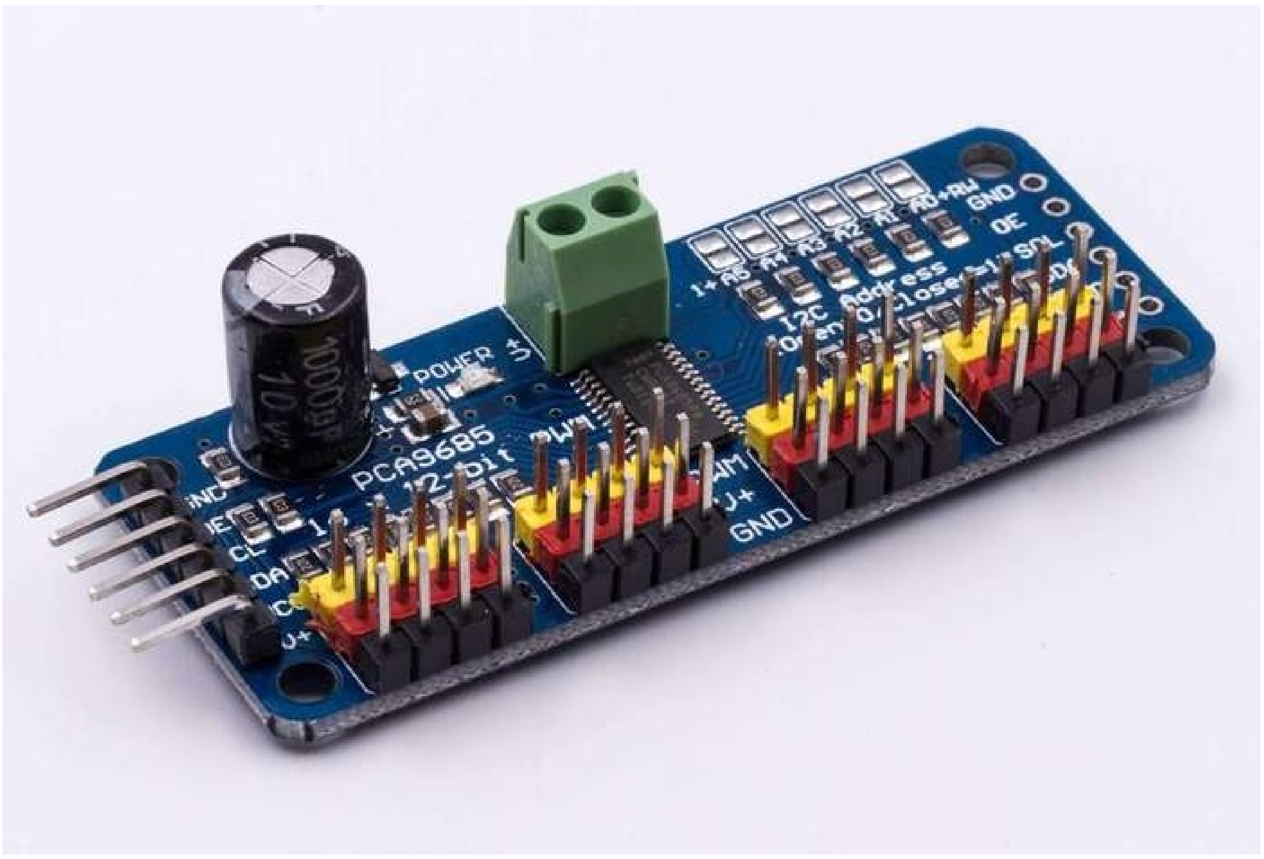
Welcome!

Thank you very much for purchasing our AZ-Delivery PCA9685 16x Servo Driver. On the following pages, we will introduce you to how to use and setup this handy device.

Have fun!



