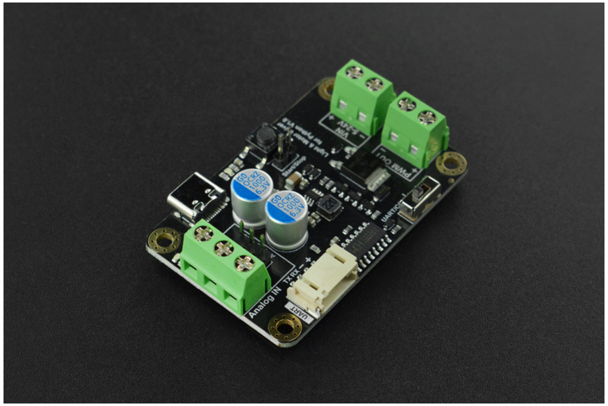
SKU:DRI0050 (https://www.dfrobot.com/product-2429.html)



(https://www.dfrobot.com/product-2429.html)

1. Introduction

This is a PWM driver board with four adjustment methods, namely, Python code direct control, PC host computer control, UART communication programming control, and potentiometer manual control. It is used in application scenarios such as DC motor speed control and light adjustment.

This product supports 5V~24V wide voltage input, and has a load capacity of 10A, which can drive a DC motor of about 50W or a LED strip light of about 5 meters. It is suitable for water pump water volume control, cooling fan speed control, light brightness adjustment, power tool transformation, motor/LED strip product testing, motor speed automation control, lighting effect automation control and other scenes.

2. Features

- Python code programming, no motherboard or adapter required. Directly connected to PC via a USB cable, simple and easy to use.
- Windows software control, no programming required, plug and play.
- UART communication control, convenient for long-term use of embedded equipment.
- External potentiometer control, manual control, no programming required.
- Standard MODBUS protocol. The protocol is public so you can flexibly customize your own control method.

- 5V~24V wide voltage support, suitable for various motors and light strips.
- Driven by high current MOS tube, load capacity of 10A

3. Applications

- Control by Python code

1. Connect the USB interface, you can directly use Python code to automatically control the motor speed and lighting effect. No additional motherboard or adapter is required.
2. Can be very conveniently used on Windows computers, Apple computers, Raspberry Pi, industrial computers or LattePanda.

- Control by Windows computer

1. When remaking a PC cooling fan, cooling water pump, etc, you can manually control the device on PC, or make it be automatically adjusted by reading the CPU temperature.
2. Because the host computer can accurately adjust the frequency and duty ratio, this driver can be used to test dc motor or light strip products, or to optimize small equipment.
3. Make a small fan whose wind speed can be automatically adjusted or controlled on the

computer. Just plug in the USB to use.

- Control by programming mainboard

1. Create a project with special effects, or turn it on at a specific time, or adjust lighting effects, or use it with sensors. For example, a fan that automatically adjusts the wind force, and a light strip that automatically changes its brightness.
2. Automatic control of the water output of the fish tank or watering pump.
3. Automatic control of festival atmosphere lights.

- Manual control by potentiometer

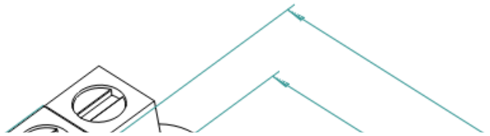
1. Home appliance renovation, making adjustable fans.
2. Electric tool production, transformation, adjustable speed.
3. Lighting effects control of light strip.

