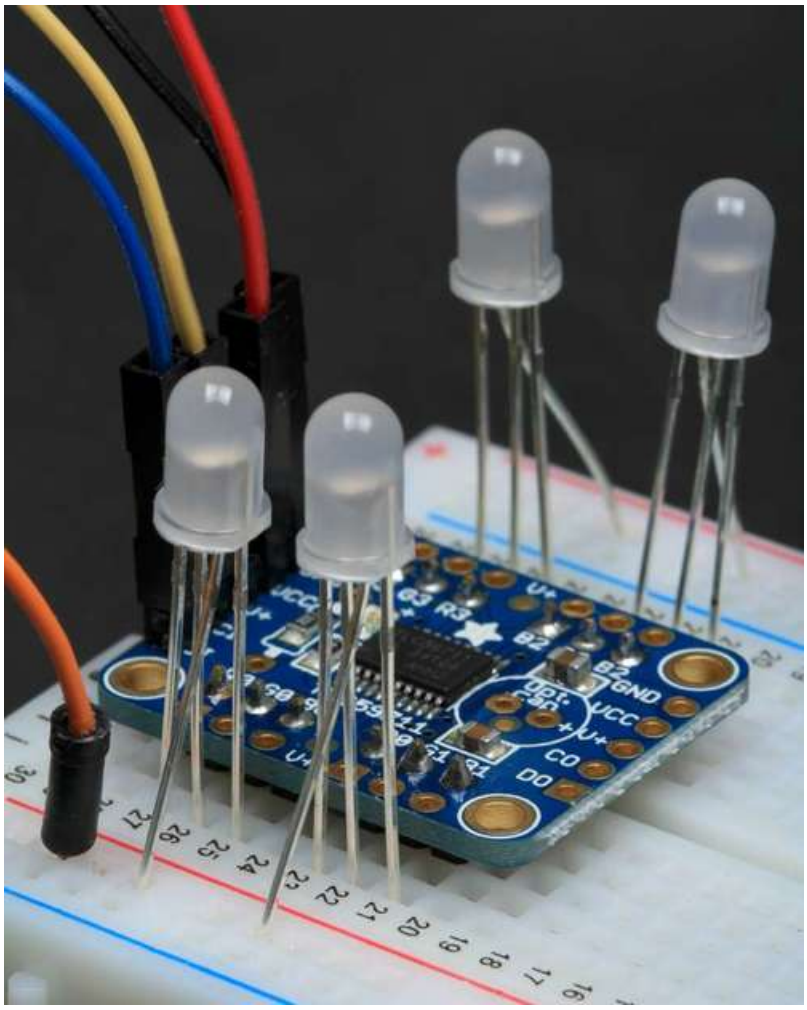
adafruit learning system

TLC5947 and TLC59711 PWM LED Driver Breakouts

Created by Bill Earl

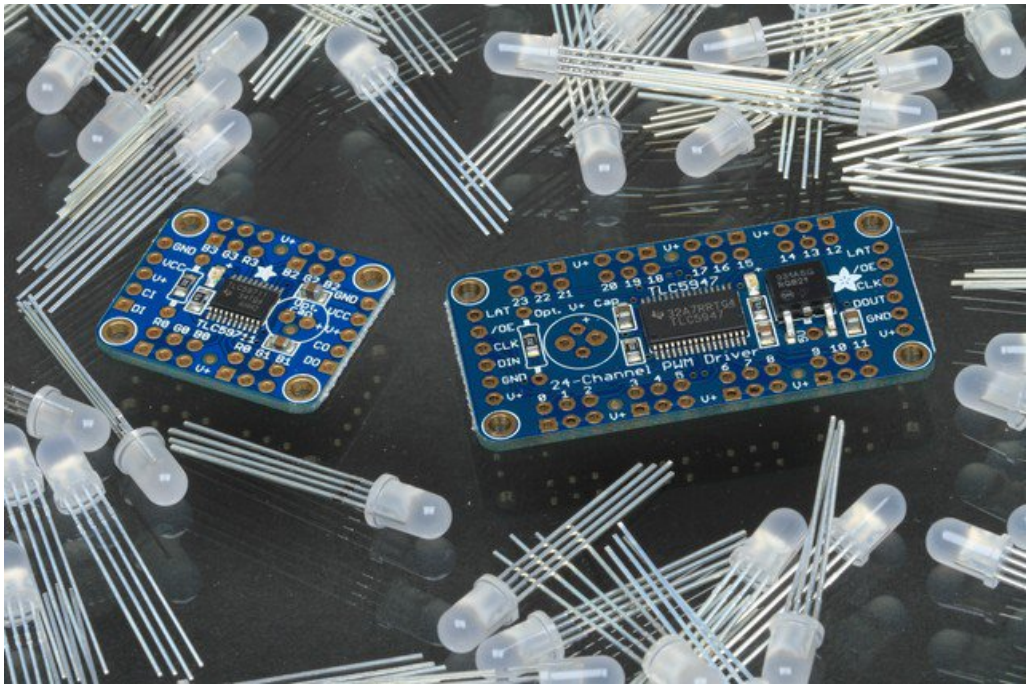


Last updated on 2018-02-03 06:08:45 PM UTC

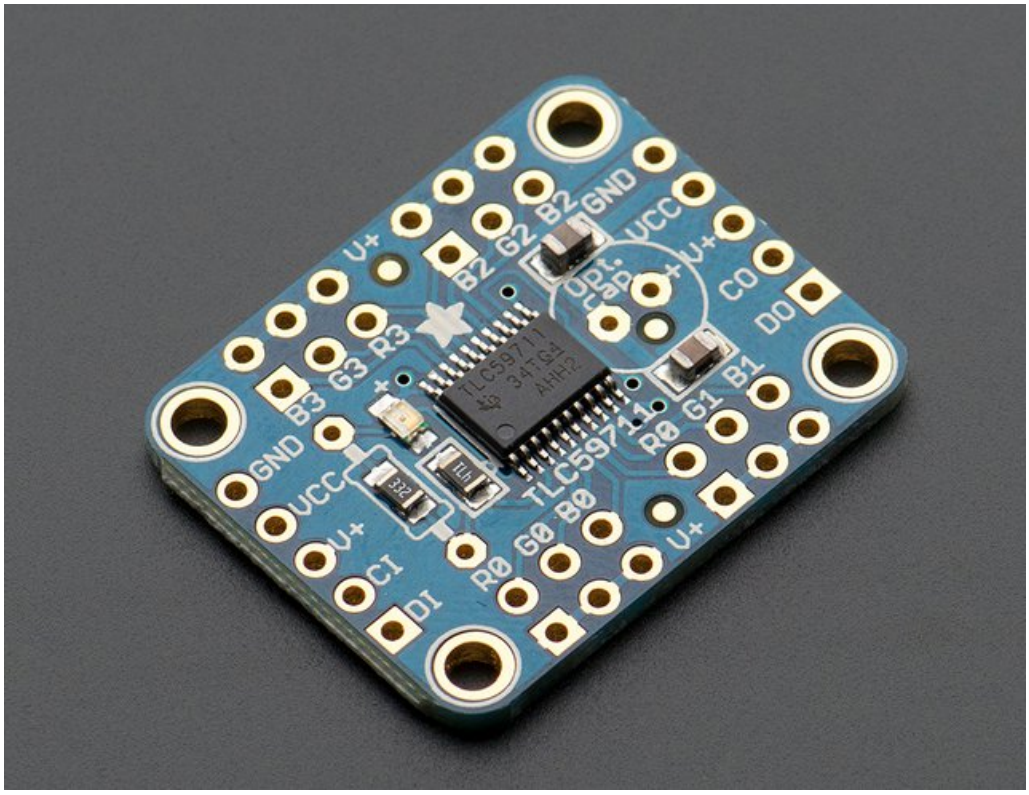
Guide Contents

Overview

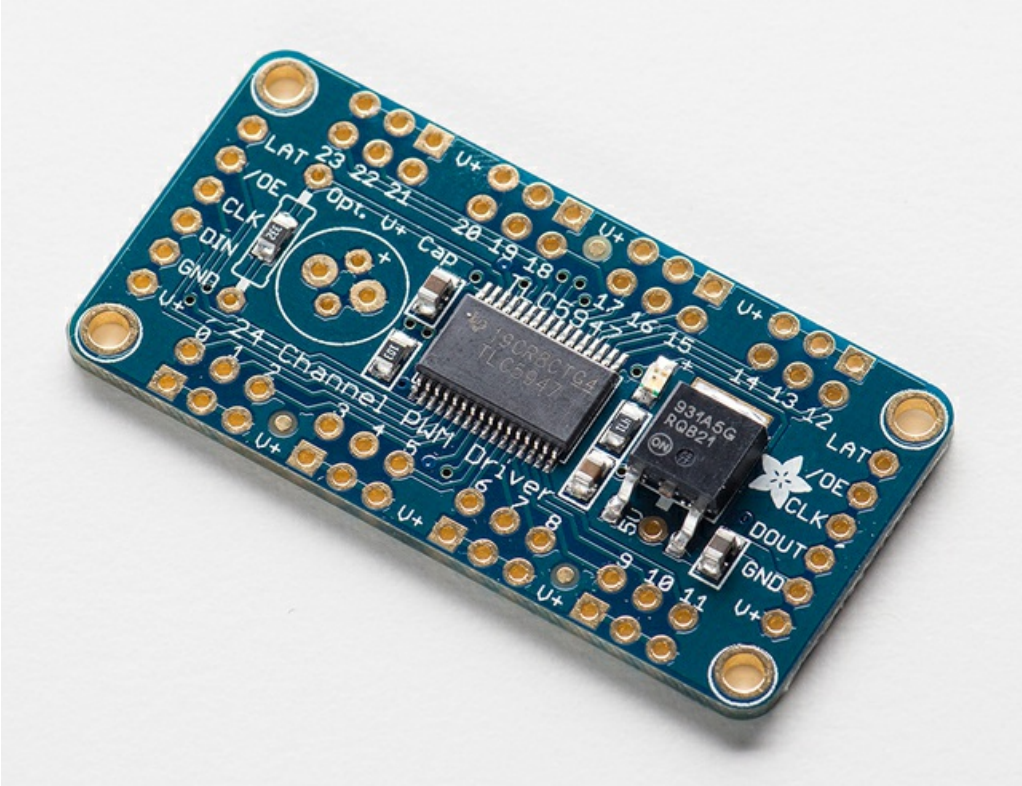
The TLC59711 and TLC5947 breakout boards are ideal for applications requiring precise control of lots of LEDs.



The **TLC59711** can control 12 separate channels of 16-bit PWM output. This is the highest-resolution PWM board we've seen!



The **TLC5947** has even more channels. It can control 24 separate channels with 12-bit PWM output.



Both boards have a 2 or 3-pin SPI interface. Our library lets you use any two (TLC59711) or three (TLC5947) free pins to drive them. Best of all, you can chain multiple boards together to control hundreds or thousands of LEDs!

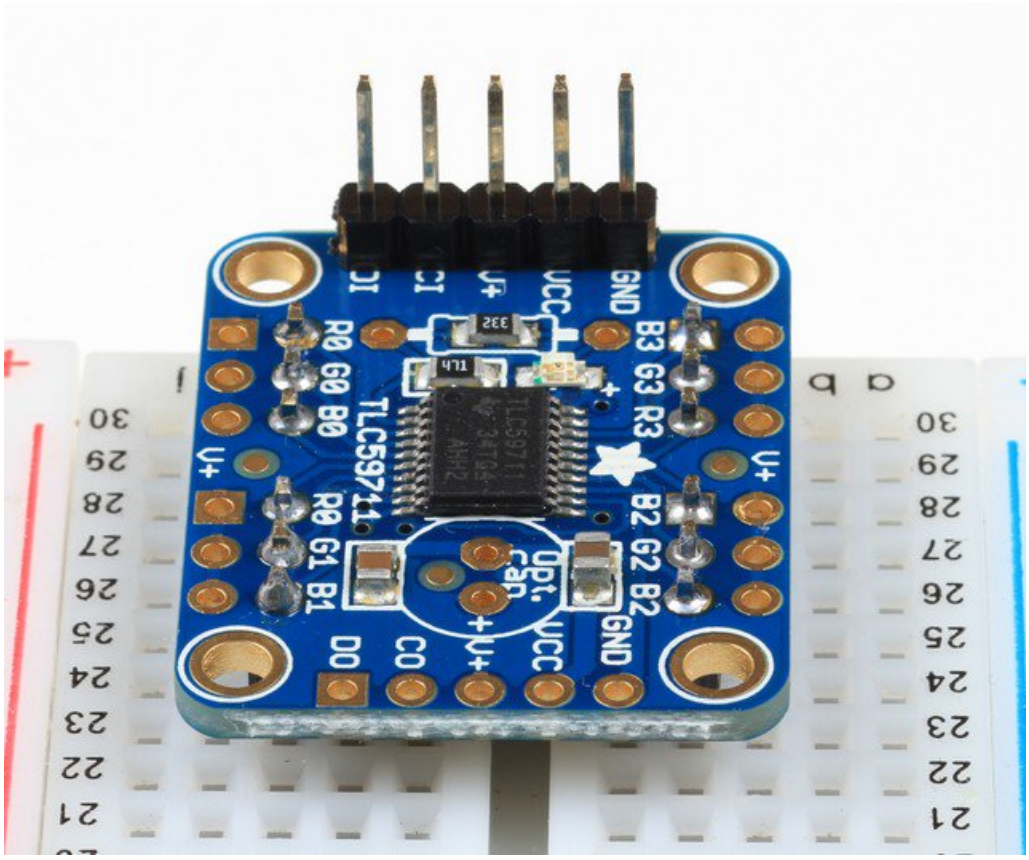