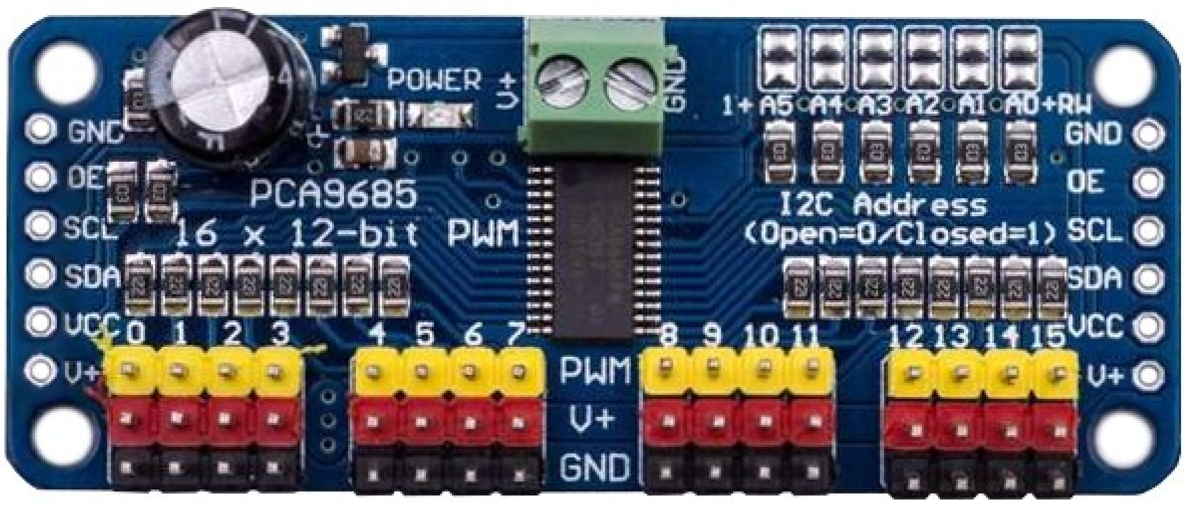
The PCA9685 module is an I2C bus controlled 16 channel PWM controller module for Arduino and Raspberry Pi applications with servo motors. Each PWM servo output has a separate internal PWM controller with 12 bit resolution (4096 steps).

The PWM output driver can be programmed to be either a drain with 25mA current sink at 5V or a totem pole with 25mA sink and 10mA current source at 5V.

The module operates with a supply voltage range from 2.3V to 5.5V and the inputs and outputs are 5V tolerant.

The servo motors can be connected up to 25mA and 5V directly to the PWM outputs of the module or controlled with external drivers and a minimum number of discrete components with higher current or voltage.

The switch-on and switch-off times are independently programmable for each of the 16 channels. The power-on reset (POR) status of all 16 PWM output pins is LOW. Six hardware address pins theoretically allow up to 62 devices on the same bus.



The module has a total of 6 connections. These are occupied starting from the top left:

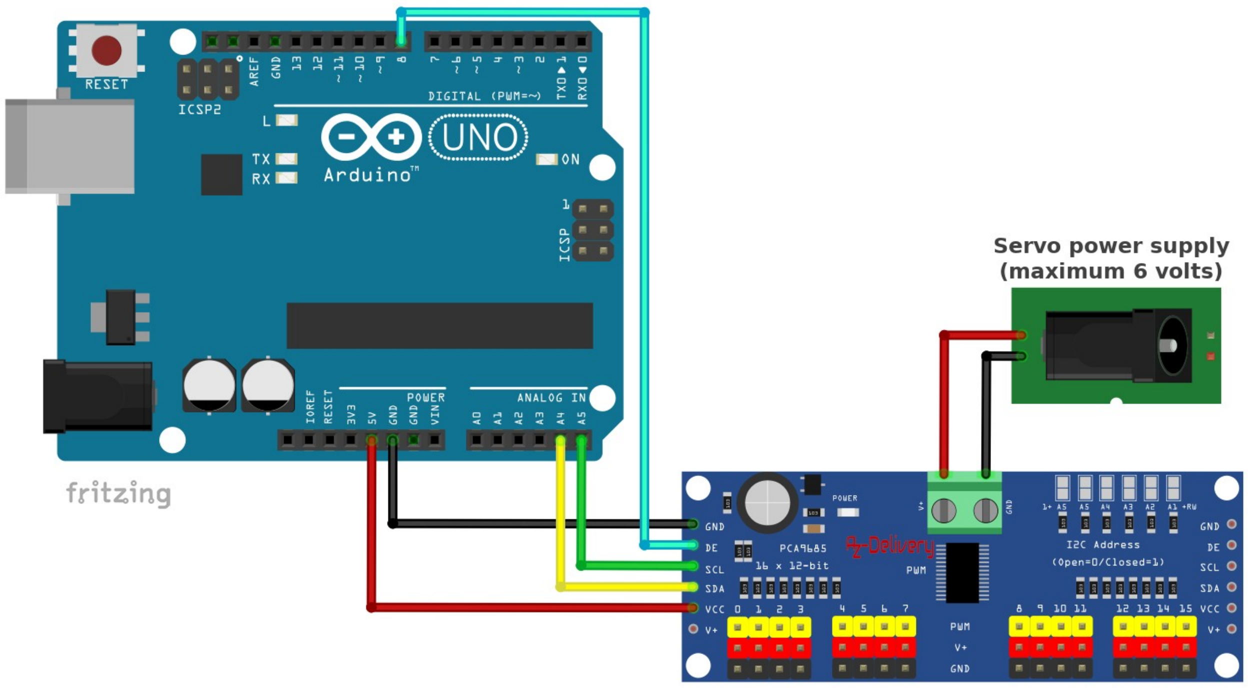
Pin	>	Description	Additional information:
1	>	GND	Ground
2	>	OE Input	Output Enable - LOW active
3	>	SCL Input	I2C Bus - Clock line
4	>	SDA Input	I2C Bus - Data line
5	>	VCC	Power supply max. 5V
6	>	V+	Power supply max. 6V

Pin 6 should remain unconnected.

Output Enable

The active LOW Output Enable input pin (OE) allows asynchronous control of the PWM outputs and can be used to set all outputs to a defined, programmable I2C bus logic state. The OE can also be used to externally "pulse width modulate" the outputs. This is useful when multiple devices need to be dimmed or flashed together using software control.

The basic circuit with the Arduino Uno is as follows:



The pin connection is:

A4	>	SDA	**Yellow wire**
A5	>	SCL	**Green wire**
8	>	OE	**Cyan wire**

PWM - basic frequency

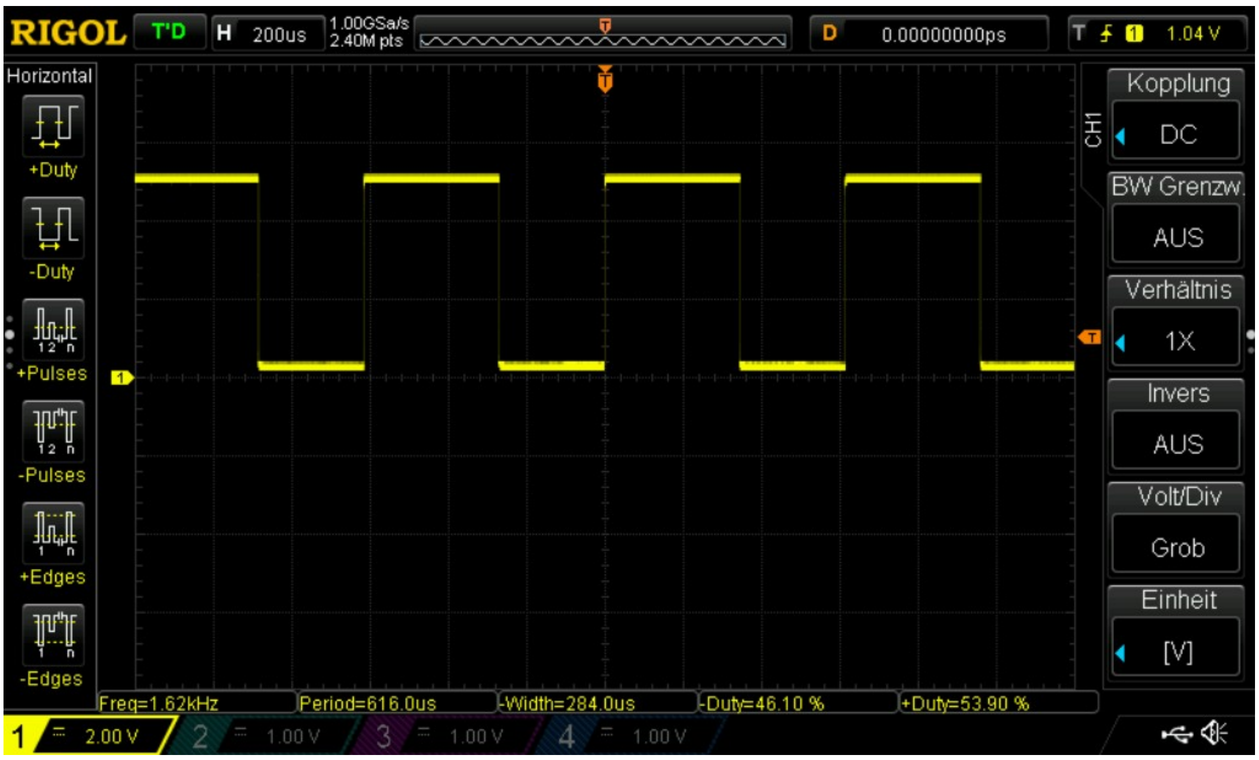
The PCA9685 has an adjustable PWM base frequency from 24Hz to 1526Hz, whereby the accuracy of the PWM base frequency is not very high, since the internal oscillator is used as clock generator. The PWM base frequency is determined using an internal programmable prescaler value written to register *0xFEh*. All outputs always work with the same PWM frequency, whereby the duty cycle can be set between 0% and 100%.

Here are a few Prescaler values to try out for yourself:

Prescaler value:	PWM base frequency:
0x03h	1,526 kHz
0x06h	1 kHz
0x0Ch	500 Hz
0x1Eh	200 Hz
0x7Ah	50 Hz
0xFFh	24 Hz

The prescaler value can be adjusted in the code in the *Set_Prescaler* function. The PWM base frequency with which your servo motor works best, can either be taken from the servo datasheet or simply tried out for yourself.