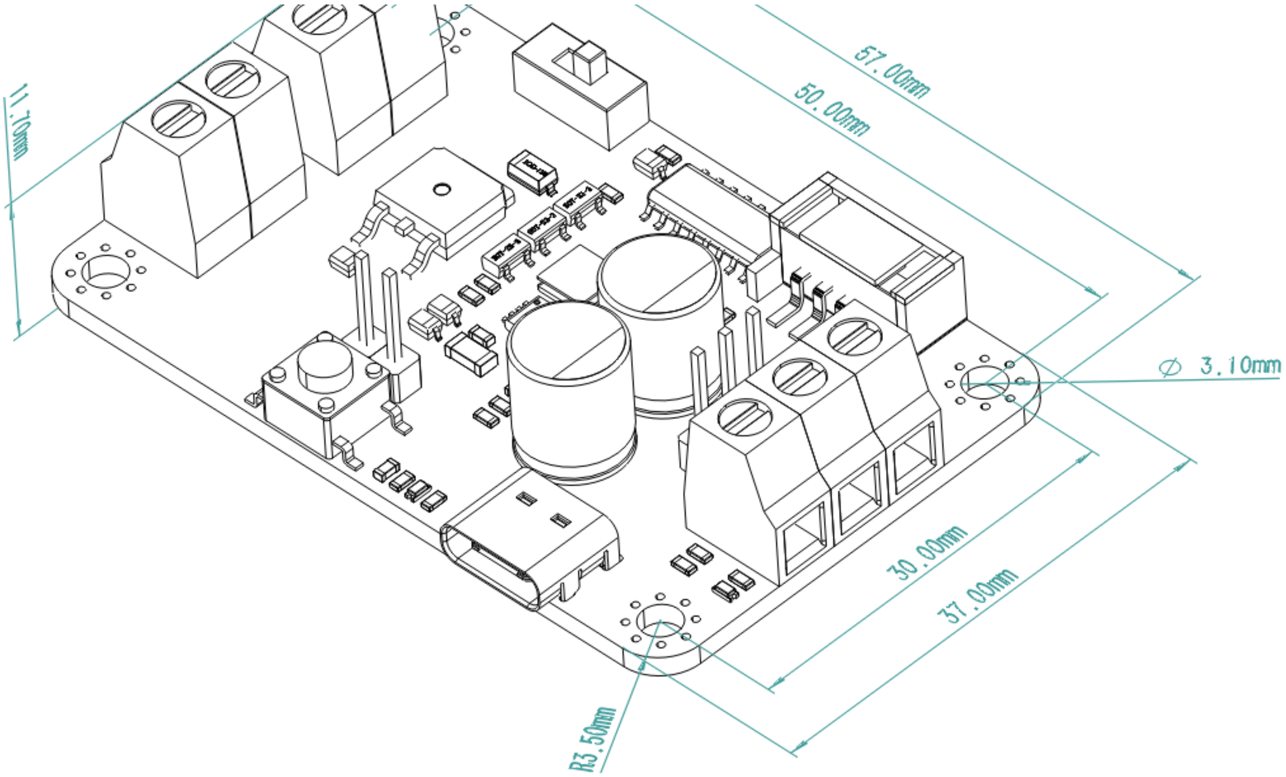
4. Specification

- Input Voltage Range: 5~24V
- Max Control Current: 10A

- Max Control Current: 10A
- PWM Duty Ratio Adjustment Range: 0~255
- PWM Frequency Adjustment Range: 183Hz ~ 46875Hz
- Number of PWM Channels:1 channel
- Stop/Start Button ×1
- Control Mode: External potentiometer adjustment, UART communication control, USB port host computer control, Python control
- USB Interface: Type-C
- UART Interface: PH2.0-4P
- External Potentiometer Interface: 2.54 pin header, binding post
- Mounting Hole Size: 30mm × 50mm, diameter 3.1mm
- PCB Size: 37 × 57mm / 1.46 × 2.24 inches

5. Product Dimension





6. Board Overview

USB Interface which can connect to PC, LattePanda and Raspberry Pi
PC software or serial port assistant can be used to control PWM output value.

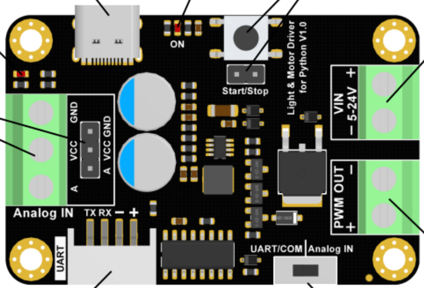
Power Indicator

ON/ OFF button and corresponding external interface
Click switch to start or stop PWM output and the default state is startup.
Start and stop can be controlled by Pin header with an external switch.

PWM Status Indication

Power Input Interface
Support 5V-24V

Analog Input Interface
Two Interface Modes: Binding Post and Pin Header
The potentiometer or potentiometer module can be connected to control the PWM output value.



PWM Output Interface
It can control motor speed or LED strip brightness.

UART Interface
Master control can be connected to control PWM parameters.

Switch is controlled by serial port and analog port.
UART / COM : Use UART or USB to control PWM value
Analog IN: Use analog port external potentiometer to control PWM value

7. Control Peripheral via Windows

The product allows users to configure its PWM parameters by Windows so as to control motor speed or light brightness.

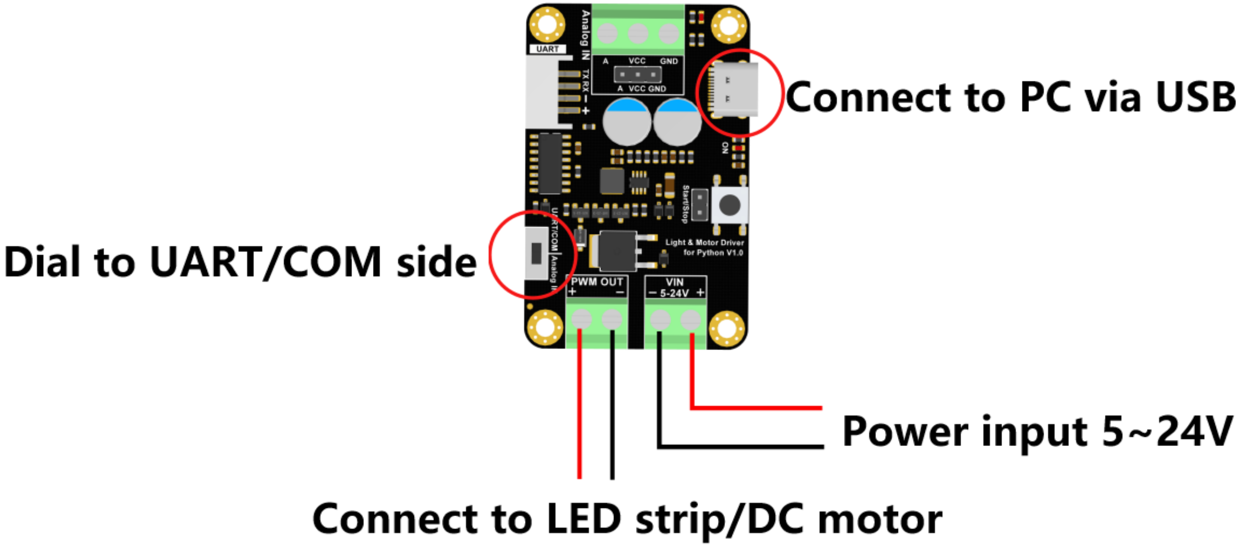
Note: Currently the product only passed the test on Windows. Other Systems are not tested yet.

