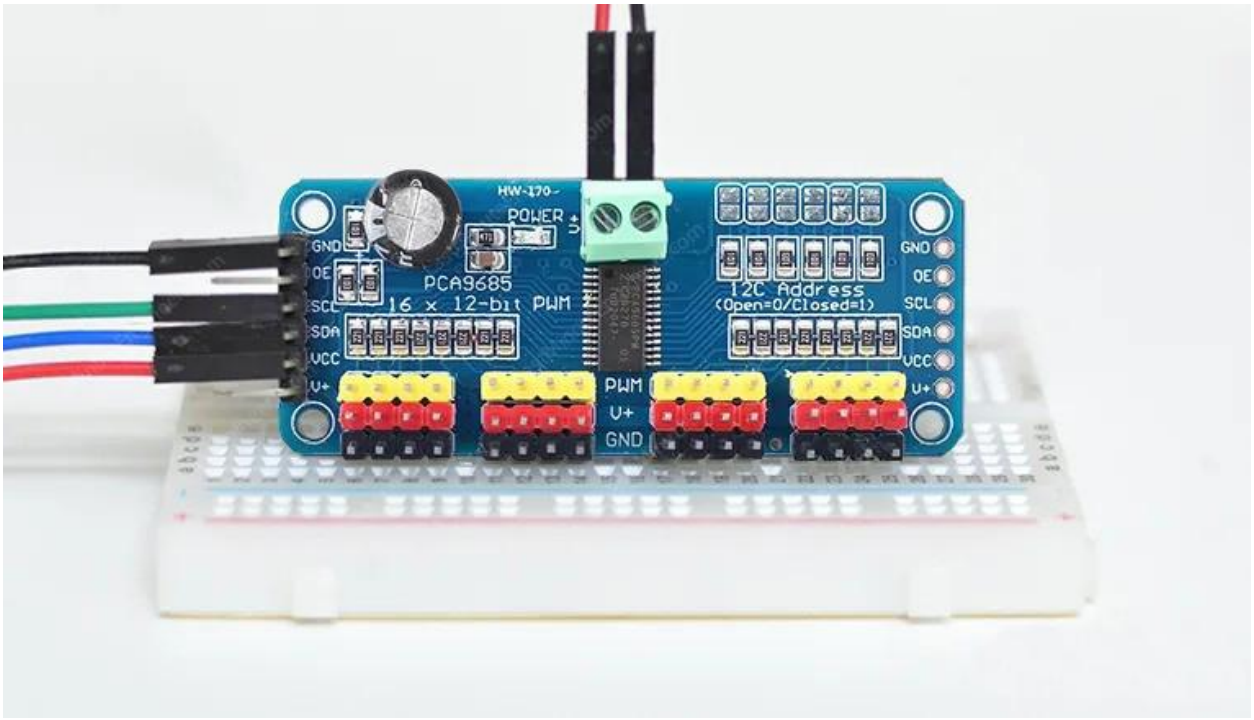
How to Control Multiple Servo Motors with a PCA9685 Module and Arduino



Sometimes you might want to connect several servo motors to your Arduino for a project. However, directly connecting multiple servos to your Arduino has its limitations. Firstly, you'll quickly run out of available PWM pins on the Arduino, especially if your project includes other components that also require PWM control. Secondly, generating precise PWM signals for the servos can consume some of your Arduino's processing power.

A better solution is to use a separate servo driver board, such as the PCA9685 module. This module allows you to drive up to 16 servo motors using just two pins on your Arduino. What's more, you can chain up to 62 of these modules to control up to 992 servo motors, all while using only those two pins! And most

importantly, the PCA9685 chip handles the generation of the PWM signals, which frees up your Arduino's processing power for other important tasks.

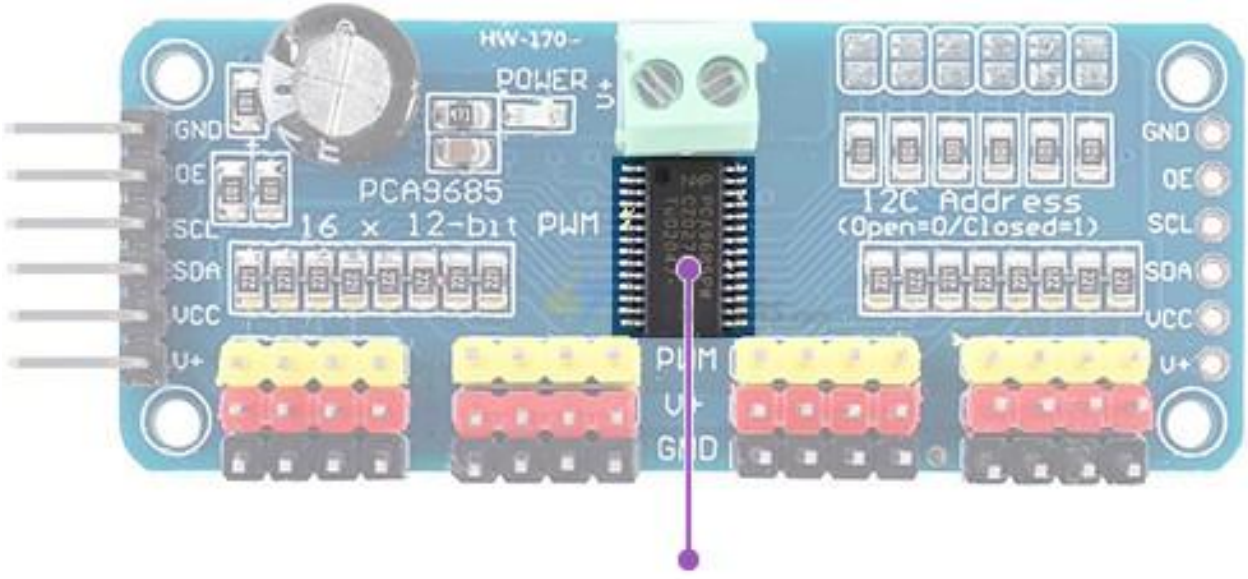
Isn't that awesome? Let's learn how to connect and use a PCA9685 module with your Arduino.

