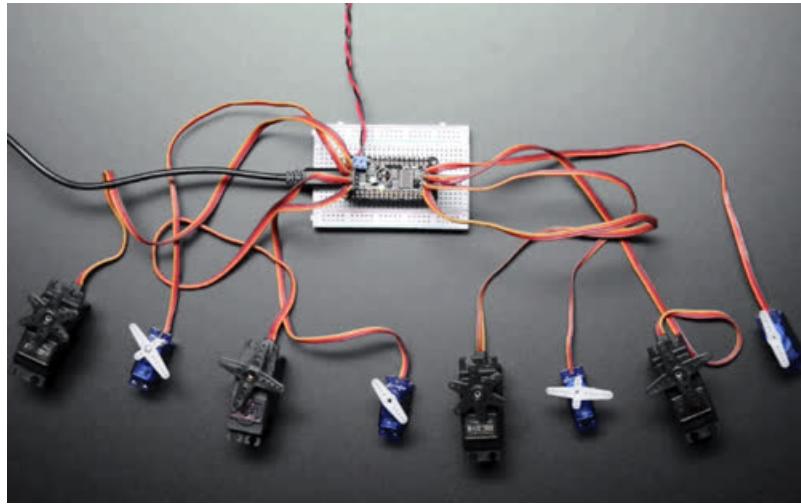




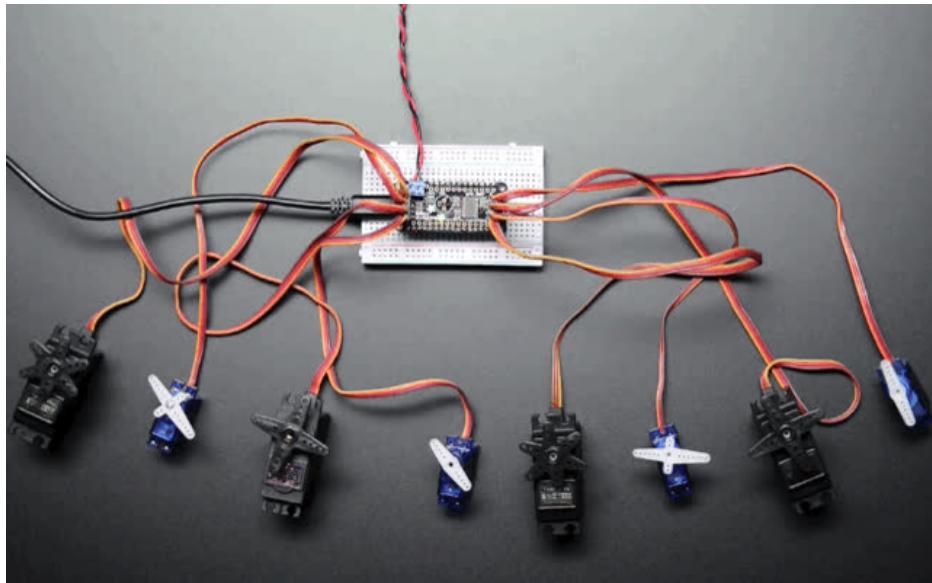
Adafruit 8-Channel PWM or Servo FeatherWing

Created by lady ada

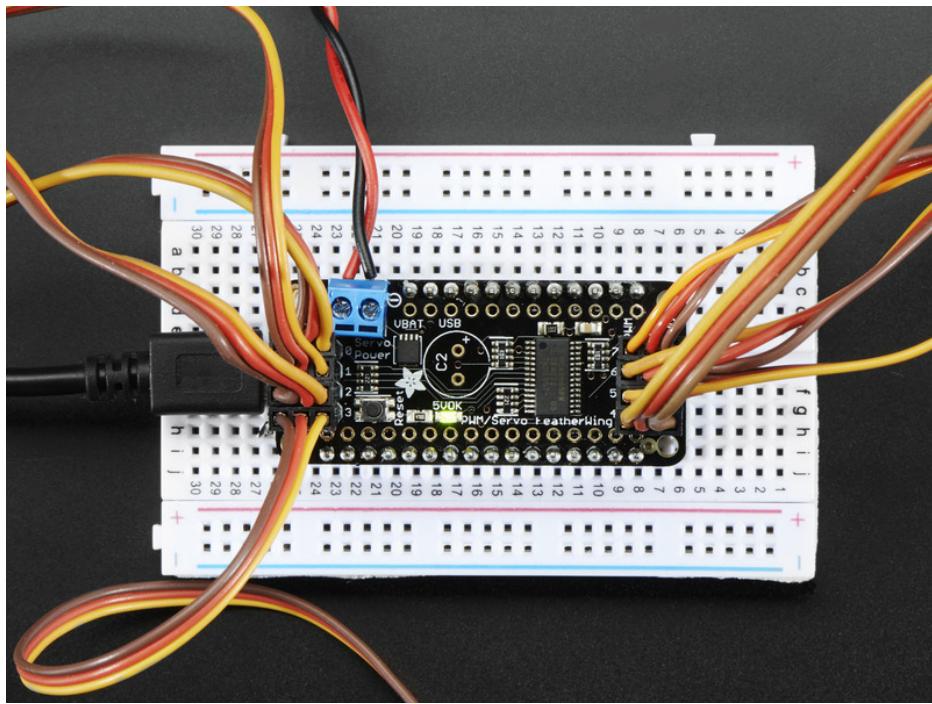


Last updated on 2019-11-18 09:57:14 PM UTC

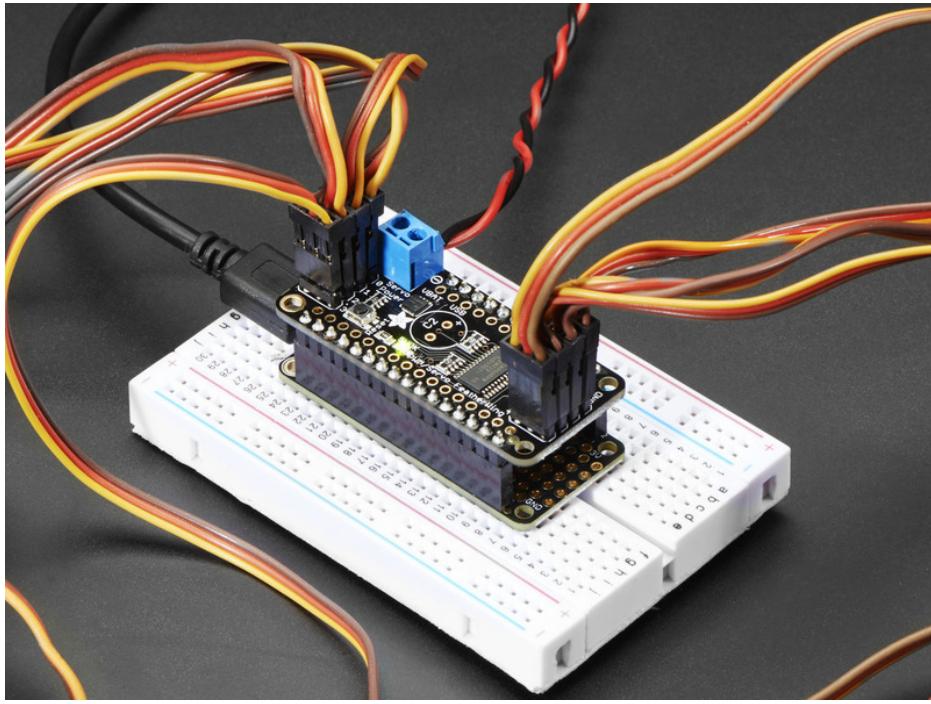
Overview



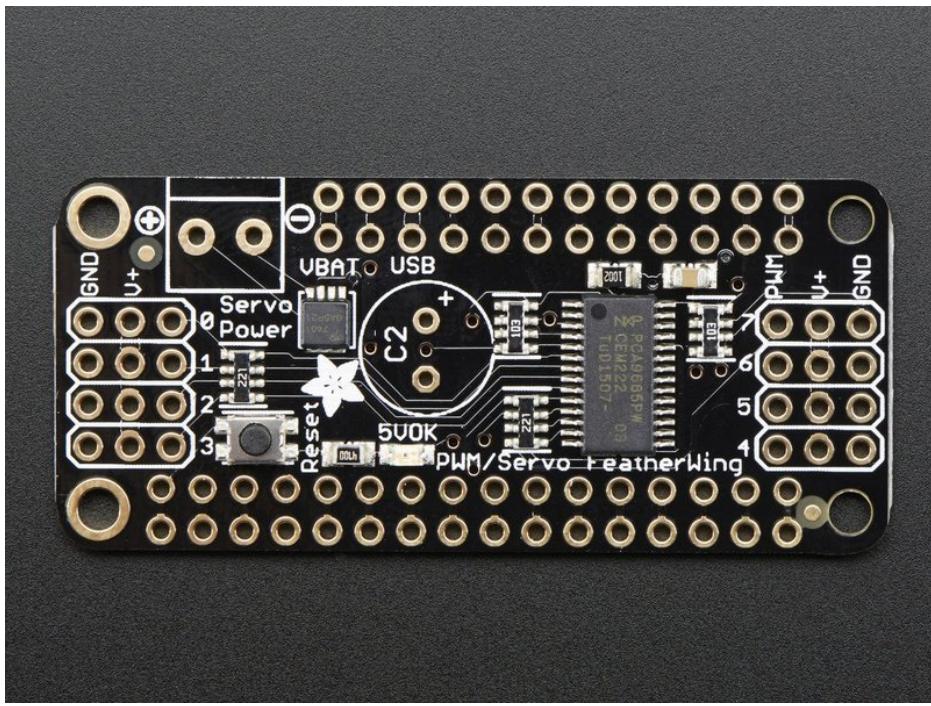
A Feather board without ambition is a Feather board without FeatherWings! This is the **8-Channel PWM or Servo FeatherWing**, you can add 8 x 12-bit PWM outputs to your Feather board. Using our [Feather Stacking Headers](http://adafru.it/2830) (<http://adafru.it/2830>) or [Feather Female Headers](http://adafru.it/2886) (<http://adafru.it/2886>) you can connect a FeatherWing on top or bottom of your Feather board and let the board take flight!



You want to make a cool robot, maybe a hexapod walker, or maybe just a piece of art with a lot of moving parts. Or maybe you want to drive a lot of LEDs with precise PWM output. What now? You could give up OR you could just get our handy PWM and Servo FeatherWing. It's a lot like our popular [PWM/Servo Shield](http://adafru.it/1411) (<http://adafru.it/1411>) but with half the channels & squished into a nice small portable size and works with any of our Feather boards.