1. Connect a device

Before using the product, you have to prepare the following hardware:

- PC Windows 10
- Motor or LED strip
- Type-C cable
- Power Source(Voltage: 5V~24V Power>2A Select the suitable power according to the devcie you used)

Connect all parts together as shown below, dial the switch to UART/COM side.



2. Install USB driver

- Driver Download

When using this product for the first time, you may need to download and install the USB driver program.

USB Driver download link(Select one of the links below to download):

https://github.com/DFRobot/CH_Driver (https://github.com/DFRobot/CH_Driver)

http://download.dfrobot.top/CH_Drive/ (http://download.dfrobot.top/CH_Drive/)

- Driver Description

Windows Driver: CH340/CH341 USB to Serial Windows Driver program. DLL and non-standard buad rate settings are included. It supports 32/64 bits Windows 10/8.1/8/7/VISTA/XP, SERVER 2016/2012/2008/2003, 2000/ME/98. Certificated by microsoft digital signature. Supporting USB to serial port 3-wire and 9-wire.

Linux Driver: CH340/CH341 USB to serial port Linux Driver program, supporting 32/64bits.

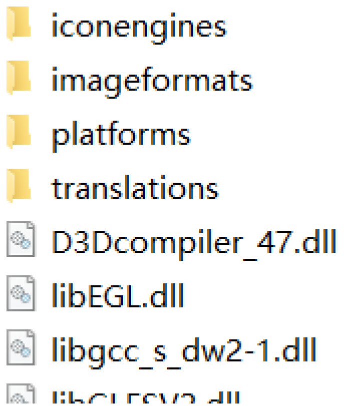
MAC Driver: CH340/CH341 USB to serial port MAC OS Driver program, supporting 32/64bits. Manual included.

Android Driver: CH340/CH341 USB to serial port Android driver-free library, used for the USB Host mode of Android OS 3.1 and above. No need to load Android core driver and root operation privilege is unnecessary. It includes apk installtion program, lib files(Java Driver), and APP Demo(USB to UART Demo Engineering SDK).

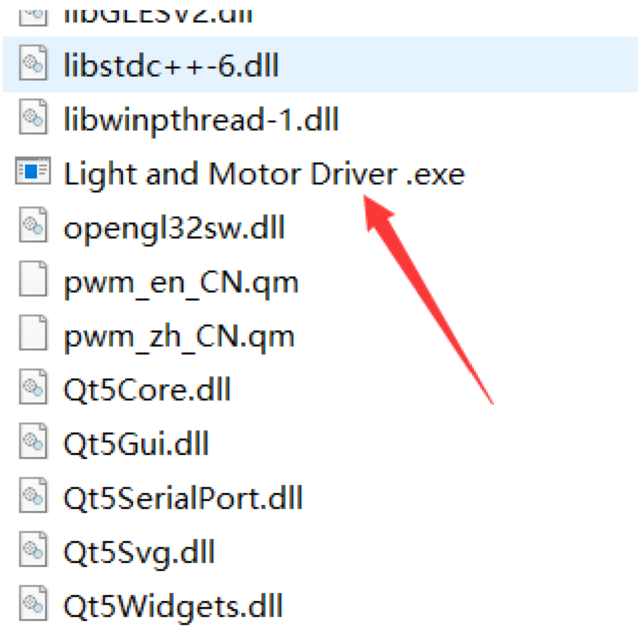
3. Download program package for Windows

Download link: https://github.com/DFRobot/DRI0050_soft_V1.0 (https://github.com/DFRobot/DRI0050_soft_V1.0)