Outputs from these boards are constant-current and open drain. You can drive multiple LEDs in series. One resistor is used to set the current for each of the outputs, the constant current means that the LED brightness doesn't vary if the power supply dips.

We supply these with a 3.3K resistor for about 15mA per channel. But you can solder a thru-hole resistor over it if you'd like to change that value

Assembly

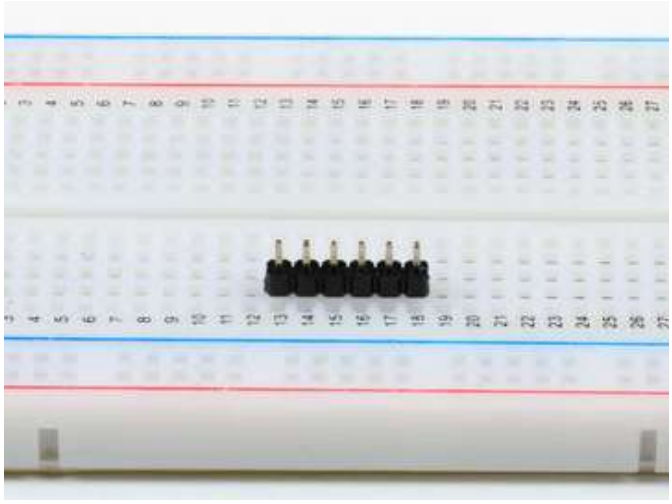
Assembly:

These boards come fully assembled and tested. We include optional header strips in case you want to use these on a breadboard. These take just a few minutes to install. For use in a breadboard, it is easiest to have the control connections on top of the board and the led connections on the bottom as shown below:



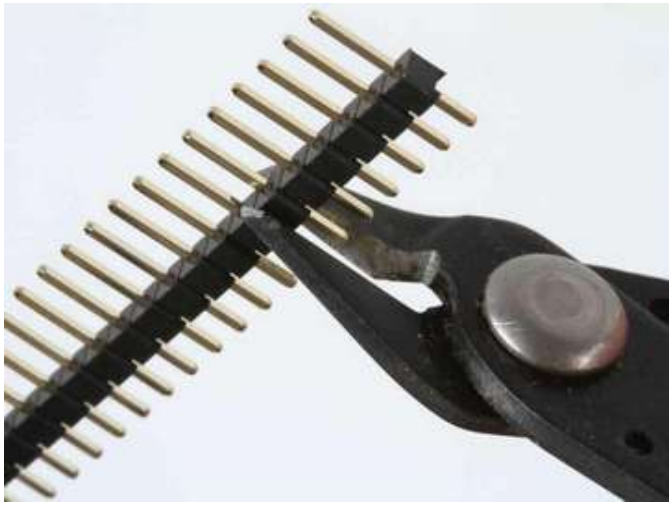
Soldering the Headers

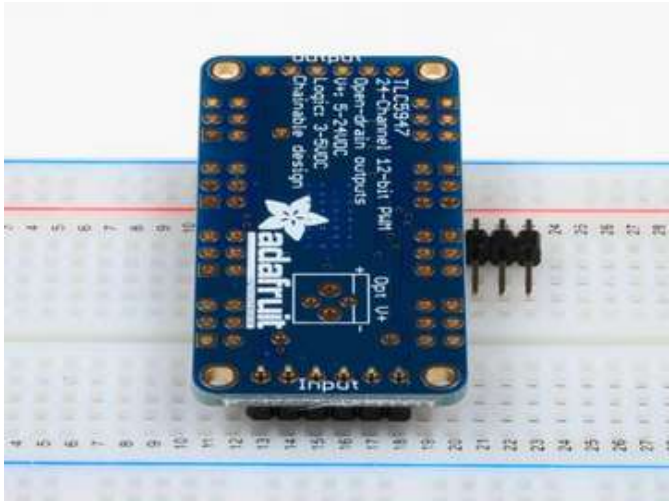
Use the breadboard to hold the parts in position for soldering.



