CircuitPython Hardware: PCA9685 PWM & Servo Driver

Created by Tony DiCola
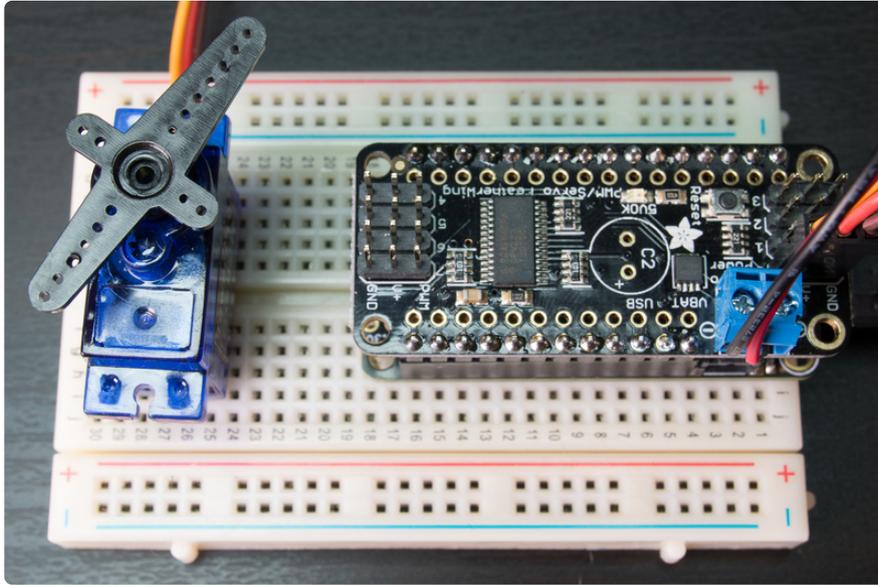

https://learn.adafruit.com/micropython-hardware-pca9685-pwm-and-servo-driver
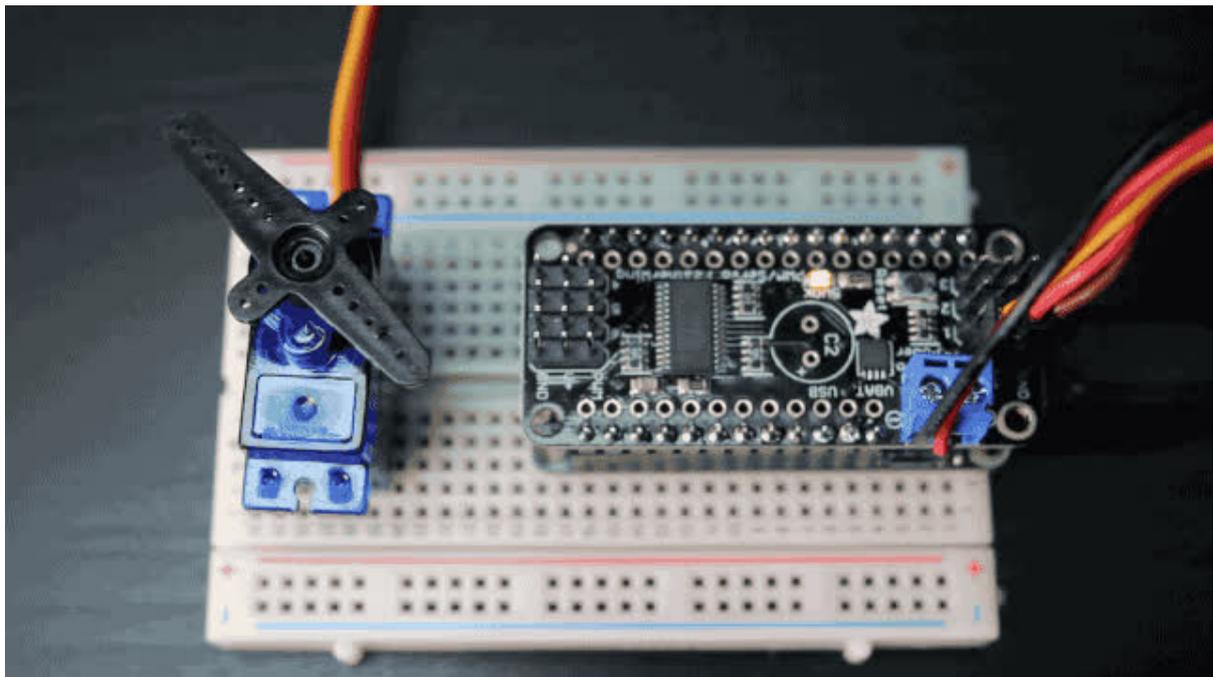
Last updated on 2024-06-03 02:01:04 PM EDT