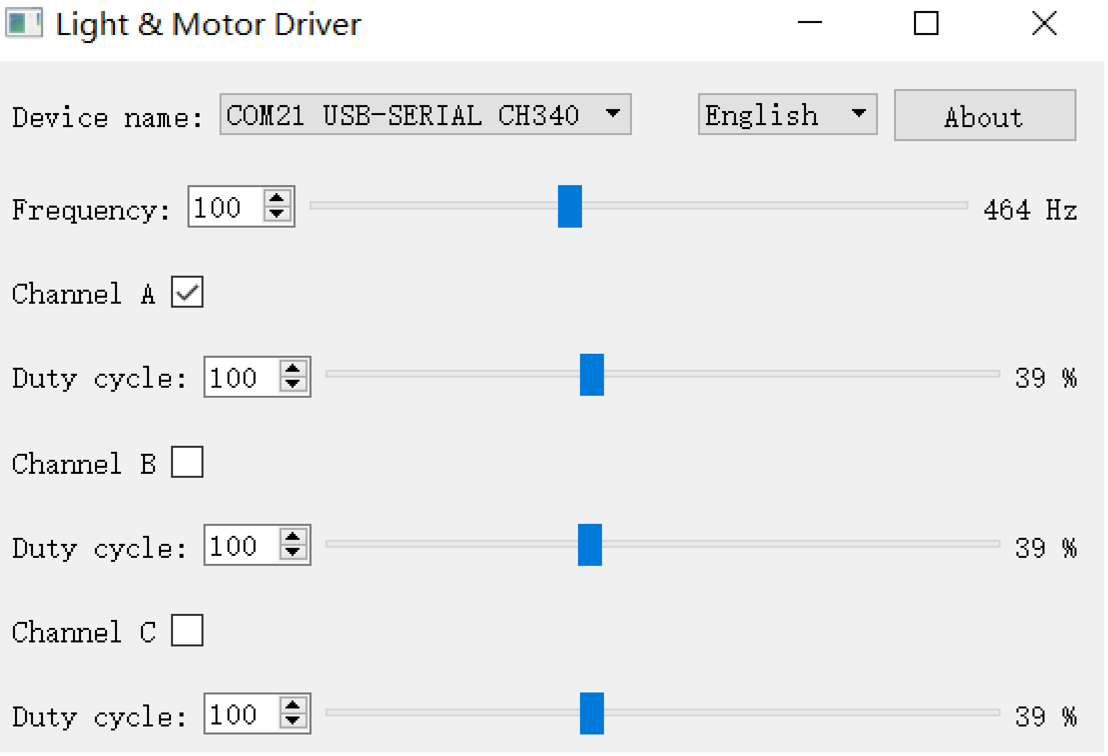
Download the file, unzip it, click the program "Light and Motor Driver.exe" to run. No need to install.



iconengines
imageformats
platforms
translations
D3Dcompiler_47.dll
libEGL.dll
libgcc_s_dw2-1.dll


libGLESv2.dll
libstdc++-6.dll
libwinpthread-1.dll
Light and Motor Driver .exe
opengl32sw.dll
pwm_en_CN.qm
pwm_zh_CN.qm
Qt5Core.dll
Qt5Gui.dll
Qt5SerialPort.dll
Qt5Svg.dll
Qt5Widgets.dll

Software Interface:


Light & Motor Driver

Device name: COM21 USB-SERIAL CH340 ▾ English ▾ About

Frequency: 100 464 Hz

Channel A ☑

Duty cycle: 100 39 %

Channel B ☐

Duty cycle: 100 39 %

Channel C ☐

Duty cycle: 100 39 %

1. Connect the product to a PC via a USB Type-C cable. If the driver program is installed already, the software will automatically recognize it.

2. Click the start button, the PC communicates successfully with PWM module, and now the PWM frequency and duty ratio can be adjusted in real-time.

3. Adjustable Frequency ranges from 0 to 255. You can input manually or drag the control bar to revise. The frequency will be displayed on the right side of the interface.

4. The product supports single-channel PWM control, so only the duty ratio of Channel 1 can be adjusted. Enter the duty ratio into the input box.

5. Tick/untick to enable/disable PWM control.

8. Control Peripherial via Python Programming

When connecting this board to a PC via a USB cable, you can not only directly control it on PC, but also program in Python on multiple platforms like Windows, MAC, Linux, etc. to control motor or LED strip.

Note: Windows platform needs to install Python3.5+.(The newest version is recommended.)

Most Linux platforms has pre-installed Python3, if not, please install Python3.5+.

There are two ways provided for Python programming:

1. **Program by Python Pinpong library:** pinpong library is a Python library that integrates abundant hardware control functions, which aims to greatly simplify coding.

2. **Program by Python directly:** The codes will be a little bit cumbersome, but still, same functions can be realized in this way.

Control prepherial by Python_pinpong library (All platforms)

Input the following commands to install pinpong library:

```
pip3 install pinpong
```

Use pinpong library to control device (For Linux and Windows)

Use pinpong library to control device (For Linux and Windows)