Since the FeatherWing only uses the I2C (SDA & SCL pins), it works with any and all Feathers- ATmega32u4, ATSAM M0 or ESP8266-based. You can stack it with any other FeatherWing or with itself (just make sure you have each wing with a unique I2C address) [Check out our range of Feather boards here.](https://adafru.it/I7B) (<https://adafru.it/I7B>)



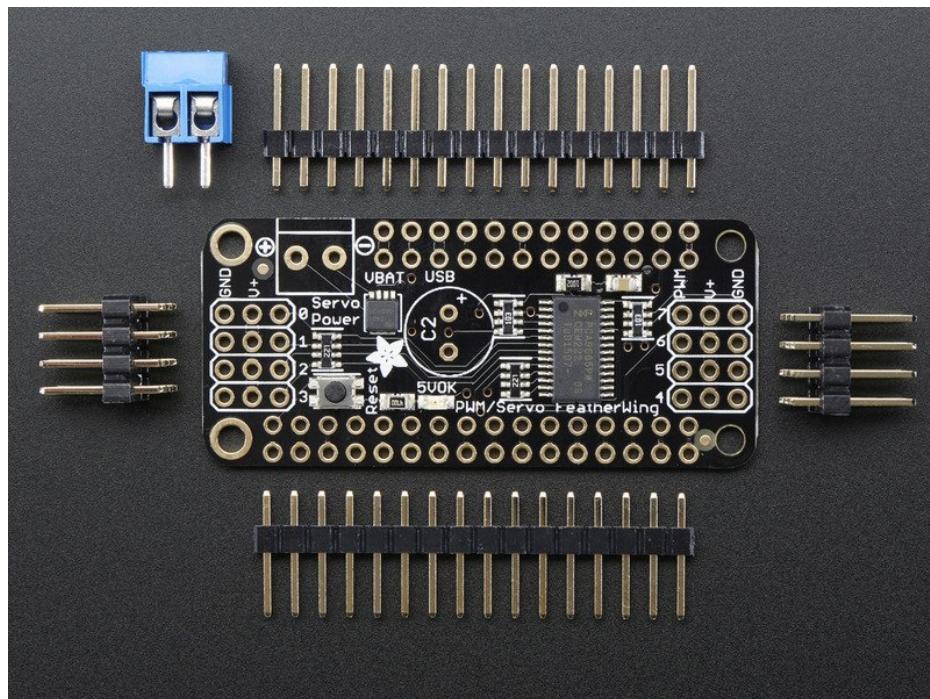
Specs:

- There's an I2C-controlled PWM driver with a built in clock. That means that, unlike the TLC5940 family, you do not need to continuously send it signal tying up your microcontroller, its completely free running!
- It is 5V compliant, which means you can control it from a 3.3V Feather and still safely drive up to 6V outputs (this is good for when you want to control white or blue LEDs with 3.4+ forward voltages)

- 6 address select pins so you can stack up to 62 of these on a single i2c bus, a total of 496 outputs - that's a lot of servos or LEDs
- Adjustable frequency PWM up to about 1.6 KHz
- 12-bit resolution for each output - for servos, that means about 4us resolution at 60Hz update rate
- Configurable push-pull or open-drain output

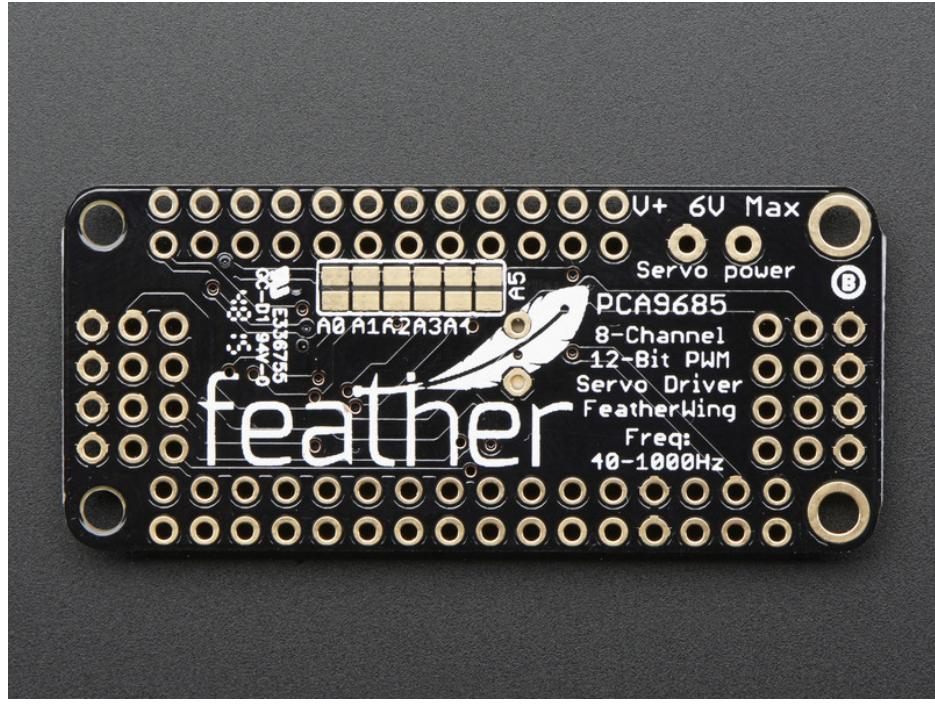
We wrapped up this lovely chip into a FeatherWing with a couple nice extras:

- Terminal block for power input (or you can use the 0.1" breakouts on the side)
- Reverse polarity protection on the terminal block input
- Green power-good LED
- Two groups of 4 outputs on either side, 8 total.
- Stackable design. You'll need to pick up stacking headers and right angle 3x4 headers in order to stack on top of this shield without the servo connections getting in the way.
- A spot to place a big capacitor on the V+ line (in case you need it)
- 220 ohm series resistors on all the output lines to protect them, and to make driving LEDs trivial
- Solder jumpers for the 6 address select pins

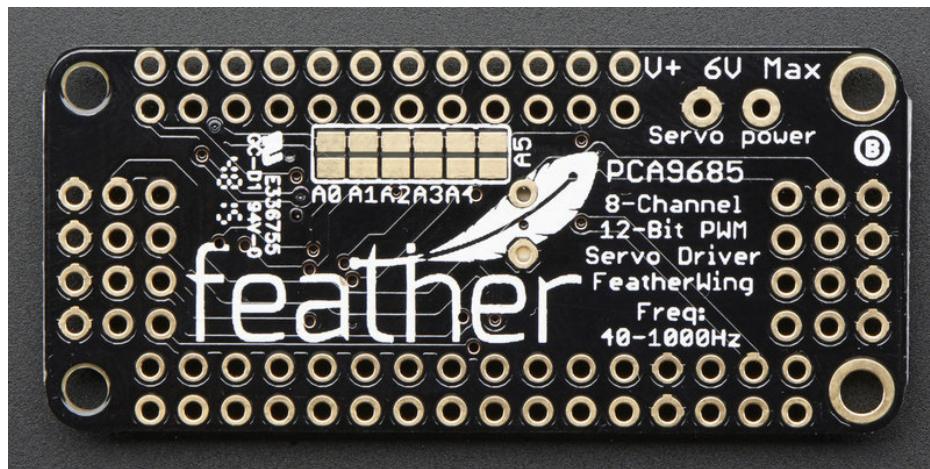
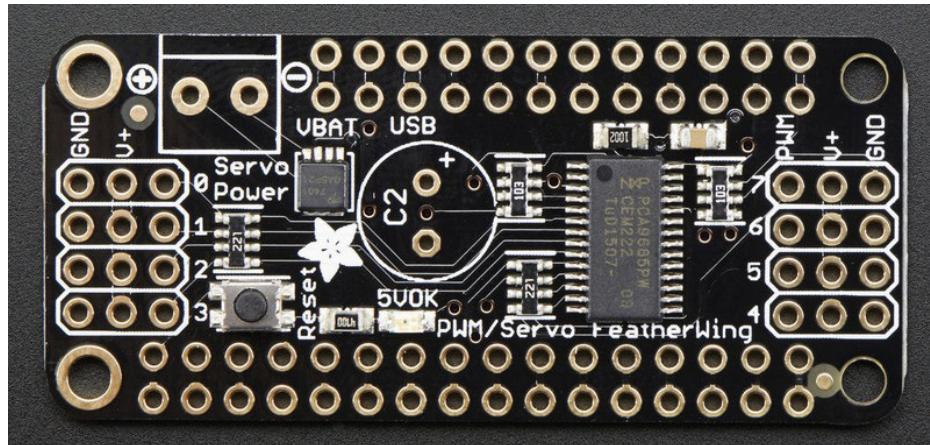


This product comes with a fully tested and assembled wing as well as 2 pieces of 3x4 male straight header (for servo/LED plugs), a 2-pin terminal block (for power) and a stick of 0.1" header so you can plug into a Feather. A little light soldering will be required to assemble and customize the board by attaching the desired headers but it is a 15 minute task that even a beginner can do.

If you want to use right-angle 3x4 headers, we also carry a 4 pack in the shop. (<http://adafru.it/816>) **Servos and Feather not included**, but we have lots of servos in the shop (<https://adafru.it/oke>).

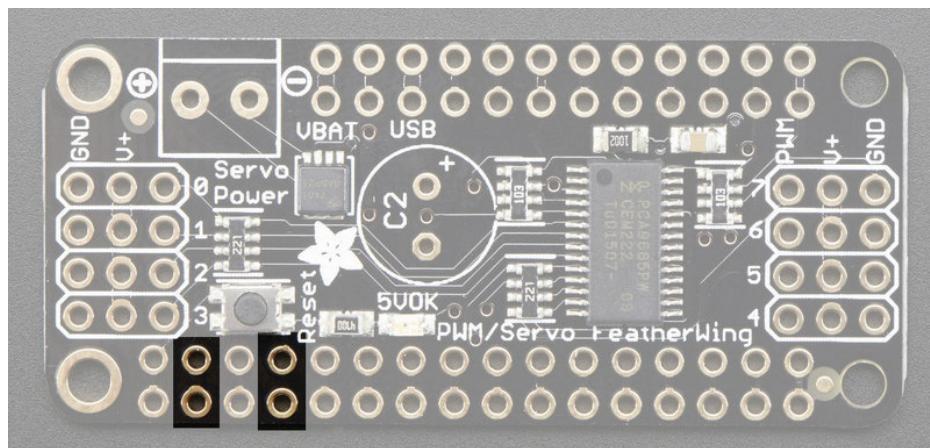


Pinouts



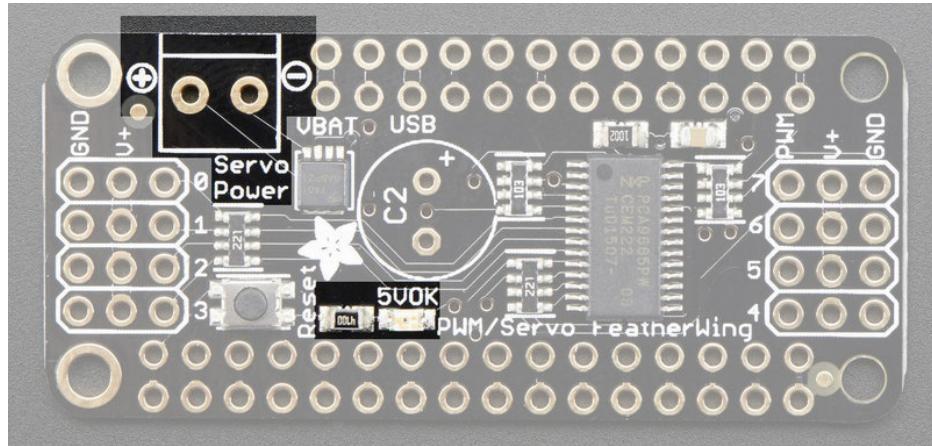
Power Pins

This shield has **two** power supplies. One is logic level power - that is the 3.3V power from the Feather, it is used to power the PWM chip and determines the I2C logic level and the PWM signal logic level.



To power servos you will need to also connect the 5V Servo power supply- this is the power supply for the servos. (If you are lighting up single LEDs you may not need this power supply.) This power supply should be 5 to 6VDC. You can

connect this power through the blue terminal block. There is reverse-polarity protection in case you hook up power backwards.



When the servo power pin is powered, the **5VOK** LED will be lit. If this LED is not lit, the **V+** pins will not have any voltage on them and the servos won't be powered.

Nearly all servos are designed to run on about 5 or 6v. Keep in mind that a lot of servos moving at the same time (particularly large powerful ones) will need a lot of current. Even micro servos will draw several hundred mA when moving. **Some high-torque servos will draw more than 1A each under load.**

Good power choices are:

- [5v 2A switching power supply](http://adafru.it/276) (<http://adafru.it/276>) (up to perhaps 4 servos)
- [5v 10A switching power supply](http://adafru.it/658) (<http://adafru.it/658>) (up to perhaps 16 servos)
- [4xAA Battery Holder](http://adafru.it/830) (<http://adafru.it/830>) - 6v with Alkaline cells. 4.8v with NiMH rechargeable cells, portable!
- 4.8 or 6v Rechargeable RC battery packs from a hobby store.