Table of Contents

Overview

Note the video above was made showing the MicroPython version of this library. Follow the guide to see both CircuitPython and MicroPython versions of the PCA9685 library.

Servo motors are one way to make projects come to life with exciting movements like steering robots, flipping switches, and more. To control a servo you need to generate a special PWM, or pulse-width modulation, signal. Most development boards can generate a few basic PWM signals, but what if you need to control a lot of servos or don't have a board with PWM support? The PCA9685 PWM driver board comes to the rescue! This driver board can generate up to 16-channels (or 8-channels in FeatherWing form) of PWM signals--perfect for driving many servos!

This guide explores how to use the PCA9685 PWM & Servo driver with CircuitPython and MicroPython. You'll learn how to connect the PCA9685 to a board and use it to control LED brightness and move servos from Python code.

For more background check out these guides on servo motors:

- Adafruit Motor Selection Guide: Servos (https://adafru.it/scS)

- Arduino Lesson 14: Servos (https://adafru.it/scT) (focus on the general servo information, less on the Arduino details)

Hardware

The examples in this guide are no longer supported. Check out the PCA9685 driver guide for CircuitPython and Python usage: https://learn.adafruit.com/16-channel-pwm-servo-driver

Parts

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

CircuitPython board. This guide focuses on the ESP8266 (http://adafru.it/2821) and Feather M0/SAMD21-based boards (http://adafru.it/2772), but any CircuitPython board that supports I2C should work.
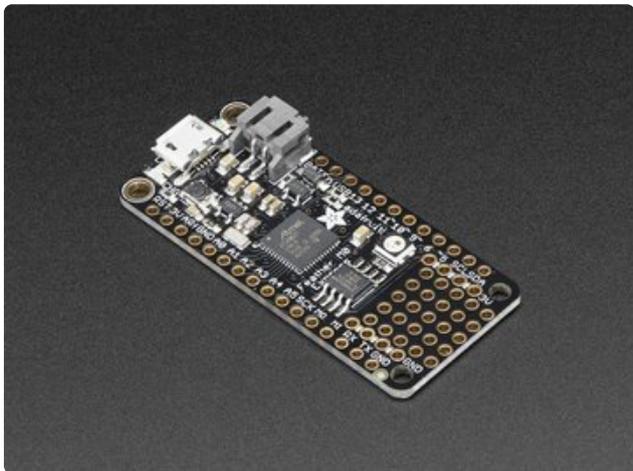
If your board doesn't come with CircuitPython running on it already then check out your board's guide for how to load CircuitPython firmware. For example the Feather M0 express guide (https://adafru.it/wbv) is a good reference.

