regulator and I2C level shifting so its safe to use with any 3V or 5V microcontroller/processor like Arduino. We even added an LED onto the IRQ line so it will blink when 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
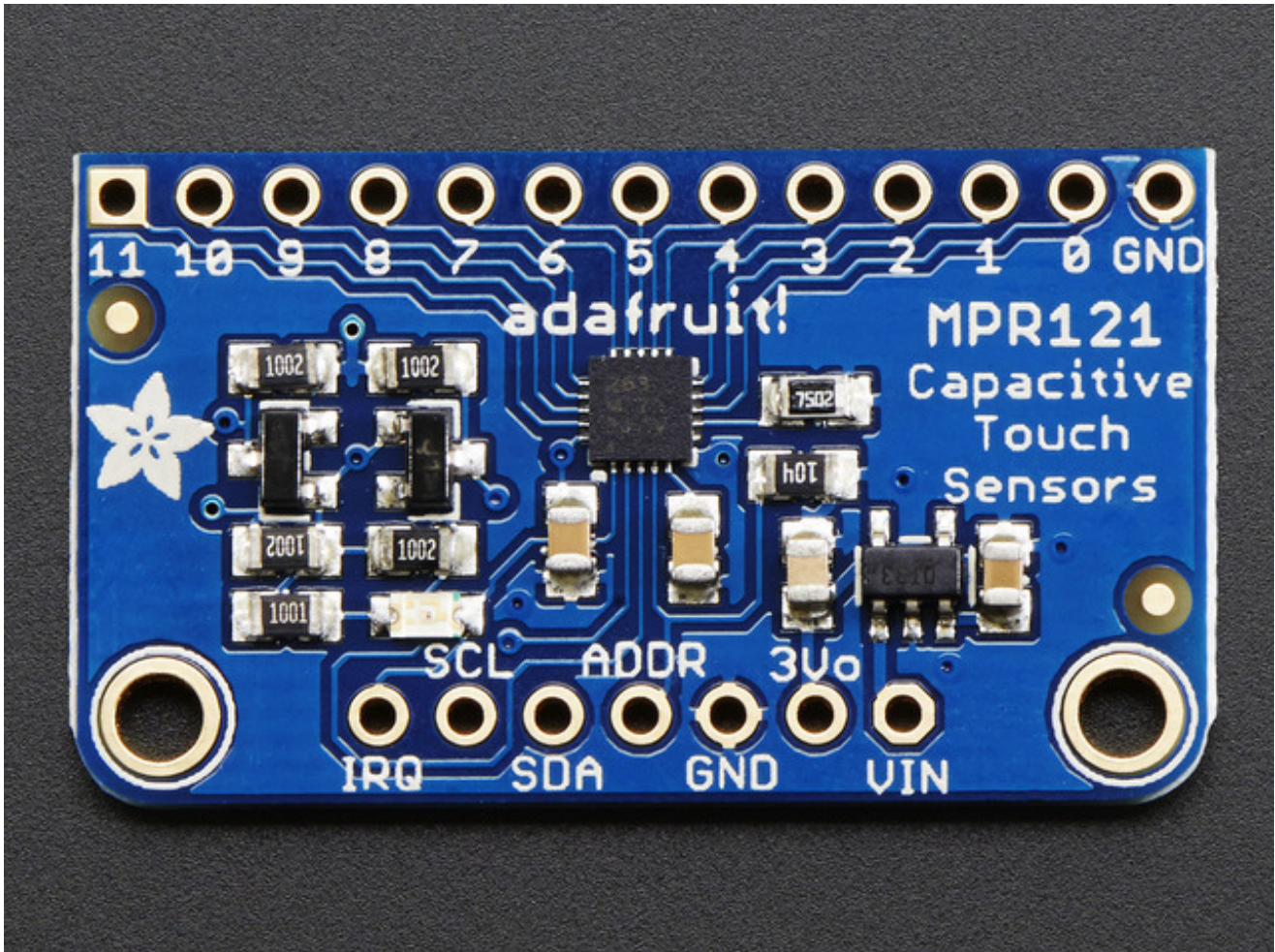
The little chip in the middle of the PCB is the actual MPR121 sensor that does all the capacitive sensing and filtering. We add all the extra components you need to get started, and 'break out' all