Hardware Overview

The PCA9685 module is a popular breakout board designed primarily for driving multiple servo motors. Here's a detailed overview of its various components:

PCA9685 Chip

At the heart of the module lies the PCA9685—A 16-channel, 12-bit PWM I2C-bus controlled LED/servo motor driver developed by NXP Semiconductors.



PCA9685 Chip

This chip is responsible for generating the PWM signals that control the devices (in our case, servo motors) connected to the module. It features 16 independent PWM channels that means you can control up to 16 servo motors simultaneously.

Each channel’s frequency and duty cycle can be individually adjusted, with the frequency range spanning from about 24 Hz to approximately 1526 Hz.

What’s even more impressive is that you can chain up to 62 of these modules to control up to 992 servo motors.

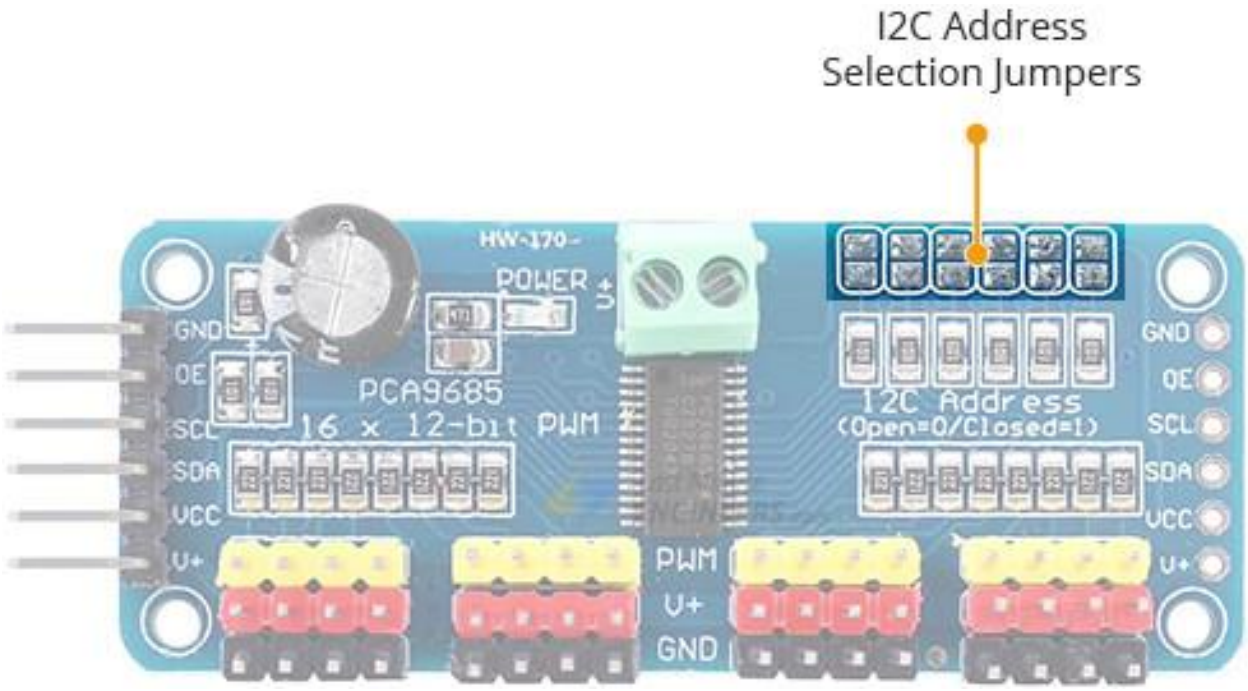
The chip communicates with a microcontroller (like an Arduino) through the I2C protocol, using only two pins (SDA and SCL). Once you send a simple I2C command to set a servo to a desired position, the PCA9685 handles all the necessary updates to maintain that position. This significantly reduces the workload on your microcontroller, freeing it up to focus on other important tasks within your project.

Additionally, the PCA9685 has an internal oscillator, so it doesn’t require an external clock source to generate the PWM signals.