If you're using a Feather board and FeatherWing you probably want a Feather female header set (http://adafru.it/2886) or Feather stacking female header set (http://adafru.it/2830).
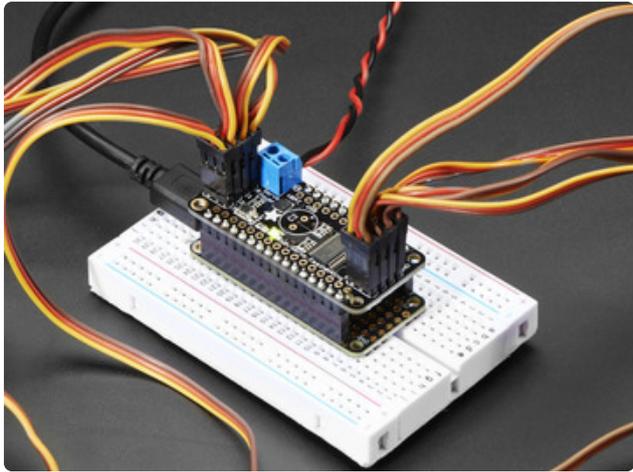
PCA9685 PWM & Servo Driver Board. If you're using a Feather the 8-channel PWM & Servo FeatherWing (http://adafru.it/ 2928) is the perfect option. If you need more channels or aren't using a Feather grab the 16-channel PCA9685 breakout board (http://adafru.it/815) or even 16-channel PCA9685 Arduino shield (http:// adafru.it/1411).

5V Power Supply. To power servos you almost always need an external 5V power supply. Servos can pull a lot of power and could damage your board if powered directly from it! A simple power option is a 4x AA battery pack (http://adafru.it/ 830) or a 5V power supply (http://adafru.it/ 276) and barrel jack screw terminal connector (http://adafru.it/368).
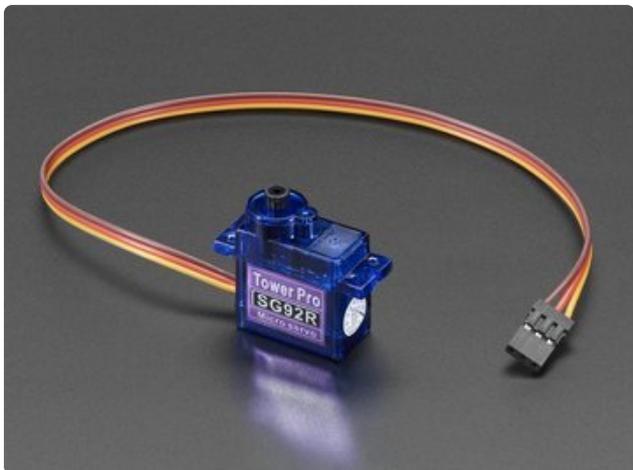
Servos or LEDs. You'll want something to drive with the PCA9685, like servos (https://adafru.it/scU) you can move or LEDs (https://adafru.it/dLA) you can dim. Remember the PCA9685 only drives **servos**, not stepper or other motors.

Breadboard **(http://adafru.it/64)** and jumper wires **(http://adafru.it/153)**. If you aren't using a Feather and FeatherWing you'll need a breadboard and jumper wires to connect the components.

Soldering tools **(http://adafru.it/136)**. You'll need to solder headers to the boards Check out the guide to excellent soldering (https://adafru.it/dxy) if you're new to soldering.

Make sure to follow the board and PCA9685 driver board product guides to assemble and verify they work before continuing.

Wiring

If you're using a Feather and the PCA9685 FeatherWing just slide the wing into the Feather's headers and you're done! The wing will use an I2C connection to talk to the Feather board.

If you're using a PCA9685 breakout you'll need to connect its I2C and power pins as follows:

Fritzing Source

https://adafru.it/zev
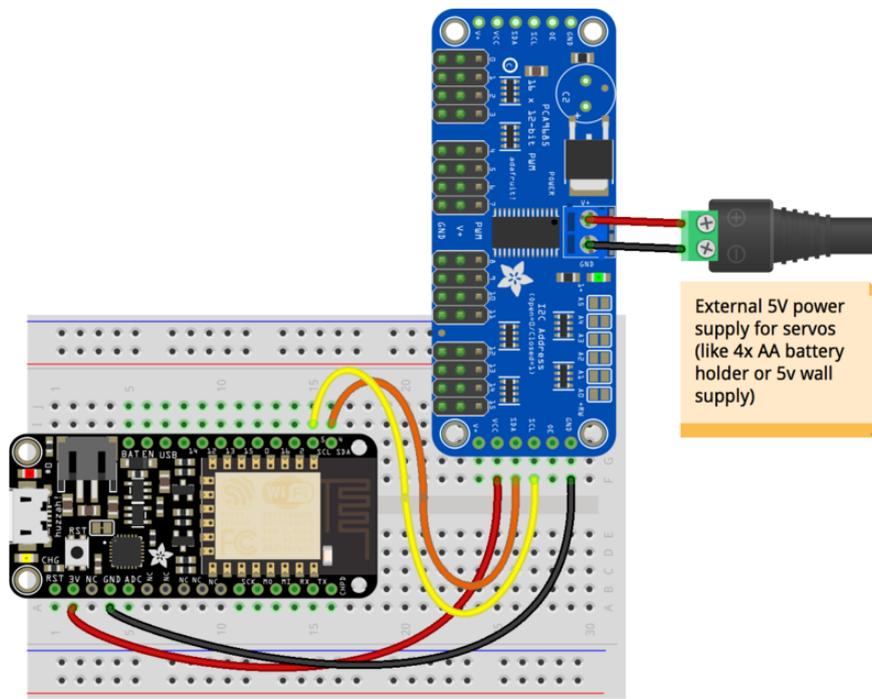
- **Board SCL / I2C clock** to **PCA9685 SCL**.
- **Board SDA / I2C data** to **PCA9685 SDA**.
- **Board 3.3V power** to **PCA9685 VCC**.
- **Board GND / ground** to **PCA9685 GND / ground**.

Once your board is wired to the PCA9685 continue on to learn how to use a MicroPython module to control servos and LEDs!

CircuitPython

The examples in this guide are no longer supported. Check out the PCA9685 driver guide for CircuitPython and Python usage: https://learn.adafruit.com/16-channel-pwm-servo-driver

Adafruit CircuitPython Module Install

To use the PCA9685 with your Adafruit CircuitPython (https://adafru.it/tCy) board you'll need to install the Adafruit_CircuitPython_PCA9685 (https://adafru.it/

tZF) module on your board. **Remember this module is for Adafruit CircuitPython firmware and not MicroPython.org firmware!**

First make sure you are running the latest version of Adafruit CircuitPython (https://adafru.it/tBa) for your board. Next you'll need to install the necessary libraries to use the hardware--read below and carefully follow the referenced steps to find and install these libraries from Adafruit's CircuitPython library bundle (https://adafru.it/zdx) with a version 20180110 or newer.

Bundle Install

For express boards that have extra flash storage, like the Feather/Metro M0 express and Circuit Playground express, you can easily install the necessary libraries with Adafruit's CircuitPython bundle (https://adafru.it/zdx). This is an all-in-one package that includes the necessary libraries to use the PCA9685 with CircuitPython. To install the bundle follow the steps in your board's guide, like these steps for the Feather M0 express board (https://adafru.it/zco).

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

- **adafruit_pca9685**
- **adafruit_bus_device**
- **adafruit_register**
- **adafruit_motor**

If your board supports USB mass storage, like the M0-based boards, then simply drag the files to the board's file system. **Note on boards without external SPI flash, like a Feather M0 or Trinket/Gemma M0, you might run into issues on Mac OSX with hidden files taking up too much space when drag and drop copying,** see this page for a workaround **(https://adafru.it/u1d).**

If your board doesn't support USB mass storage, like the ESP8266, then use a tool like ampy to copy the file to the board (https://adafru.it/s1f). You can use the latest version of ampy and its new directory copy command (https://adafru.it/q2A) to easily move module directories to the board.

