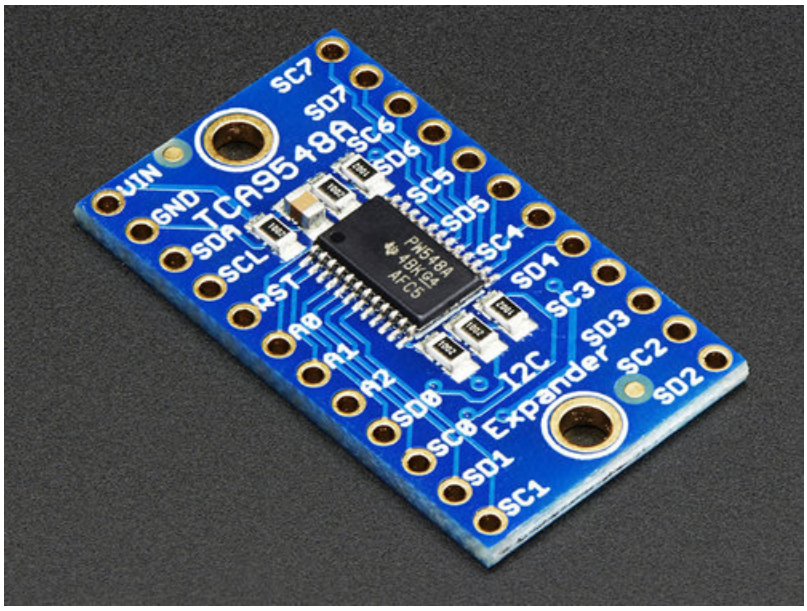
adafruit learning system

Adafruit TCA9548A 1-to-8 I2C Multiplexer Breakout

Created by lady ada



Last updated on 2018-08-22 03:50:09 PM UTC

Guide Contents

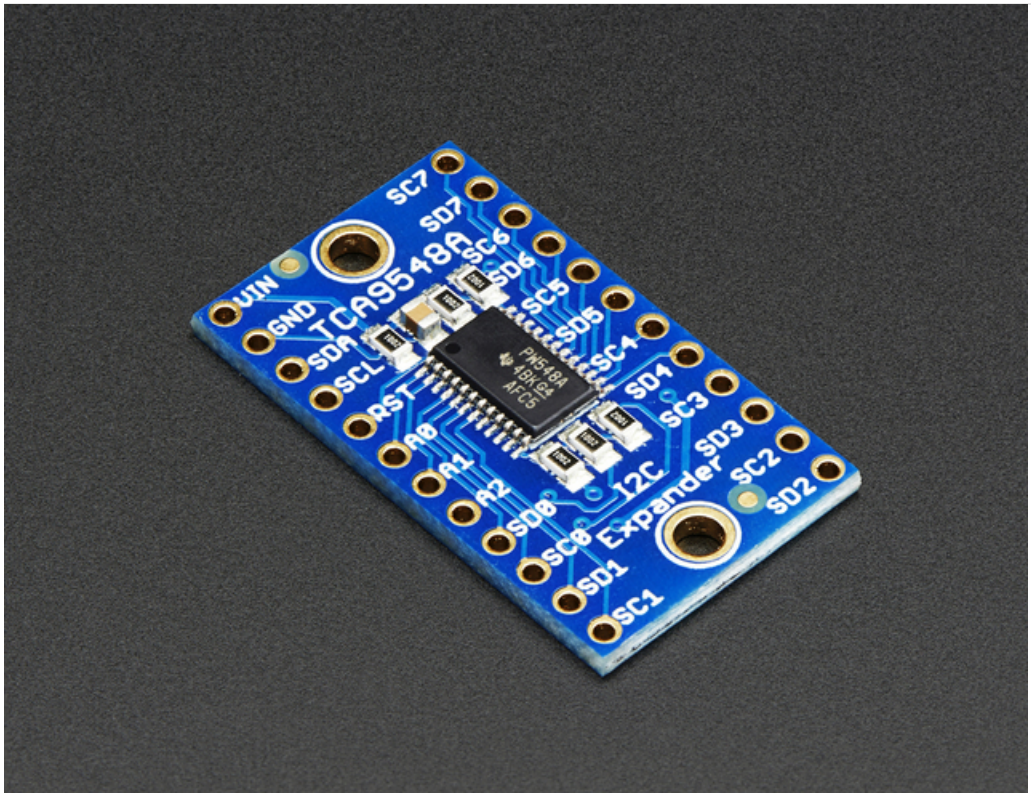
Guide Contents 2
Overview 3
Pinouts 6
- Power Pins: 6
- I2C Control-Side pins: 6
- I2C Multiplexed-Side pins: 6

Assembly 8
- Prepare the header strip: 8
- Add the breakout board: 9
- And Solder! 9

