Motor Driver HAT User Manual

OVERVIE

This module is a motor driver board for Raspberry Pi. Use I2C interface, could be used for Robot applications.

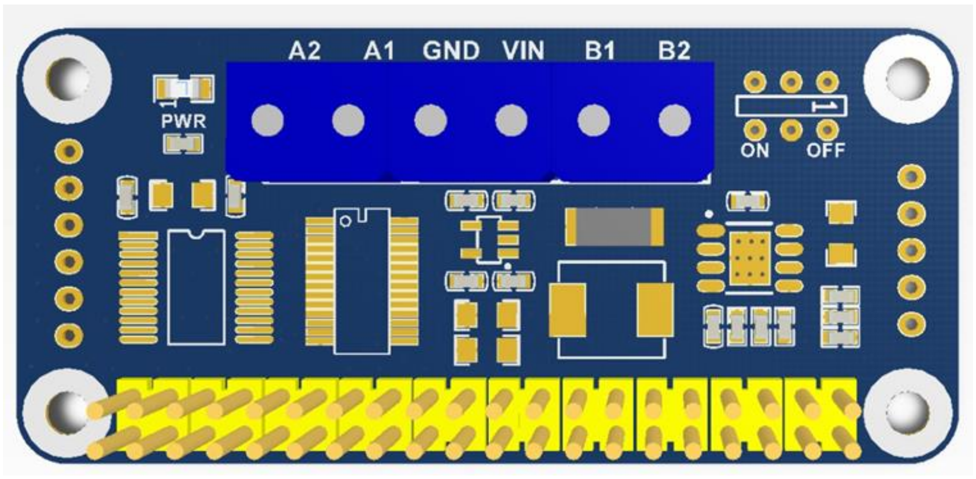
FEATURES

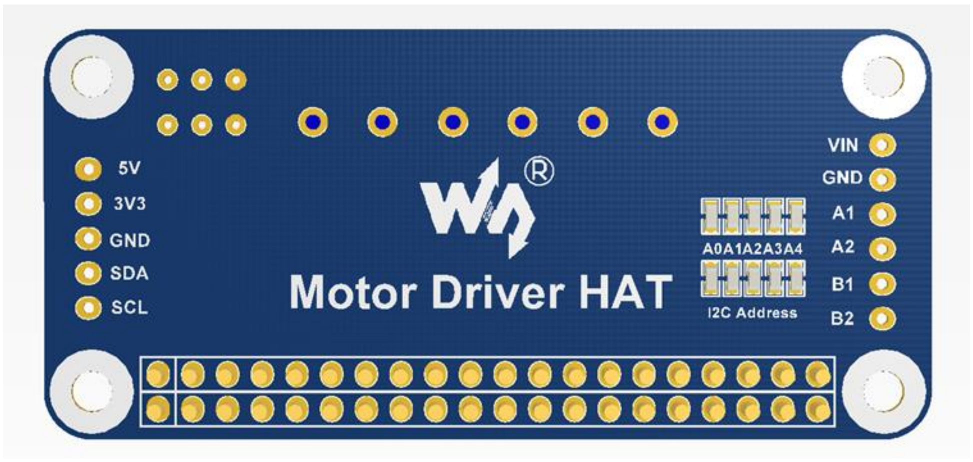
- Compatible with Raspberry Pi
- I2C interface. Slave address hardware configurable makes your pi able to connect 32 motors at the same time
- Integrate PCA9685, supports 12bits PWM output
- Integrate TB6612FNG, high performance
- Integrate 5V regulator, output current up to 3A.
- Pins out I2C interface, could connect to other development board
- Come with sample codes and user manual

SPECIFICATIONS

Operating voltage:	6V~12V (VIN port)
Logic voltage:	3.3V
PWM controller:	PCA9685
Interface:	I2C
Motor controller:	TBA6612FNG
Dimension:	65mm x 30mm
Holes size:	3.0mm

