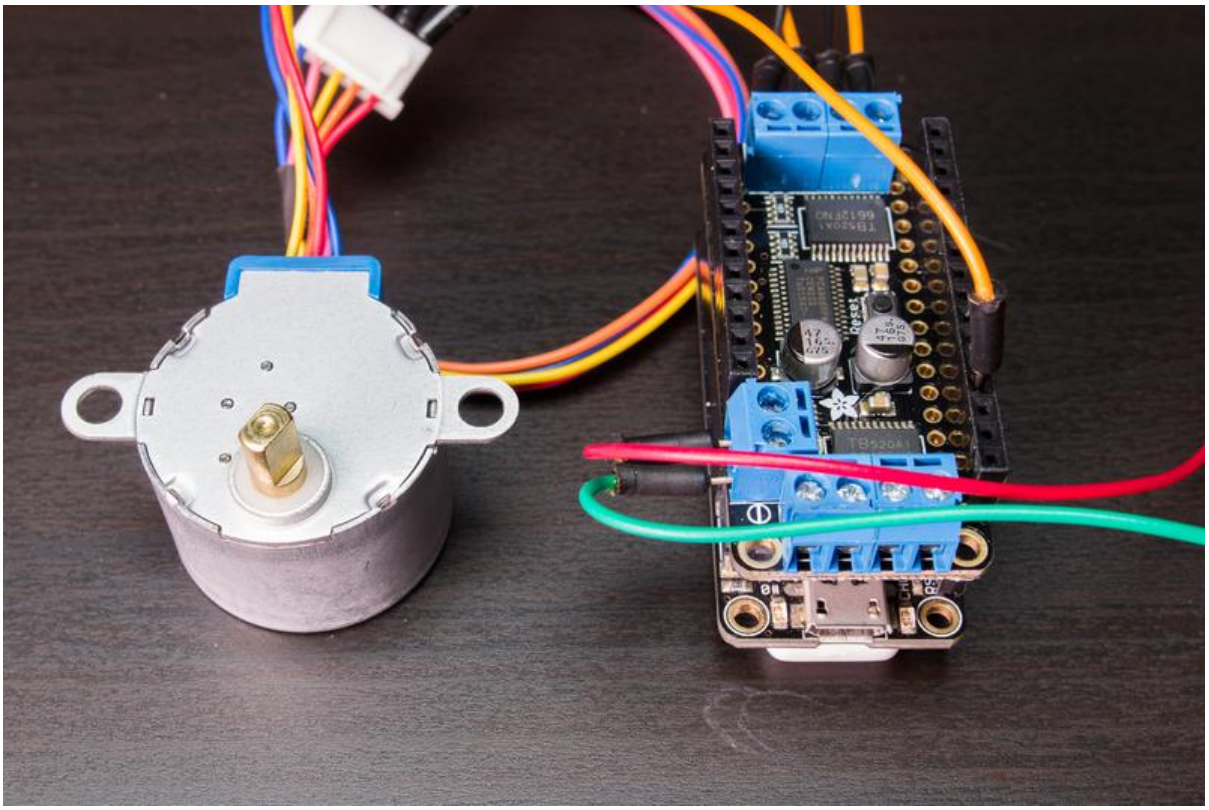
CircuitPython Hardware: PCA9685 DC Motor & Stepper Driver

Created by Tony DiCola



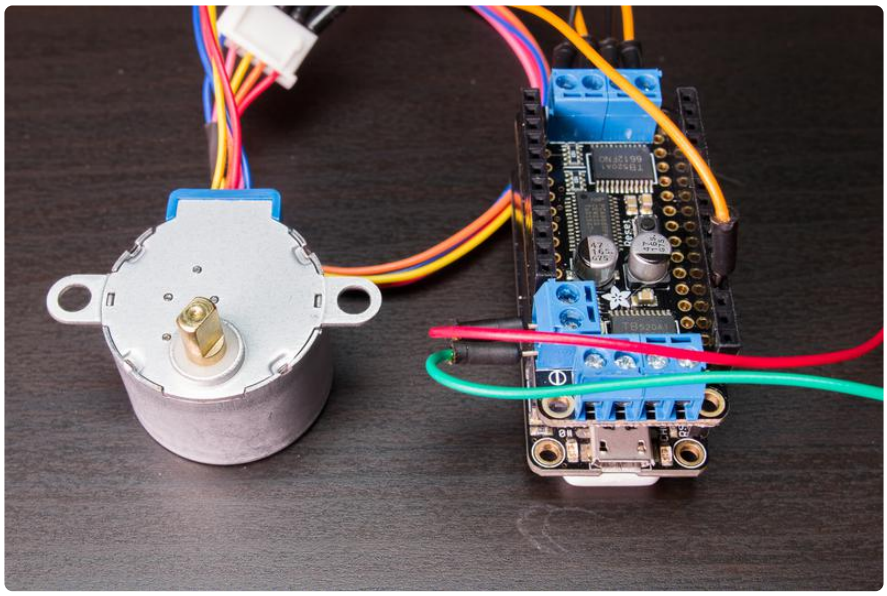
https://learn.adafruit.com/micropython-hardware-pca9685-dc-motor-and-stepper-driver