Position the header

Cut to size first if necessary

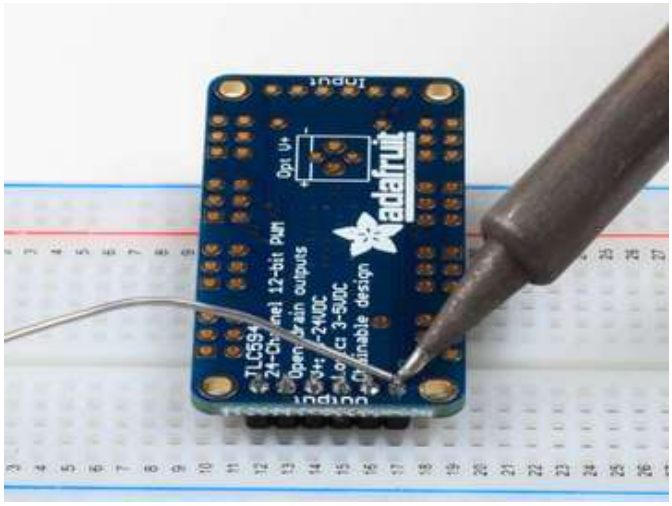




Position the board

Remember to place it upside down if you want the pins to end up on top.

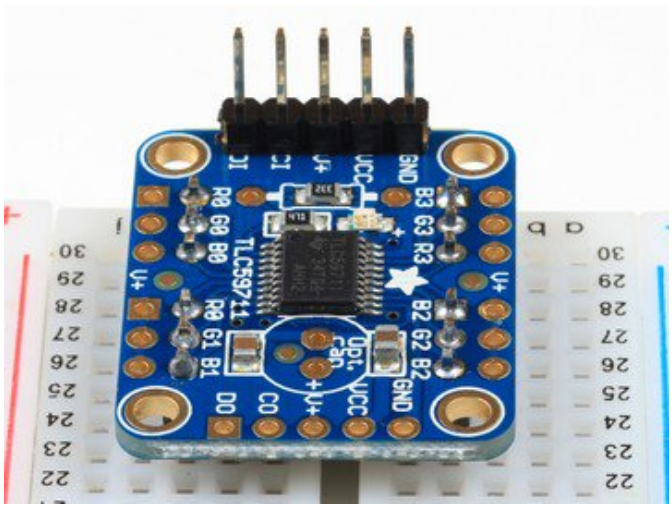
***Hint**: A spare piece of header placed under the breakout board will help keep things aligned for soldering.*



And solder!

Be sure to solder all pins for good electrical connection.

If you are new to soldering, check out the Adafruit Guide to Excellent Soldering (https://adafru.it/aTk)



Repeat for the other side

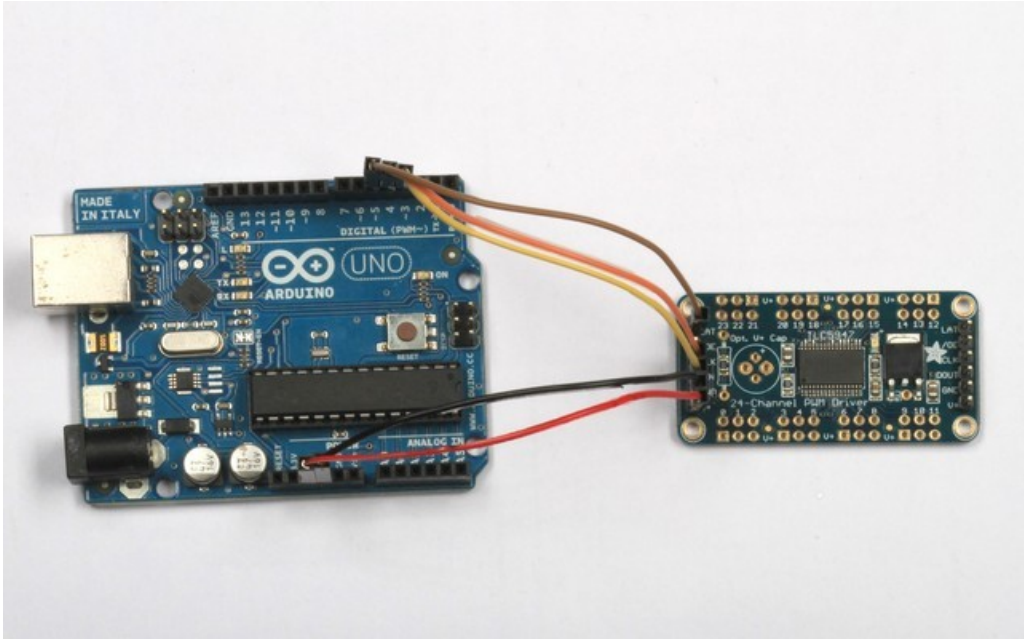
Flip the board over and repeat the process for the LED headers on the bottom of the board.

Connecting to the Arduino

These boards communicate using an SPI protocol. The wiring is slightly different for the two boards, so we will describe them separately. For making breadboard connections with the header pins on top of the board, a set of male-female jumpers are handy.

The pin configurations below are consistent with the example code supplied with the libraries:

24-Channel TLC5947



Connect to the Arduino as follows:

- **DIN -> Digital 4**
- **CLK -> Digital 5**
- **LAT -> Digital 6**
- **GND -> GND**
- **V+ -> VIN**

Here we show V+ connected to the Arduino VIN pin. This will power the breakout board and LED directly from the supply connected to the DC power jack. The TLC5947 can accept a V+ of 5v-30v. Higher voltages allow you to drive multiple LEDs in series from each channel.

The **DIN/CLK/LAT** pins can be changed to any other pins later

> If you need V+ voltages higher than 12v, you will need to use a separate power supply. If you use a separate supply, be sure to connect the ground wire to the Arduino ground.

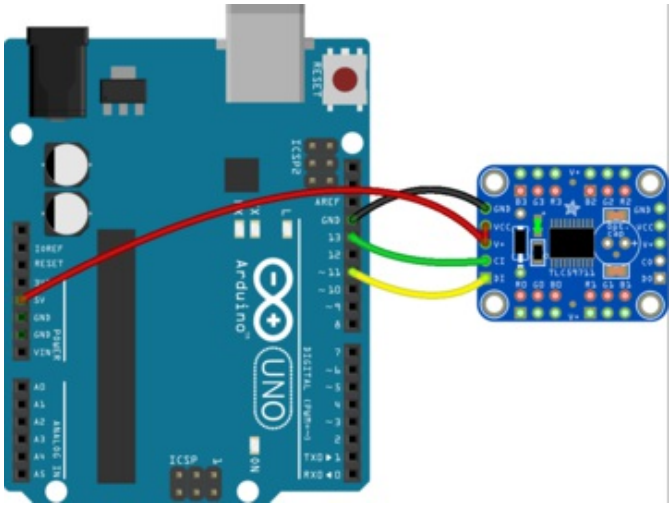
12-Channel TLC59711

You've got *two options* for powering/wiring your TLC59711

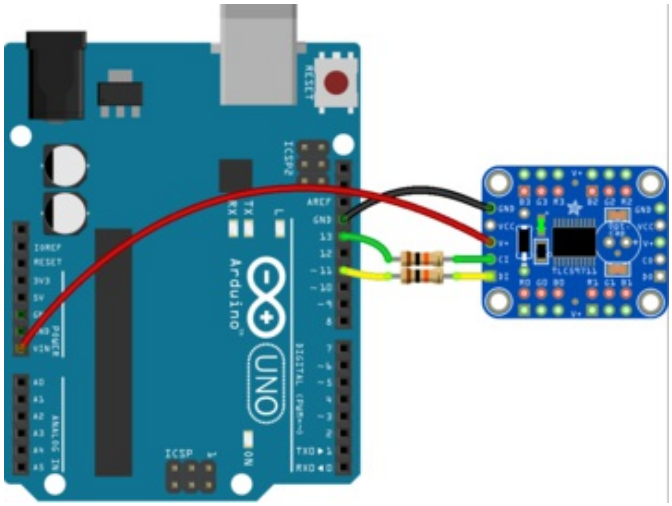
1. LEDs **V+** and logic level **VCC** connected together to 3 - 5V

2. LEDs **V+** at 4-17V and logic level **VCC** at 3.3V

If you need to use it with a 5V logic UNO, you can **do either of the following two wiring diagrams:**

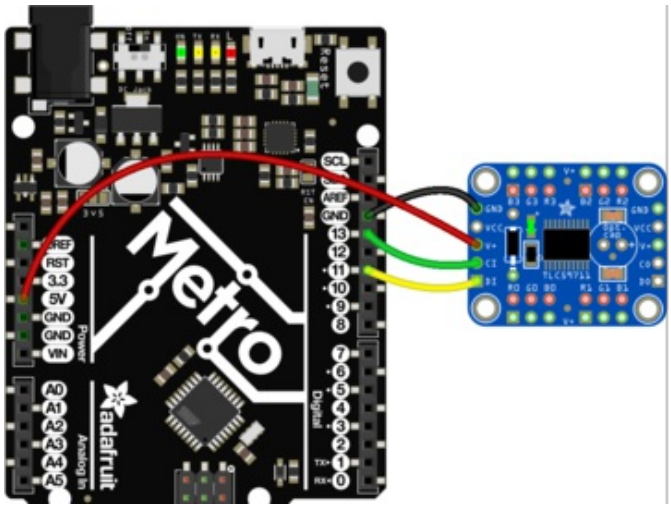


Connect V+ and VCC to 5VDC (this is the best and easiest option)