The program provides the basic controllable commands. You can select your program control method and coding platform, etc, as your actual needs.

```
# -*- coding: utf-8 -*-
'''
PWM frequency higher than 2K, there may be relatively large difference between the frequen
For frequency higher than 2K, please refer to the following frequency value:
46875HZ, 23437HZ, 15625HZ, 11718HZ,
9375HZ, 7812HZ, 6696HZ, 5859HZ, 5208HZ, 4687HZ, 4261HZ,
3906HZ, 3605HZ, 3348HZ, 3125HZ,
2929HZ, 2757HZ, 2604HZ, 2467HZ, 2343HZ, 2232HZ, 2130HZ, 2038HZ,
'''

import time
from pinpong.board import Board
from pinpong.libs.dfrobot_dri0050 import DRI0050 # Import DRI0050 library from libs

#Board("RPi").begin()  #RPi Linux platform
Board("Win").begin() #windows platform

#pwmd = DRI0050(port="/dev/ttyUSB0") #RPi Linux platform
pwmd = DRI0050(port="COM12")  #Windows platform
pwmd = DR10050(port='COM12')  #windows platform

print("version=0x{:x}, addr=0x{:x}".format(pwmd.get_version(), pwmd.get_addr()))
print("pid=0x{:x}, vid=0x{:x}".format(pwmd.get_vid(), pwmd.get_pid()))


while True:
  print("\n--------Inital Value------")
  print("freq={}, duty={:.2f} enable={}".format(pwmd.get_freq(), pwmd.get_duty(), pwmd.get

  print("--------Set a new value------")
  #pwmd.pwm(freq=860,duty=0.82) # freq(183HZ-46875HZ) duty(0%-100%)
  pwmd.set_freq(860) #(183HZ-46875HZ)
  pwmd.set_duty(0.82)#(0%-100%)
  pwmd.set_enable(1)
  print("freq={}, duty={:.2f} enable={}".format(pwmd.get_freq(), pwmd.get_duty(), pwmd.get

  print("--------Restore to factory settings (366HZ, duty ratio 50%, disable output)-----
  pwmd.pwm(freq=366,duty=0.5) # freq(183HZ-46875HZ) duty(0%-100%)
  pwmd.set_enable(0)
  time.sleep(5)
```

Control prepherial via Python (All platforms)

Use the following commands to install serial library and modbus library:

```
pip3 install serial
pip3 install modbus_tk
```

The program provides the basic controllable commands. You can select your program control method and coding platform, etc, as your actual needs.

```
# -*- coding: utf-8 -*-
'''
PWM frequency higher than 2K, there may be relatively large difference between the frequer
For frequency higher than 2K, please refer to the following frequency value:
46875HZ, 23437HZ, 15625HZ, 11718HZ,
9375HZ, 7812HZ, 6696HZ, 5859HZ, 5208HZ, 4687HZ, 4261HZ,
3906HZ, 3605HZ, 3348HZ, 3125HZ,
2929HZ, 2757HZ, 2604HZ, 2467HZ, 2343HZ, 2232HZ, 2130HZ, 2038HZ,
'''

import time
import serial
import modbus_tk
import modbus_tk.defines as cst
from modbus_tk import modbus_rtu

PORT="COM12" #Windows platform
#PORT="/dev/ttyUSB0" #Linux platform
BAUDRATE=9600
```

```
BAUDRATE=9600
SLAVE_ADDR=0x32

PID_REG       =  0x00
VID_REG       =  0x01

ADDR_REG      =  0x02
VER_REG       =  0x05
DUTY_REG      =  0x06
FREQ_REG      =  0x07
PWM_EN_REG    =  0x08

ser = serial.Serial(port=PORT,baudrate=BAUDRATE, bytesize=8, parity='N', stopbits=1)
master = modbus_rtu.RtuMaster(ser)
time.sleep(0.5)

def get_pid():
  data = master.execute(SLAVE_ADDR, cst.READ_HOLDING_REGISTERS, PID_REG, 1)
  time.sleep(0.03)
  return data[0]

def get_vid():
  data = master.execute(SLAVE_ADDR, cst.READ_HOLDING_REGISTERS, VID_REG, 1)
  time.sleep(0.03)
  return data[0]
```

```
def get_addr():
  data = master.execute(SLAVE_ADDR, cst.READ_HOLDING_REGISTERS, ADDR_REG, 1)
  time.sleep(0.03)
  return data[0]


def get_version():
  data = master.execute(SLAVE_ADDR, cst.READ_HOLDING_REGISTERS, VER_REG, 1)
  time.sleep(0.03)
  return data[0]

def get_duty():
  data = master.execute(SLAVE_ADDR, cst.READ_HOLDING_REGISTERS, DUTY_REG, 1)
  time.sleep(0.03)
  return data[0]/255

def get_freq():
  data = master.execute(SLAVE_ADDR, cst.READ_HOLDING_REGISTERS, FREQ_REG, 1)
  time.sleep(0.03)
  return int(12*1000*1000/256/(data[0]+1))

def get_enable():
  data = master.execute(SLAVE_ADDR, cst.READ_HOLDING_REGISTERS, PWM_EN_REG, 1)
  time.sleep(0.03)
  return data[0]

def set_duty(duty):
  master.execute(SLAVE_ADDR, cst.WRITE_SINGLE_REGISTER, DUTY_REG, output_value=int(duty*25
  time.sleep(0.03)


def set_freq(freq):
  master.execute(SLAVE_ADDR, cst.WRITE_SINGLE_REGISTER, FREQ_REG, output_value=int(12*1000
  time.sleep(0.03)

def set_enable(enable):
  master.execute(SLAVE_ADDR, cst.WRITE_SINGLE_REGISTER, PWM_EN_REG, output_value=enable)
  time.sleep(0.03)

def pwm(freq, duty):
  v=[]
  v.append(int(duty*255))
  v.append(int(12*1000*1000/256/freq) - 1)
  master.execute(SLAVE_ADDR, cst.WRITE_MULTIPLE_REGISTERS, DUTY_REG, output_value=v)
  time.sleep(0.03)


print("version=0x{:x}, addr=0x{:x}".format(get_version(), get_addr()))
print("pid=0x{:x}, vid=0x{:x}".format(get_vid(), get_pid()))
print("\n--------Initial Value------")
print("freq={}, duty={:.2f} enable={}".format(get_freq(), get_duty(), get_enable()))

print("--------Set a new value------")
#pwm(freq=860,duty=0.82) # freq(183HZ-46875HZ) duty(0%-100%)
set_freq(860) #(183HZ-46875HZ)
set_duty(0.82)#(0%-100%)
set_enable(1)
print("freq={}, duty={:.2f} enable={}".format(get_freq(), get_duty(), get_enable()))

print("--------Restore to factory settings(366HZ, duty ratio 50%, diable output)-------\n'
pwm(freq=366, duty=0.5) # freq(183HZ-46875HZ) duty(0%-100%)
set_enable(0)
```