the other pins you may want to connect to onto the PCB. For more details you can check out the schematics 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
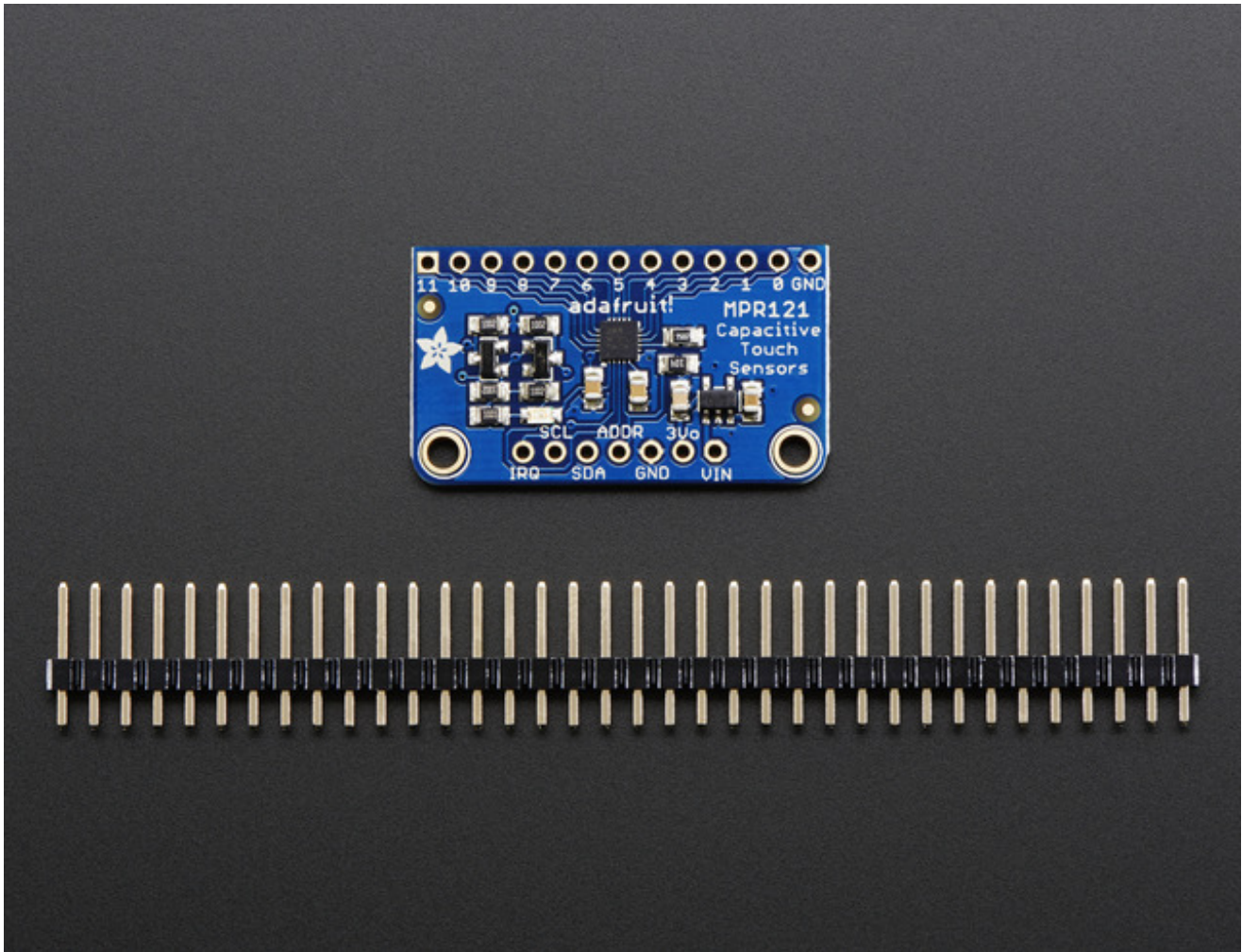
Prepare the header strip:
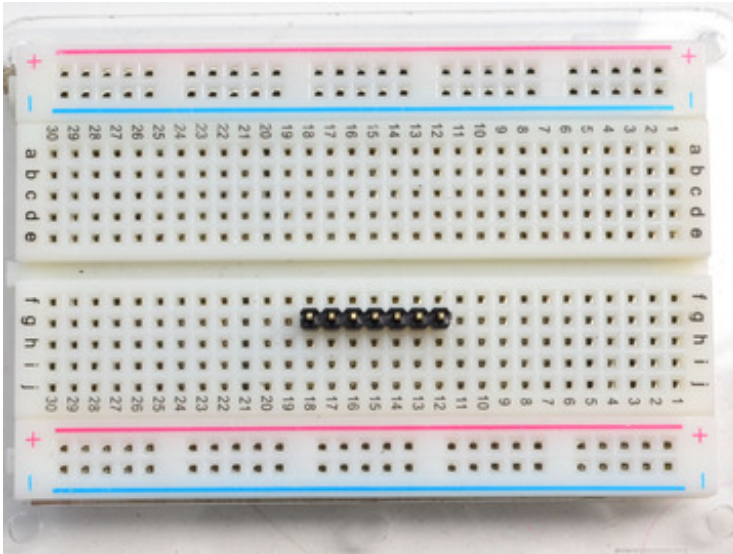
Cut the strip to length if necessary. It will be easier to solder if you insert it into a breadboard - **long pins down**
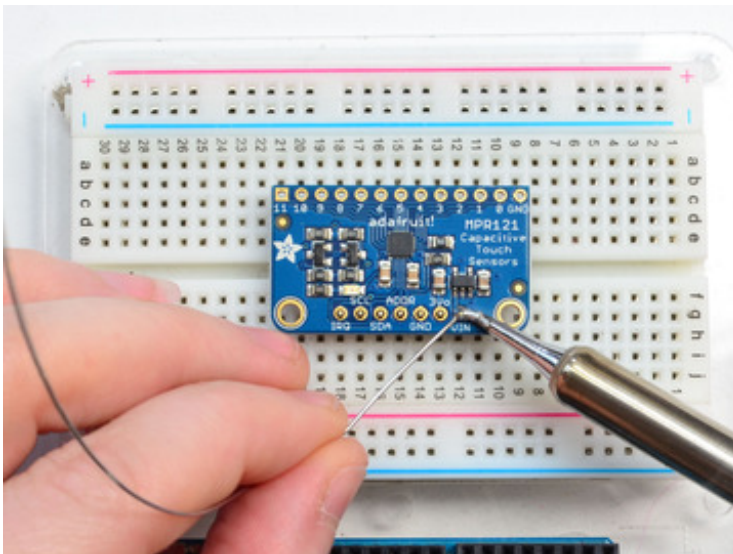
Add the breakout board:

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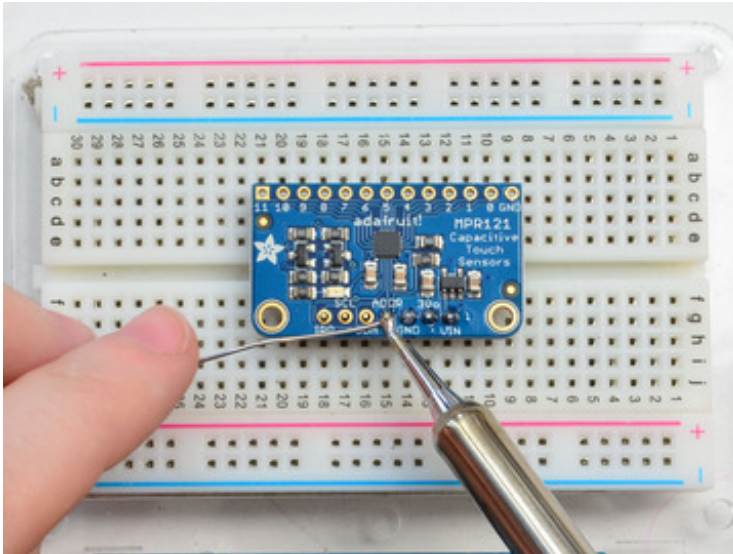
And Solder!