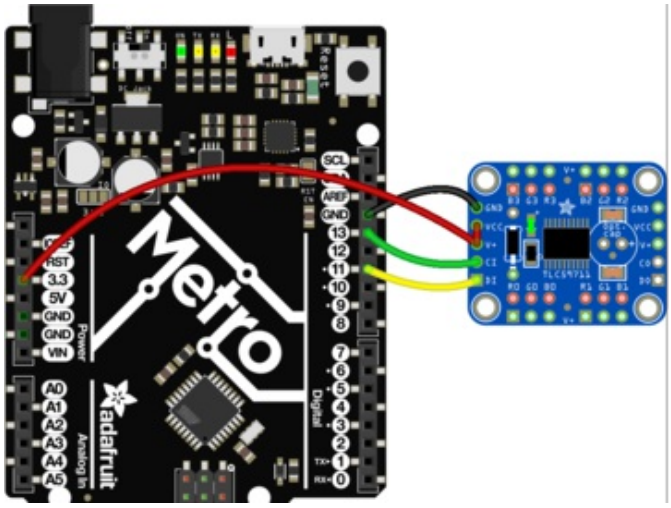


Connect V+ to 4-17V and keep VCC disconnected. Then use 10K resistors or a level shifter between the clock and data wires going into the *first* of the TLC59711

If you are using a 3.3V logic Arduino, you have two wiring options:



Keep **VCC** disconnected, and connect **V+** to **4-17V**



Connect **VCC** and **V+** together to 3.3V (if a 4V+ power source is not available)

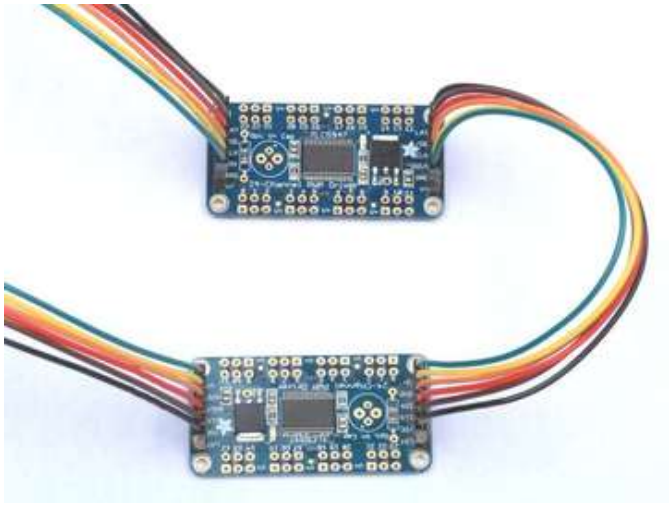
Other than that, connect to the Arduino as follows:

- **DI -> Digital 11**
- **CI -> Digital 13**
- **GND -> GND**

The DI/CI pins can be changed to any other pins later

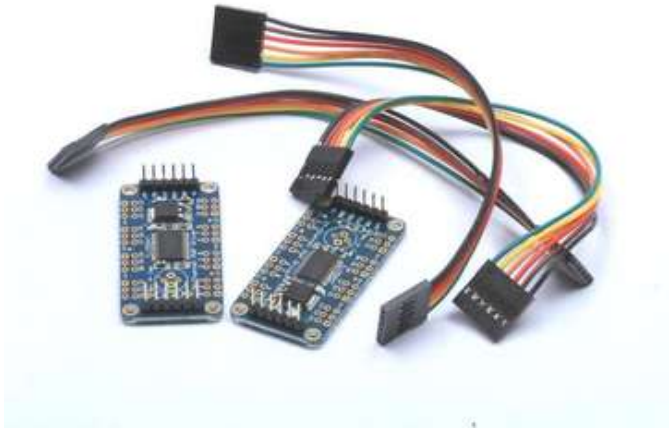
If you need V+ voltages higher than 12v, you will need to use a separate power supply. If you use a separate supply, be sure to connect the ground wire to the Arduino ground.

Chaining Boards

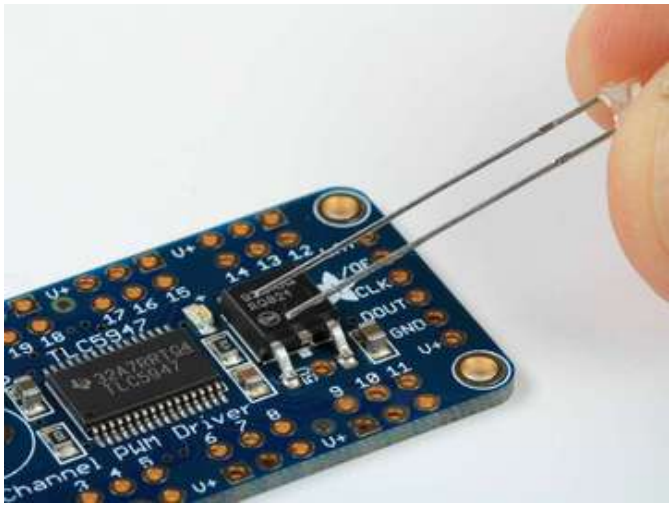


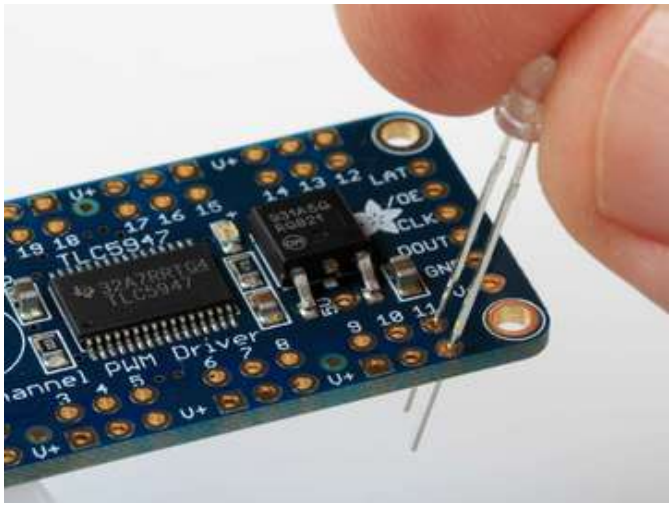
Multiple boards can be chained to control hundreds of LEDs. Using an Arduino, you will run out of memory long before you exceed the chaining capacity of these boards!

Header connections at both ends of the board make chaining simple. Our 6-Conductor 0.1" Socket-Socket Cable (http://adafru.it/206) is perfect for linking them together.



The next two pages will show you how to connect some LEDs and test them out with the library example sketch:





Connecting Monochrome LEDs

Note that one leg of the led is longer than the other. This is the Anode and should be connected to the V+ hole in the board.

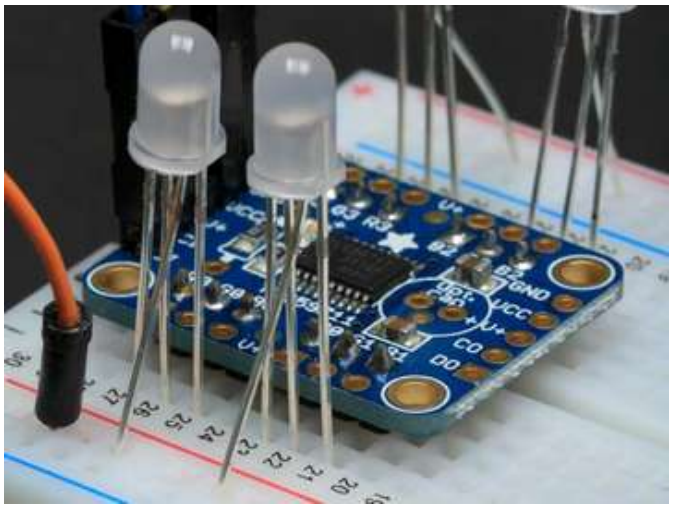
Off-Board Mounting

For more flexibility in positioning LEDs, you will probably want to connect your LEDs with wire instead of soldering them directly to your breakout board.

Our Pig-Tail Cables (http://adafru.it/1003) make remote mounting of individual LEDs very simple.