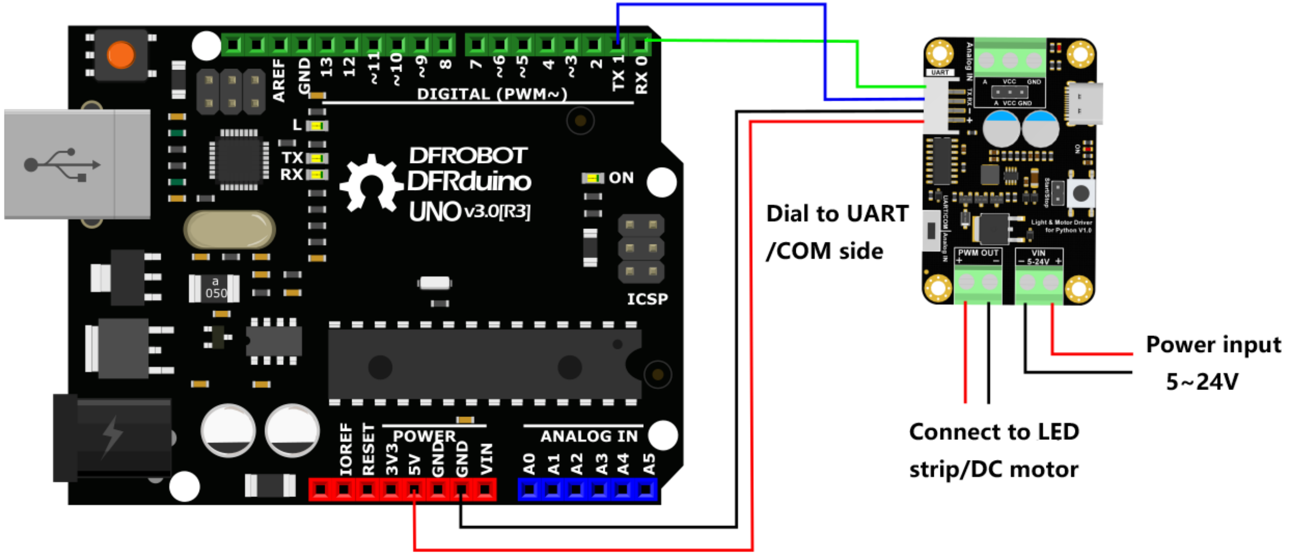
9. Control Prepherial via Arduino Programming

Before using Arduino programming, you have to prepare the following hardware:

- Arduino UNO
- Motor or LED strip

- 4PIN Gravity sensor wire
- Power source(Voltage: 5~24V, power>2A Select the suitable power according to the devcie you used)

Connect all parts together as shown below, dial the switch to UART/COM side.



Function Description: let the LED strip flash every 2s.

```
#define  PWM_ENABLE           0x01
#define  PWM_DISENABLE        0x00
#define  DEV_ADDR             0x32
#define  DUTY_REG_ADDR        0x0006
#define  FREQ_REG_ADDR        0x0007
#define  PWM_EN_REG_ADDR      0x0008


static uint16_t CheckCRC(uint8_t *data, uint8_t len){
  uint16_t crc = 0xFFFF;
  for(uint8_t pos = 0; pos < len; pos++){
    crc ^= (uint16_t)data[pos];
    for(uint8_t i = 8; i != 0; i-- ){
      if((crc & 0x0001) != 0){
        crc >>= 1;
        crc ^= 0xA001;
      }else{
        crc >>= 1;
      }
    }
  }
  crc = ((crc & 0x00FF) << 8) | ((crc & 0xFF00) >> 8);
  return crc;

}

static void WriteRegValue(uint16_t regAddr, uint16_t value){
  uint8_t tempData[8];
  uint16_t crc;
  tempData[0] = DEV_ADDR;
  tempData[1] = 0x06;
  tempData[2] = (regAddr >> 8) & 0xFF;
  tempData[3] = regAddr & 0xFF;
  tempData[4] = (value >> 8) & 0xFF;
  tempData[5] =  value & 0xFF;
  crc = CheckCRC(tempData, 6);
  tempData[6] = (crc >> 8) & 0xFF;
  tempData[7] = crc & 0xFF;
  for(uint8_t i = 0 ;i < 8; i++){
    Serial.print((char)tempData[i]);
  }
  Serial.flush();
}
```

```
static void setPwmDuty(uint8_t duty){
  WriteRegValue(DUTY_REG_ADDR, (uint16_t)duty);
}


static void setPwmFreq(uint8_t freq){
  WriteRegValue(FREQ_REG_ADDR, (uint16_t)freq);
}

static void setPwmEnable(uint8_t pwmStatus){
  WriteRegValue(PWM_EN_REG_ADDR, (uint16_t)pwmStatus);
}

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  delay(1000);
  setPwmFreq(10);
  delay(50);
  setPwmDuty(0);
  delay(50);
  setPwmEnable(PWM_ENABLE);
  delay(50);
}

void loop() {
    setPwmDuty(200);
    delay(2000);

    setPwmDuty(0);
    delay(2000);
}
```