Be sure to solder all pins for reliable electrical contact.

(For tips on soldering, be sure to check out our [Guide to Excellent Soldering](http://adafru.it/aTk) (http://adafru.it/aTk)).
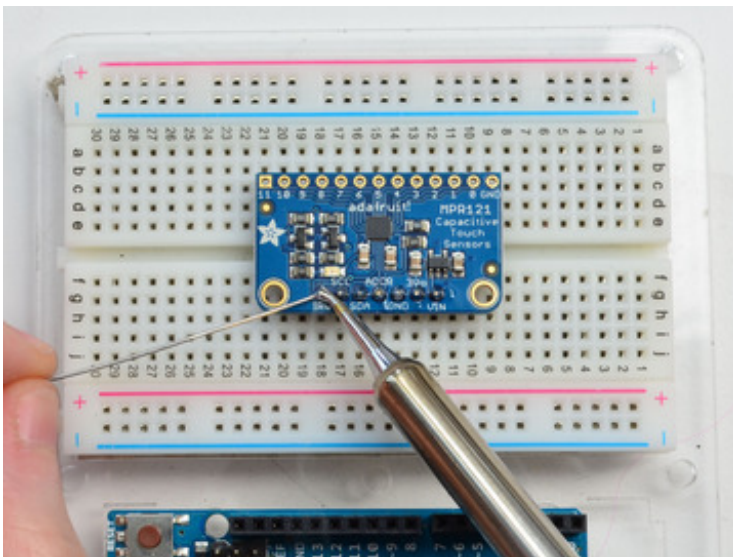
You're done! Check your solder joints visually and continue onto the next steps

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!
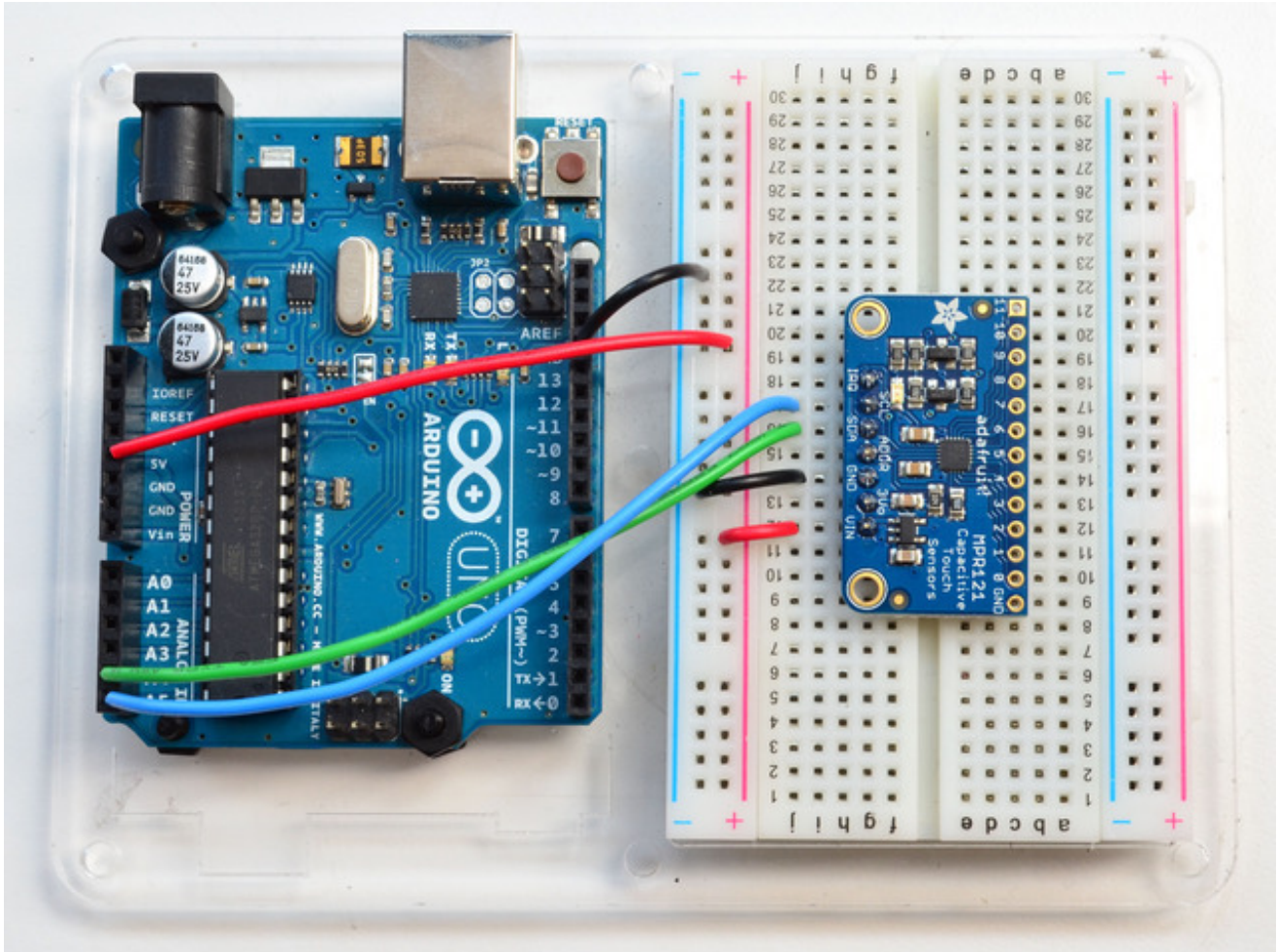
- Connect **Vin** to the power supply, 3-5V is fine. Use the same voltage that the microcontroller logic is based off of. For most Arduinos, that is 5V
- Connect **GND** to common power/data ground
- Connect the **SCL** pin to the I2C clock**SCL** pin on your Arduino. On an UNO & '328 based Arduino, 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 (http://adafru.it/dKE). 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
http://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:
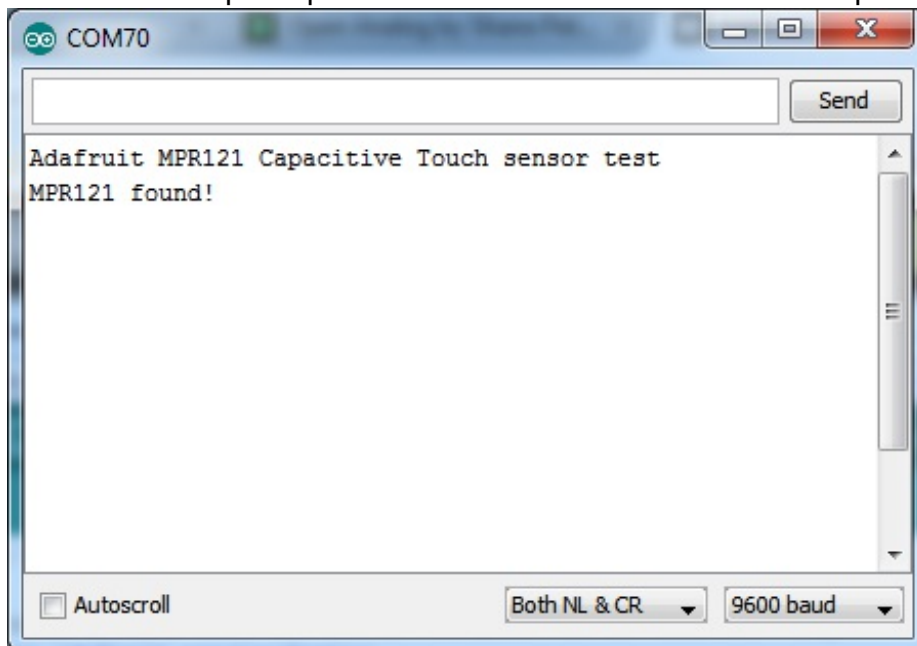http://learn.adafruit.com/adafruit-all-about-arduino-libraries-install-use (http://adafru.it/aYM)