You can also install headers onto the PCBs, then plug the pig-tail socket there, and solder the LED legs to the wire ends - whatever makes sense for your project

Tip: *One strand of the pig-tail is marked in gray. Solder this to the V+ side to mark the polarity.*

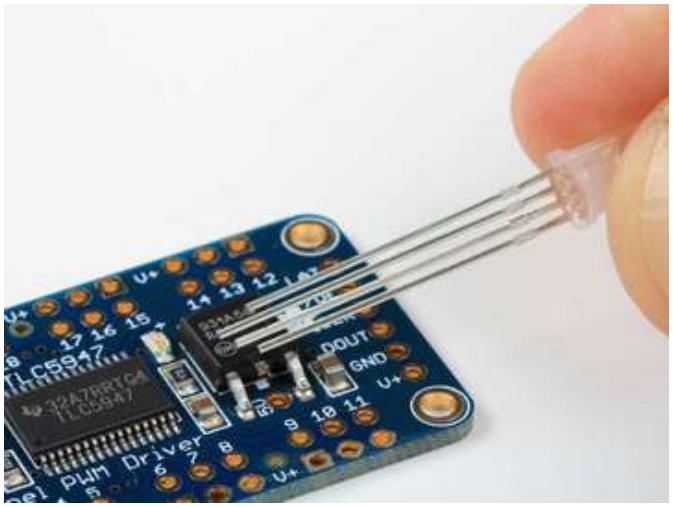


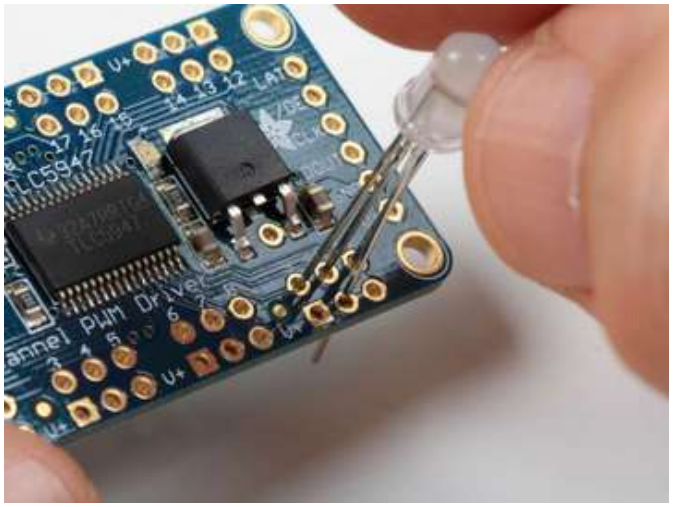
Connecting RGB LEDs

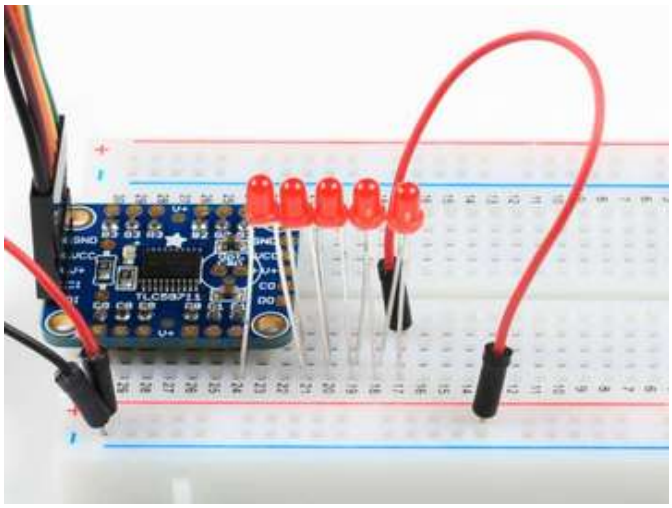
Output channels are conveniently arranged in groups of 3 to simplify connecting RGB leds.

The common anode (longest lead) should be soldered to one of the V+ holes (any one will do)

When working with a breadboard, connect V+ to the bus on the edge of the breadboard and plug the anode into the bus.







Multiple LEDs in Series

The wide voltage range and constant current drive makes it simple to drive multiple LEDs in series from any channel. Just be sure to choose a supply voltage is higher than the sum of the Vfs of all the leds in series.

For example: To drive 5 blue LEDs in series, you would need at least **3.2v** * **5** = **16v**. This voltage is within the range of either the TLC59711 or the TLC5947.

Powering your LEDs

For running as many as 8 channels of RGB or 24 channels of single LEDs, you can usually get by borrowing power from the Arduino as shown. If you plan on running more LEDs with these boards, you should start thinking about how to power them.

Constant Current

Both the TLC59711 and TLC5947 are constant current drivers. This is just what LEDs need. Since they are constant current, you have some flexibility with the supply voltage and you don't don't need to add current limiting resistors. The TLC59xxx drivers will adjust automatically to power supply fluctuations. Your LEDs won't flicker and you don't have to worry about burning them out. We have configured these breakouts to set the current level at 15mA per channel. This is a safe level for virtually all leds you might want to connect to them.

To operate at different currents, it is possible to replace the on-board reference resistor with a through-hole resistor. These drivers are capable of driving up to 60mA (TLC59711) or 30mA (TLS5947) per channel. The graphs in the data sheets show the relationship between resistance and output current.

TLC59711 Data Sheet
TLC5947 Data Sheet

Choosing a Supply Voltage

Since these are constant current drivers, the voltage selection is not so critical. It just needs to be slightly higher than the forward votage (Vf)of your LEDs.

Supply Voltage Range:

- **TLC5947 - 5v to 30v**
- **TLC59711 - 5v to 17v**

Typical LED Vf by Color:

- **Red 2.1v**
- **Yellow 2.2v**
- **Green 3.2v**
- **Blue 3.2v**
- **White 3.2v**

Connecting an External Supply

If you do decide you need an external supply for your LEDs:

- First remove the connection between V+ and VIN from the Arduino.
- **Be sure to keep the connection between GND and the Arduino GND.**
- Next connect the negative terminal of your supply to GND on the breakout.
- Finally, connect the positive terminal of your supply to V+ on the breakout.

Programming - Library Reference

Install The Library

To use these boards, first download and install the library using one of the buttons below. If you are new to Arduino Libraries, check this guide for instructions on how to install them:

http://learn.adafruit.com/adafruit-all-about-arduino-libraries-install-use

Adafruit TLC59711 Library Download

https://adafru.it/cZ2

Adafruit TLC5947 Library Download

https://adafru.it/cZ3

Run the Example Code

Wire your breakout and connect some LEDs as shown on the previous pages. Load the example code for your breakout board:

File->Examples->Adafruit_59711->tlc59711test
or:
File->Examples->Adafruit_5947->tlc5947test

And run. If you are using RGB LEDs, the code will cycle through some colors to demonstrate the capabilities of the board.

TLC5947 Library Reference:

Adafruit_TLC5947(uint8_t n, uint8_t c, uint8_t d, uint8_t l)
Call the constructor to create an instance of the TLC5947 PWM breakout driver.

- n = Number of Drivers (>1 if the drivers are chained)
- c = Clock pin
- d = Data pin
- l = Latch pin

boolean begin(void);
Call begin just once in your setup() function to initialize the devices.

void setPWM(uint8_t chan, uint16_t pwm);
Call this to set the PWM level for a channel.

- chan = Channel
- pwm = PWM level (0 = minimum, 4095 = maximum)

void setLED(uint8_t lednum, uint16_t r, uint16_t g, uint16_t b);
Call this to set the RGB value for a group of 3 channels

- lednum = LED number (channel number of the "red" pin divided by 3)
- r = red level

- g = green level
- b = blue level

void write(void);
Call this after every change to write the new PWM levels to the device.

TLC59711 Library Reference:

Adafruit_TLC59711(uint8_t n, uint8_t c, uint8_t d);
Call the constructor to create an instance of the TLC59711 PWM breakout driver.

- n = Number of Drivers (>1 if the drivers are chained)
- c = Clock Pin
- D = Data Pin

Adafruit_TLC59711(uint8_t n);
Alternate consturctor for hardware SPI. Assumes hardware SPI pins for MOSI and SCK.

boolean begin(void);
Call begin just once in your setup() function to initialize the devices.

void setPWM(uint8_t chan, uint16_t pwm);
Call this to set the PWM level for a channel.