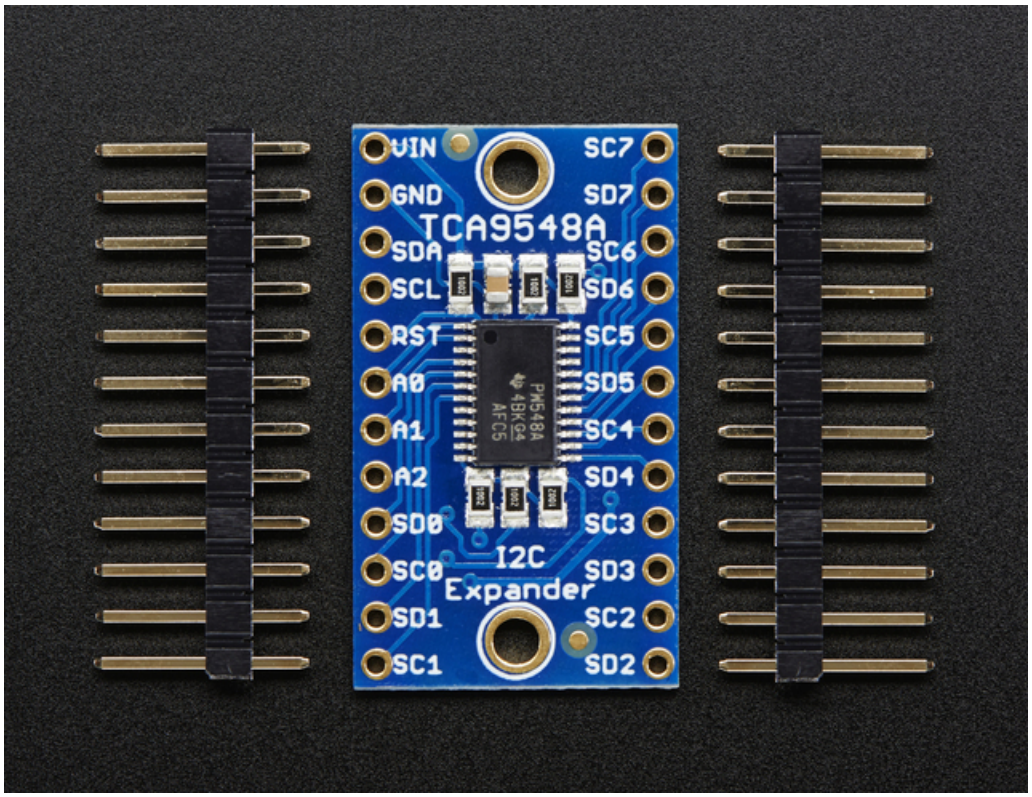
Wiring & Test 12
Example Multiplexing 12
- Multiple Multplexers 16

Downloads 18
Datasheets 18
Schematic 18
Fabrication Print 18

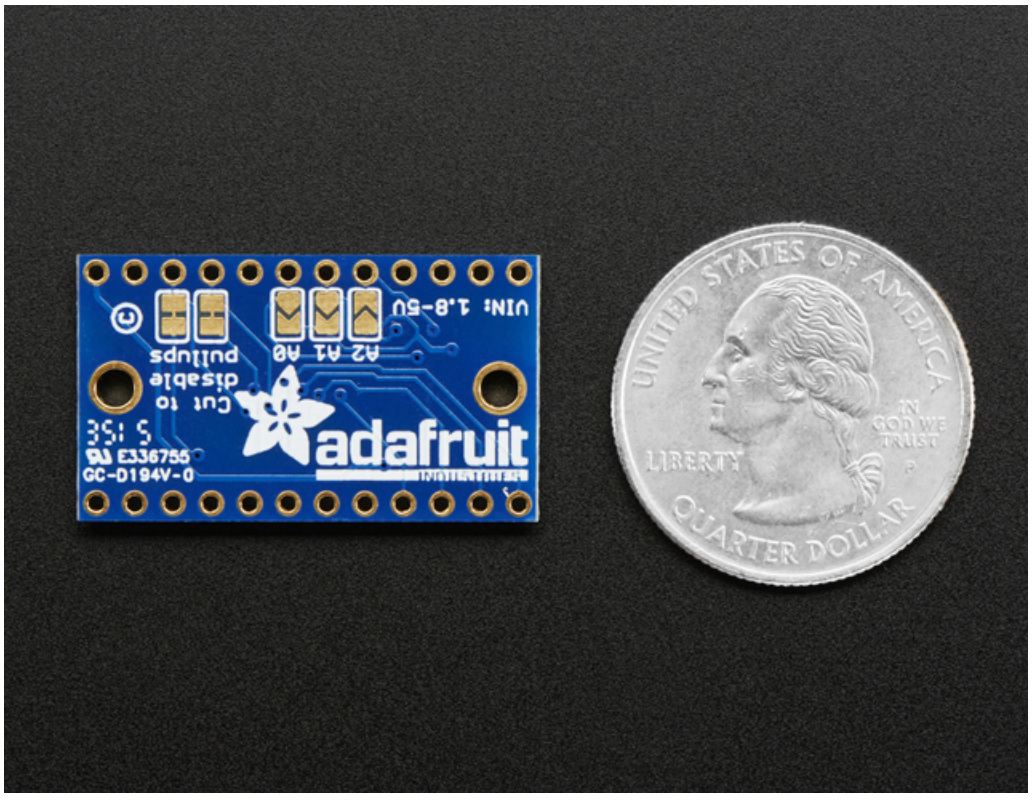
Overview



You just found the perfect I2C sensor, and you want to wire up two or three or more of them to your Arduino when you realize "Uh oh, this chip has a fixed I2C address, and from what I know about I2C, you cannot have two devices with the same address on the same SDA/SCL pins!" Are you out of luck? You would be, if you didn't have this ultra-cool **TCA9548A 1-to-8 I2C multiplexer!**