Before continuing make sure your board's lib folder or root filesystem has the **adafruit_pca9685**, **adafruit_bus_device**, **adafruit_register** and **adafruit_motor** folders/modules copied over.
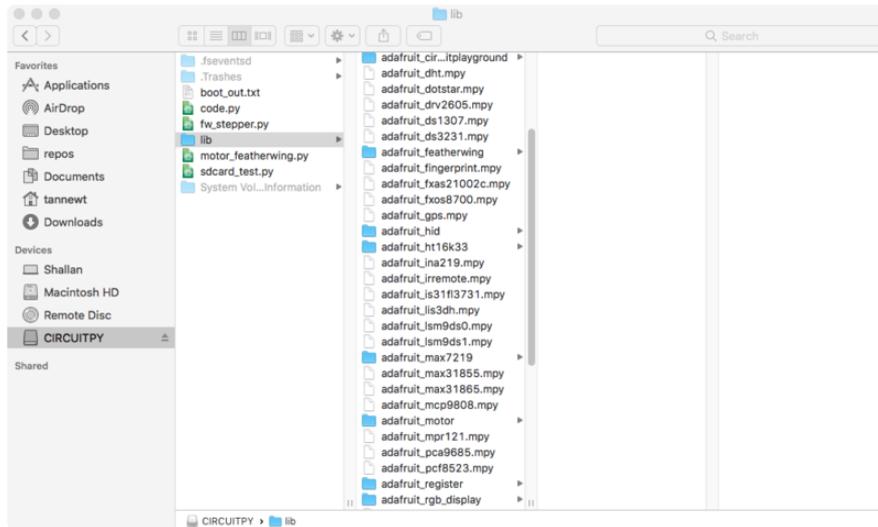
Usage

The following section will show how to control the PCA9685 from the board's Python prompt / REPL. You'll learn how to interactively control servos and dim LEDs by typing in the code below.

First connect to the board's serial REPL (https://adafru.it/pMf)so you are at the CircuitPython >>> prompt.

I2C Initialization

First you'll need to initialize the I2C bus for your board. First import the necessary modules:

```
import board
import busio
```

Now for either board run this command to create the I2C instance using the default SCL and SDA pins (which will be marked on the boards pins if using a Feather or similar Adafruit board):

```
i2c = busio.I2C(board.SCL, board.SDA)
```

After initializing the I2C interface you need to import the PCA9685 module to use it in your own code:

```
import adafruit_pca9685
```

Dimming LED's

Each channel of the PCA9685 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 wire a LED to the board as follows. Note you don't need to use a resistor to limit current through the LED as the PCA9685 will limit the current to around 10mA:

Fritzing Source

https://adafru.it/zew
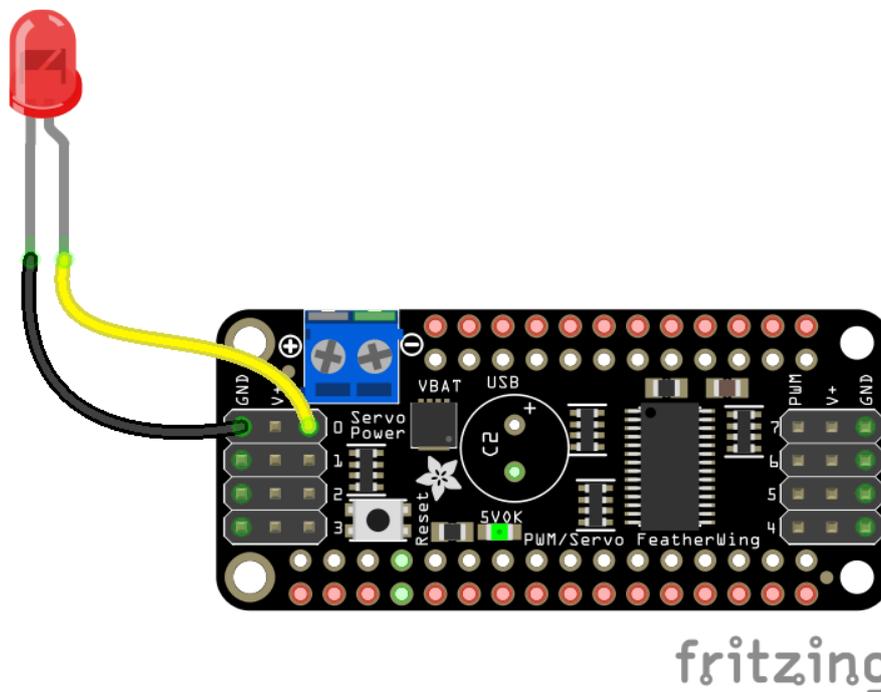
- **LED cathode / shorter leg** to **PCA9685 channel GND / ground**.
- **LED anode / longer leg** to **PCA9685 channel PWM**.