For more information on PCA9685, please refer to the datasheet.

I2C Address Selection Jumpers

The PCA9685 module includes I2C address selection jumpers in the top right corner. These jumpers are essential for assigning a unique I2C address to each PCA9685 module when you connect multiple modules to the same I2C bus. This prevents conflicts and ensures proper communication.



By default, a PCA9685 module without any jumpers connected has a base address of 0x40. The binary address you set using the jumpers is then added to this base I2C address. For example, if you bridge the A0 solder pad, the address becomes 0x41. If you bridge A1 instead, the address becomes 0x42. Bridging both A0 and A1 results in an address of 0x43. This system allows you to configure the unique addresses necessary for controlling multiple PCA9685 modules simultaneously.

Refer to the image below for a visual representation of different jumper combinations and their corresponding I2C addresses.

= 0x40

= 0x41

= 0x42

= 0x43

= 0x44

= 0x45

= 0x46

= 0x47

= 0x48

= 0x49

= 0x4A

= 0x4B

= 0x4C

= 0x4D

= 0x4E

= 0x4F

= 0x50

= 0x51

= 0x52

= 0x53

= 0x54

= 0x55

= 0x56

= 0x57

= 0x58

= 0x59

= 0x5A

= 0x5B

= 0x5C

= 0x5D

= 0x5E

= 0x5F

= 0x60

= 0x61

= 0x62

= 0x63

= 0x64

= 0x65

= 0x66

= 0x67

= 0x68

= 0x69

= 0x6A

= 0x6B

= 0x6C

= 0x4D

= 0x6E

= 0x6F

= 0x70

= 0x71

= 0x72

= 0x73

= 0x74

= 0x75

= 0x76

= 0x77

= 0x78

= 0x79

= 0x7A

= 0x7B

= 0x7C

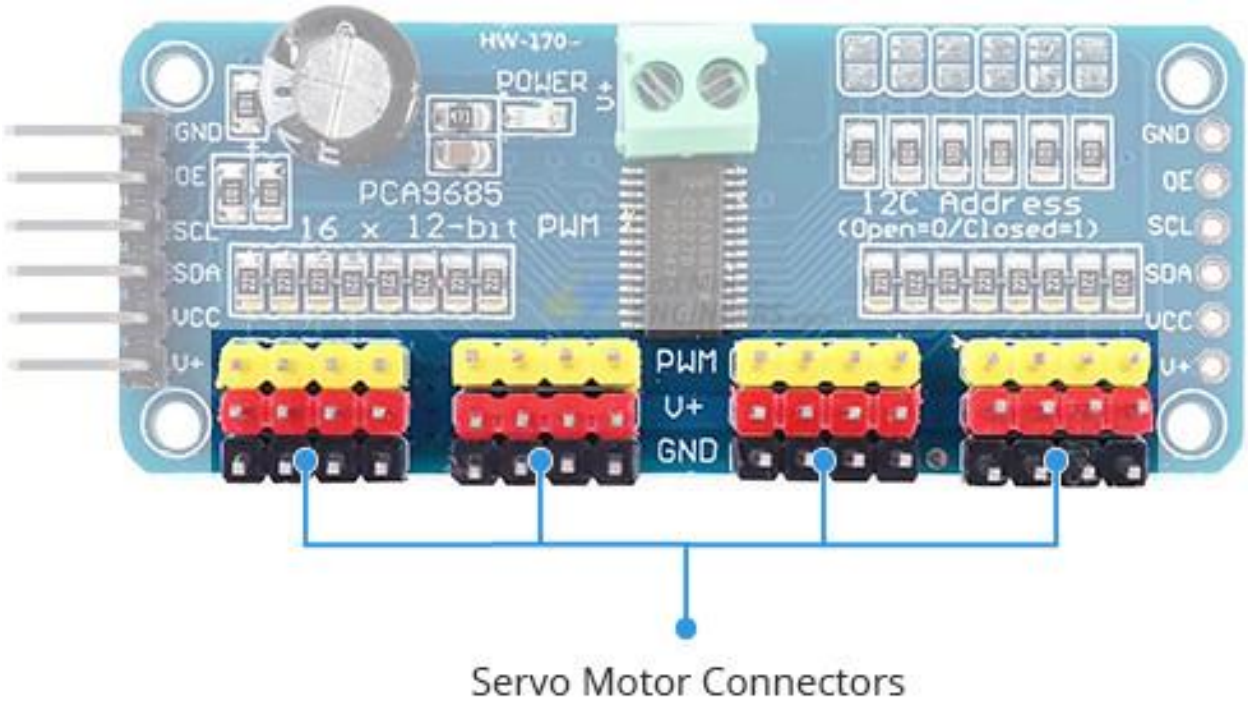
= 0x7D

= 0x7E

= 0x7F

Servo Motor Connectors

The module includes 16 sets of three-pin connectors, designed for easy connection of your servo motors. Each set provides a ground pin, a power pin, and a PWM output pin, corresponding to one of the 16 available channels on the module.