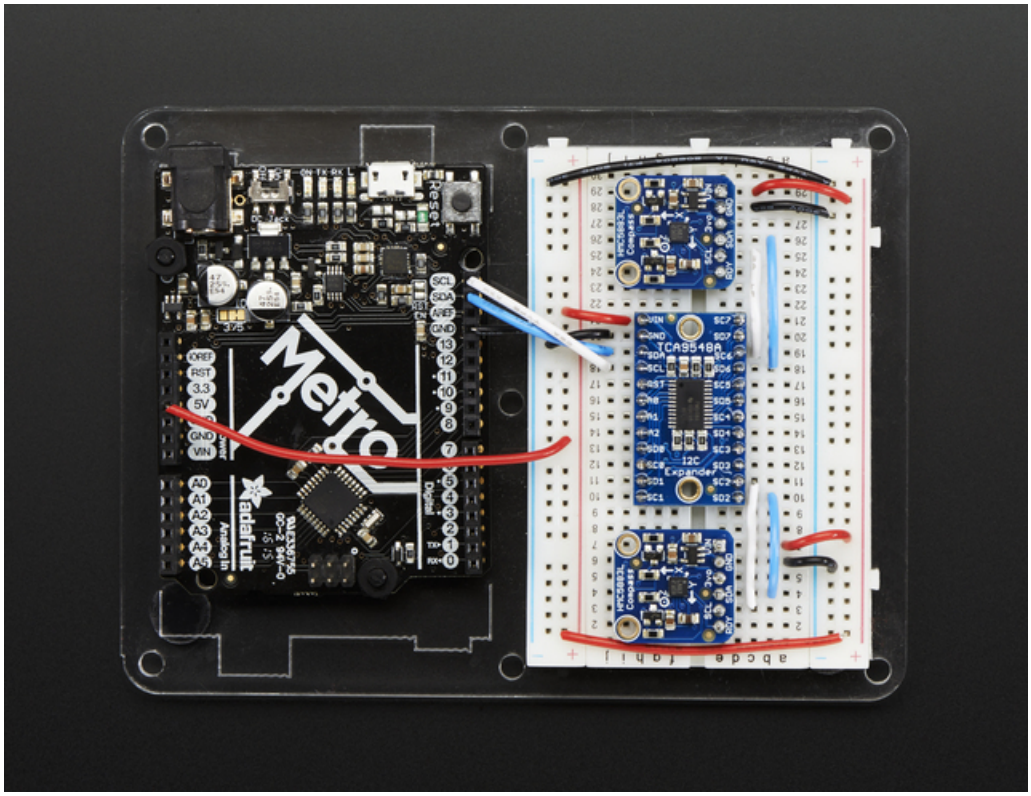


Finally, a way to get up to 8 same-address I2C devices hooked up to one microcontroller - this multiplexer acts as a gatekeeper, shuttling the commands to the selected set of I2C pins with your command.



Using it is fairly straight-forward: the multiplexer itself is on I2C address 0x70 (but can be adjusted from 0x70 to 0x77)

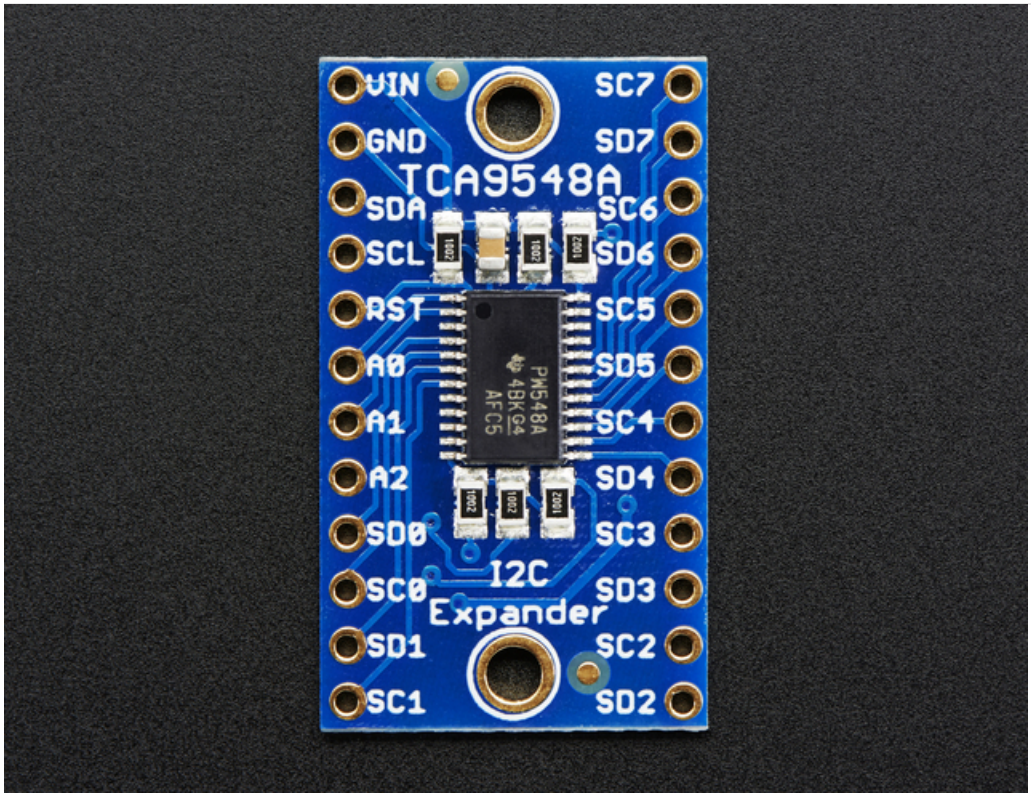
and you simply write a single byte with the desired multiplexed output number to that port, and bam - any future I2C packets will get sent to that port. In theory, you could have 8 of these multiplexers on each of 0x70-0x77 addresses in order to control 64 of the same-I2C-addressed-part.



Like all Adafruit breakouts, we put this nice chip on a breakout for you so you can use it on a breadboard with capacitors, and pullups and pulldowns to make usage a snap. Some header is required and once soldered in you can plug it into a solderless-breadboard. The chip itself is 1.8V - 5V compliant so you can use it with any logic level.