- chan = Channel
- pwm = PWM level (0 = minimum, 65535= maximum)

void setLED(uint8_t lednum, uint16_t r, uint16_t g, uint16_t b);
Call this to set the RGB value for a group of 3 channels

- lednum = LED number (channel number of the "red" pin divided by 3)
- r = red level
- g = green level
- b = blue level

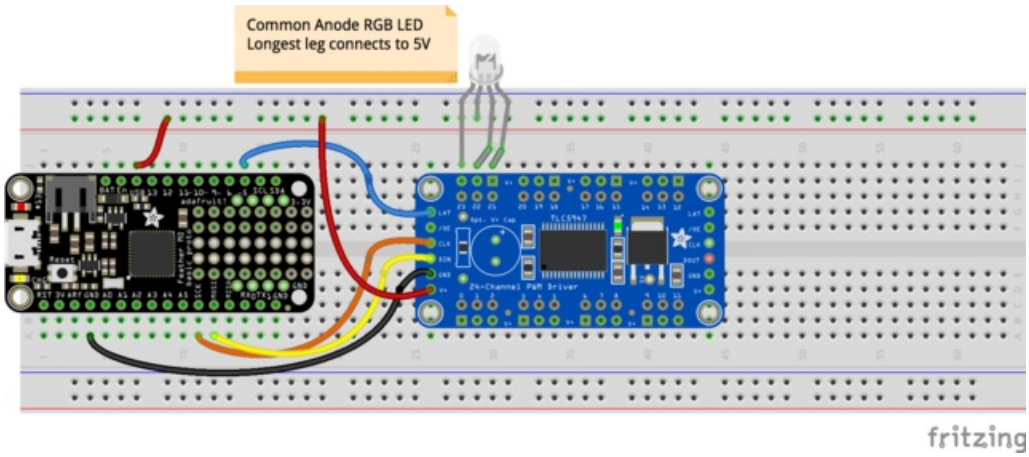
void write(void);
Call this after every change to write the new PWM levels to the device.

CircuitPython

It's easy to use the TLC5947 and TLC59711 with CircuitPython and the Adafruit CircuitPython TLC5947 and Adafruit CircuitPython TLC59711 modules. These modules allows you to easily write Python code that controls the PWM outputs of these boards.

Wiring

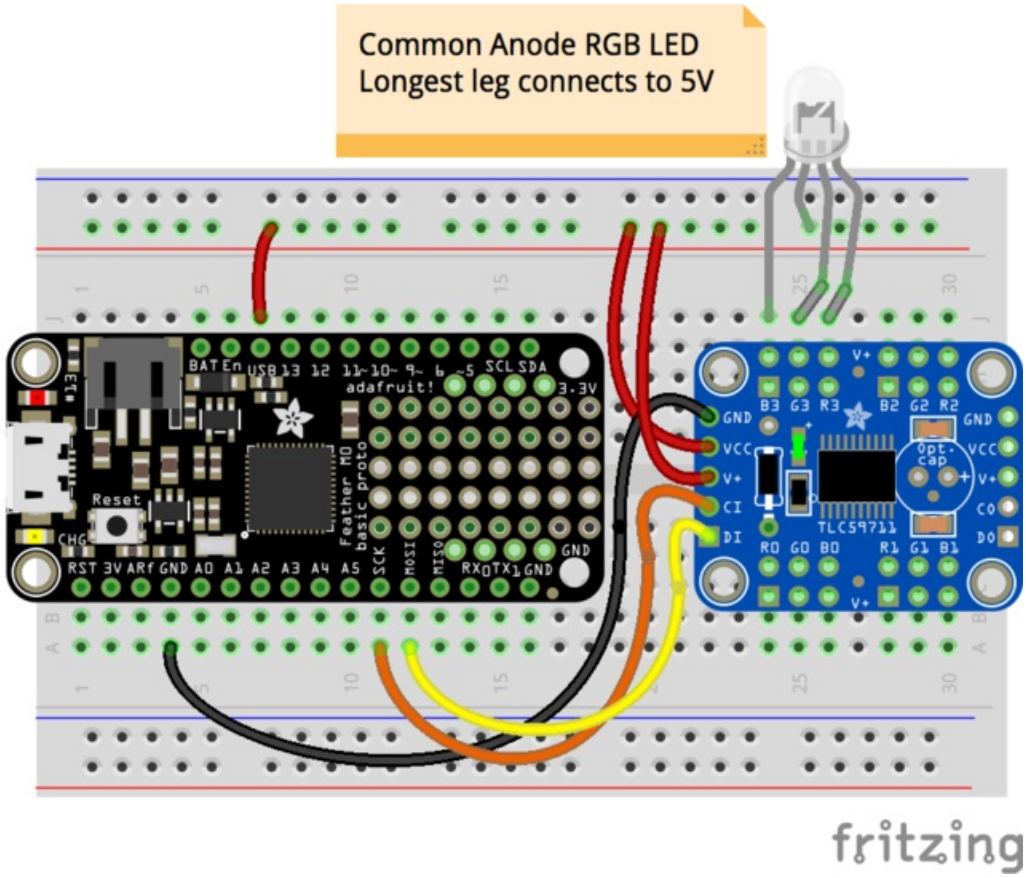
First wire up your board exactly as shown on the previous pages for an Arduino. For example here's how to wire a TLC5947 to a Feather M0 using a standard hardware SPI connection:



- **Board GND/ground** to **TLC5947 GND/ground**.
- **Board VUSB/5V** to **TLC5947 V+** and **RGB LED anode** (longest leg).
- **Board SCK** to **TLC5947 CLK/clock input**.
- **Board MOSI** to **TLC5947 DIN/data input**.
- **Board D5** to **TLC5947 LAT/latch input** (or use any other free digital I/O).
- **RGB LED red leg** to **TLC5947 output 21** (or any other output).
- **RGB LED green leg** to **TLC5947 output 22** (or any other output).
- **RGB LED blue leg** to **TLC5947 output 23** (or any other output).

You might need to check your RGB LED's datasheet to find which legs are the red, green, blue inputs (or just experiment with turning them on off to find out). The longest leg will always be the anode, or positive voltage input. Be sure you're using a **common anode** and **not common cathode** RGB LED!

If you're using a TLC59711 here's how to wire it to a Feather M0 again using a standard SPI connection:



- **Board GND/ground** to **TLC59711 GND/ground**.
- **Board VUSB/5V** to **TLC59711 V+, TLC59711 VCC,** and **RGB LED anode** (longest leg).
- **Board SCK** to **TLC59711 CI/clock input**.
- **Board MOSI** to **TLC59711 DI/data input**.
- **RGB LED red leg** to **TLC5947 output R3** (or any other output).
- **RGB LED green leg** to **TLC5947 output G3** (or any other output).
- **RGB LED blue leg** to **TLC5947 output B3** (or any other output).

Again you might need to check your RGB LED's datasheet to find which legs are the red, green, blue inputs (or just experiment with turning them on off to find out). The longest leg will always be the anode, or positive voltage input. Be sure you're using a **common anode** and **not common cathode** RGB LED!

> If your project needs to be portable, you may want to connect V+ and VCC to VBAT (battery power) or 3.3V (less power available but will work on both USB or battery power)

Module Install

Next you'll need to install either the Adafruit CircuitPython TLC5947 or Adafruit CircuitPython TLC59711 module on your CircuitPython board, depending on which board you're using.

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. Our introduction 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, you'll need to manually install the necessary libraries from the bundle, either:

- **adafruit_tlc5947.mpy**

Or:

- **adafruit_tlc59711.mpy**

You can also download the **adafruit_tlc5947.mpy** from its releases page on Github, or**adafruit_tlc59711.mpy** from its releases page on Github.

Before continuing make sure your board's lib folder or root filesystem has the **adafruit_tlc5947.mpy** or **adafruit_tlc59711.mpy** files copied over.

Next connect to the board's serial REPL so you are at the CircuitPython >>> prompt.

TLC5947 Usage

To demonstrate the usage of the TLC5947 we'll initialize it and control PWM outputs using the Python REPL. First run the following code to initialize the SPI bus and other connections to the chip:

```
import board
import busio
import digitalio
import adafruit_tlc5947
spi = busio.SPI(clock=board.SCK, MOSI=board.MOSI)
latch = digitalio.DigitalInOut(board.D5)
tlc5947 = adafruit_tlc5947.TLC5947(spi, latch)
```

Be sure to set **latch** to the right digital I/O pin connected to your TLC5947's latch input. The wiring and code above use pin **D5**.

Now you're ready to control any of the 24 PWM outputs from the board. There are two ways to control these outputs, the first is with direct access to each channels 12-bit PWM duty cycle. You just index into the TLC5947 object like an array and set or get the duty cycle value, for example to set output 21 to a 100% duty cycle (which should turn the LED on to full red intensity if wired as shown):

```
tlc5947[21] = 4095
```

Notice the LED turns bright red! You can drop it down to half red brightness by setting a smaller duty cycle. Any value from 0 (off/no intensity) to 4095 (maximum/full intensity) can be set. Try a value in-between like 2048:

```
tlc5947[21] = 2048
```

You can change the other colors of the LED too, like to set full green (output 22) and quarter intensity blue (output 23):

```
tlc5947[22] = 4095
tlc5947[23] = 1024
```

The other way to control the TLC5947 outputs is with a syntax similar to CircuitPython's built-in PWMOut object. This is useful if you have code meant to work with PWMOut objects, like a motor controller or other higher-level class. You can pass in a special TLC5947 PWMOut object and it will be controlled just like a regular CircuitPython PWMOut.