Now in the Python REPL you can create an instance of the basic PCA9685 class which provides low-level PWM control of the board's channels:

```
pca = 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:

```
pca.frequency = 60
```

The PCA9685 supports 16 separate channels that share a frequency but can have independent duty cycles. That way you could dim 16 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 = pca.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, 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.

Control Servos

Servo motors use a PWM signal to control the position of their arm or horn. Using the PCA9685 and the Motor library (https://adafru.it/C5o) you can easily plug in servos and control them with Python. If you aren't familiar with servos be sure to first read this intro to servos page (https://adafru.it/scW) and this in-depth servo guide page (https://adafru.it/scS).

First connect the servo to a channel on the PCA9685. Be sure to plug the servo connector in the correct way! Check your servo's datasheet to be sure, but typically the brown wire is connected to ground, the red wire is connected to 5V power, and the yellow pin is connected to PWM:
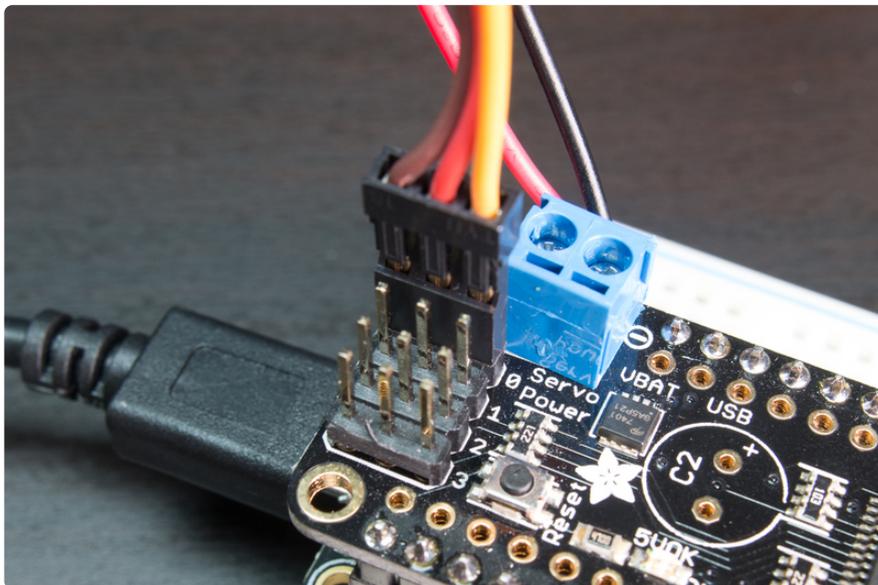
Be sure you've turned on or plugged in the external 5V power supply to the PCA9685 board too!

Now in the Python REPL as above with the led, save a variable for its channel:

```
servo_channel = pca.channels[0]
```

Servos typically operate at a frequency of 50 hz so update pca accordingly.

```
pca.frequency = 50
```

Now that the PCA9685 is set up for servos lets make a Servo object so that we can adjust the servo based on **angle** instead of **duty_cycle**.

By default the Servo class will use actuation range, minimum pulse-width, and maximum pulse-width values that should work for most servos. However check the Servo class documentation (https://adafru.it/BNE) for more details on extra parameters to customize the signal generated for your servos.