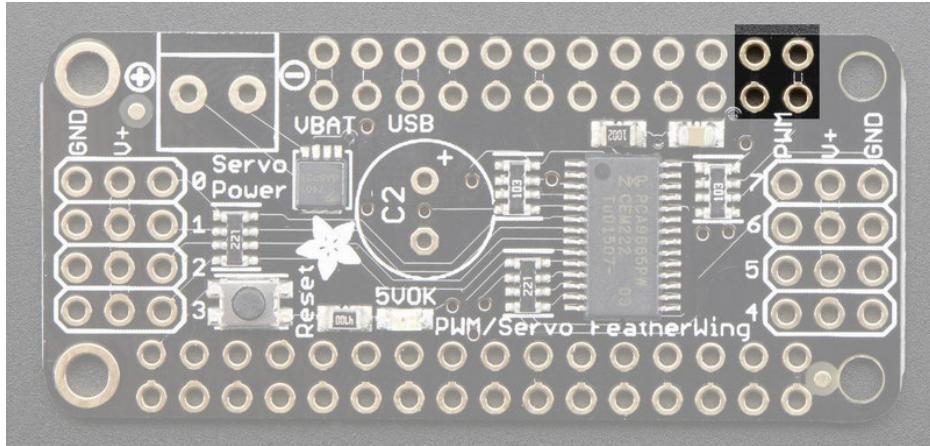


SERVOS CAN USE A LOT OF POWER! It is not a good idea to use the Feather USB pin to power your servos. Electrical noise and 'brownouts' from excess current draw can cause your Feather to act erratically, reset and/or overheat.

You can add an extra large-value electrolytic capacitor to the servo power supply to help stabilize the power supply, see the [Usage](#) page for details

I2C Data Pins

The PWM driver does all of the data transfer over the I2C pins, highlighted above **SDA** and **SCL**. No other pins are required. There are two 10K pullups to 3V on each.

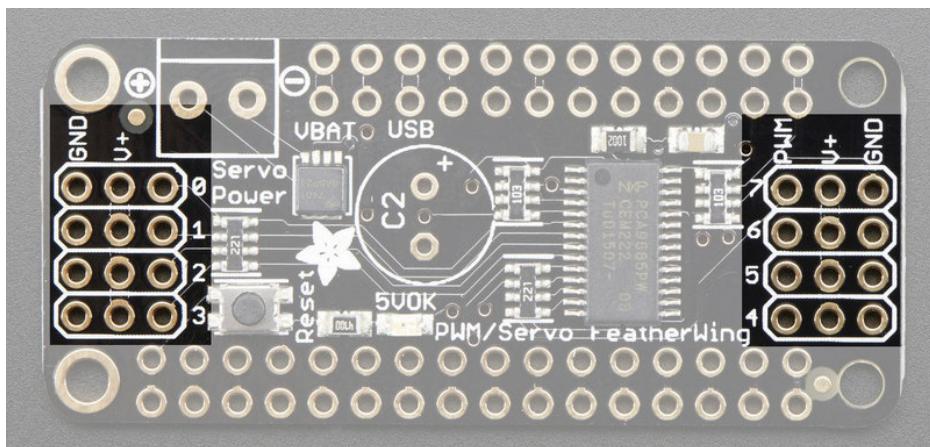


These pins can be shared with other I2C devices.

The default I2C address is **0x40** and can be changed by closing jumpers on the bottom. See the Advanced Usage page for more details

Servo / PWM Pins

OK now we get to the fun part. These are the pins we can use for driving LEDs or Servos. There are 8 outputs, each in a 3-pin "port"

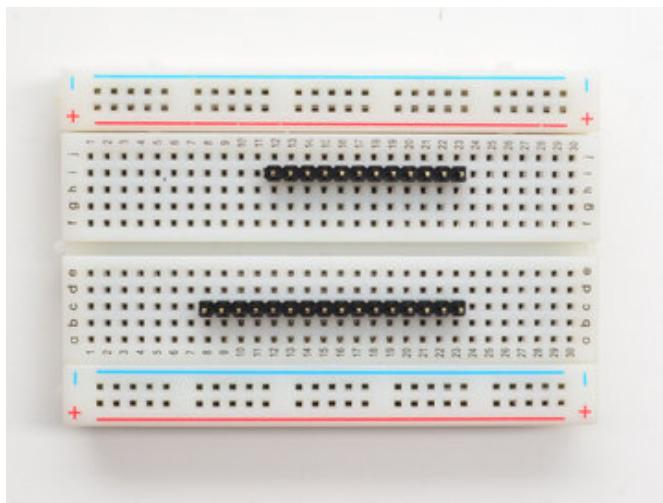
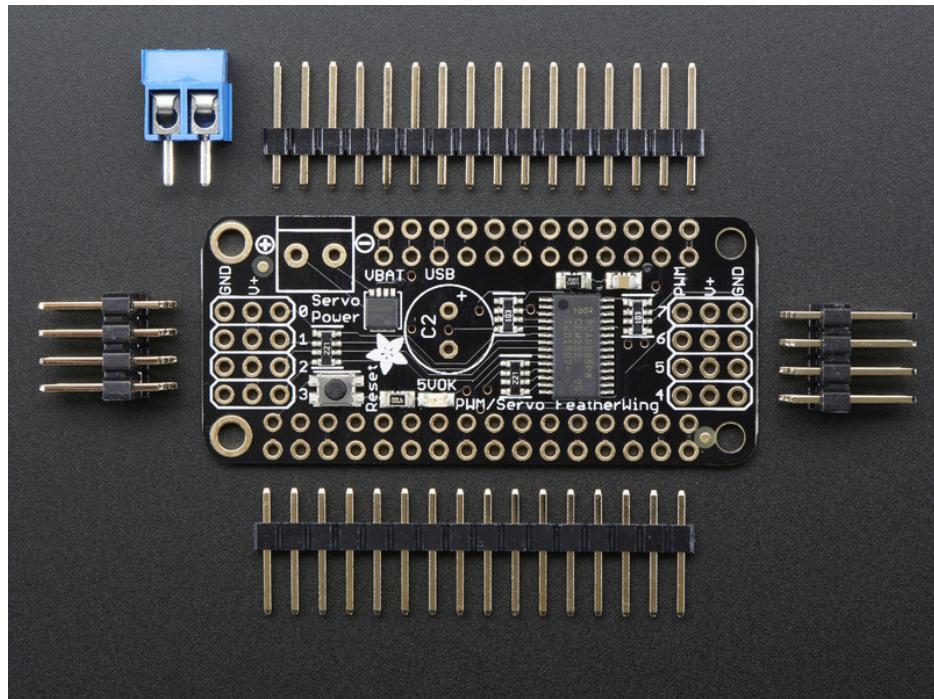


Each port contains three pins:

1. **GND** - power and signal ground
2. **V+** - 5V power from the terminal block for powering servos or LEDs that are common anode or require 5V
3. **Signal** - 3.3V logic signal from the PCA9685 PWM generator

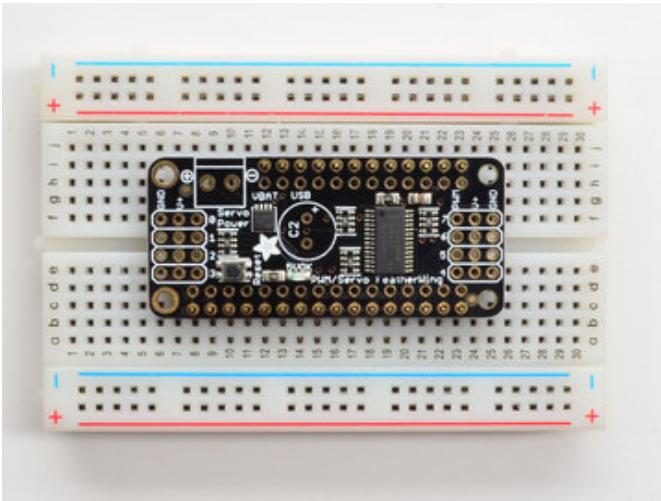
If you're driving LEDs you can probably get away with just using either GND or V+ and the signal. For servos you will need all three pins.

Assembly



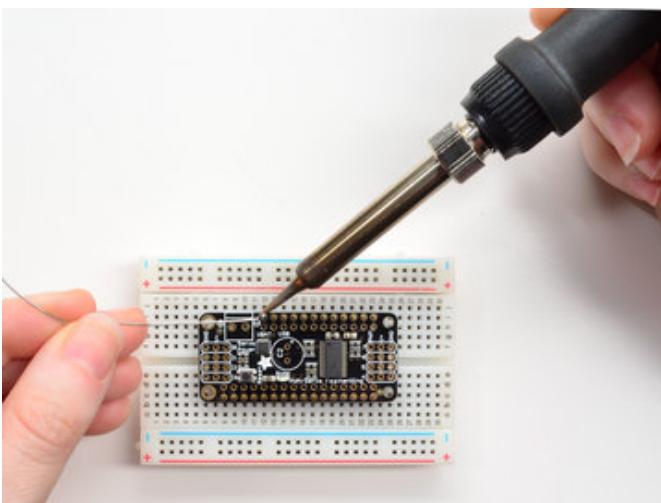
Prepare the header strip:

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 FeatherWing:

Place the featherwing over the pins so that the short pins poke through the two rows of breakout pads

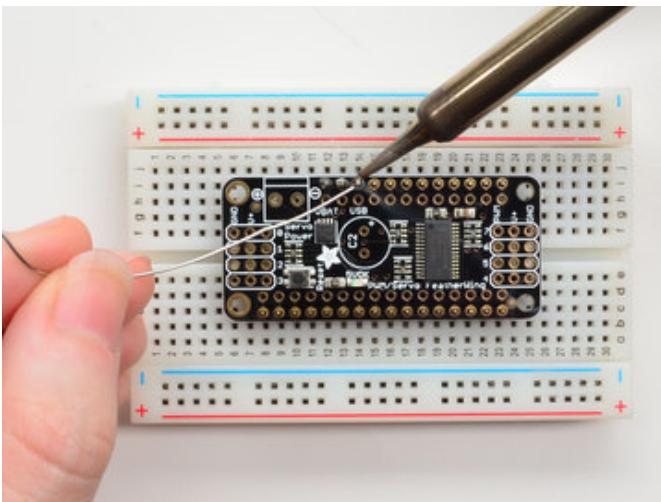


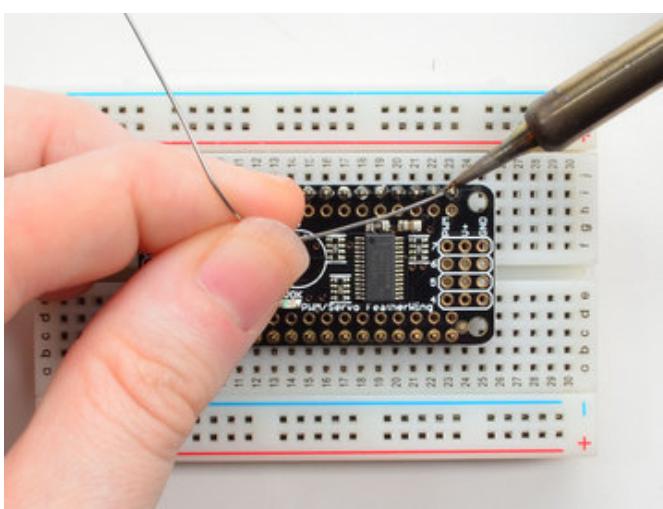
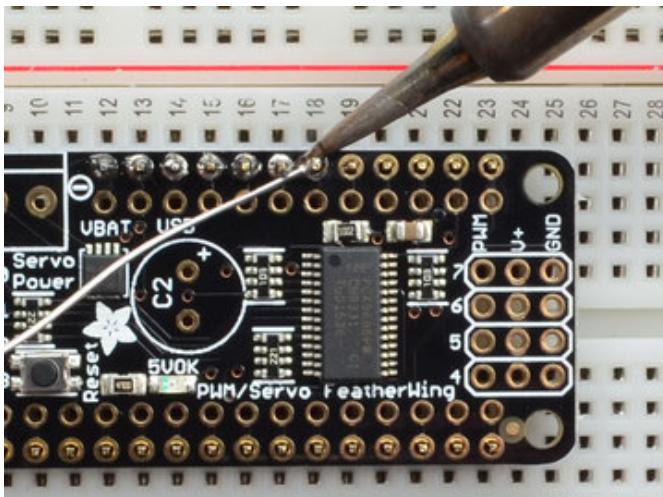
And Solder!