INTERFACES



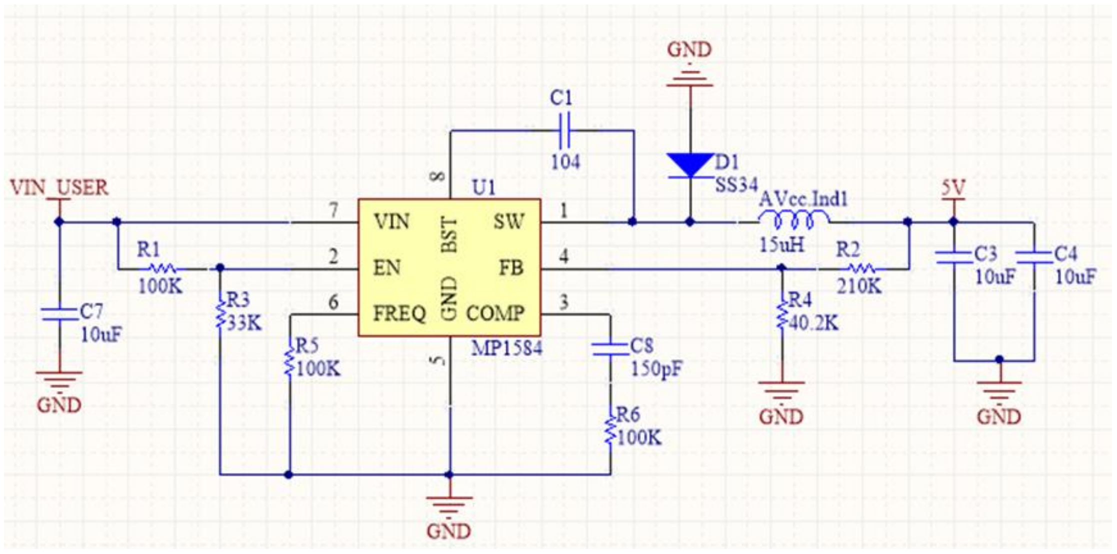


PIN	Description
5V	5V
3V3	3.3V
GND	Ground
SDA	I2C Data
SCL	I2C Clock
VIN	Driver voltage for motor（6-12V）
A1	positive pole of motor A
A2	negative pole of motor A
B1	positive pole of motor B
B2	negative pole of motor B

HARDWARES

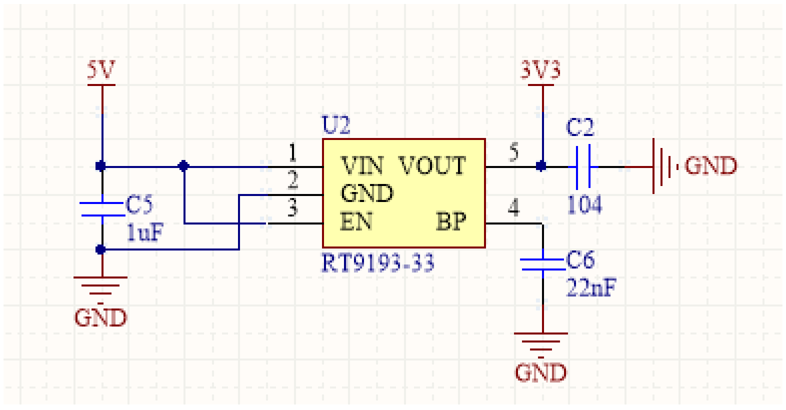
The module includes three part in hardware: Power, PWM and Motor driver

POWER



MP1854 regulator is used to convert the input voltage (VIN_USER), has wide 4.5V to 28V input range, could provide 3A output. Even the chip supports 28V input, however, VIN_USER is also the power supply for motor, so the actual working voltage of this module is 6-12V.