Load Demo

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

MPR121test	Arduino 1.0.5		Adafruit_MAX21100 ▸

File Edit Sketch Tools Help

			Adafruit_MAX21100	▸

Adafruit_MAX21100 ▸
Adafruit_MAX31855 ▸
Adafruit_MCP23008 ▸
Adafruit_MCP23017 ▸
Adafruit_MCP4725 ▸
Adafruit_MCP9808 ▸
Adafruit_MiniMLX9014 ▸ .sor
Adafruit_MLX9014 ▸
Adafruit_MMA8451 ▸ .afruit shop
Adafruit_MotorShield ▸
Adafruit_MPL115A2 ▸
Adafruit_MPL3115A2 ▸
Adafruit_MPR121 ▸   MPR121test
Adafruit_MPU9150 ▸
Adafruit_NECremote ▸
Adafruit_NeoMatrix ▸
Adafruit_NeoPixel ▸ .on
Adafruit_NFCShield_I2C ▸
Adafruit_nRF8001 ▸
Adafruit_OV7670 ▸
Adafruit_PCD8544 ▸

New   Ctrl+N
Open...   Ctrl+O
Sketchbook   ▸
Examples   ▸
Close   Ctrl+W
Save   Ctrl+S
Save As...   Ctrl+Shift+S
Upload   Ctrl+U
Upload Using Programmer Ctrl+Shift+U
Page Setup   Ctrl+Shift+P
Print   Ctrl+P
Preferences   Ctrl+Comma
Quit   Ctrl+Q