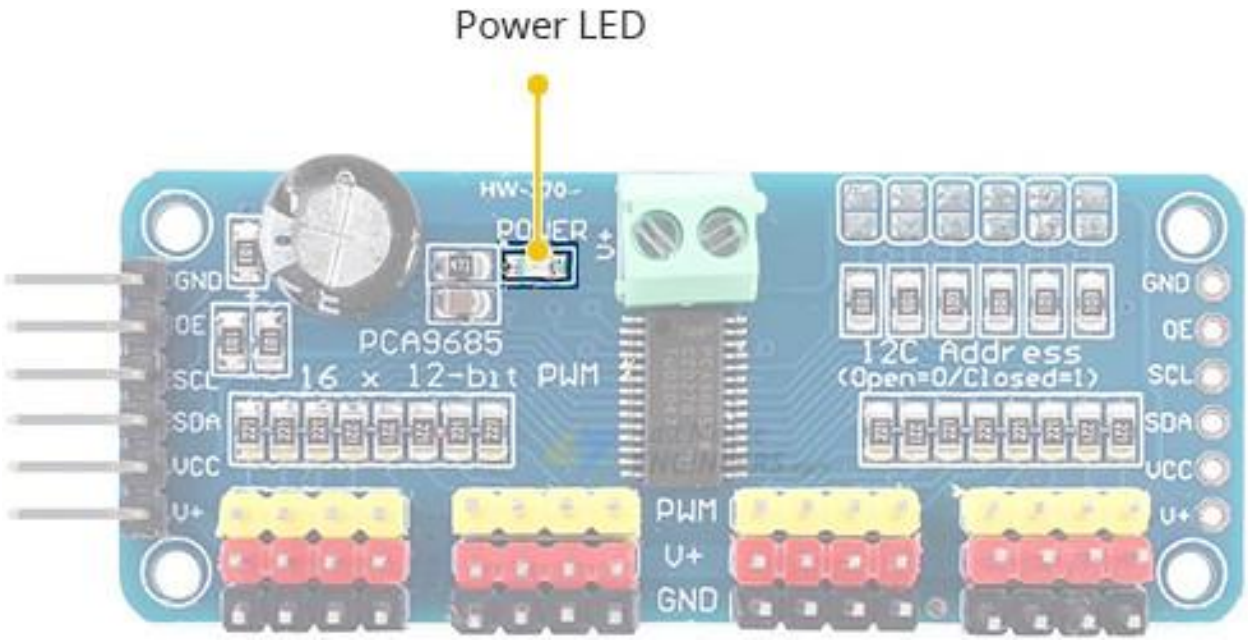


Servo Motor Connectors

While each PWM channel operates independently, it’s important to remember that all 16 channels must have the same PWM frequency.

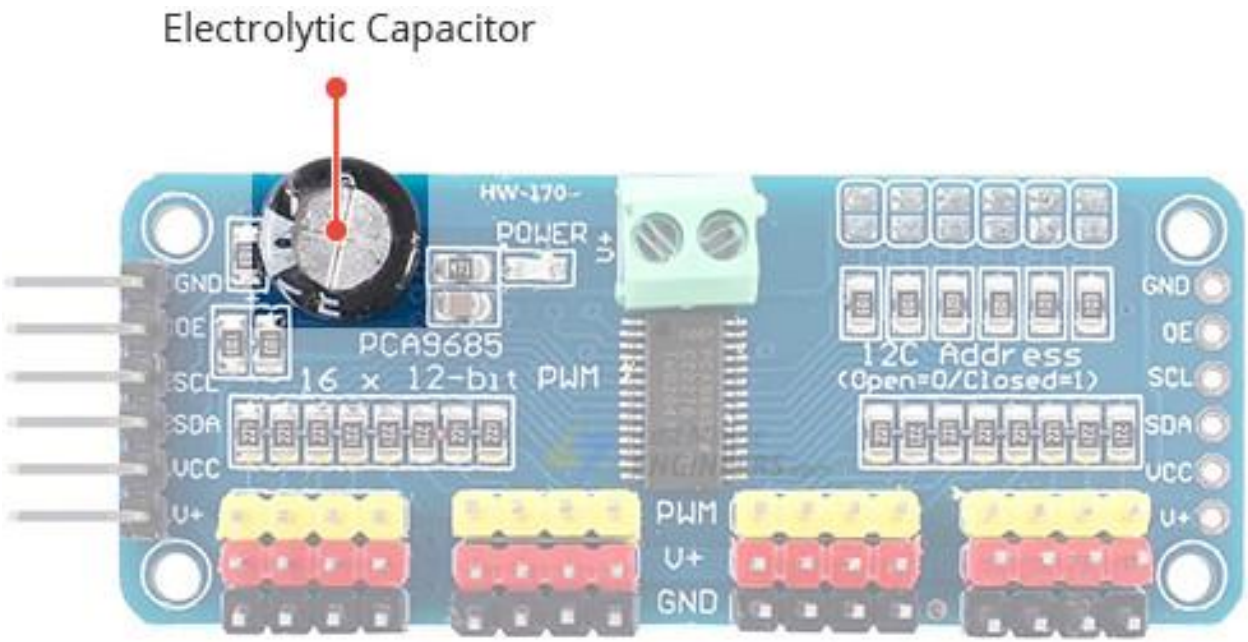
LED Indicator

There’s an LED indicator on the module to let you know that the module is powered up.