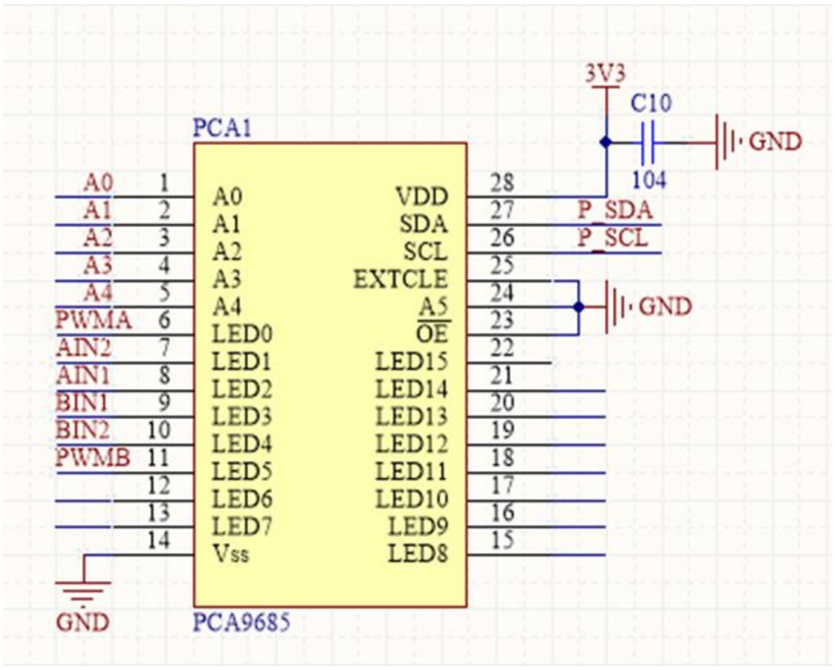
MP1854 outputs stable 5V, which is used to power Raspberry Pi, and supplies 3.3V logic level for PWM and motor driver via RT9193-33.



PWM

Raspberry Pi only has one hardware PWM pin (GPIO.1), and the software PWM of WiringPi and Python will cost CPU resources. This module, we use PCA9685, I2C bus controlled, support 16-channel 12-bits PWM output. The frequency range of PWM is 40Hz to 1000Hz.

It is very simply to use, to output PWM signal you only need to control corresponding registers value of chip.



Refer the recommend circuit, LED0-LED5 are motor control pins.

From Page6 to Page8 of datasheet:

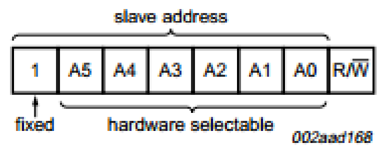
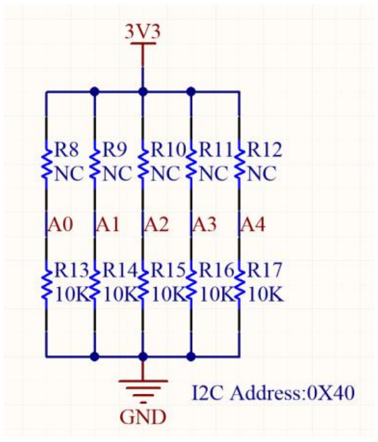


Fig 4. Slave address

The slave address of I2C bus is 7-bits, highest is fixed 1. A5-A0 are hardware selectable.

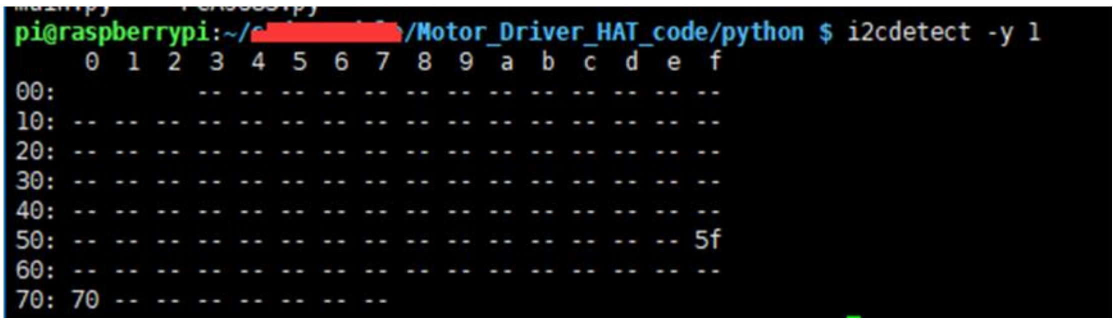
This Motor Driver HAT, A5 is connected to ground (0) by default, you can change resistors of A0-A4 to configure the slave address. If you weld a resistor or short it, means 1, otherwise 0. The address range from 0x40 to 0x5F.