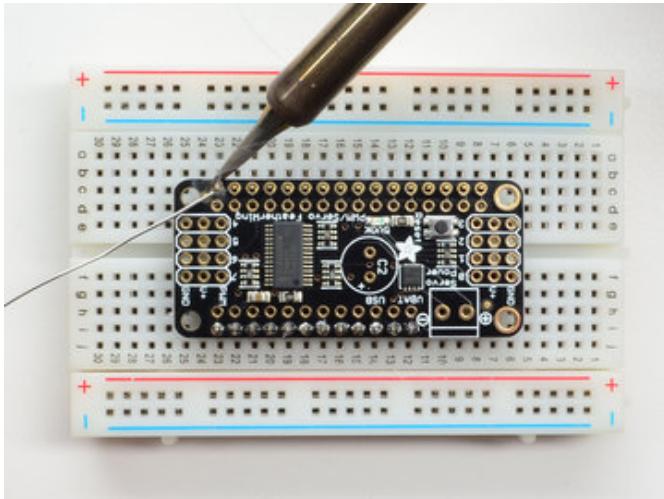
Be sure to solder all pins for reliable electrical contact.

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

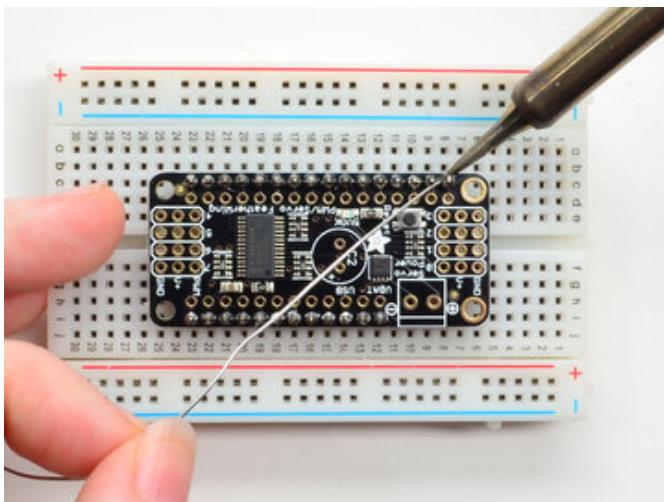
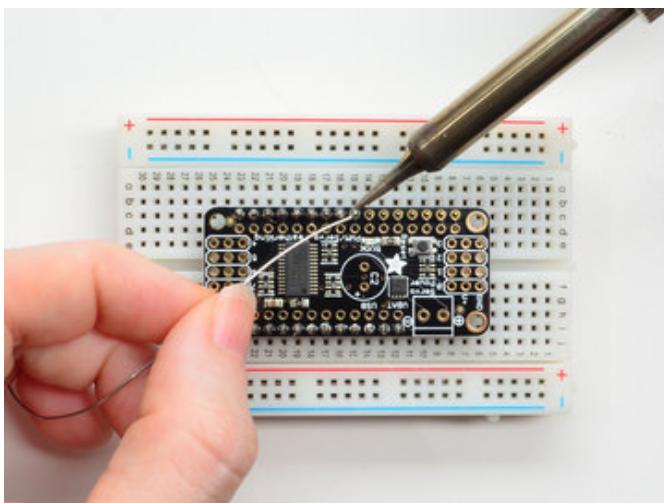
Start by soldering the first row of header

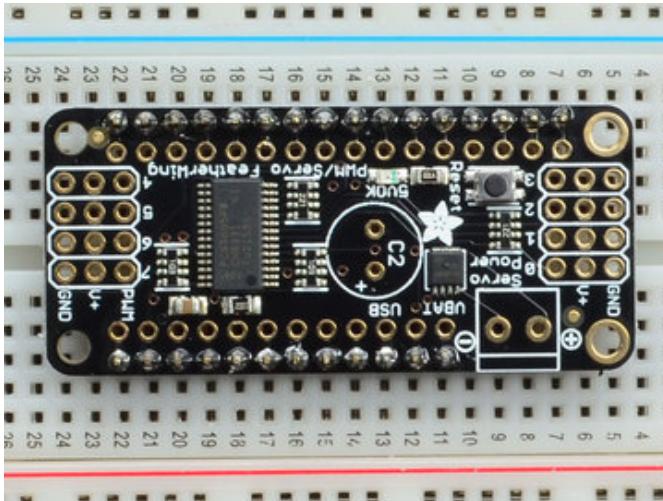






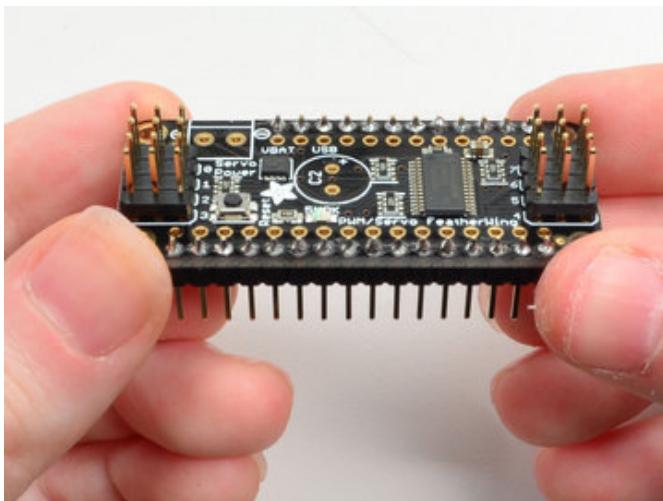
Now flip around and solder the other row completely





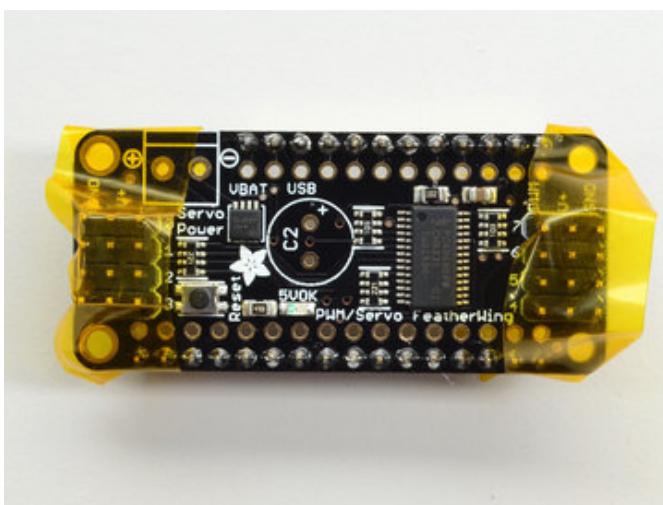
You're done with the two header strips.

Check your solder joints visually and continue onto the next steps



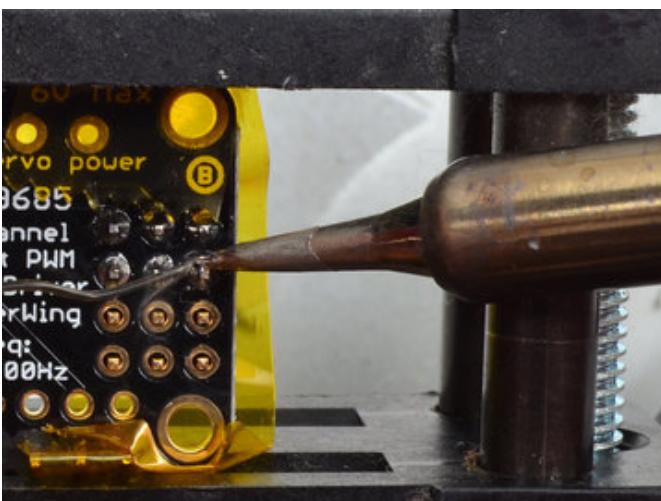
Next we will solder in the larger header blocks used to plug in servos into the FeatherWing. There are two blocks, each are 3x4 headers in size.

The go on either end. Make sure the **long side of the headers** is facing up!



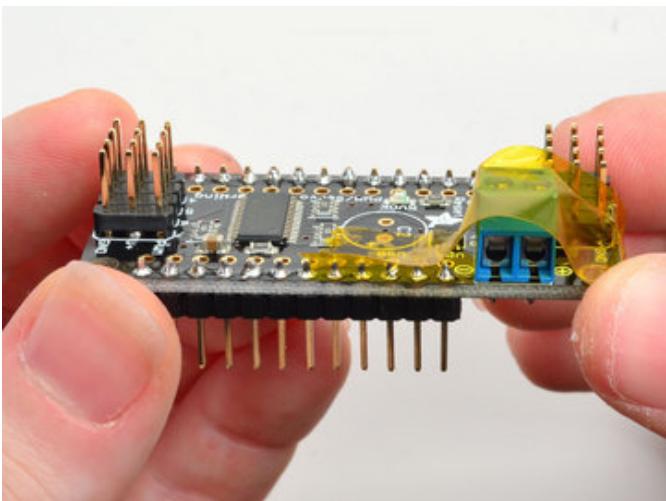
To make it easier to keep these in place, you can use some tape to hold down the two header pieces. Tacky clay also works, whatever you've got handy!

Solder in each block, make sure you get to each of the 12 pins



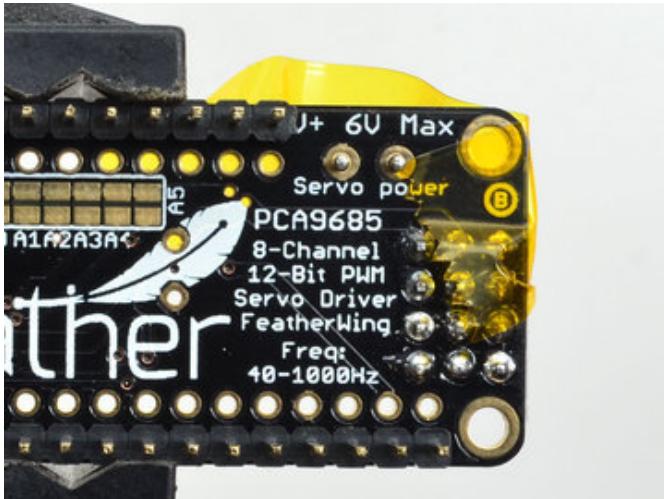


Now that you're done with the header blocks, check your work make sure that each solder joint is done and looks shiny.

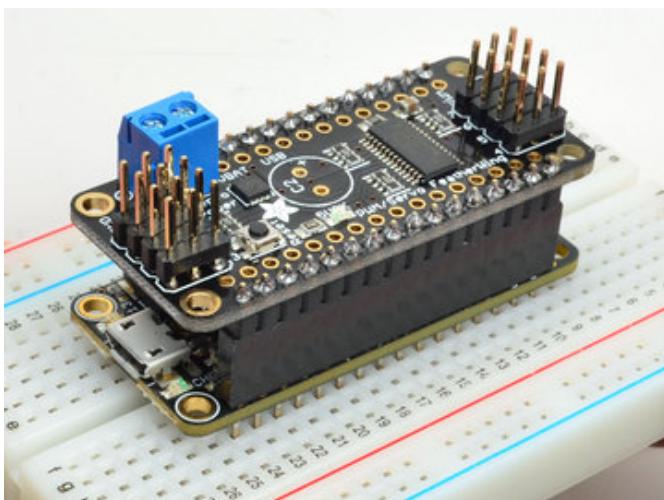


Last is the 3.5mm terminal block used for power. This is how you will provide the large amount of current that servos require.