Big Electrolytic Capacitor

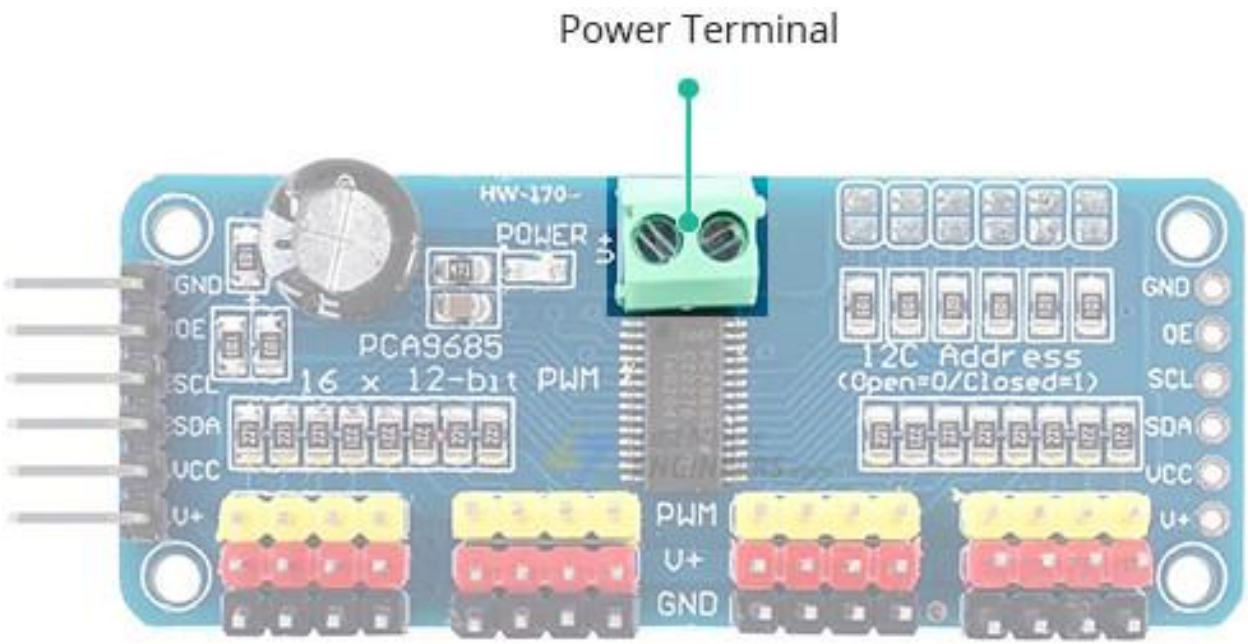
Motors have a tendency to draw higher currents, which can create fluctuations in the power supply. That's why the PCA9685 module includes a large electrolytic capacitor.



This capacitor acts as a power supply filter and stabilizer, smoothing out voltage fluctuations and spikes. This ensures that both the PCA9685 chip and connected devices, especially the motors, operate reliably and without issues.

2-Pin Screw Terminal for Power

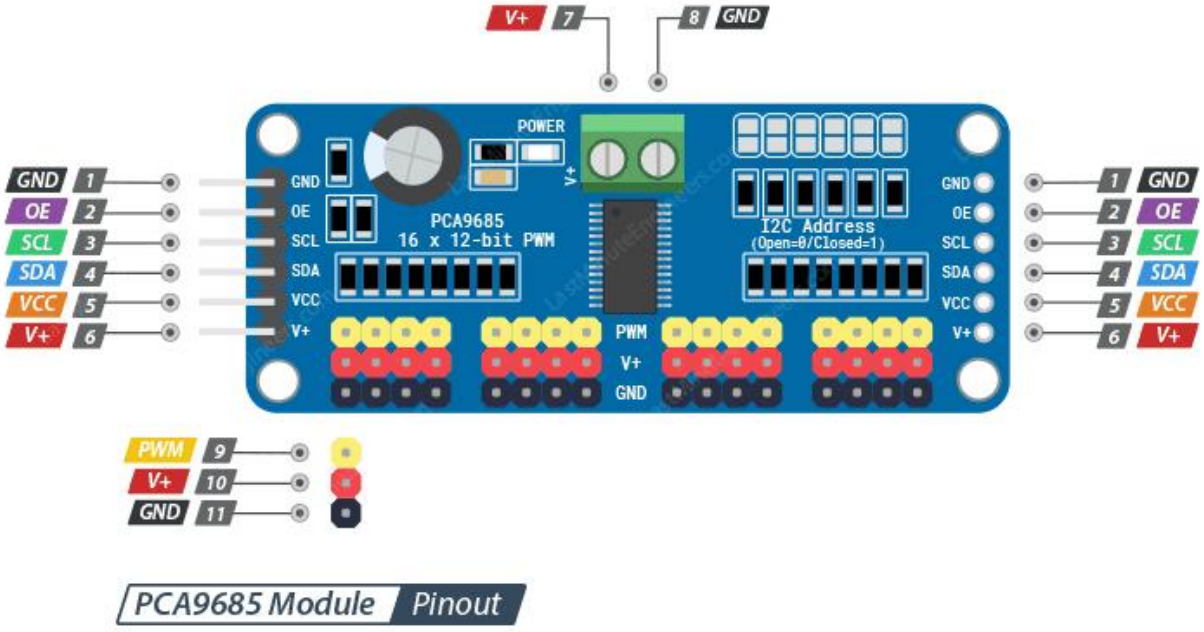
The module includes a two-pin screw terminal for connecting an external power source.