For example:

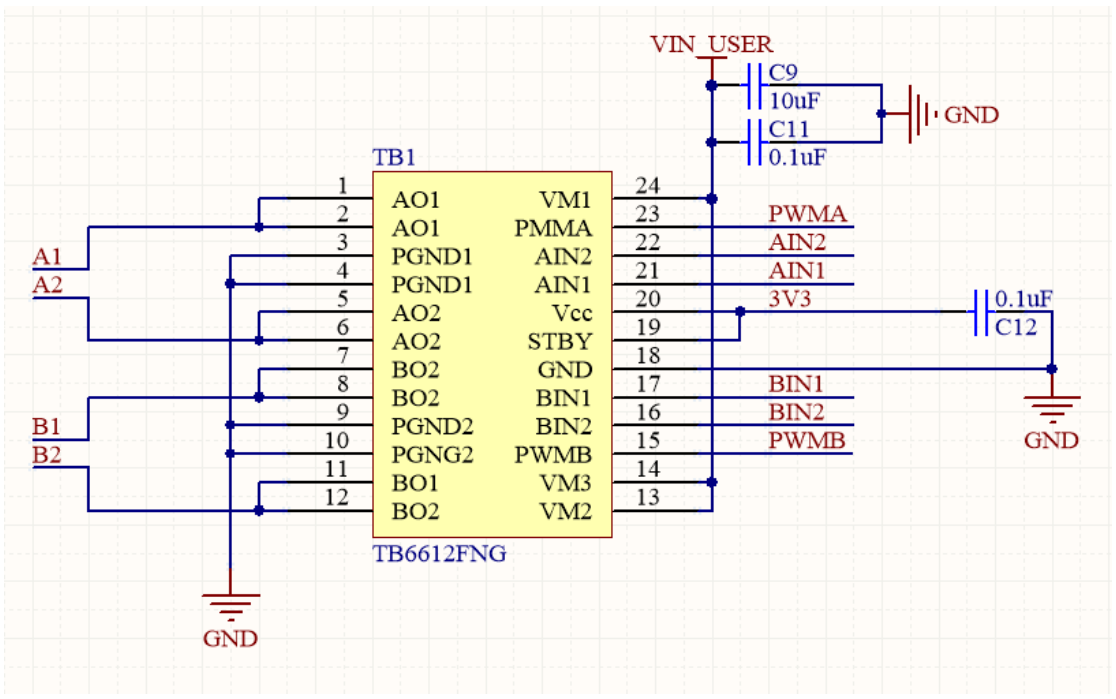
1. A5 is 0, A0-A4 are disconnected (don' t weld), so they are 0 as well. The final I2C slave address is 0x40

2. Welding 0Ω resistors to A0-A4 or shorting them, the final I2C slave address is 0x5F. You can use i2cdetect tool to detect i2C devices on Raspberry Pi as below: execute command: **i2cdetect -y 1**



```
pi@raspberrypi:~/[redacted]/Motor_Driver_HAT_code/python $ i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- 5f
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: 70 -- -- -- -- -- -- --
```

MOTOR DRIVER



TB6612FNG is a driver IC for DC motor with high performance.

VIN_USER is input voltage, in theory, motor speed up if this voltage is increased.

Recommend input voltage is 6~12V

PWMA and PWMB control speed of motors, AIN1 and AIN2, BIN1 and BIN2 control rotate direction of motors.

A1 and A2, B1 and B2 are connect to positive/negative poles of two motors separately.