Make sure the two open parts of the terminal face outwards so you can easily connect wires



Solder in both pins of the terminal block. You can remove the tape when done.



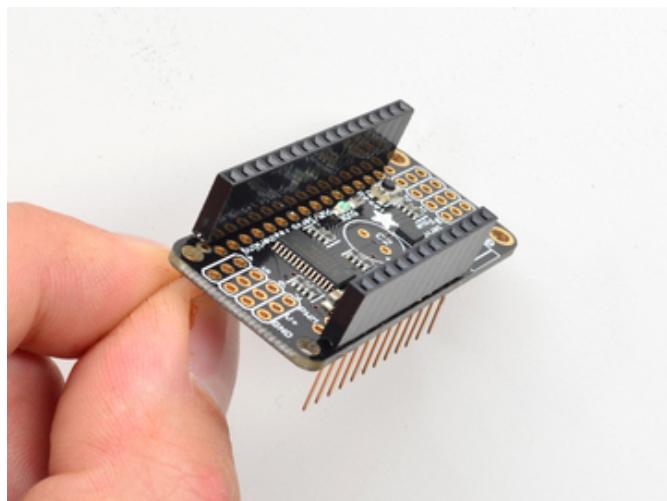
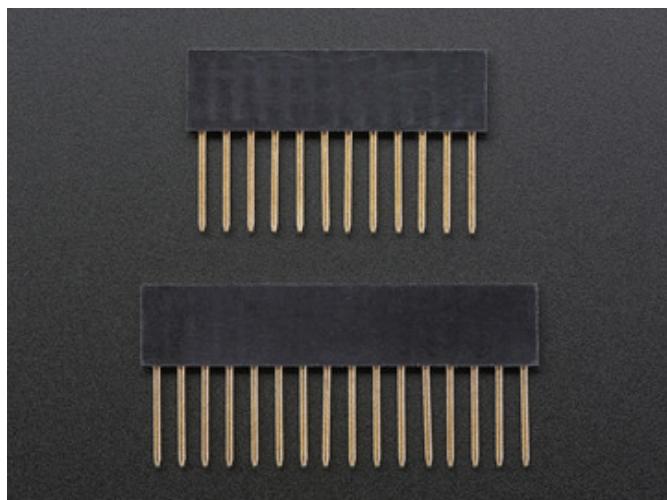
OK You're done! You can now plug in your FeatherWing into your Feather and get servo'ing

Stacking Assembly



For more controlling than 8 servos, you can stack Servo FeatherWings, **but the assembly is a little different**. You'll need to grab these products from the shop:

- [3x4 Right Angle Male Header - 4 pack](https://adafru.it/oAO) (<https://adafru.it/oAO>)
- [Feather Stacking Headers](https://adafru.it/oAP) (<https://adafru.it/oAP>)



Add the FeatherWing:

Place the stacking headers into the FeatherWing so that the long pins poke through the two rows of breakout pads. **Make sure the long pins are sticking out underneath the FeatherWing.**

To make it easier to keep these in place, you can use some tape to hold down the two header pieces. Tacky clay also works, whatever you've got handy!



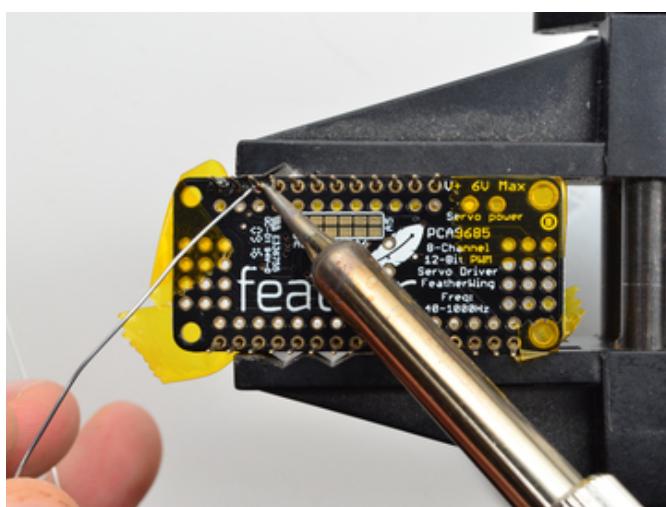


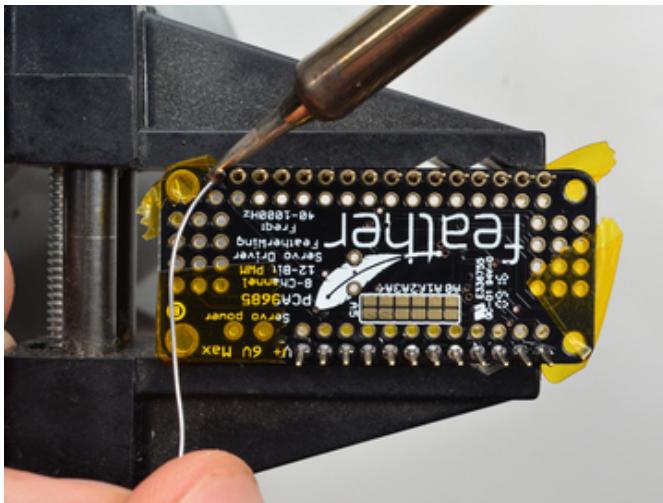
And Solder!

Be sure to solder all pins for reliable electrical contact.

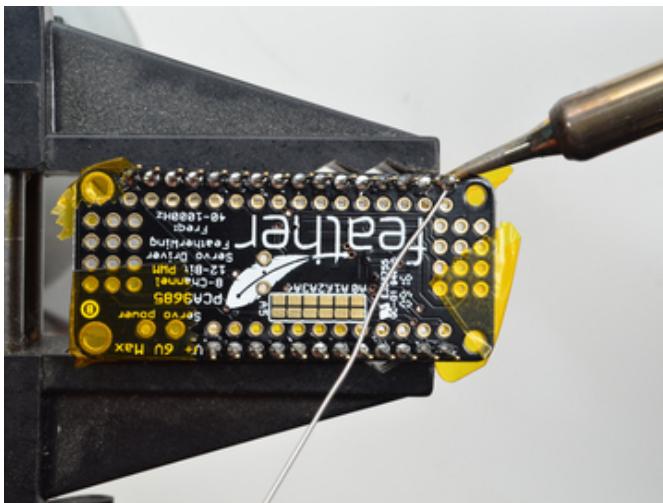
(For tips on soldering, be sure to check out our [Guide to Excellent Soldering](#) (<https://adafru.it/aTk>)).

Start by soldering the first row of header

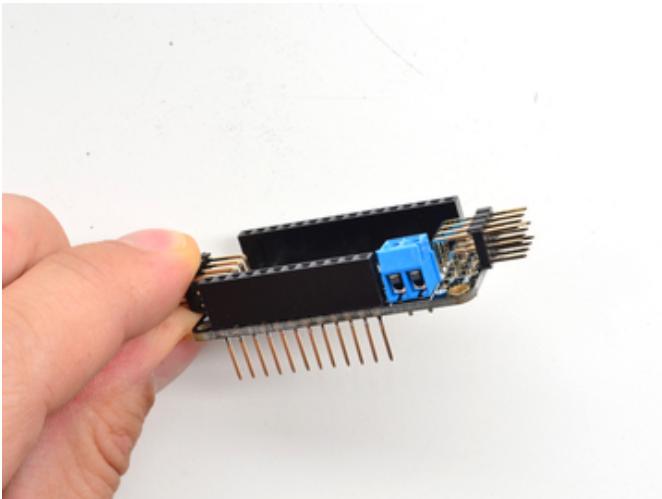




Now flip around and solder the other row completely

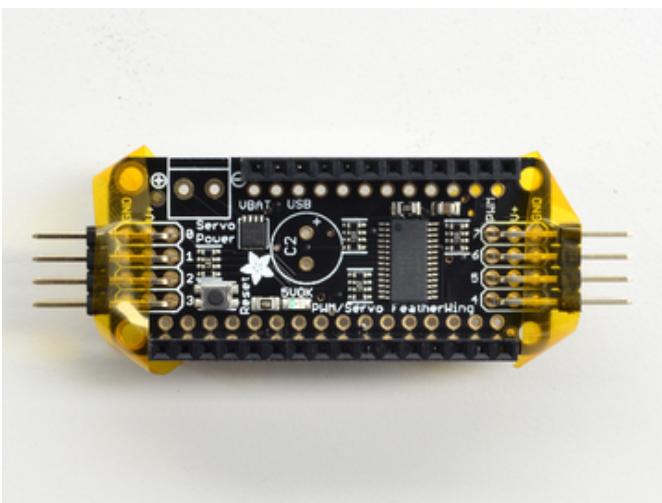


When you are finished, check that your soldered joints are nice and shiny, then continue to the next step

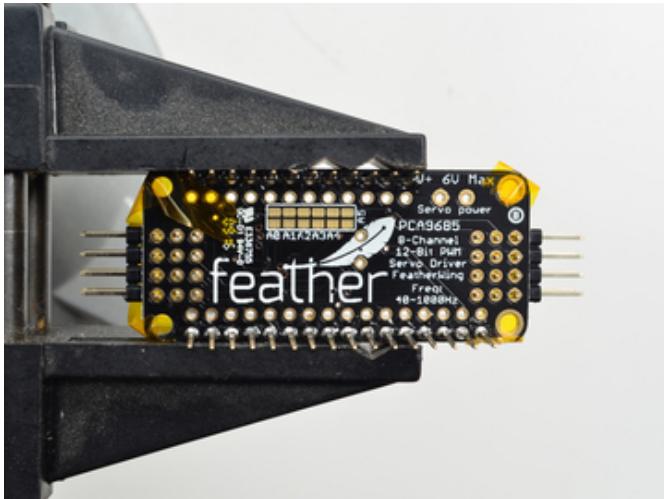


Next we will solder in the right angle header blocks used to plug in servos into the FeatherWing. There are two blocks, each are 3x4 headers in size.

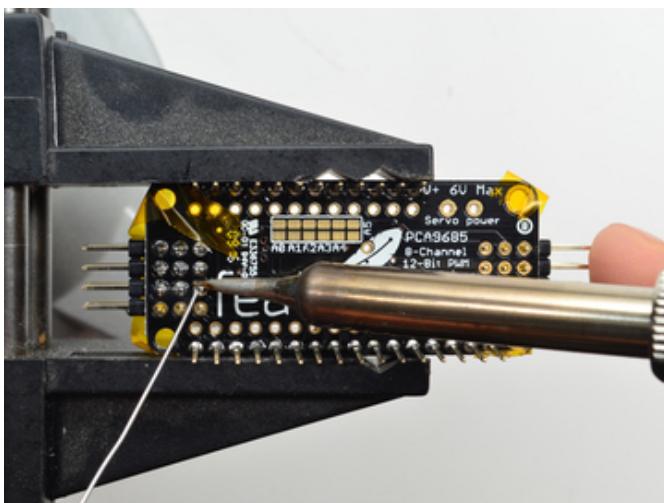
They go on either end. Make sure the **long side of the headers** is sticking off the top left & top right sides!

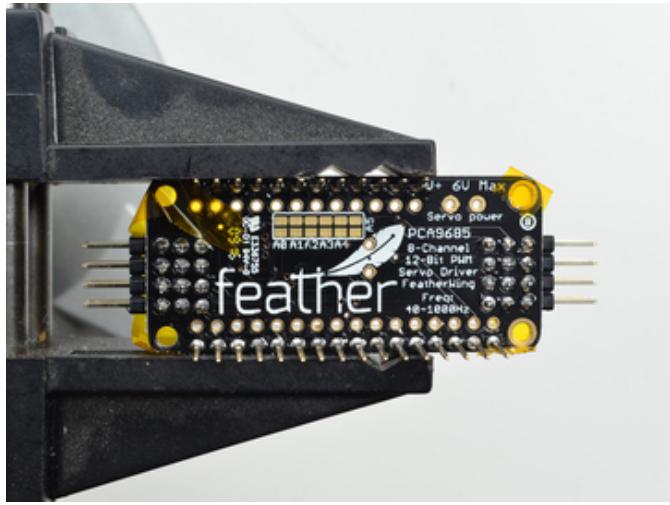


Again, use some tape to hold down the two header pieces in place to make soldering easier.