Create a PWMOut object by calling the **create_pwm_out** function and pass it the number of the output. For example to make a PWM out for the red channel of the LED (output 21) you would run:

```
red = tlc5947.create_pwm_out(21)
```

Now you can change the **duty_cycle** property of the red object to control the duty cycle of that channel on the TLC5947.

One thing to note is that the duty cycle for this PWMOut expects a 16-bit value just like the regular CircuitPython PWMOut! This means you instead need to use a range of values from 0 (off/no intensity) to 65535 (max/full intensity). Also there's a small chance you'll get little math errors because the TLC5947 only supports a 12-bit duty cycle so be aware if you need the most accurate PWM output you might want to use the direct channel access shown above.

To set the red channel to maximum intensity run:

```
red.duty_cycle = 65535
```

Or set any value from 0 - 65535 to change to an in-between duty cycle!

You might notice the PWMOut object also has a **frequency** property. Unlike CircuitPython's built-in PWMOut this property actually does nothing and cannot be changed. The TLC5947 has a fixed PWM frequency in the high megahertz range which is good for driving LEDs. If you need control of PWM frequency you might need to use a native PWM output from your development board!

That's all there is to using the TLC5947 with CircuitPython! Here's a complete example of initializing the chip and changing the PWM outputs in a few different ways as mentioned above. Save this as **main.py** on your board to run it:

```
# Simple demo of controlling the TLC5947 12-bit 24-channel PWM controller.
# Will update channel values to different PWM duty cycles.
# Author: Tony DiCola
import board
import busio
import digitalio

import adafruit_tlc5947


# Define pins connected to the TLC5947
SCK = board.SCK
MOSI = board.MOSI
LATCH = digitalio.DigitalInOut(board.D5)

# Initialize SPI bus.
spi = busio.SPI(clock=SCK, MOSI=MOSI)

# Initialize TLC5947
tlc5947 = adafruit_tlc5947.TLC5947(spi, LATCH)
# You can optionally disable auto_write which allows you to control when
# channel state is written to the chip.  Normally auto_write is true and
# will automatically write out changes as soon as they happen to a channel, but
# if you need more control or atomic updates of multiple channels then disable
# and manually call write as shown below.
#tlc5947 = adafruit_tlc5947.TLC5947(spi, LATCH, auto_write=False)

# There are two ways to channel channel PWM values.  The first is by getting
# a PWMOut object that acts like the built-in PWMOut and can be used anywhere
# it is used in your code.  Change the duty_cycle property to a 16-bit value
# (note this is NOT the 12-bit value supported by the chip natively) and the
# PWM channel will be updated.
pwm0 = tlc5947.create_pwm_out(0)

# Set the channel 0 PWM to 50% (32767, or half of the max 65535):
pwm0.duty_cycle = 32767
# Note if auto_write was disabled you need to call write on the parent to
# make sure the value is written (this is not common, if disabling auto_write
# you probably want to use the direct 12-bit raw access instead shown below).
#tlc5947.write()

# The other way to read and write channels is directly with each channel 12-bit
# value and an item accessor syntax.  Index into the TLC5947 with the channel
# number (0-23) and get or set its 12-bit value (0-4095).
# For example set channel 1 to 50% duty cycle.
tlc5947[1] = 2048
# Again be sure to call write if you disabled auto_write.
#tlc5947.write()
```

TLC59711 Usage

To demonstrate the usage of the TLC59711 we'll initialize it and control PWM outputs using the Python REPL. First run the following code to initialize the SPI bus and other connections to the chip:

```
import board
import busio
import digitalio
import adafruit_tlc59711
spi = busio.SPI(clock=board.SCK, MOSI=board.MOSI)
tlc59711 = adafruit_tlc59711.TLC59711(spi)
```

Now you're ready to start controlling the PWM outputs of the TLC59711. There are two ways to control these outputs. The first way is with a format that's very similar to controlling a strip of NeoPixels. For each of the R0, G0, B0, R1, G1, B1, etc. sets of PWM outputs they can be adjusted at once by indexing into the TLC59711 object. For example to set the R3, G3, B3 output to full red intensity run:

```
tlc59711[3] = (65535, 0, 0)
```

Notice you set the outputs using a 3-tuple of 16-bit values. These are the PWM duty cycle values to assign to the R3, G3, and B3 channels respectively. You could set the same outputs to moderate green/blue (half intensity of each) by running:

```
tlc59711[3] = (0, 32767, 32767)
```

The other way to control PWM outputs is with direct access to each output pin. The TLC59711 object has a property for each R0, G0, B0, R1, etc. output and you can set or get its 16-bit PWM duty cycle directly. For example to change R3 to full intensity:

```
tlc59711.r3 = 65535
```

This is handy if you're not controlling RGB LEDs and instead just want to control each channel independently.

Finally the TLC59711 supports a global red, green, and blue channel brightness control. These values adjust all of the associated color outputs, like the red brightness channel control changes the R0, R1, R2, and R3 outputs. Set this to a 7-bit value (0-127) with 0 being low intensity and 127 being maximum intensity. Both the channel brightness and the individual channel PWM duty cycle are used to set the final output of a channel. For example to dim all the red channels by half you can run:

```
tlc59711.red_brightness = 63
```

There's a similar **green_brightness** property that controls G0, G1, G2, G3 and **blue_brightness** property that controls B0, B1, B2, B3.

That's all there is to controlling the TLC59711 with CircuitPython! Below is a complete example of initializing communication with the chip and changing some of the PWM outputs as shown above. Save this as **main.py** on your board to have it run:

```
# Simple demo of the TLC59711 16-bit 12 channel LED PWM driver.
# Shows setting channel values in a few ways.
# Author: Tony DiCola
import board
import busio

import adafruit_tlc59711


# Define SPI bus connected to chip.  You only need the clock and MOSI (output)
# line to use this chip.
spi = busio.SPI(board.SCK, MOSI=board.MOSI)

# Define the TLC59711 instance.
leds = adafruit_tlc59711.TLC59711(spi)
# Optionally you can disable the auto_show behavior that updates the chip
# as soon as any channel value is written.  The default is True/on but you can
# disable and explicitly call show to control when updates happen for better
# animation or atomic RGB LED updates.
#leds = adafruit_tlc59711.TLC59711(spi, auto_show=False)

# There are a couple ways to control the channels of the chip.
# The first is using an interface like a strip of NeoPixels.  Index into the
# class for the channel and set or get its R, G, B tuple value.  Note the
# component values are 16-bit numbers that range from 0-65535 (off to full
# brightness).  Remember there are only 4 channels available too (0 to 3).
# For example set channel 0 (R0, G0, B0) to half brightness:
leds[0] = (32767, 32767, 32767)
# Dont forget to call show if you disabled auto_show above.
#leds.show()

# Or to set channel 1 to full red only (green and blue off).
leds[1] = (65535, 0, 0)

# You can also explicitly control each R0, G0, B0, R1, B1, etc. channel
# by getting and setting its 16-bit value directly with properties.
# For example set channel 2 to full green (i.e. G2 to maximum):
leds.g2 = 65535
# Again don't forget to call show if you disabled auto_show above.
#leds.show()

# The chip also supports global brightness channels to change the red, green,
# blue colors of all 4 channels at once.  These are 7-bit values that range
# from 0-127.  Get and set them with the red_brightness, green_brightness,
# and blue_brightness properties and again be sure to call show if you
# disabled auto_show.
# For example set the red channel to half brightness globally.
leds.red_brightness = 63
# Don't forget to call show if you disabled auto_show above.
#leds.show()
```

Downloads and Links

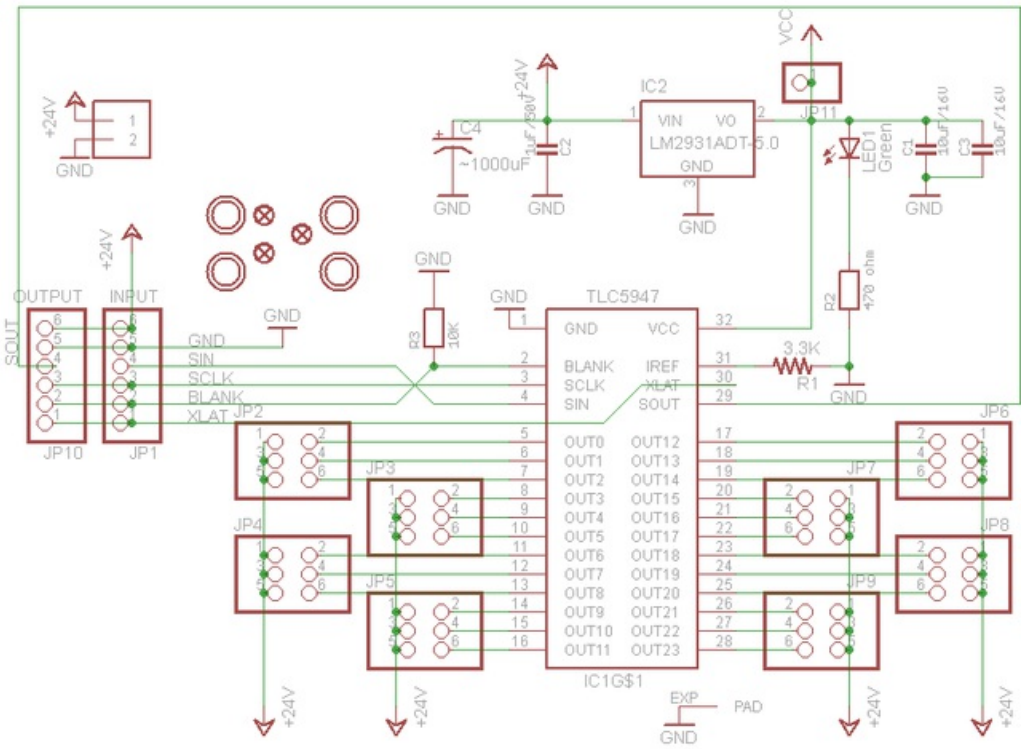
Libraries:

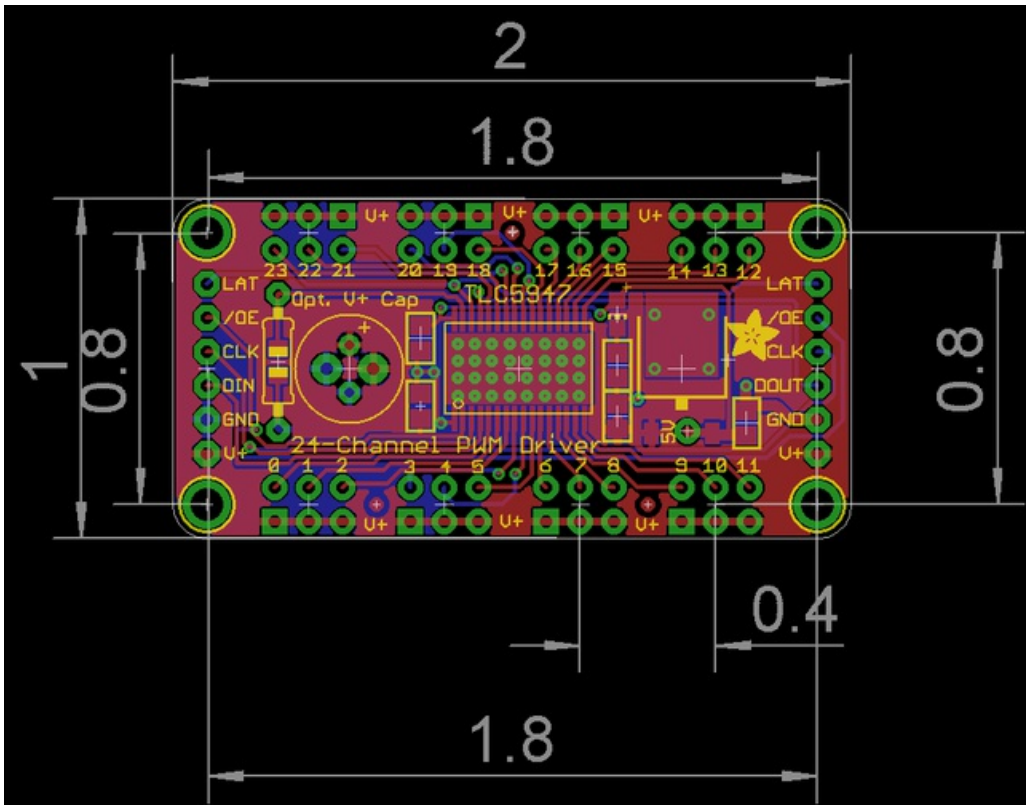
TLC59711 Library (12 channel breakout)
TLC5947 Library (24 channel breakout)
For tips on installing Arduino Libraries, see our guide: All About Arduino Libraries.

Data Sheets & Files

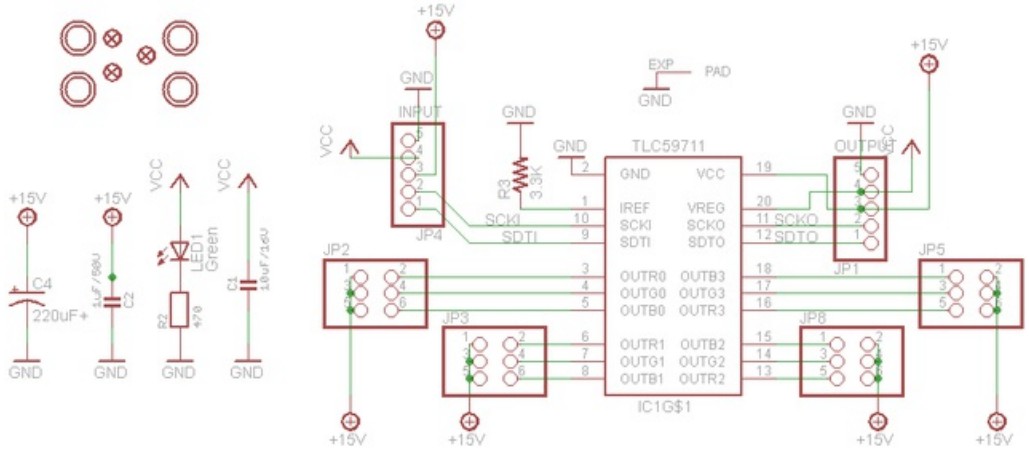
- TLC59711 Data Sheet
- TLC5947 Data Sheet
- Fritzing objects in Adafruit Fritzing Library
- EagleCAD PCB files for TLC5947 on GitHub
- EagleCAD PCB files for TLC59711 on GitHub

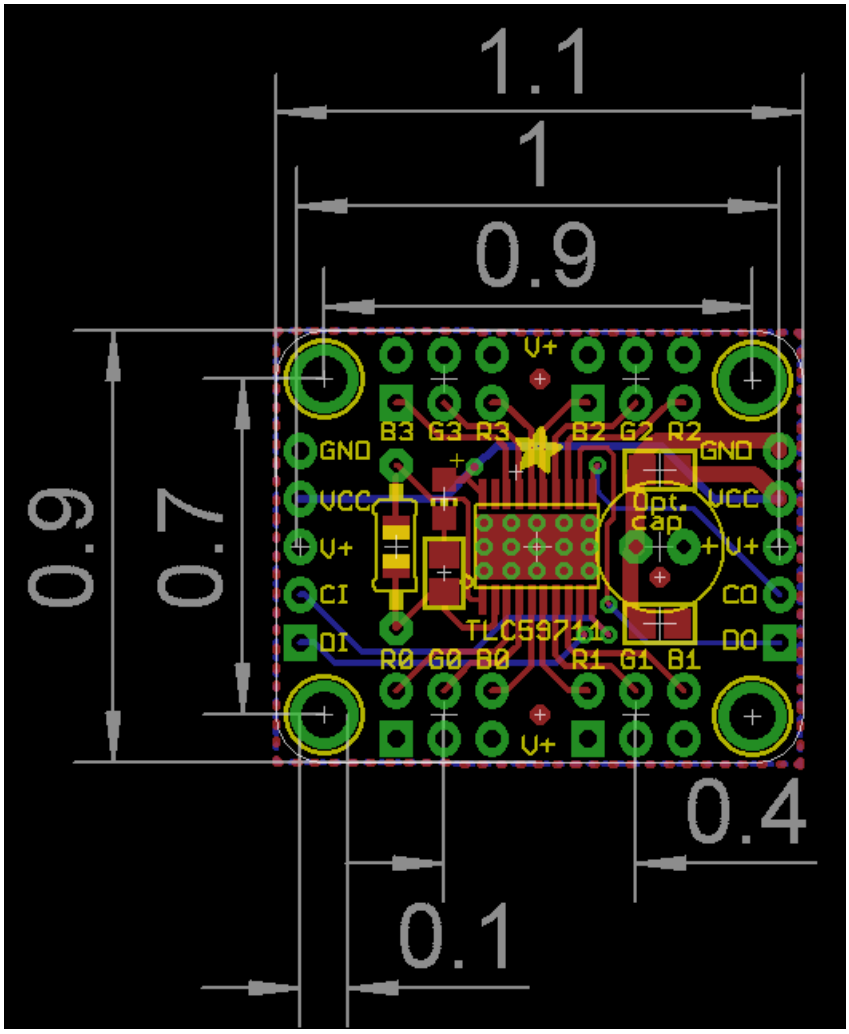
TLC5947 Schematic & Print





TLC59711 Schematic and Print




1.1
1
0.9
0.9
0.7
V+
GND
VCC
V+
CI
DI
B3 G3 R3
B2 G2 R2
GND
VCC
Opt.
Cap
V+
CO
DO
TLC59711
R0 G0 B0
R1 G1 B1
V+
0.4
0.1