****

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

// You can have up to 4 on one i2c bu    ting!
```

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

COM70		
		Send

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

☐ Autoscroll   Both NL & CR ▾   9600 baud ▾

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

https://learn.adafruit.com/adafruit-mpr121-12-key-capacitive-touch-
sensor-breakout-tutorial

```
COM70                                          ─ □ X

[                                        ]  [ Send ]

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
                                                    ▼

☐ Autoscroll          [ Both NL & CR ▼ ] [ 9600 baud ▼ ]
```

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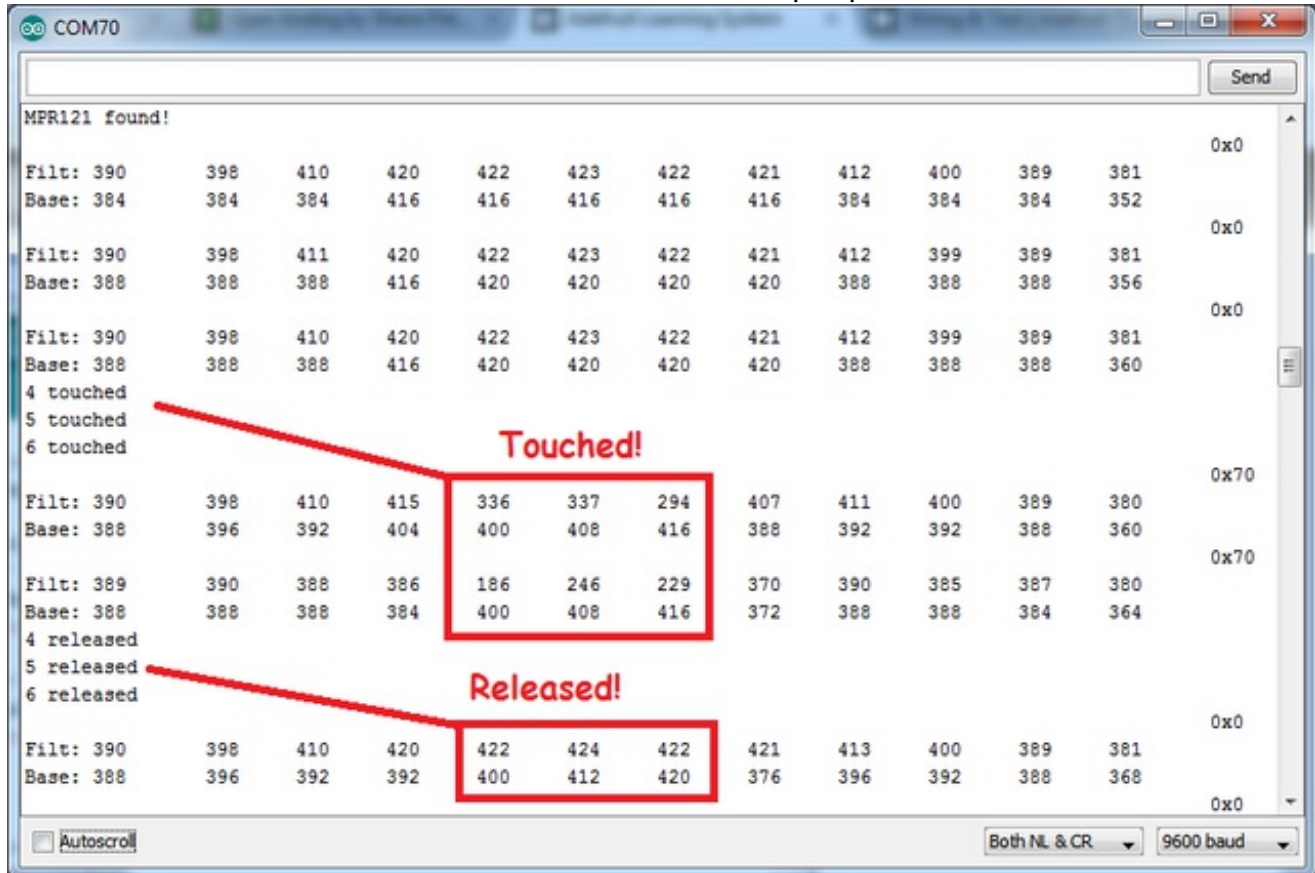
Most people don't need raw data too much, but it can be handy if doing intense debugging. It can be helpful if you are tweaking your sensors to get good responsivity.

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.

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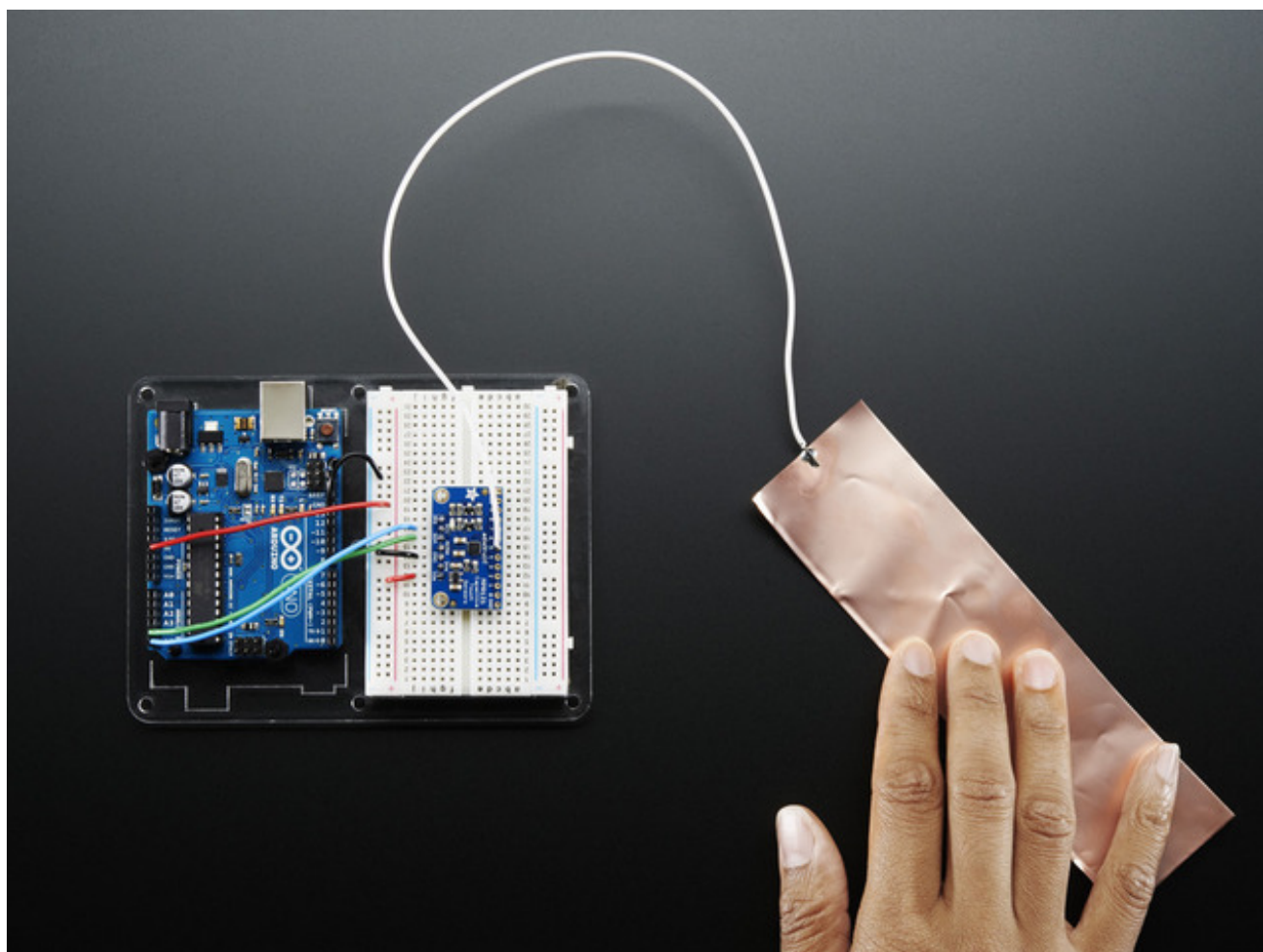
Once you have the MPR121 breakout working you'll want to construct electrodes. These are large conductive piece of copper, foil, paint, etc that will act as the "thing you touch"