We even wrote up a nice tutorial with wiring diagrams, schematics and examples to get you running in 10 minutes! (https://adafru.it/jhC)

Pinouts



Power Pins:

- **Vin** - this is the power pin. Since the sensor chip uses 3-5 VDC. 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
- **GND** - common ground for power and logic

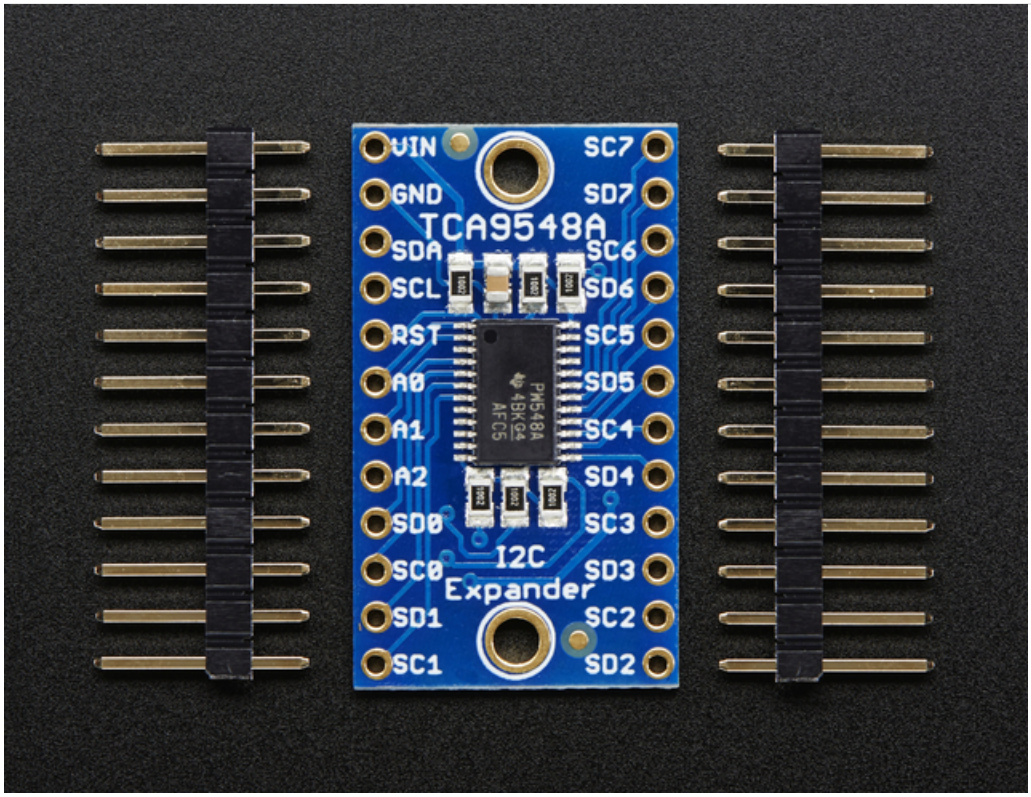
I2C Control-Side pins:

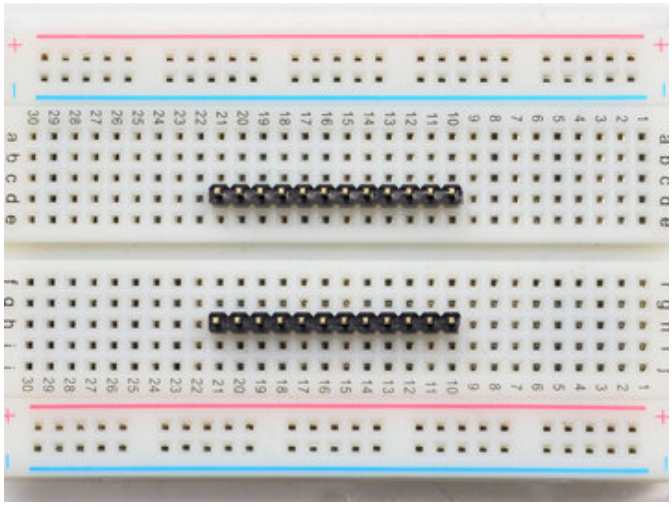
- **SCL** - this is the I2C clock pin for the chip itself, connect to your microcontrollers I2C clock line.
- **SDA** - this is the I2C data pin for the chip itself, connect to your microcontrollers I2C data line.
- **RST** - this is the reset pin, for resetting the multiplexer chip. Pulled high by default, connect to ground to reset
- **A0 A1 A2** - these are the address selection pins *for the multiplexer*. By default the multiplexer is at address **0x70** and these three pins are pulled low. Connect them to **Vin** to set the address to **0x71** - **0x77**.
- **A0** is the lowest-significant bit (if it is pulled high, it will increase the address by 1).
- **A1** is the 2nd-lowest-significant bit (if it is pulled high, it will increase the address by 2).
- **A2** is the 3rd-lowest-significant bit (if it is pulled high, it will increase the address by 4).

I2C Multiplexed-Side pins:

- **SDx** and **SCx**: There are 8 sets of **SDx** and **SCx** pins, from **SD0/SC0** to **SD7/SC7**. These are the multiplexed pins. Each one is a completely seperate I2C bus set. So you have have 8 I2C devices with identical addresses, as long as they are on one I2C bus each.
  These pins do not have any pullups installed, so if you are using a chip or breakout without i2c pullups be sure to add them! Nicely, you can have **Vin** be 3.3V and have these pins pulled up to 5V (that is, they are 5V compliant)