10. Control Peripherial via Potentiometer

We can also connect a "Potentiometer module" or "Potentiometer element" to control PWM value so as to control motor speed or light brightness.

Before using potentiometer to control, you need to prepare the following hardware and tool:

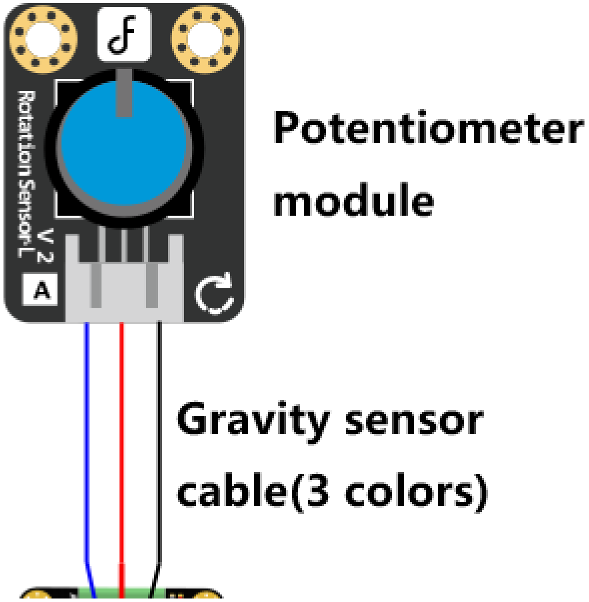
- Gravity Potentiometer module or potentiometer element with wires soldered.
- Motor or LED strip
- Screwdriver
- Power source(Voltage: 5~24V, power>2A Select the suitable power according to the devcie you used)

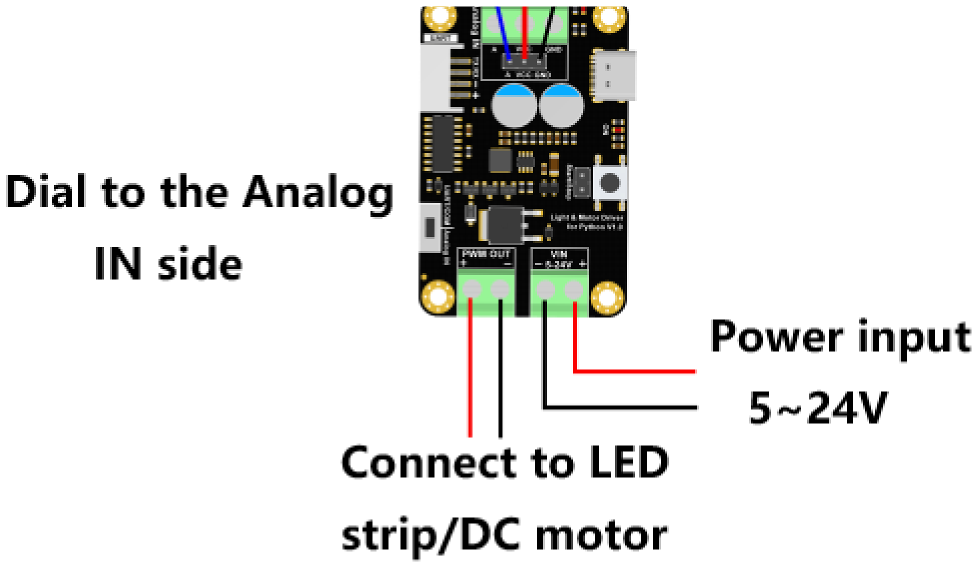
you used)

Connect all parts together as shown below, dial the switch to Analog IN side.

Connect with Gravity Potentiometer Module

Use the Gravity 3PIN sensor cable to connect the DFRobot Potentiometer with the driver board. (Pay attention to the wire order when connecting.)