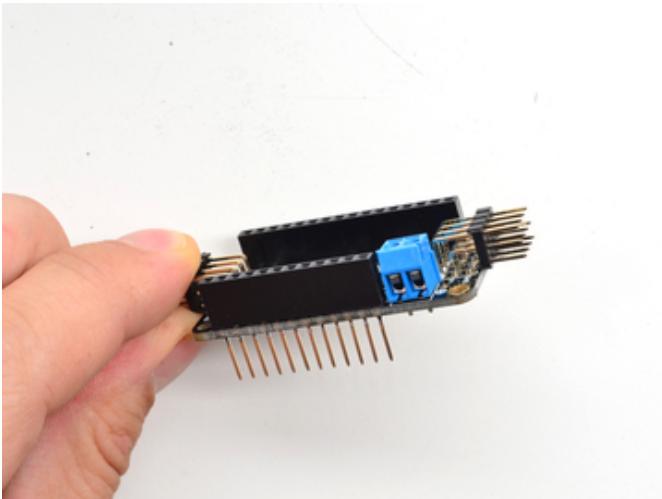


Flip it over and solder in each block, make sure you get to each of the 12 pins



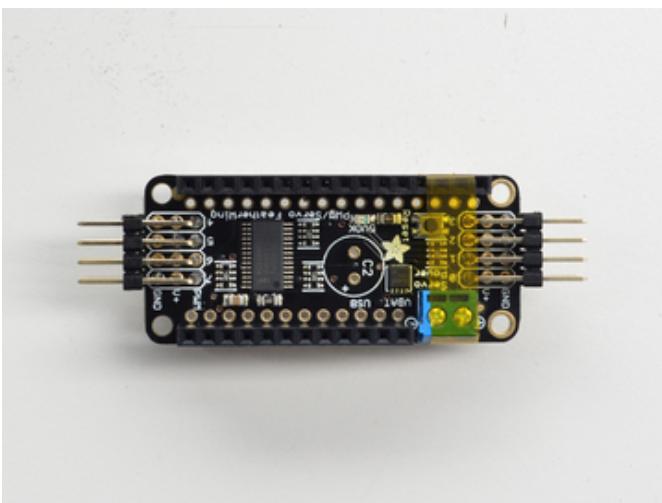


Now that you're done with the header blocks, check your work make sure that each solder joint is done and looks shiny.



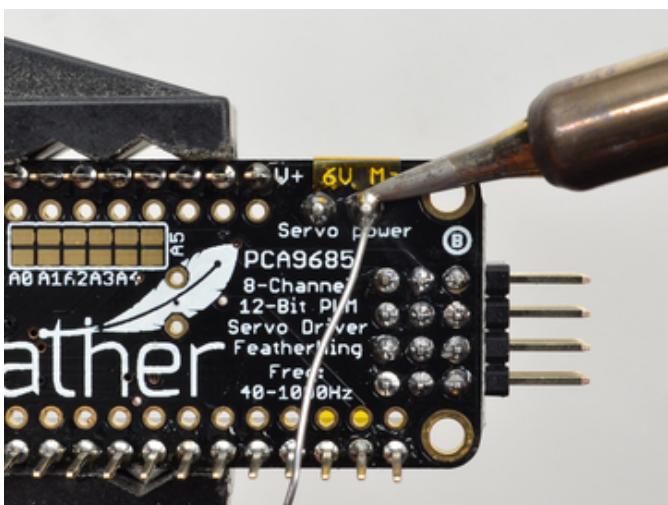
Last is the 3.5mm terminal block used for power. This is how you will provide the large amount of current that servos require.

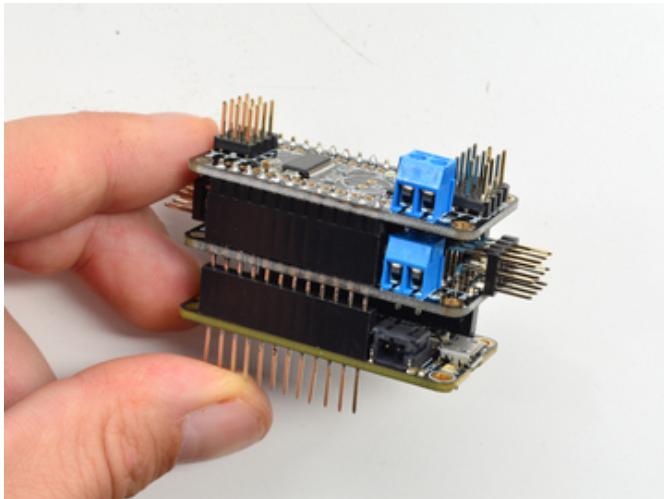
Make sure the two open parts of the terminal face outwards so you can easily connect wires. You'll want to tack down this one too.



Solder in both pins of the terminal block.

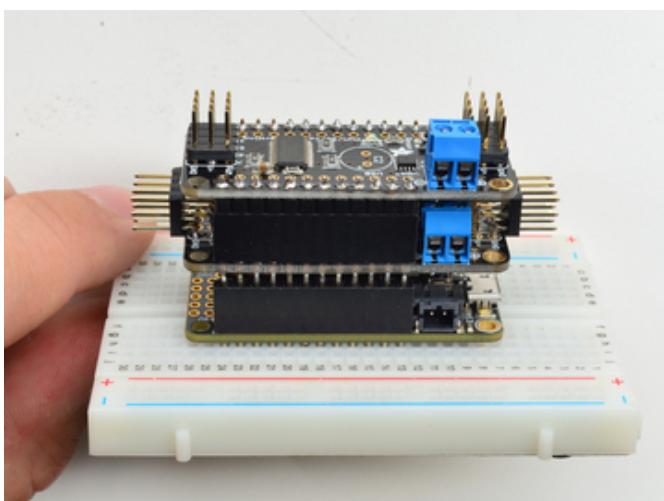
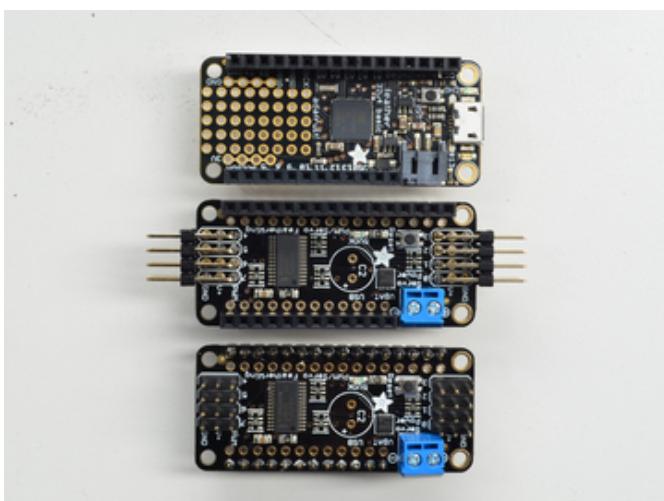
Check your work and remove the tape when done.





OK You're done! Check out the [Advanced Usage](https://adafru.it/oAQ) (<https://adafru.it/oAQ>) page for information on selecting an address for each board!

You can now stack your FeatherWings and get servo'ing



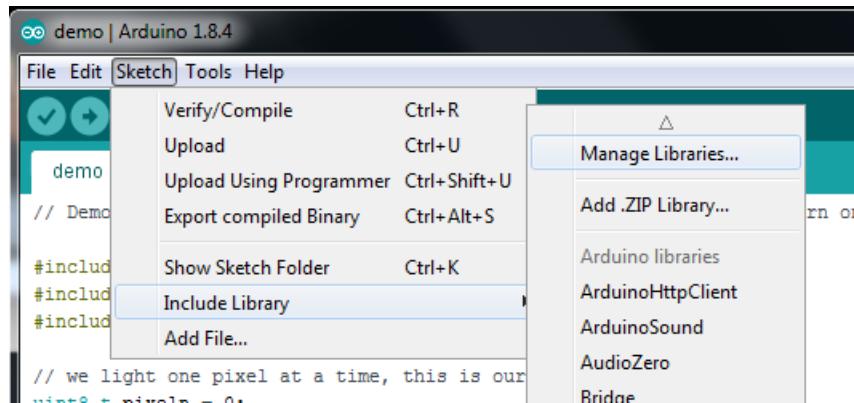
Using the Adafruit Library

Since the PWM Servo Driver is controlled over I2C, its super easy to use with any microcontroller or microcomputer. In this demo we'll show using it with the Arduino IDE but the C++ code can be ported easily

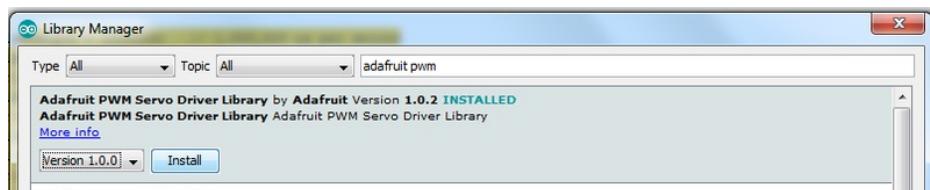
Install Adafruit PCA9685 library

To begin reading sensor data, you will need to [install the Adafruit_PWM servo library \(code on our github repository\) \(<https://adafru.it/aQl>\)](#). It is available from the Arduino library manager so we recommend using that.

From the IDE open up the library manager...



And type in **adafruit pwm** to locate the library. Click **Install**



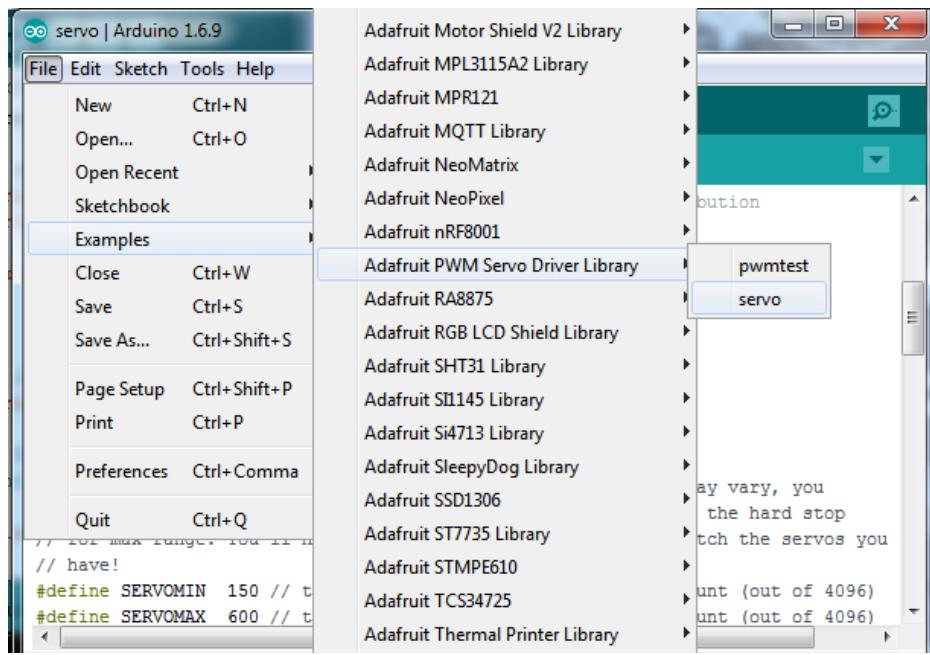
We also have a great tutorial on Arduino library installation at:

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

Test with the Example Code:

First make sure all copies of the Arduino IDE are closed.