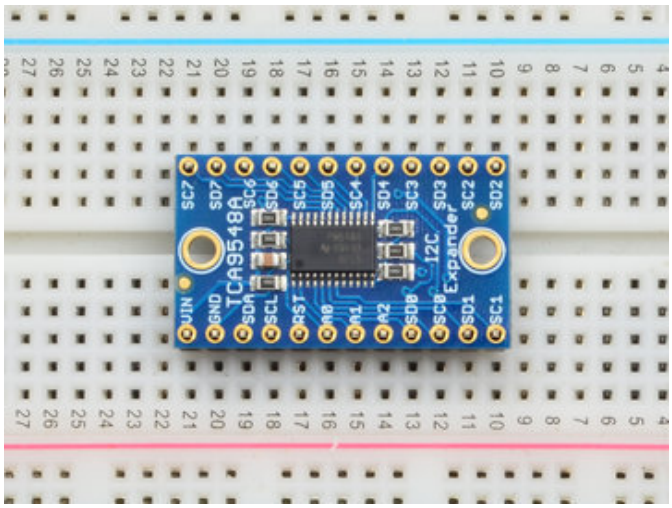
Assembly





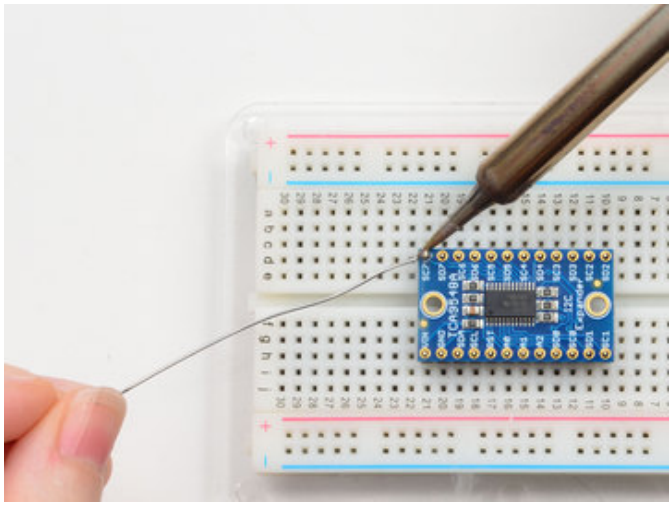
Prepare the header strip:

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:

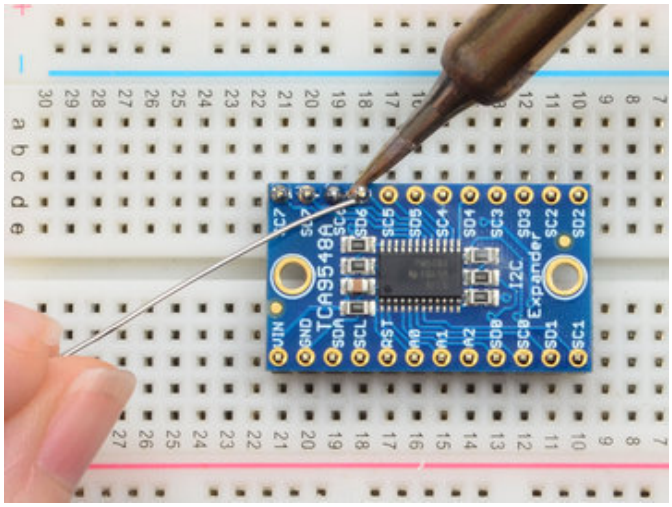
Place the breakout board over the pins so that the short pins poke through the breakout pads

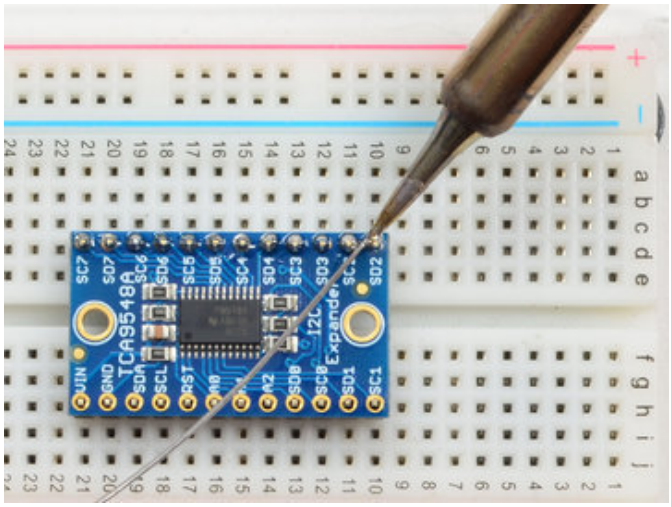


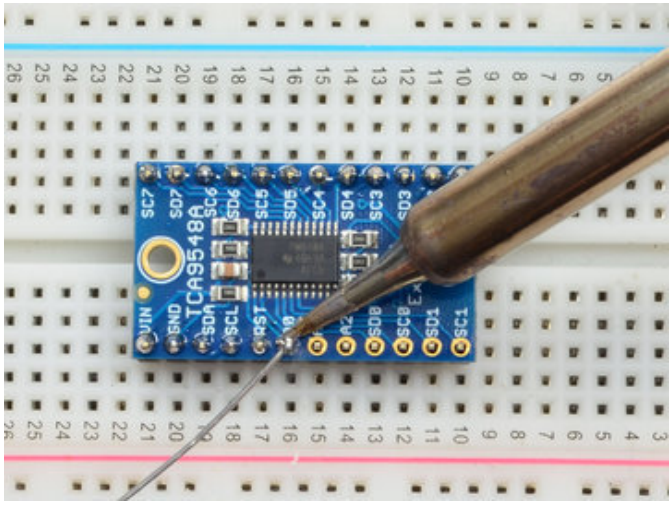
And Solder!