Last updated on 2021-11-15 06:48:51 PM EST

Table of Contents

Overview

The examples in this guide are no longer supported. Check out the PCA9685 guide if you're using the breakout: https://learn.adafruit.com/16-channel-pwm-servo-driver or the FeatherWing guide if you're using the Wing: https://learn.adafruit.com/adafruit-stepper-dc-motor-featherwing/circuitpython



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.

Motors make the world spin around, and now you can easily control motors with CircuitPython and the PCA9685 DC Motor & Stepper driver (https://adafru.it/sci)! Simple DC motors can moved forwards and backwards, perfect for moving the wheels on a robot or vehicle. Stepper motors can precisely move in small increments, like moving the nozzle of a 3D printer up and down with millimeter accuracy. Using a simple CircuitPython or MicroPython library and the PCA9685 DC Motor & Stepper driver board you can drive both DC motors and stepper motors and add movement to any project.

For more background check out these guides on DC and stepper motors:

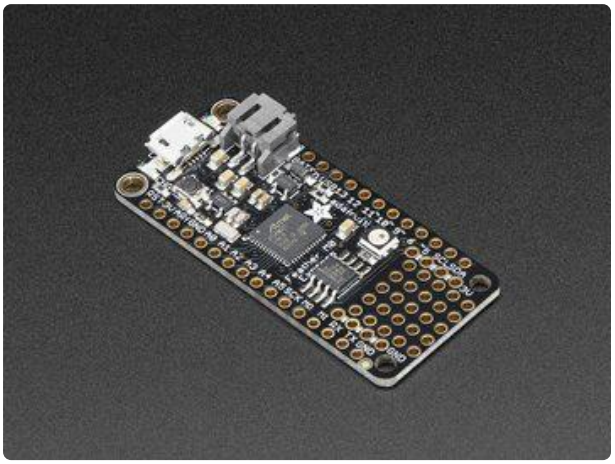
- Adafruit Motor Selection Guide (https://adafru.it/scj)
- All About Stepper Motors (https://adafru.it/sck)

Hardware

The examples in this guide are no longer supported. Check out the PCA9685 guide if you're using the breakout: https://learn.adafruit.com/16-channel-pwm-servo-driver or the FeatherWing guide if you're using the Wing: https://learn.adafruit.com/adafruit-stepper-dc-motor-featherwing/circuitpython

Parts

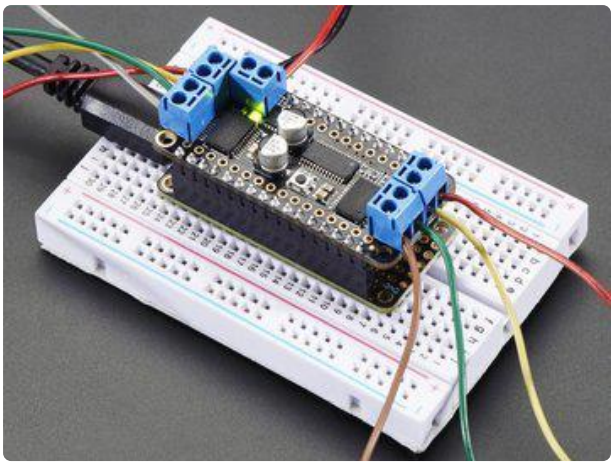
You'll need the following parts to follow this guide:



CircuitPython board. This guide focuses on the ESP8266 (https://adafru.it/n6A) and Feather M0/SAMD21-based boards (http://adafru.it/2772), but any CircuitPython board that supports I2C should work.