Next open the Arduino IDE and select **File->Examples->Adafruit_PWMservoDriver->Servo**. This will open the example file in an IDE window.



If using a Breakout:

Connect the driver board and servo as shown on the previous page. Don't forget to provide power to both **Vin** (3-5V logic level) and **V+** (5V servo power). **Check the green LED is lit!**

If using a Shield:

Plug the shield into your Arduino. Don't forget you will also have to provide 5V to the V+ terminal block. **Both red and green LEDs must be lit.**

If using a FeatherWing:

Plug the FeatherWing into your Feather. Don't forget you will also have to provide 5V to the V+ terminal block. **Check the green LED is lit!**

Connect a Servo

A single servo should be plugged into the **PWM #0** port, the first port. You should see the servo sweep back and forth over approximately 180 degrees.

Calibrating your Servos

Servo pulse timing varies between different brands and models. Since it is an analog control circuit, there is often some variation between samples of the same brand and model. For precise position control, you will want to calibrate the minimum and maximum pulse-widths in your code to match known positions of the servo.

Find the Minimum:

Using the example code, edit SERVOMIN until the low-point of the sweep reaches the minimum range of travel. It is best to approach this gradually and stop before the physical limit of travel is reached.

Find the Maximum:

Again using the example code, edit SERVOMAX until the high-point of the sweep reaches the maximum range of travel. Again, is best to approach this gradually and stop before the physical limit of travel is reached.

- Use caution when adjusting SERVOMIN and SERVOMAX. Hitting the physical limits of travel can strip the gears and permanently damage your servo.

Converting from Degrees to Pulse Length

The [Arduino "map\(\)" function](#) (<https://adafru.it/aQm>) is an easy way to convert between degrees of rotation and your calibrated SERVOMIN and SERVOMAX pulse lengths. Assuming a typical servo with 180 degrees of rotation; once you have calibrated SERVOMIN to the 0-degree position and SERVOMAX to the 180 degree position, you can convert any angle between 0 and 180 degrees to the corresponding pulse length with the following line of code:

```
pulseLength = map(degrees, 0, 180, SERVOMIN, SERVOMAX);
```

Library Reference

setPWMFreq(freq)

Description

This function can be used to adjust the PWM frequency, which determines how many full 'pulses' per second are generated by the IC. Stated differently, the frequency determines how 'long' each pulse is in duration from start to finish, taking into account both the high and low segments of the pulse.

Frequency is important in PWM, since setting the frequency too high with a very small duty cycle can cause problems, since the 'rise time' of the signal (the time it takes to go from 0V to VCC) may be longer than the time the signal is active, and the PWM output will appear smoothed out and may not even reach VCC, potentially causing a number of problems.

Arguments

- **freq:** A number representing the frequency in Hz, between 40 and 1600

Example

The following code will set the PWM frequency to 1000Hz:

```
pwm.setPWMFreq(1000)
```

setPWM(channel, on, off)

Description

This function sets the start (on) and end (off) of the high segment of the PWM pulse on a specific channel. You specify the 'tick' value between 0..4095 when the signal will turn on, and when it will turn off. Channel indicates which of the 16 PWM outputs should be updated with the new values.

Arguments

- **channel:** The channel that should be updated with the new values (0..15)
- **on:** The tick (between 0..4095) when the signal should transition from low to high
- **off:** The tick (between 0..4095) when the signal should transition from high to low

Example

The following example will cause channel 15 to start low, go high around 25% into the pulse (tick 1024 out of 4096), transition back to low 75% into the pulse (tick 3072), and remain low for the last 25% of the pulse:

```
pwm.setPWM(15, 1024, 3072)
```

Using as GPIO

There's also some special settings for turning the pins fully on or fully off

You can set the pin to be fully on with

```
pwm.setPWM(pin, 4096, 0);
```

You can set the pin to be fully off with

```
pwm.setPWM(pin, 0, 4096);
```

Arduino Library Docs

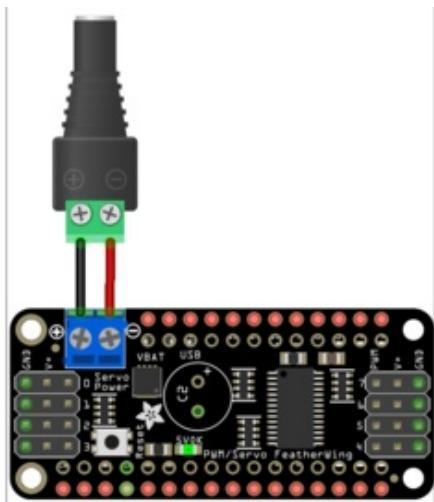
[Arduino Library Docs \(https://adafru.it/Au7\)](https://adafru.it/Au7)

CircuitPython Usage

It's easy to control PWM or servos with the Adafruit 8-Channel PWM or Servo FeatherWing. There are multiple CircuitPython libraries available to work with the different features of this board including [Adafruit CircuitPython PCA9685](https://adafru.it/tZF) (<https://adafru.it/tZF>), and [Adafruit CircuitPython ServoKit](https://adafru.it/Dpu) (<https://adafru.it/Dpu>). These libraries make it easy to write Python code to control PWM and servo motors.

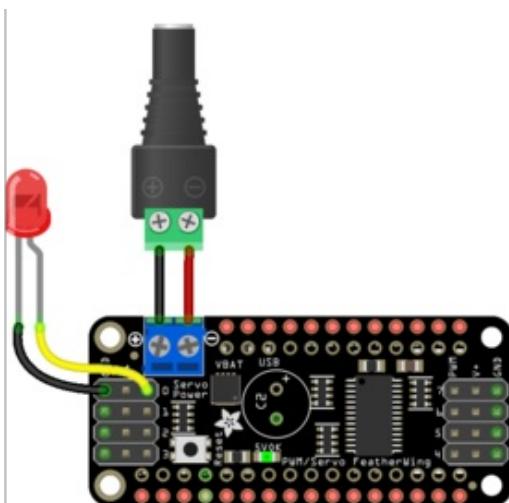
CircuitPython Microcontroller Wiring

First assemble the FeatherWing exactly as shown in the previous pages. There's no wiring needed to connect the FeatherWing to the Feather. The example below shows the FeatherWing attached to a Feather using the stacking assembly method, and wiring a barrel jack to the power terminal to attach an appropriate external power source to the FeatherWing. **The FeatherWing will not power servos without an external power source!**



- Connect the **positive side of the power terminal** to the **positive side of the barrel jack**.
- Connect the **negative side of the power terminal** to the **negative side of the barrel jack**.

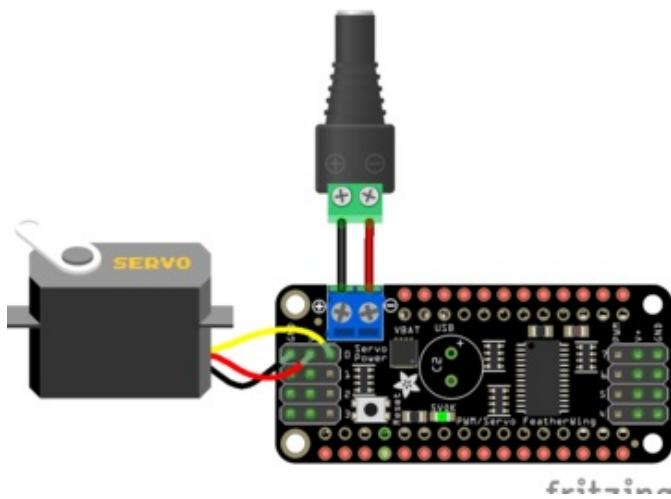
To dim an LED, wire it to the board as follows. Note: you don't need to use a resistor to limit current through the LED as the FeatherWing will limit the current to around 10mA.



- Connect **LED cathode / shorter leg** to **FeatherWing channel GND / ground**.
- Connect **LED anode / longer leg** to **FeatherWing channel PWM**.

External power is not necessary to PWM an LED.

To control a servo, wire it to the board as follows.



Check your servo datasheet to be certain.

Typically:

- Connect servo brown wire to the **FeatherWing channel GND**
- Connect servo red wire to the **FeatherWing channel V+**
- Connect servo yellow wire to the **FeatherWing channel PWM**
- Connect the **positive side of the power terminal** to the **positive side of the barrel jack**.
- Connect the **negative side of the power terminal** to the **negative side of the barrel jack**.

You must have an external power source to run servos!

CircuitPython Installation of ServoKit and Necessary Libraries

You'll need to install a few libraries on your Feather 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>).

If you choose, you can manually install the libraries individually on your board:

- `adafruit_pca9685`
- `adafruit_bus_device`
- `adafruit_register`
- `adafruit_motor`
- `adafruit_servokit`

Before continuing make sure your board's lib folder or root filesystem has the `adafruit_pca9685.mpy`, `adafruit_register`, `adafruit_motor`, `adafruit_bus_device` and `adafruit_servokit` 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.

CircuitPython Usage

To demonstrate the usage, we'll use Python code to control PWM to dim an LED and to control servo motors from the board's Python REPL.

Dimming LEDs

This FeatherWing uses the PCA9685. Each channel of the FeatherWing can be used to control the brightness of an LED. The PCA9685 generates a high-speed PWM signal which turns the LED on and off very quickly. If the LED is

turned on longer than turned off it will appear brighter to your eyes.

First you'll need to import the necessary modules, initialize the I2C bus for your board, and create an instance of the class.

```
import board
import busio
import adafruit_pca9685
i2c = busio.I2C(board.SCL, board.SDA)
wing = adafruit_pca9685.PCA9685(i2c)
```

The PCA9685 class provides control of the PWM frequency and each channel's duty cycle. Check out the [PCA9685 class documentation](#) (<https://adafru.it/C5n>) for more details.

For dimming LEDs you typically don't need to use a fast PWM signal frequency and can set the board's PWM frequency to 60hz by setting the `frequency` attribute:

```
wing.frequency = 60
```

The FeatherWing supports 8 separate channels that share a frequency but can have independent duty cycles. That way you could dim 8 LEDs separately!

The PCA9685 object has a `channels` attribute which has an object for each channel that can control the duty cycle. To get the individual channel use the `[]` to index into `channels`.

```
led_channel = wing.channels[0]
```

Now control the LED brightness by controlling the duty cycle of the channel connected to the LED. The duty cycle value should be a 16-bit value, i.e. `0` to `0xffff` (65535), which represents what percent of the time the signal is on vs. off. A value of `0xffff` is 100% brightness, `0` is 0% brightness, and in-between values go from 0% to 100% brightness.

For example set the LED completely on with a `duty_cycle` of `0xffff`:

```
led_channel.duty_cycle = 0xffff
```

After running the command above you should see the LED light up at full brightness!

Now turn the LED off with a `duty_cycle` of `0`:

```
led_channel.duty_cycle = 0
```

Try an in-between value like `1000`:

```
led_channel.duty_cycle = 1000
```

You should see the LED dimly lit. Try experimenting with other duty cycle values to see how the LED changes brightness!

For example make the LED glow on and off by setting `duty_cycle` in a loop:

```
# Increase brightness:  
for i in range(0xffff):  
    led_channel.duty_cycle = i  
  
# Decrease brightness:  
for i in range(0xffff, 0, -1):  
    led_channel.duty_cycle = i
```

These for loops take a while because 16-bits is a lot of numbers. **CTRL-C** to stop the loop from running and return to the REPL.

That's all there is to dimming LEDs using CircuitPython and the PWM/Servo FeatherWing!

Controlling Servos

We've written a handy CircuitPython library for the various PWM/Servo kits called [Adafruit CircuitPython ServoKit](#) (<https://adafru.it/Dpu>) that handles all the complicated setup for you. All you need to do is import the appropriate class from the library, and then all the features of that class are available for use. We're going to show you how to import the `ServoKit` class and use it to control servo motors with the Adafruit PWM/Servo FeatherWing.