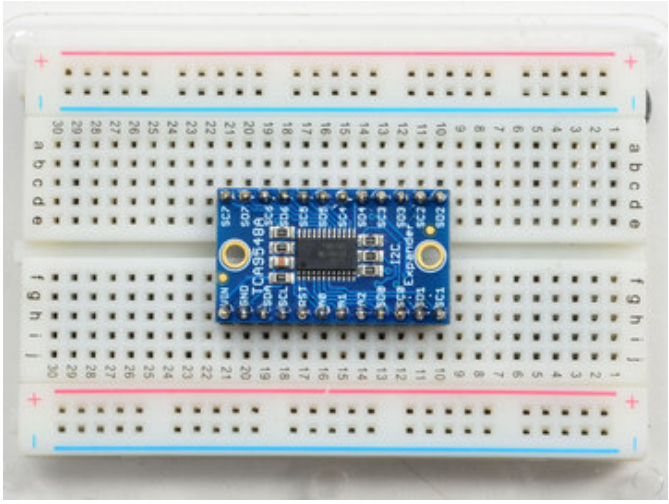
Be sure to solder all pins for reliable electrical contact.

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


TCA9548A
I2C Expander
VIN
GND
SDA
SCL
RST
A0
A1
A2
SD0
SC0
SD1
SC1
SD2
SC2
SD3
SC3
SD4
SC4
SD5
SC5
SD6
SC6
SD7
SC7


TCA9548A
I2C Expander
VIN
GND
SDA
SCL
RST
A0
A1
A2
SD0
SC0
SD1
SC1
SD2
SC2
SD3
SC3
SD4
SC4
SD5
SC5
SD6
SC6
SD7
SC7


TCA9548A
VIN
GND
SDA
SCL
RST
A0
A1
A2
SD0
SC0
SD1
SC1
SD2
SC2
SD3
SC3
SD4
SC4
SD5
SC5
SD6
SC6
SD7
SC7



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

Wiring & Test

The TCA9548A multiplexer is interesting in that it has an I2C address (0x70 by default) - and you basically send it a command to tell it which I2C multiplexed output you want to talk to, then you can address the board you want to address.

We suggest using this little helper to help you select the port

```
#define TCAADDR 0x70

void tcaselect(uint8_t i) {
  if (i > 7) return;

  Wire.beginTransmission(TCAADDR);
  Wire.write(1 << i);
  Wire.endTransmission();
}
```

You can then call tcaselect(0) thru tcaselect(7) to set up the multiplexer.

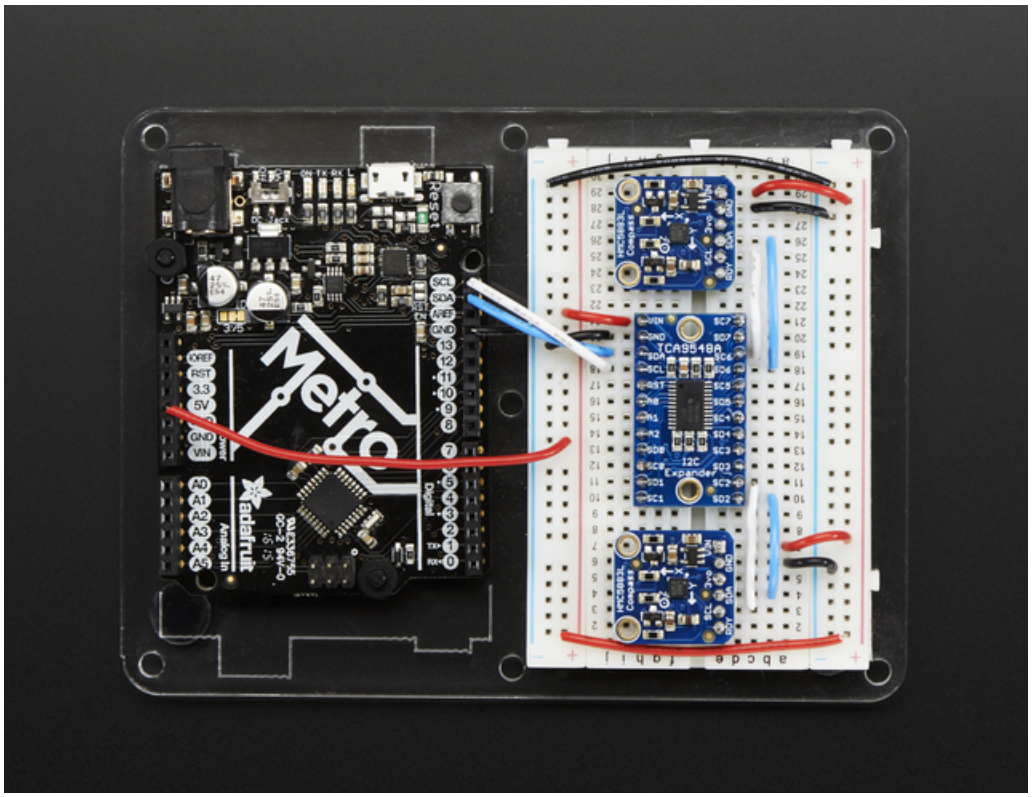
Note that you if you happen to have I2C devices with I2C address 0x70, you will need to short one of the **Addr** pins on the TCA9548 breakout to **Vin** in order to make it not conflict. Given that you can have 0x70 thru 0x77, just find one that's free and you're good to go!