This power input is necessary because driving multiple servos requires more current than what a microcontroller can supply. By using the dedicated power input on the module, you ensure that your servos have the necessary power to function reliably.

PCA9685 Module Pinout

Let’s have a look at the pinout:



Motor Power Input (2-Pin Screw Terminal)

This is where you connect the external power supply for driving servo motors. Here’s a breakdown of the pins:

GND is the common ground connection.

V+ supplies distributed power to the servos. It’s recommended to supply 5-6VDC to this pin.

Logic Connections (1×6 Headers)

This header is where you establish the connections between the PCA9685 module and your microcontroller (MCU). The controls are 3.3 and 5V compatible.

GND connects to the ground pin on your MCU.

OE (Output Enable) can be used to quickly disable all outputs. When this pin is pulled LOW, all outputs are enabled. If you pull it HIGH, the outputs are disabled. Since the module has a built-in pull-down resistor, the outputs are enabled by default.

SCL I2C clock pin, connect to your microcontroller's I2C clock line. Can use 3V or 5V logic, and has a weak pullup to VCC

SDA I2C data pin, connect to your microcontrollers I2C data line. Can use 3V or 5V logic, and has a weak pullup to VCC

VCC is the logic power supply pin. It typically accepts a voltage between 3.3V and 5V. Ensure it matches the logic level of your microcontroller.

V+ supplies distributed power to the servos. It's recommended to supply 5-6VDC to this pin. While you can directly connect a power supply to this pin, it's preferable to use the 2-pin terminal block. The V+ pins are useful for cascading multiple PCA9685 modules and powering all the servos from a single power supply.

The unpopulated 1×6 header on the opposite side of the module is meant for daisy-chaining multiple PCA9685 modules. Both headers have identical connections, so you can add pins to the unpopulated side and use it as the input if it's more convenient for your project's layout.

Servo Connections (16 1×3 Color Coded Headers)

The output headers are color-coded to match the standard cables supplied with servo motors for easy hookup. They are numbered 0 through 15 on the board, and each connection includes the following pins:

PWM connects to the PWM pin on the servo motor.

V+ connects to the power pin on the servo motor.

GND connects to the ground pin on the servo motor.

Connecting Servo Motors to PCA9685 Module and Interfacing with an Arduino

Let’s first connect servo motors to the PCA9685 module

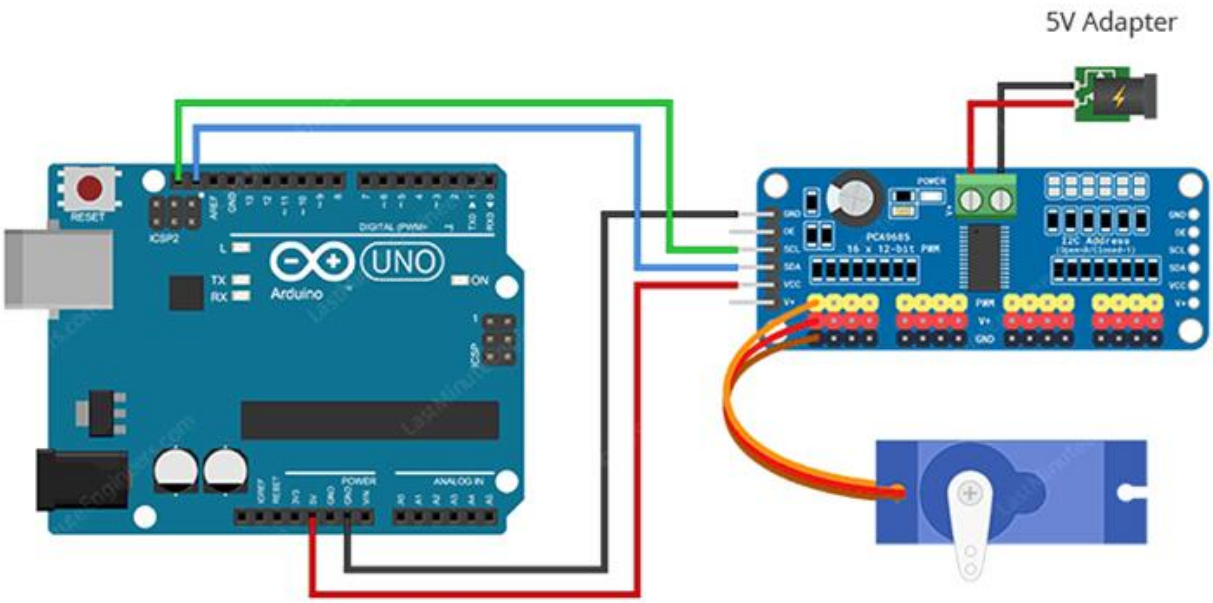
Each servo motor has three wires: brown or black for ground (GND), red for power (usually 5V), and orange, yellow, or white for the signal (PWM). Plug the servo connectors into the 3-pin headers on the PCA9685 module. Be sure to align the plug with the ground wire in the bottom row and the signal wire on the top.