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. (http://adafru.it/dKl)

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...

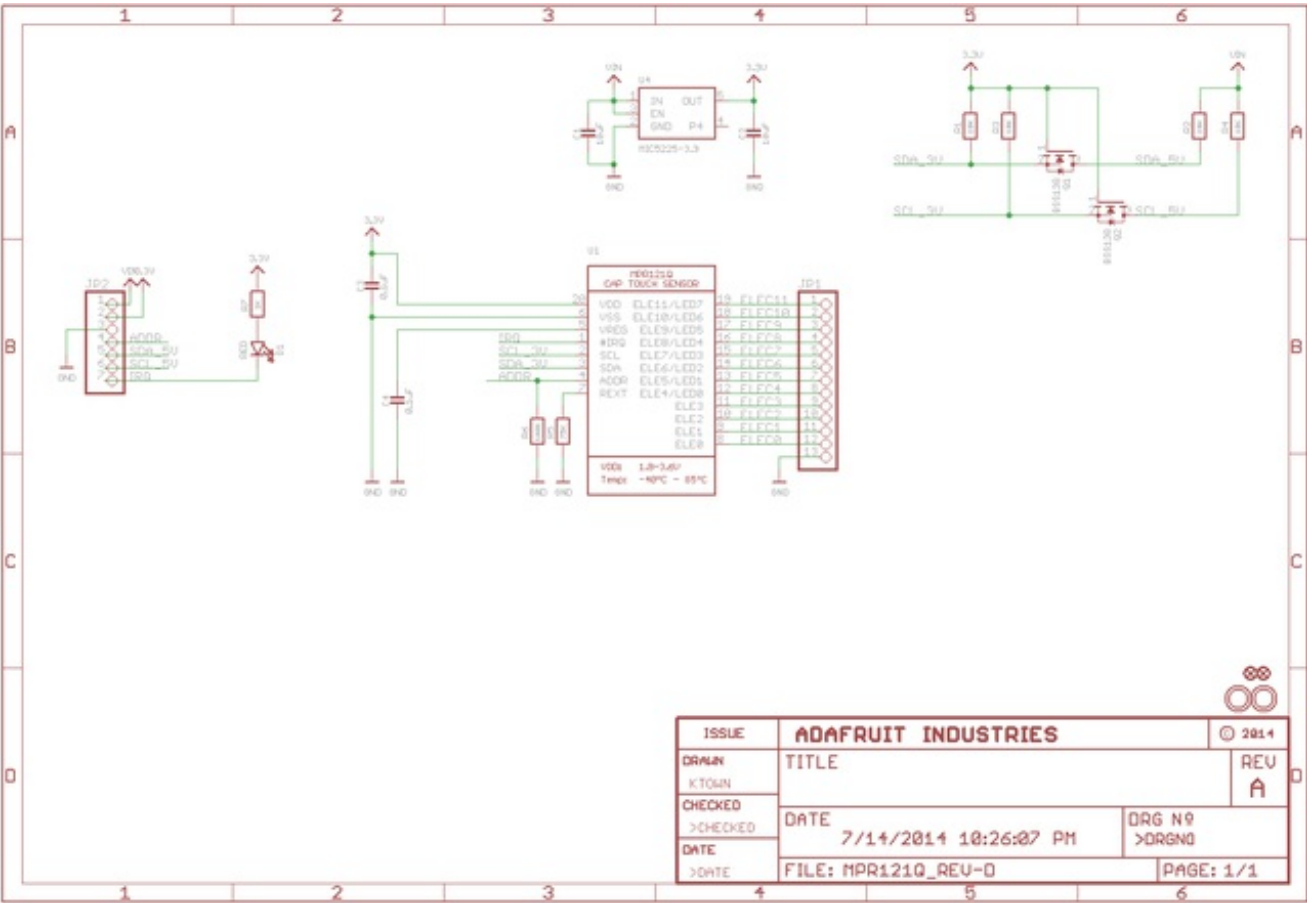**connect all your wires, electrodes, fruit, etc..*then* start up the capacitive touch program!**

Downloads

Datasheets & Files

- MPR121 Datasheet (http://adafru.it/dKG)
- EagleCAD PCB files on GitHub (http://adafru.it/pJa)
- Fritzing object in Adafruit Fritzing library (http://adafru.it/c7M)

Breakout Board Schematic

Fabrication Print

Dimensions in Inches