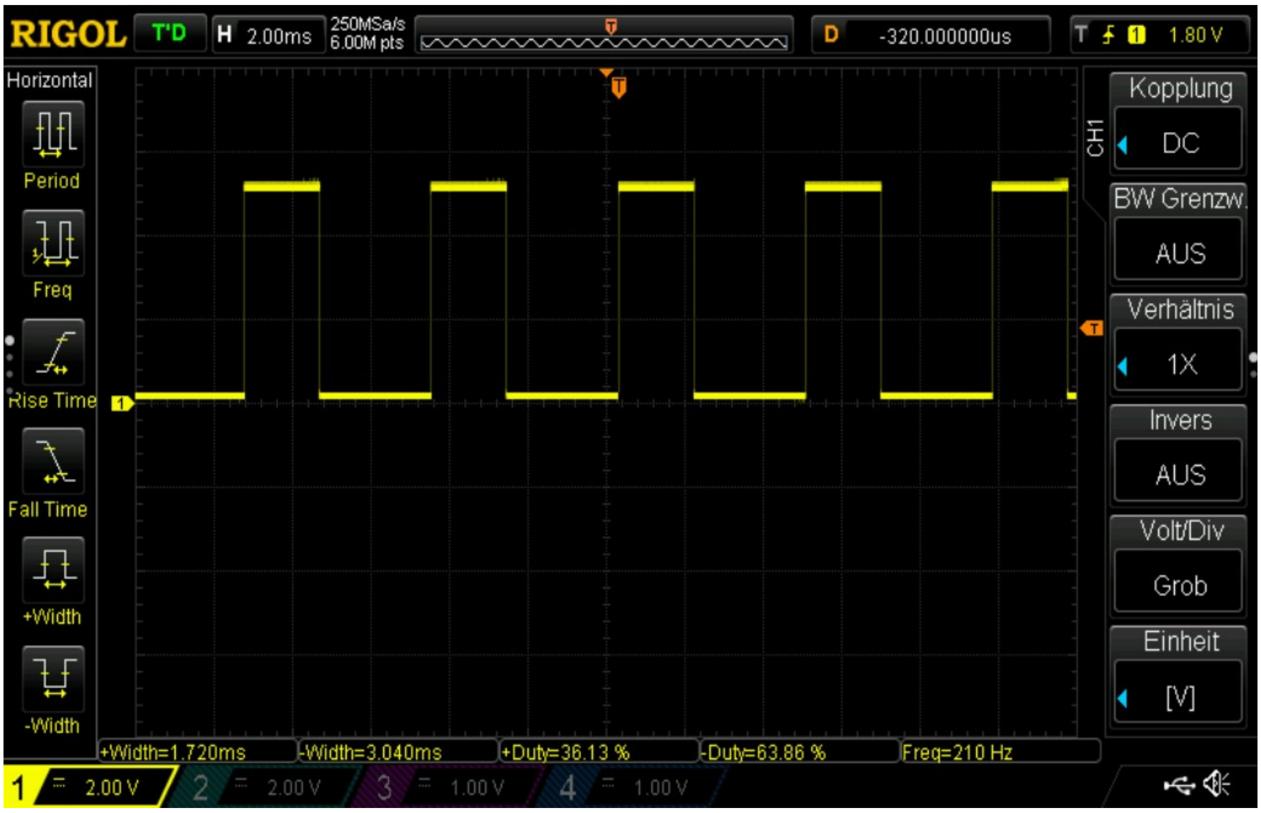
In our example sketch in this manual, we select the maximum possible PWM base frequency of 1.5kHz using the *Set_Prescaler* function. On subsequent oscilloscope images, the effect of changing the prescaler value becomes visible.


RIGOL T'D H 200us 1.00GSa/s 2.40M pts D 0.00000000ps T 1.04 V

Freq=1.62kHz Period=616.0us -Width=284.0us -Duty=46.10 % +Duty=53.90 %

1 2.00 V 2 1.00 V 3 1.00 V 4 1.00 V

Max. PWM base frequency with a prescaler value of 0x03h. (Here due to oscillator inaccuracy: 1.62 Khz)


RIGOL T'D H 2.00ms 250MSa/s 6.00M pts D -320.000000us T 1.80 V

+Width=1.720ms -Width=3.040ms +Duty=36.13 % -Duty=63.86 % Freq=210 Hz

1 2.00 V 2 2.00 V 3 1.00 V 4 1.00 V

If the Prescaler value is not set, the PWM base frequency is approx. 200Hz.

Power supply of the module / output current limitation

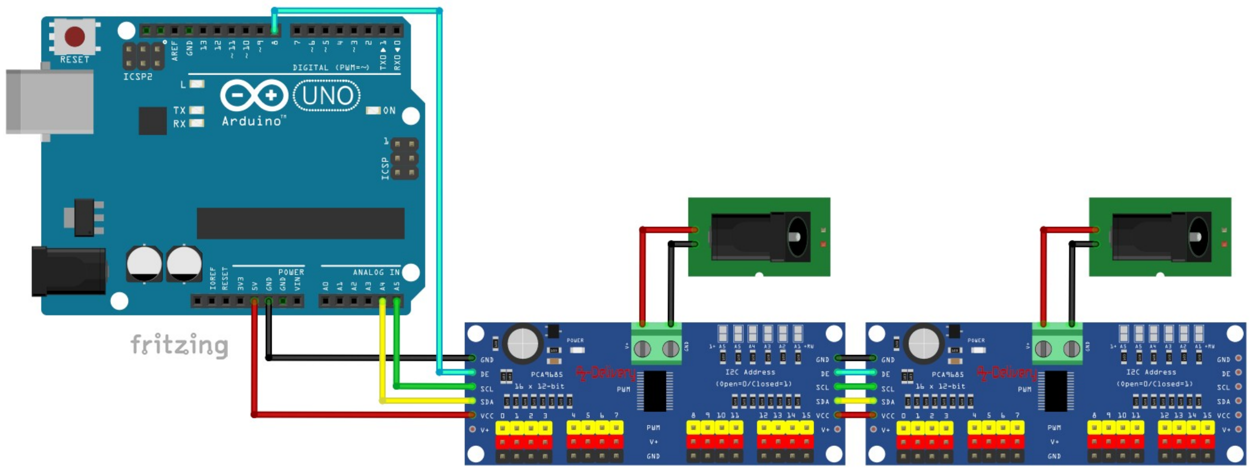
On the left connector strip of the module there are two positive supply pins and one ground pin. The positive supply pins are described with Vcc and V+. Vcc is the supply for the PWM chip itself and not for the servos!