```
import adafruit_motor.servo
servo = adafruit_motor.servo.Servo(servo_channel)
```

With Servo, you specify a position as an angle. The angle will always be between 0 and the actuation range given when Servo was created. The default is 180 degrees but your servo might have a smaller sweep--change the total angle by specifying the **actuation_angle** parameter in the Servo class initializer above.

Now set the angle to 180, one extreme of the range:

```
servo.angle = 180
```

Or to sweep back to the minimum 0 degree position:

```
servo.angle = 0
```

Often the range an individual servo recognizes varies a bit from other servos. If the servo didn't sweep the full expected range, then try adjusting min_pulse and max_pulse. Lower min_pulse until the servo stops moving or moves irregularly when angle is changed to 0. Raise max_pulse util the servo stops moving or moves irregularly when angle is change to the actuation range.

```
servo = adafruit_motor.servo.Servo(servo_channel, min_pulse=800, max_pulse=2200)
```

That's all there is to controlling servos with the PCA9685 and CircuitPython! Using the **angle** attribute you can sweep and move servos in any way. This is perfect for building robots, actuating switches, or other fun mechanical projects!

MicroPython

The examples in this guide are no longer supported. Check out the PCA9685 driver guide for CircuitPython and Python usage: https://learn.adafruit.com/16-channel-pwm-servo-driver

Note this page describes how to use a MicroPython.org version of this library with MicroPython boards. Skip back to the previous page if you're using a CircuitPython board like the Feather M0 express!

In addition to CircuitPython there's an older MicroPython version of the PCA9685 library that you can use with some MicroPython boards. Before you get started it will help to be familiar with these guides for working with MicroPython:

- MicroPython Basics: What is MicroPython? (https://adafru.it/pXa)
- MicroPython Basics: How to Load MicroPython on a Board (https://adafru.it/pNB)
- MicroPython Basics: Load Files & Run Code (https://adafru.it/s1f)
- MicroPython Hardware: Analog I/O (https://adafru.it/s2d) (see the PWM page (https://adafru.it/scR) specifically)

See all the MicroPython guides in the learning system (https://adafru.it/qzD) for more information.

MicroPython Module Install

To use the PCA9685 with your MicroPython board you'll need to install the micropython-adafruit-pca9685 MicroPython module (https://adafru.it/scq) on your board. **Remember this module is for MicroPython.org firmware and not Adafruit CircuitPython!**

First make sure you are running the latest version of MicroPython for your board. If you're using the **ESP8266 MicroPython** port you **must** be running version 1.8.5 or higher **(https://adafru.it/sas)** as earlier versions do not support using .mpy modules as shown in this guide.

Next download the latest **pca9685.mpy, servo.mpy, motor.mpy, and stepper.mpy** file from the releases page (https://adafru.it/scr) of the micropython-adafruit-pca9685 GitHub repository (https://adafru.it/scq). You'll need to copy **all** of the files to your

MicroPython board's file system and can use a tool like ampy to copy the files to the board (https://adafru.it/r2B).

Usage

The following section will show how to control the PCA9685 from the board's Python prompt / REPL.   First connect to the board's serial REPL (https://adafru.it/pMf)so you are at the MicroPython >>> prompt.

I2C Initialization

First you'll need to initialize the I2C bus for your board. On MicroPython.org firmware which uses the machine API you can initialize I2C like the MicroPython I2C guide mentions (https://adafru.it/sau). For example on a board like the ESP8266 you can run (assuming you're using the default SDA gpio #4 and SCL gpio #5 pins like on a Feather & PCA9685 FeatherWing):

```
import machine
i2c = machine.I2C(scl=machine.Pin(5), sda=machine.Pin(4))
```

Then import the PCA9685 module as follows:

```
import pca9685
```

LED & Servo Control

At this point you're ready to use the PCA9685 module to dim LEDs and controls servos. Using the module with MicroPython is exactly the same as with CircuitPython so check out the CircuitPython usage section (https://adafru.it/zex) to see details on using the board.