To connect the PCA9685 module to your Arduino, you’ll use the I2C pins. Connect the PCA9685’s SDA pin to your Arduino’s SDA pin (A4) and the SCL pin to the Arduino’s SCL pin (A5). Next, connect the PCA9685’s GND pin to your Arduino’s GND pin and the VCC pin to the Arduino’s 5V pin.

Most servos are designed to run on about 5 or 6 volts. Keep in mind that driving multiple servos simultaneously can result in a significant current draw. Even smaller servos consume several hundred milliamps (mA) when in motion, and high-torque servos can demand over 1 amp each under load. It is not a good idea to use the Arduino 5V pin to power your servos. Electrical noise and voltage drops (“brownouts”) from excess current draw can make your Arduino behave erratically, reset, and/or overheat.

Therefore, it’s strongly recommended to use a separate power supply for your servos. Connect this power supply to the 2-pin screw terminal on the PCA9685 module.

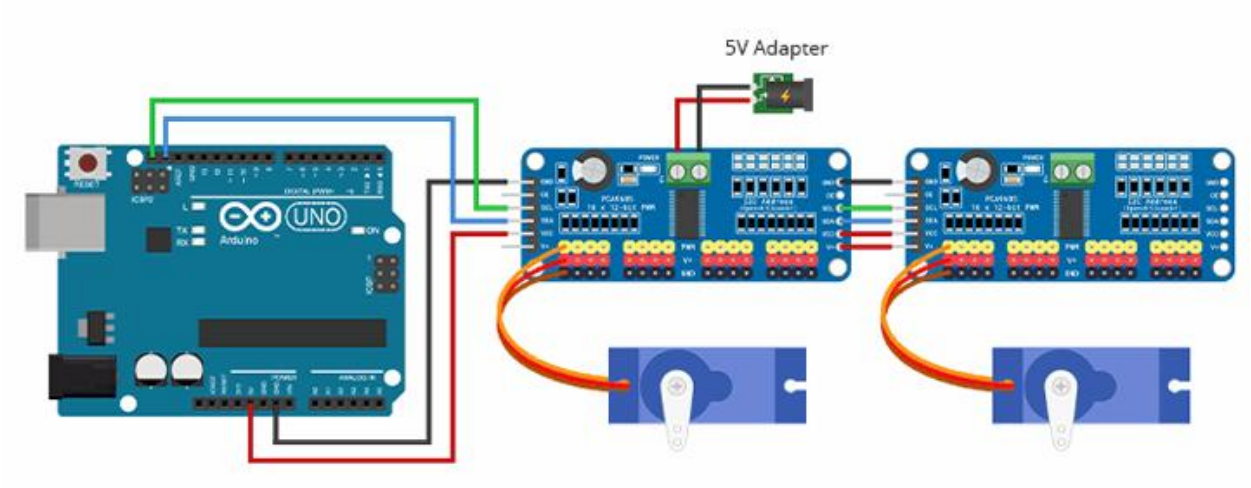
The image below shows how to build the circuit.



Chaining PCA9685 Modules

If your project requires controlling more than 16 servo motors, you can easily daisy-chain multiple PCA9685 modules. Up to 62 boards can be chained together to control up to 992 servo motors.

Connecting them is straightforward, as each board features headers on both ends. You simply need to use a 6-pin parallel cable to link one board to the next. However, it's crucial to remember to set the address jumpers on each module to a unique address. This prevents conflicts on the I2C bus, ensuring that each module can be addressed and controlled individually.

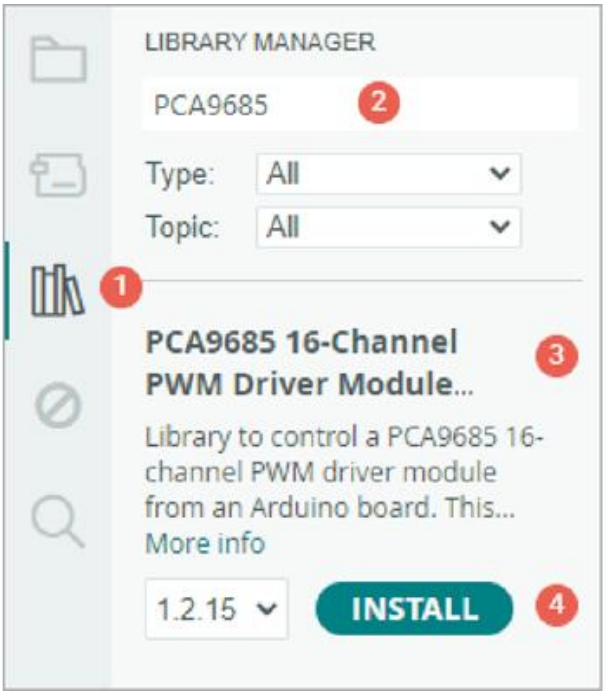

5V Adapter
UNO
Arduino

Library Installation

You'll need an Arduino library to interact with the PCA9685. The [PCA9685 library by NachtRaveVL] is a popular choice.