Example Multiplexing

For example, say we want to talk to two HMC5883 breakouts. These magnetometers have a fixed address of **0x1E** so you cannot have two on one I2C bus. Wire up the TCA9548 breakout so that:

- **Vin** is connected to **5V** (on a 3V logic Arduino/microcontroller, use **3.3V**)
- **GND** to ground
- **SCL** to I2C clock
- **SDA** to I2C data

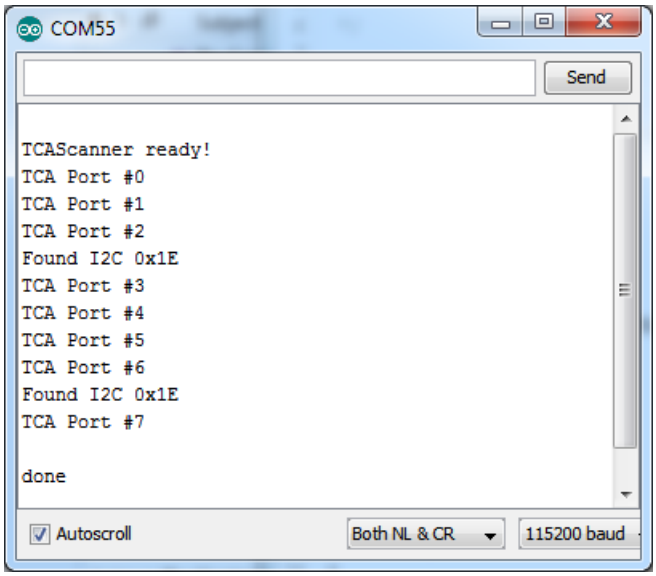
Then wire up each of the other sensor breakouts to **Vin**, **Ground** and use one of the **SC*n*** / **SD*n*** multiplexed buses:



On an Arduino, which is what we're using, we suggest running this handy scanner script which will tell you what the breakout detected

```
/**
 * TCA9548 I2CScanner.pde -- I2C bus scanner for Arduino
 *
 * Based on code c. 2009, Tod E. Kurt, http://todbot.com/blog/
 *
 */

#include "Wire.h"
extern "C" {
#include "utility/twi.h"  // from Wire library, so we can do bus scanning
}

#define TCAADDR 0x70

void tcaselect(uint8_t i) {
  if (i > 7) return;

  Wire.beginTransmission(TCAADDR);
  Wire.write(1 << i);
  Wire.endTransmission();
}


// standard Arduino setup()
void setup()
{
    while (!Serial);
    delay(1000);

    Wire.begin();

    Serial.begin(115200);
    Serial.println("\nTCAScanner ready!");

    for (uint8_t t=0; t<8; t++) {
      tcaselect(t);
      Serial.print("TCA Port #"); Serial.println(t);

      for (uint8_t addr = 0; addr<=127; addr++) {
        if (addr == TCAADDR) continue;

        uint8_t data;
        if (! twi_writeTo(addr, &data, 0, 1, 1)) {
           Serial.print("Found I2C 0x");  Serial.println(addr,HEX);
        }
      }
    }
    Serial.println("\ndone");
}

void loop()
{
}
```

For example, running it on the above setup will give you:



Next up you will have to adjust whatever code you have to select the correct multiplexed port!

Make sure before you query from the sensor that you call **tcaselect** to get the right one

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_HMC5883_U.h>

#define TCAADDR 0x70

/* Assign a unique ID to this sensor at the same time */
Adafruit_HMC5883_Unified mag1 = Adafruit_HMC5883_Unified(1);
Adafruit_HMC5883_Unified mag2 = Adafruit_HMC5883_Unified(2);

void displaySensorDetails(Adafruit_HMC5883_Unified *mag)
{
  sensor_t sensor;
  mag->getSensor(&sensor);
  Serial.println("------------------------------------");
  Serial.print  ("Sensor:       "); Serial.println(sensor.name);
  Serial.print  ("Driver Ver:   "); Serial.println(sensor.version);
  Serial.print  ("Unique ID:    "); Serial.println(sensor.sensor_id);
  Serial.print  ("Max Value:    "); Serial.print(sensor.max_value); Serial.println(" uT");
  Serial.print  ("Min Value:    "); Serial.print(sensor.min_value); Serial.println(" uT");
  Serial.print  ("Resolution:   "); Serial.print(sensor.resolution); Serial.println(" uT");
  Serial.println("------------------------------------");
  Serial.println("");
  delay(500);
}

void tcaselect(uint8_t i) {
  if (i > 7) return;

  Wire.beginTransmission(TCAADDR);
  Wire.write(1 << i);
  Wire.endTransmission();
}
```

```
void setup(void)
{
  Serial.begin(9600);
  Serial.println("HMC5883 Magnetometer Test"); Serial.println("");

  /* Initialise the 1st sensor */
  tcaselect(2);
  if(!mag1.begin())
  {
    /* There was a problem detecting the HMC5883 ... check your connections */
    Serial.println("Ooops, no HMC5883 detected ... Check your wiring!");
    while(1);
  }

  /* Initialise the 2nd sensor */
  tcaselect(6);
  if(!mag2.begin())
  {
    /* There was a problem detecting the HMC5883 ... check your connections */
    Serial.println("Ooops, no HMC5883 detected ... Check your wiring!");
    while(1);
  }

  /* Display some basic information on this sensor */
  tcaselect(2);
  displaySensorDetails(&mag1);
  tcaselect(6);
  displaySensorDetails(&mag2);
}

void loop(void)
{
  /* Get a new sensor event */
  sensors_event_t event;

  tcaselect(2);
  mag1.getEvent(&event);