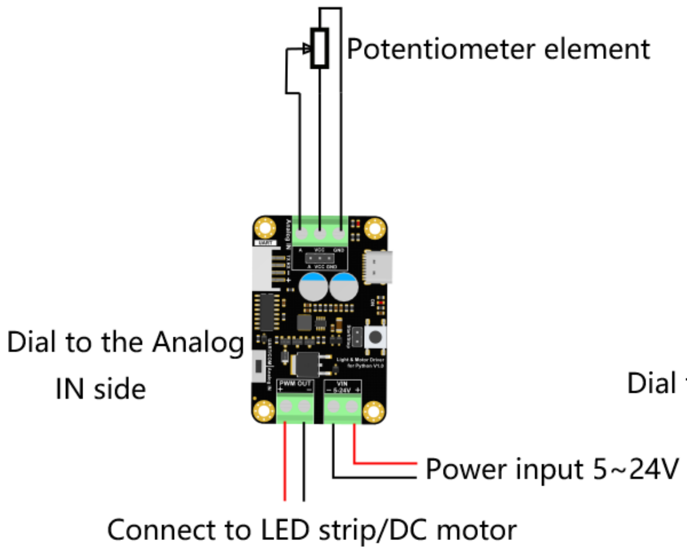




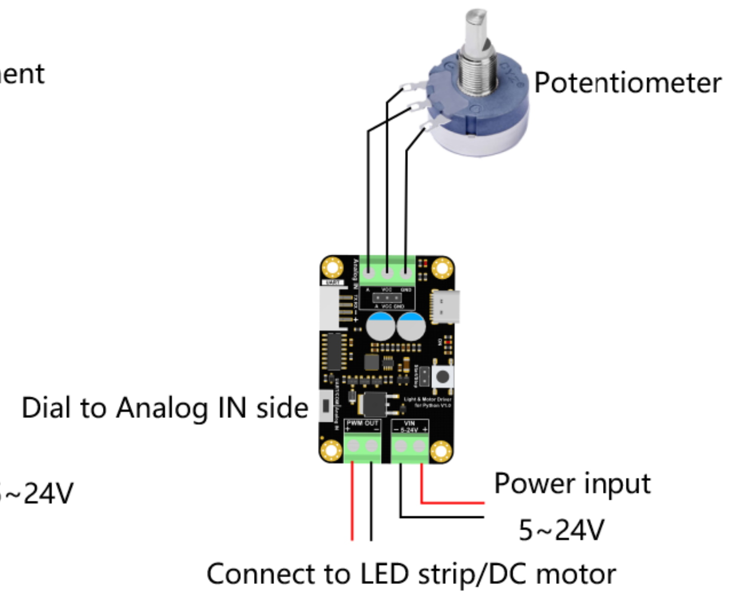
Connect with Potentiometer element

The diagram on the left is the connection for potentiometer element. You have to solder wires for the potentiometer first and then attach the wires to the binding post. The right diagram shows how to connect a real potentiometer to the driver board. Normally the middle pin of a potentiometer is the wiper that gives us the variable of resistance value, and should be connected to port "A". Find the right wiper on your potentiometer before connecting.

Available potentiometer vlaue: 4.7K~470K



Connection Schematics



Connection Diagram

11 Mind+

11. Mind+

12. Register Table (ModBus RTU Communication Protocol)

Type	Address	Name	Read/Write	Data Range	Default Value	Desc
Holding Register	0×0000	PID	R	0×C032	0×C032	PI
Holding Register	0×0001	VID	R	0×3343	0×3343	VI
Holding Register	0×0002	Device Address	R	0×0032	0×0032	Mc A
Holding Register	0×0003	Reserve	R	0×0000~0×FFFF	0×FFFF	
Holding Register	0×0004	Reserve	R	0×0000~0×FFFF	0×FFFF	

Type	Address	Name	Read/Write	Data Range	Default Value	Desc
Holding Register	0×0005	Version	R	0×0000~0×00FF	0×1000	Firm 0

Holding Register	0×0006	PWM0 Duty Ratio	R/W	0×0000~0×00FF	0×007F	PV Dut fo
Holding Register	0×0007	PWM0 Frequency	R/W	0×0000~0×00FF	0×007F	PV Frec freq fa fr 12M co th

th

Type	Address	Name	Read/Write	Data Range	Default Value	Desc
Holding Register	0×0008	PWM Output Enable/Disable Status	R/W	0×0000~0×0001	0×0000	En PV

FAQ

For any questions, advice or cool ideas to share, please visit the **DFRobot Forum** (https://www.dfrobot.com/forum/).

More Documents

- Schematics PDF

- Schematics.PDF (https://dfimg.dfrobot.com/nobody/wiki/e56242dc316eb1db47799bca7e05bff2.PDF)
- DRI0050-V1.0-3D-STEP File.rar (https://dfimg.dfrobot.com/nobody/wiki/46c98ea706e831286a9cb0c265c0f61a.rar)
- DRI0050-V1.0-2D-DXF File.rar (https://dfimg.dfrobot.com/nobody/wiki/d08ff6fcb58626bec2af453dcc113eb8.rar)

Get **Light and Motor Driver for Python** (https://www.dfrobot.com/product-2429.html) from DFRobot Store or **DFRobot Distributor**. (https://www.dfrobot.com/index.php?route=information/distributorslogo)

Turn to the Top