To install the library,

- First open your Arduino IDE program. Then click on the Library Manager icon on the left sidebar.
- Type "PCA9685" in the search box to filter your results.
- Look for the PCA9685 16-channel PWM Driver Module Library by NachtRaveVL.
- Click the Install button to add it to your Arduino IDE.



Arduino Code

Here is a simple sketch that makes two servos, connected to channels 0 and 1 of the PCA9685 module, sweep back and forth across a 180-degree range.

```
#include <Wire.h>
#include "PCA9685.h"

PCA9685 driver;

// PCA9685 outputs = 12-bit = 4096 steps
// 2.5% of 20ms = 0.5ms ; 12.5% of 20ms = 2.5ms
// 2.5% of 4096 = 102 steps; 12.5% of 4096 = 512 steps
// we will use 112 and 500 to avoid pushing the servo to its physical limits
PCA9685_ServoEval pwmServo(112, 500); // (-90deg, +90deg)

void setup() {
  Wire.begin();                 // Wire must be started first
  driver.resetDevices();        // Software resets all PCA9685 devices on Wire line
  driver.init();
  driver.setPWMFrequency(50);   // Set frequency to 50Hz
}

void loop() {
  driver.setChannelPWM(0, pwmServo.pwmForAngle(-90));
  driver.setChannelPWM(1, pwmServo.pwmForAngle(-90));
  delay(1000);
  driver.setChannelPWM(0, pwmServo.pwmForAngle(0));
  driver.setChannelPWM(1, pwmServo.pwmForAngle(0));
  delay(1000);
  driver.setChannelPWM(0, pwmServo.pwmForAngle(90));
  driver.setChannelPWM(1, pwmServo.pwmForAngle(90));
  delay(1000);
}
```

Code Explanation

The code starts by including two essential libraries:

- The `Wire.h` library enables I2C communication, which is how the Arduino will interact with the PCA9685 module.
- The `PCA9685.h` library is a custom library that simplifies the commands needed to control the PCA9685.

```
#include <Wire.h>
#include "PCA9685.h"
```

An instance of the `PCA9685` class named `driver` is then created to facilitate communication with the PCA9685 hardware.

```
PCA9685 driver;
```

Similarly, an instance of the `PCA9685_ServoEval` class named `pwmServo` is initialized with parameters 112 and 500. These parameters define the PWM pulse boundaries that correspond to -90 degrees and +90 degrees servo positions, respectively. The values are calculated based on a 20 ms cycle (50 Hz signal) typical for servo control, where 0.5 ms (~112 steps) and 2.5 ms (~500 steps) represent the pulse widths for the extreme positions of the servo.

```
// PCA9685 outputs = 12-bit = 4096 steps
// 2.5% of 20ms = 0.5ms ; 12.5% of 20ms = 2.5ms
// 2.5% of 4096 = 102 steps; 12.5% of 4096 = 512 steps
// we will use 112 and 500 to avoid pushing the servo to its physical limits
PCA9685_ServoEval pwmServo(112, 500); // (-90deg, +90deg)
```

The `setup()` function gets things ready. It starts the I2C communication, sends a signal to reset any connected PCA9685 modules, initializes the `driver` object for communicating with the PCA9685, and finally sets the PWM frequency for all channels to 50Hz, a typical frequency for controlling servo motors.

```
void setup() {
  Wire.begin();                 // Wire must be started first
  driver.resetDevices();        // Software resets all PCA9685 devices on Wire line
  driver.init();
  driver.setPWMFrequency(50);   // Set frequency to 50Hz
}
```

The main `loop()` function is where the action happens. It repeatedly sets the PWM values for channels 0 and 1 to move them through a sequence: first to -90 degrees (leftmost), then pausing for a second, then to 0 degrees (center), another pause, and finally to 90 degrees (rightmost). This cycle continues indefinitely, making the servos sweep back and forth.

```
void loop() {
  driver.setChannelPWM(0, pwmServo.pwmForAngle(-90));
  driver.setChannelPWM(1, pwmServo.pwmForAngle(-90));
  delay(1000);
  driver.setChannelPWM(0, pwmServo.pwmForAngle(0));
  driver.setChannelPWM(1, pwmServo.pwmForAngle(0));
  delay(1000);
  driver.setChannelPWM(0, pwmServo.pwmForAngle(90));
  driver.setChannelPWM(1, pwmServo.pwmForAngle(90));
  delay(1000);
}
```

Credit Amrit Prabhu