  /* Display the results (magnetic vector values are in micro-Tesla (uT)) */
  Serial.print("Sensor #1 - ");
  Serial.print("X: "); Serial.print(event.magnetic.x); Serial.print("  ");
  Serial.print("Y: "); Serial.print(event.magnetic.y); Serial.print("  ");
  Serial.print("Z: "); Serial.print(event.magnetic.z); Serial.print("  ");Serial.println("uT");

  tcaselect(6);
  mag2.getEvent(&event);
  /* Display the results (magnetic vector values are in micro-Tesla (uT)) */
  Serial.print("Sensor #2 - ");
  Serial.print("X: "); Serial.print(event.magnetic.x); Serial.print("  ");
  Serial.print("Y: "); Serial.print(event.magnetic.y); Serial.print("  ");
  Serial.print("Z: "); Serial.print(event.magnetic.z); Serial.print("  ");Serial.println("uT");

  delay(500);
}
```

However, once you add all the **tcaselect()**'s you will be able to talk to both sensors!

Multiple Multplexers

Since the TCA9548 is addressible, you can have more than one multiplexer on the bus. With 8 possible adresses, that means you can control as many as 64 separate i2c buses.

To avoid conflict between devices with the same address on different multiplexers, you can disable all channels on a multiplexer with the following code:

```
Wire.beginTransmission(TCAADDR1);
Wire.write(0);  // no channel selected
Wire.endTransmission();
```

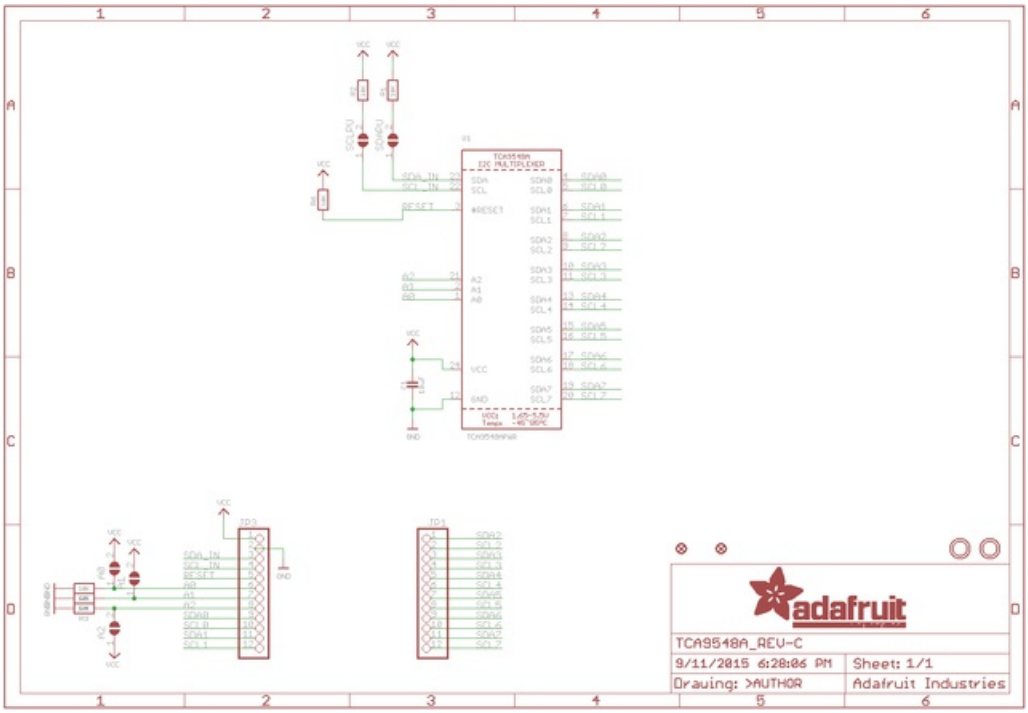
Downloads

Datasheets

- TCA9548A datasheet (https://adafru.it/id8)
- Fritzing object in the Adafruit Fritzing Library (https://adafru.it/aP3)
- EagleCAD PCB files on GitHub (https://adafru.it/rzF)

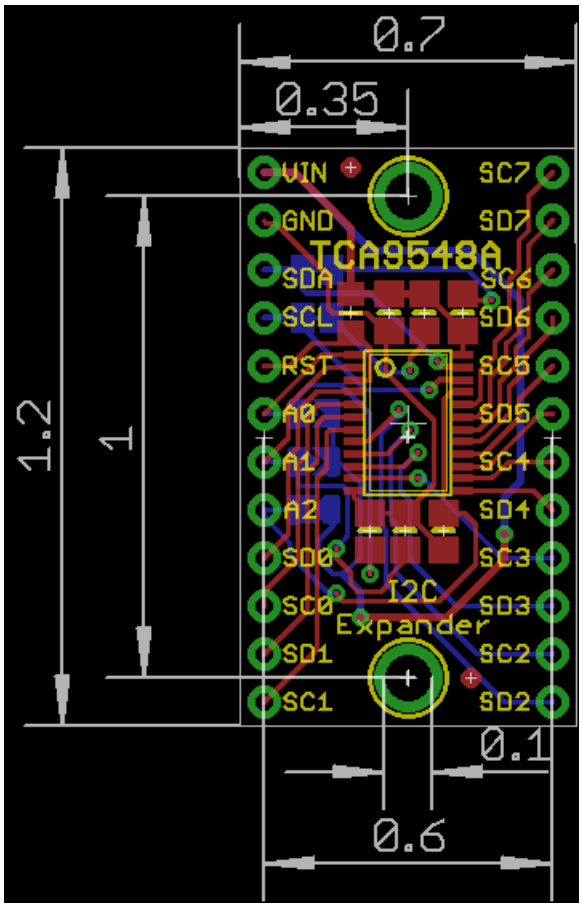
Schematic

Click to embiggen



Fabrication Print

Dimensions in Inches


0.7
0.35
VIN
GND
SDA
SCL
RST
A0
A1
A2
SD0
SC0
SD1
SC1
SC7
SD7
SC6
SD6
SC5
SD5
SC4
SD4
SC3
SD3
SC2
SD2
TCA9548A
I2C
Expander
1.2
1
0.1
0.6