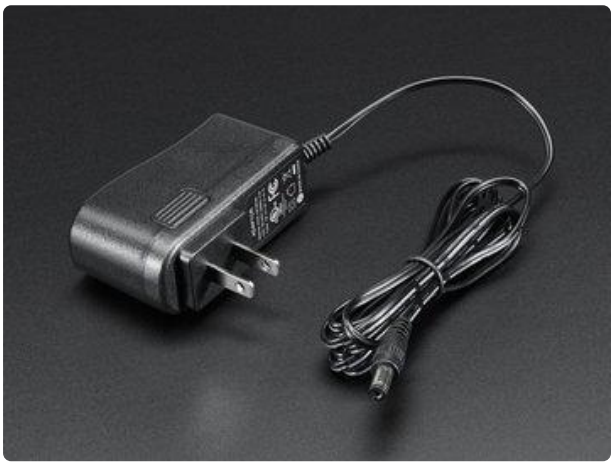
If your board doesn't come with CircuitPython running on it already then check out your board's guide for how to load CircuitPython firmware. For example the Feather M0 express guide (https://adafru.it/wbv) is a good reference.

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 DC Motor & Stepper Driver Board. If you're using a Feather the DC Motor & Stepper FeatherWing (https://adafru.it/sci) is the perfect option. The DC Motor & Stepper Arduino shield (http://adafru.it/1438) is another option that can be used with an Arduino form-factor board or on its own as a breakout for other boards.

For DC motors that are 'noisy' you might need to solder on small 0.1 microfarad decoupling capacitors, see this note from the motor guide (https://adafru.it/scl).



Power Supply. To power motors or steppers you need an external power supply. Motors can pull a lot of power and could damage your board if powered directly from it! Check your motor or stepper for the exact voltage that it requires--some motors only need 5V while others might need up to 12V.

DC or Stepper Motors. You'll want DC motors which can move continuously at different speeds, or stepper motors which can move in precise increments. Check out the motor selection guide (https://adafru.it/scm) for more details on DC and stepper 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.

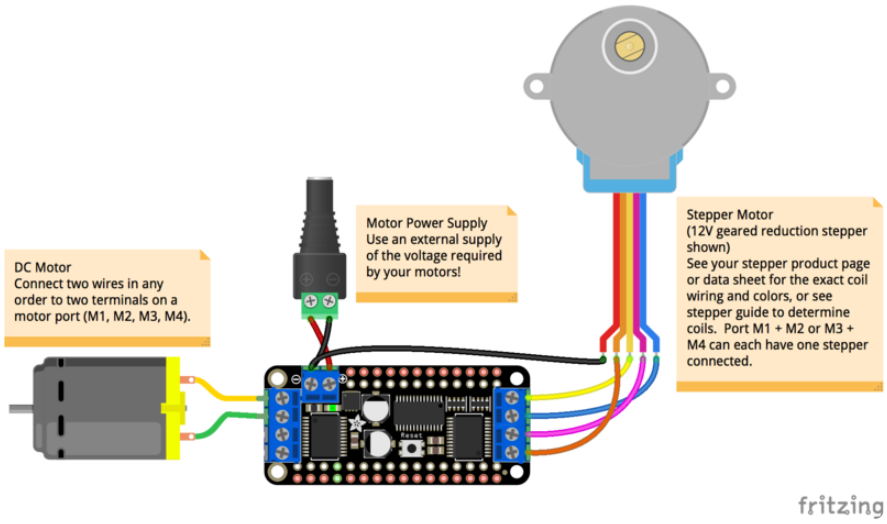
Wiring

The PCA9685 DC Motor & Stepper boards only come in FeatherWing (https://adafru.it/sci) and Arduino shield (https://adafru.it/scn) form factors. This means you don't need to do any special wiring, just slide the wing or shield onto your board and you're ready to go!

To connect a DC motor to the board see the using a DC motor section (https://adafru.it/scn) of the PCA9685 DC Motor & Stepper driver guide. Usually you just need to connect the two wires of the motor to the two terminals of an M1, M2, M3, M4 terminal block on the board.

To connect a stepper motor to the board see the using a stepper motor section (https://adafru.it/sco) of the PCA9685 DC Motor & Stepper driver guide. If you don't have product info on your stepper motor you might need to determine which wires go to which coils in the motor. See the mentioned guide page for details on determining these coils!