V+ should be left unconnected, but the power supply of the servos should be done via the green power supply terminal on the top of board, with power supply of the appropriate size.

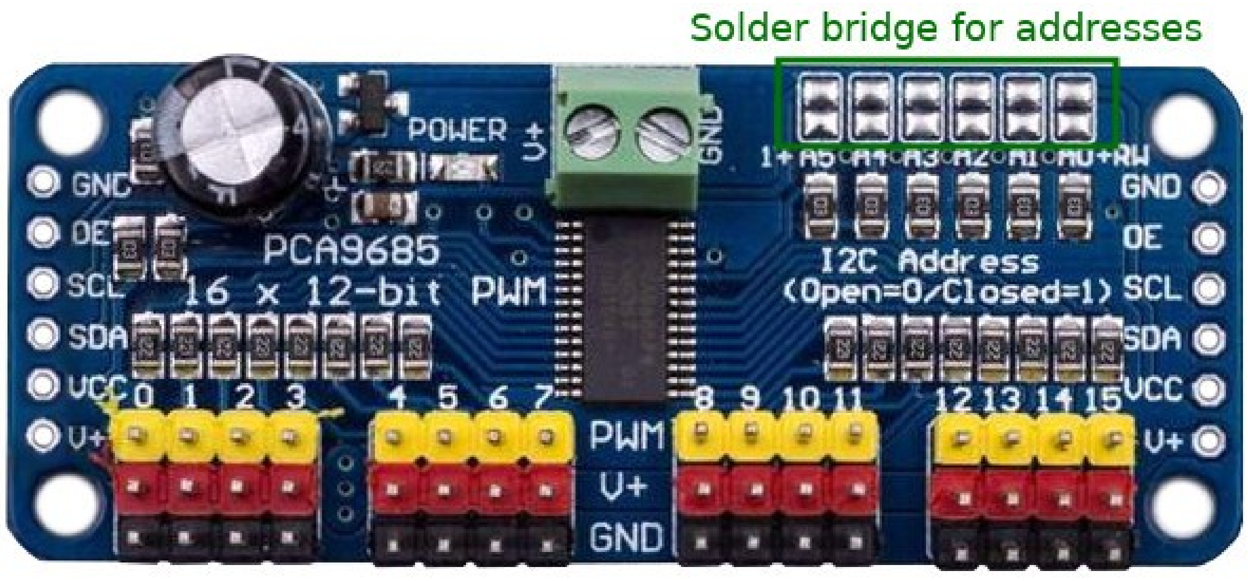
Please note that the maximum supply voltage of the servos may be at 6V. Each PWM output has a maximum positive current carrying capacity of 10mA at 5V.

Addressing several modules

If 16 outputs are not sufficient for the one project, further modules can be connected in series to the I2C bus. The wiring required for this is shown below:



It should be noted that the supply of the individual servos can no longer be taken from Arduino here at the latest, but that it is imperative that the supply voltage is externally supplied via the green terminals. One external power supply per module must be supplied.