First you'll need to import and initialize the `ServoKit` class. You must specify the number of channels available on your board. The FeatherWing has 8 channels, so when you create the class object, you will specify `8`.

```
from adafruit_servokit import ServoKit  
kit = ServoKit(channels=8)
```

Now you're ready to control both standard and continuous rotation servos.

Standard Servos

To control a standard servo, you need to specify the channel the servo is connected to. You can then control movement by setting `angle` to the number of degrees.

For example to move the servo connected to channel `0` to `180` degrees:

```
kit.servo[0].angle = 180
```

To return the servo to `0` degrees:

```
kit.servo[0].angle = 0
```

With a standard servo, you specify the position as an angle. The angle will always be between 0 and the actuation range. The default is 180 degrees but your servo may have a smaller sweep. You can change the total angle by setting `actuation_range`.

For example, to set the actuation range to 160 degrees:

```
kit.servo[0].actuation_range = 160
```

Often the range an individual servo recognises varies a bit from other servos. If the servo didn't sweep the full expected range, then try adjusting the minimum and maximum pulse widths using `set_pulse_width_range(min_pulse, max_pulse)`.

To set the pulse width range to a minimum of 1000 and a maximum of 2000:

```
kit.servo[0].set_pulse_width_range(1000, 2000)
```

That's all there is to controlling standard servos with the PWM/Servo FeatherWing, CircuitPython and [ServoKit](#)!

Continuous Rotation Servos

To control a continuous rotation servo, you must specify the channel the servo is on. Then you can control movement using `throttle`.

For example, to start the continuous rotation servo connected to channel `1` to full throttle forwards:

```
kit.continuous_servo[1].throttle = 1
```

To start the continuous rotation servo connected to channel `1` to full reverse throttle:

```
kit.continuous_servo[1].throttle = -1
```

To set half throttle, use a decimal:

```
kit.continuous_servo[1].throttle = 0.5
```

And, to stop continuous rotation servo movement set `throttle` to `0`:

```
kit.continuous_servo[1].throttle = 0
```

That's all there is to controlling continuous rotation servos with the PWM/Servo FeatherWing, CircuitPython and [ServoKit](#)!

Full Example Code

```
Temporarily unable to load content:
```

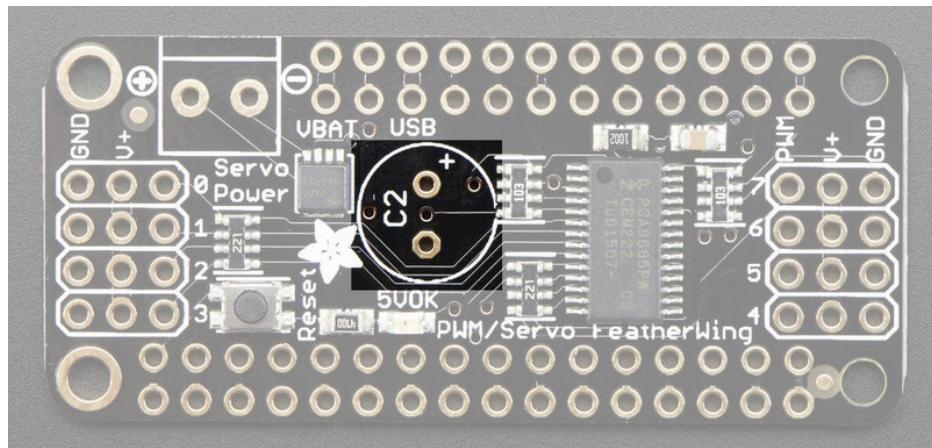
Python Docs

[Python Docs \(https://adafru.it/Dkx\)](https://adafru.it/Dkx)

Advanced Usage

Adding a Capacitor to the thru-hole capacitor slot

We have a spot on the PCB for soldering in an electrolytic capacitor. Based on your usage, you may or may not need a capacitor. If you are driving a lot of servos from a power supply that dips a lot when the servos move, $n * 100\mu F$ where n is the number of servos is a good place to start - eg **470 μF** or more for 5 servos. Since its so dependent on servo current draw, the torque on each motor, and what power supply, there is no "one magic capacitor value" we can suggest which is why we don't include a capacitor in the kit.



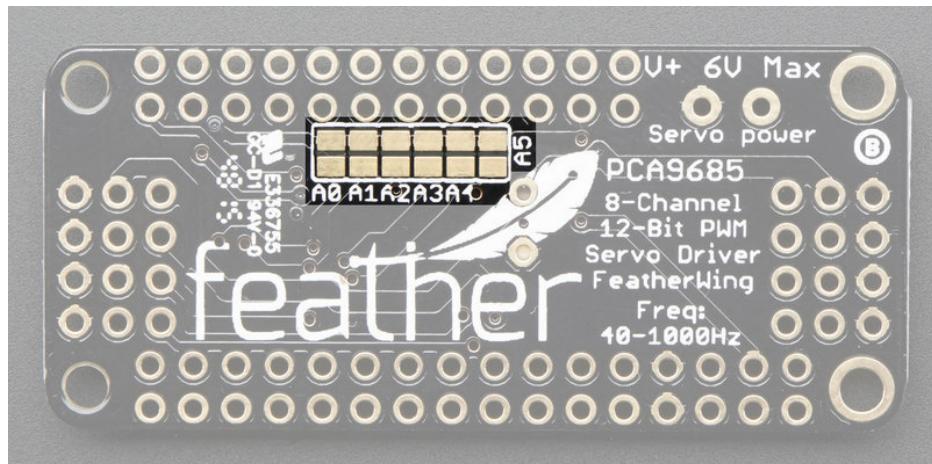
Adding/Stacking Servo Feathers - Using different i2c addresses

If you need more Servos, you can stack the FeatherWings (see [Stacking Assembly \(https://adafru.it/oAR\)](https://adafru.it/oAR)) - each new Wing will give you 8 more Servos

You'll need to solder in stacking Feather headers so you can plug more Wings on top!

You'll also need to use right-angle 3x4 headers on the Servo Wings so that the servos connect off the ends rather than straight up

Each I2C device connected to your Feather must be assigned a unique address. This is done with the address jumpers on the bottom of the board. The I2C base address for each board is **0x40**. The binary address that you program with the address jumpers is added to the base I2C address.



To program the address offset, use a drop of solder to bridge the corresponding address jumper for each binary '1' in the address.

Board 0: Address = **0x40** Offset = binary 00000 (no jumpers required)

Board 1: Address = **0x41** Offset = binary 00001 (bridge A0)

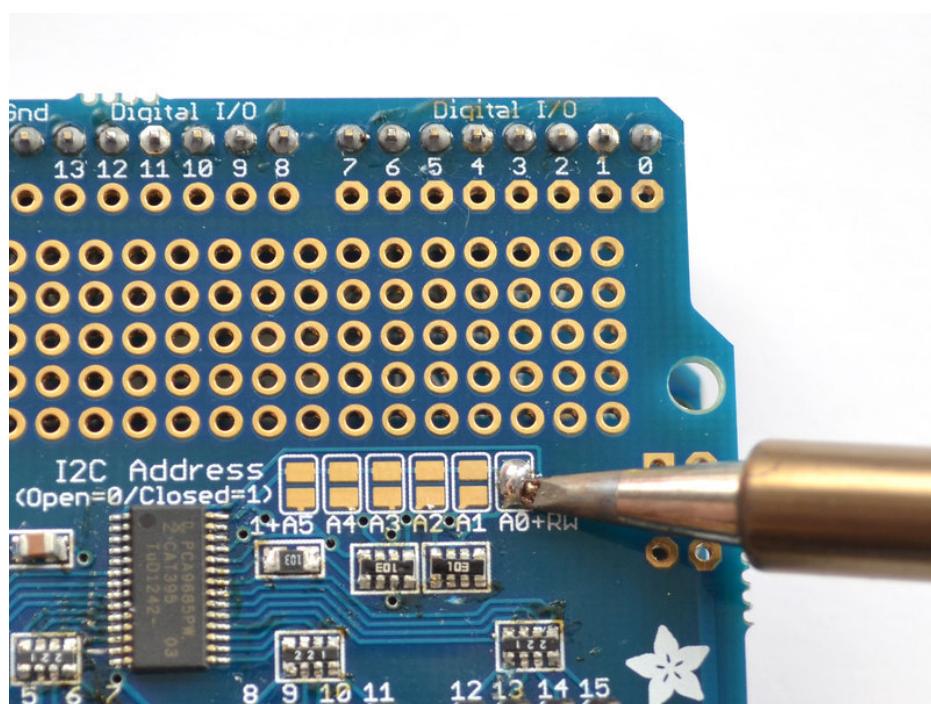
Board 2: Address = **0x42** Offset = binary 00010 (bridge A1)

Board 3: Address = **0x43** Offset = binary 00011 (bridge A0 & A1)

Board 4: Address = **0x44** Offset = binary 00100 (bridge A2)

etc.

See how we do this for the Arduino Shield version of this board, its nearly identical:



FAQ

□ Can this board be used for LEDs or just servos?

It can be used for LEDs as well as any other PWM-able device!

□ I am having strange problems when combining this shield with the Adafruit LED Matrix/7Seg Backpacks

The PCA9865 chip has an "All Call" address of 0x70. This is in addition to the configured address. Set the backpacks to address 0x71 or anything other than the default 0x70 to make the issue go away.

With LEDs, how come I cant get the LEDs to turn completely off?

If you want to turn the LEDs totally off use `setPWM(pin, 4096, 0);` not `setPWM(pin, 4095, 0);`

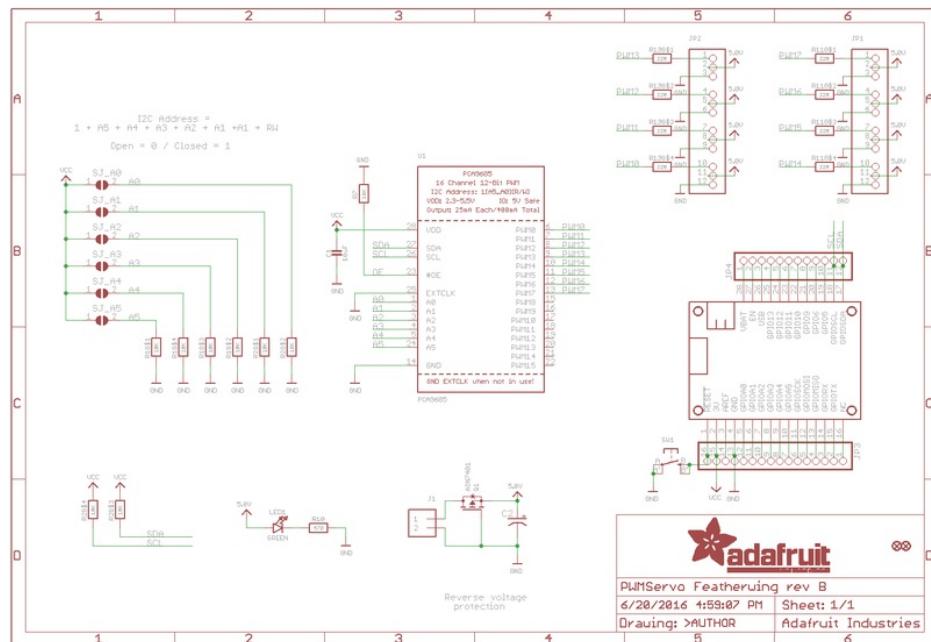
Downloads

Files

- PCA9685 Datasheet (<https://adafru.it/okB>)
 - EagleCad PCB files on GitHub (<https://adafru.it/okC>)
 - Fritzing object in Adafruit Fritzing library (<https://adafru.it/aP3>)

Schematic

[Click to embiggen](#)



Fabrication Print

Dimensions in Inches