See an example of wiring a DC motor and the 12V geared reduction stepper (https://adafru.it/scp) to a board below:



Fritzing Source

https://adafru.it/zet

CircuitPython

The examples in this guide are no longer supported. Check out the PCA9685 guide if you're using the breakout: https://learn.adafruit.com/16-channel-pwm-servo-driver or the FeatherWing guide if you're using the Wing: https://learn.adafruit.com/adafruit-stepper-dc-motor-featherwing/circuitpython

This guide is for version 3.0.0 of the PCA9685 library. Make sure to use a bundle from 20180110 or later.

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) (version 20180110 or later).

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 and motors 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).

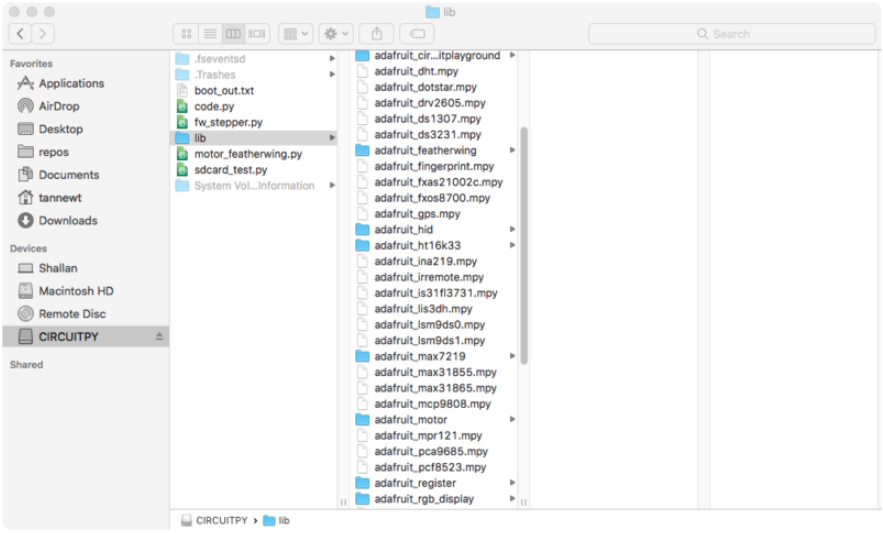
Remember for non-express boards like the Trinket M0, Gemma M0, and Feather/Metro M0 basic you'll need to manually install the necessary libraries from the bundle:

- 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_motor, and adafruit_register 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 DC Motors and Steppers 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 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)
```

DC Motors

After initializing the I2C interface you need to import and initialize the PCA9685 class to use it in your own code. We'll use it as our PWM source.

We provide the I2C address here explicitly because the Adafruit Motor FeatherWing (https://adafru.it/sci) adjusts its default address to 0x60 so it doesn't conflict with the Adafruit Servo FeatherWing (https://adafru.it/w0A).

Setting the frequency to 1600 helps smooth the motor control.

```
import adafruit_pca9685
pca = adafruit_pca9685.PCA9685(i2c, address=0x60)
pca.frequency = 1600
```

Next, we'll import and initialize the DCMotor class from adafruit_motor. This class uses the pwm channels to control the throttle of the motor. Think of it as the gas pedal from a car.

To control the motor, DCMotor needs two PWM channels, one for each wire from the motor. These should never be connected directly to PCA9685 or microcontroller because DCMotors use lots of current. Instead, use a motor driver chip, such as the TB6612 (https://adafru.it/sdc), to connect to the motor. The Motor FeatherWing has these built in. This chip is a little weird because it provides a third PWM pin marked PWMX in addition to XIN1 and XIN2. They make this available in case the controller can only PWM on a single pin. Since we can PWM two pins, we hold PWMX high.

Keeping track of which three channels of the PCA9685 to use can be tricky. If you are using the FeatherWing use the MotorFeatherWing class in the Adafruit CircuitPython FeatherWing library to make your life easier. If not, here is the convention for many motor related shields:

- Motor 1 is channels 10 and 9 with 8 held high.
- Motor 2 is channels 11 and 12 with 13 held high.
- Motor 3 is channels 4 and 3 with 2 held high.
- Motor 4 is channels 5 and 6 with 7 held high.

We'll use motor 1 in our example:

```
from adafruit_motor import motor
pwm_channel = pca.channels[8]
channel1 = pca.channels[10]
channel2 = pca.channels[9]

pwm_channel.duty_cycle = 0xffff # hold high
motor1 = motor.DCMotor(channel1, channel2)
```

Note: For small DC motors like sold in the shop you might run into problems with electrical noise they generate and erratic behavior on your board. The SAMD21

Feather M0 boards in particular have been susceptible to this issue. If you see erratic behavior like the motor not spinning or the board resetting at high motor speeds this is likely the problem. See this motor guide FAQ page for information on capacitors you can solder to the motor to reduce noise (https://adafru.it/scl).

Now to move a motor you can set the throttle attribute. We don't call it speed because it doesn't correlate to a particular number of revolutions per minute (RPM). RPM depends on the motor and the voltage which is unknown.

For example to drive motor M1 forward at a full speed you set it to 1.0:

```
motor1.throttle = 1.0
```

To run the motor at half throttle forward use a decimal:

```
motor1.throttle = 0.5
```

Or to reverse the direction use a negative throttle:

```
motor1.throttle = -0.5
```

You can stop the motor with a throttle of zero:

```
motor1.throttle = 0
```

To let the motor coast and then spin freely set throttle to None.

```
motor1.throttle = None
```

That's all there is to controlling DC motors with CircuitPython! With DC motors you can build fun moving projects like robots or remote controlled cars that glide around with ease.

Stepper Motors

To control stepper motors you'll need to import the stepper module from adafruit_motor and create an instance of the Stepper class inside of it. Unipolar and Bipolar stepper motors function by PWMing four wires in a particular sequence.

Like DCMotors, stepper motors use a lot of current to function and therefore also need a driver chip. Again, we'll need to hold the pwm pins high for each pair of control wires.

Keeping track of which six channels of the PCA9685 to use can be tricky. If you are using the FeatherWing use the MotorFeatherWing class in the Adafruit CircuitPython FeatherWing library to make your life easier. If not, the convention for many stepper related shields is:

- Stepper 1 is a combination of Motor 1 and Motor 2 as above.
- Stepper 2 is a combination of Motor 3 and Motor 4 as above.

```
from adafruit_motor import stepper
# First part of Stepper 1 is Motor 1 above.
pwma = pca.channels[8]
ain1 = pca.channels[10]
ain2 = pca.channels[9]
# Second part of Stepper 1 is Motor 2 above.
pwmb = pca.channels[13]
bin1 = pca.channels[11]
bin2 = pca.channels[12]
stepper1 = stepper.Stepper(ain1, ain2, bin1, bin2)

# Hold PWM pins high for TB6612 driver
pwma.duty_cycle = 0xffff
pwmb.duty_cycle = 0xffff
```

Now you can call the onestep function to move the stepper one 'step'. This function takes two keyword arguments:

- Direction, this should be the constant value stepper.FORWARD or stepper.BACKWARD.
- Style, this should be one of the values:
    - stepper.SINGLE for a full step rotation to a position where one single coil is powered
    - stepper.DOUBLE for a full step rotation to position where two coils are powered providing more torque
    - stepper.INTERLEAVED for a half step rotation interleaving single and double coil positions and torque
    - stepper.MICROSTEP for a microstep rotation to a position where two coils are partially active.

A single coil forward step is default:

```
stepper1.onestep()
```

The function returns the current step 'position' in microsteps which can be handy to understand how far the stepper has moved, or you can ignore the result.

To take a double-coil step backward call:

```
stepper1.onestep(direction=stepper.BACKWARD, style=stepper.DOUBLE)
```

You can even use a loop to continuously call onestep and move the stepper, for example a loop of 200 microsteps forward for smooth movement:

```
for i in range(200):
    stepper1.onestep(style=stepper.MICROSTEP)
```

That's all there is to controlling a stepper motor from CircuitPython! Steppers are handy motors for when you need smooth or precise control of something--for example 3D printers and CNC machines use steppers to precisely move tools around surfaces.

MicroPython

The examples in this guide are no longer supported. Check out the PCA9685 guide if you're using the breakout: https://learn.adafruit.com/16-channel-pwm-servo-driver or the FeatherWing guide if you're using the Wing: https://learn.adafruit.com/adafruit-stepper-dc-motor-featherwing/circuitpython

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)

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 stepper and motor modules as follows:

```
import stepper
import motor
```

DC & Stepper Motor Control

At this point you're ready to use the PCA9685 module to control DC and stepper motors. Using the module with MicroPython is exactly the same as with CircuitPython so check out the CircuitPython usage section (https://adafru.it/zeu) to see details on using the board.