The addressing solder bridges A0 to A5 on the module should then be connected differently for each module according to the binary code counting upwards:

PWM Module:	Soldering bridge for:	I2C address of the module:
1	A0	0x41
2	A1	0x42
3	A0 and A1	0x43
4	A2	0x44
5	A2 and A1	0x45

The control register addresses start anew with each module. This means that the basic control of channels 0 - 15 can remain the same, only the I2C address of the module must be changed. In the code this address is set by the variable *PWM_ModuleAddr*.

```
#include <Wire.h>
#define PWM_ModuleAddr 0x40 //  10000000b The last bit of the address
                            //  byte defines the operation to be performed.
#define OE_Pin 8            // Pin for Output Enable
byte Pwm_Channel;

void Set_Prescaler(int Precaler) {
  Wire.beginTransmission(PWM_ModuleAddr); // Initiate data transfer
  Wire.write(0x00);         // Select Mode 1 Register (Command Register)
  Wire.write(0x10);         // Configure SleepMode
  Wire.endTransmission();   // Communication stop - Send Stop Bit

  Wire.beginTransmission(PWM_ModuleAddr); // Initiate data transfer
  Wire.write(0xFE);         // Select PRE_SCALE register (Command Register)
  Wire.write(Precaler);
  Wire.endTransmission();   // Communication stop - Send Stop Bit

  Wire.beginTransmission(PWM_ModuleAddr); // Initiate data transfer
  Wire.write(0x00);   // Select Mode 1 Register (Command Register)
  Wire.write(0xA1);   // Configure Chip: All Call I2C addresses,
                      // Use internal clock, Allow Auto Increment Feature
  Wire.endTransmission();   // Communication stop - Send Stop Bit
}

void setup() {              // Initialization
  IpinMode(OE_Pin, OUTPUT);
  digitalWrite(OE_Pin, LOW);// Active LOW Output Activation Pin (OE)
  Wire.begin();             // Initialize I2C Bus A4 (SDA), A5 (SCL)
  Wire.beginTransmission(PWM_ModuleAddr); // Initiate data transfer
  Wire.write(0x01);         // Select Mode 2 Register (Command Register)
  Wire.write(0x04);         // Configure chip: 0x04:
                            // dead pole output 0x00: open drain output
  Wire.endTransmission();   // Communication stop - Send Stop Bit
  Set_Prescaler(0x03);      // The maximum PWM frequency is 1526 Hz
                            // if the PRE_SCALEer Regsiter
                            // Is set to "0x03h". Standard : 200 Hz
  Pwm_Channel = 0;          // Output to PWM port 0 ... 15
}
```

```
void loop() {
  // Continuously increase PWM output value at PWM (servo) Port 0
  Wire.beginTransmission(PWM_ModuleAddr);
  Wire.write(Pwm_Channel*4 + 6);    // Select PWM_Channel_0_ON_L Register
  Wire.write(0x00);                 // Value for above mentioned register
  Wire.endTransmission();

  Wire.beginTransmission(PWM_ModuleAddr);
  Wire.write(Pwm_Channel*4 + 7);    // Select PWM_Channel_0_ON_H Register
  Wire.write(0x00);                 // Value for above mentioned register
  Wire.endTransmission();

  for(int i = 0; i < 4096; i++) {
    Wire.beginTransmission(PWM_ModuleAddr);
    Wire.write(Pwm_Channel*4 + 8);     // Select PWM_Channel_0_OFF_L Register
    Wire.write((byte)i & 0xFF);        // Value for above mentioned register
    Wire.endTransmission();

    Wire.beginTransmission(PWM_ModuleAddr);
    Wire.write(Pwm_Channel*4 + 9);     // Select PWM_Channel_0_OFF_H Register
    Wire.write((i >> 8));              // Value for above mentioned
    Wire.endTransmission();

    delay(5);
  }
  delay(4000);
}
```

You've done it, you can now use and program your module for your projects.

Now it is time to learn and make the Projects on your own. You can do that with the help of many example scripts and other tutorials, which you can find on the internet.

If you are looking for the high quality products for Arduino and Raspberry Pi, AZ-Delivery Vertriebs GmbH is the right company to get them from. You will be provided with numerous application examples, full installation guides, eBooks, libraries and assistance from our technical experts.

https://az-delivery.de

Have Fun!

Impressum

https://az-delivery.de/pages/about-us