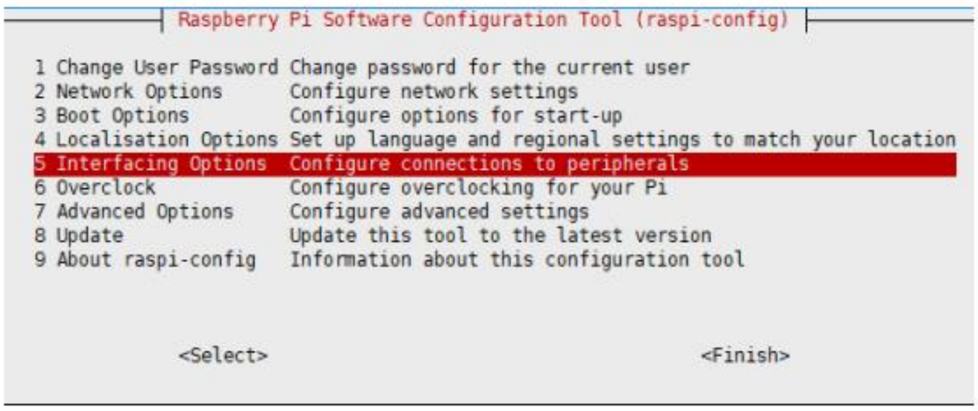
HOW TO USE

ENABLE I2C INTERFACE

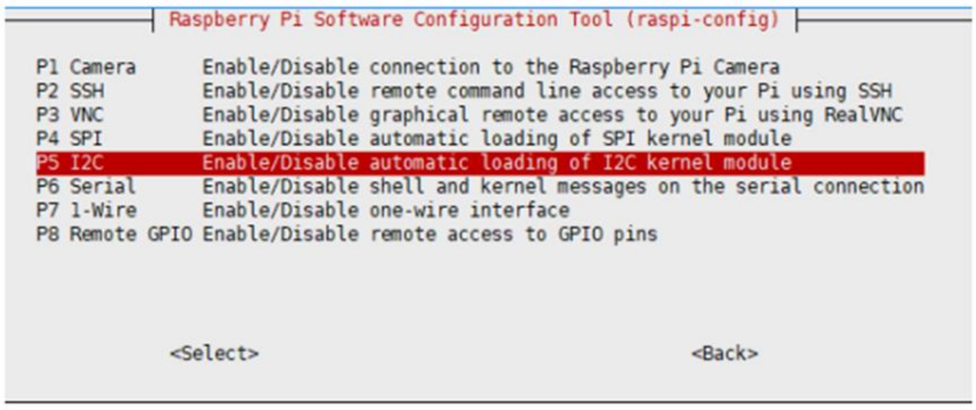
To works with Raspberry Pi, you should first enable I2C interface as below:

sudo raspi-config

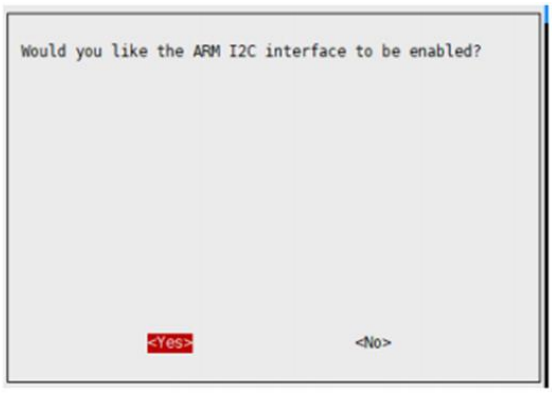
Then choose Interfacing Options -> I2C -> Yes



```
Raspberry Pi Software Configuration Tool (raspi-config)

1 Change User Password Change password for the current user
2 Network Options      Configure network settings
3 Boot Options         Configure options for start-up
4 Localisation Options Set up language and regional settings to match your location
5 Interfacing Options  Configure connections to peripherals
6 Overclock            Configure overclocking for your Pi
7 Advanced Options     Configure advanced settings
8 Update               Update this tool to the latest version
9 About raspi-config   Information about this configuration tool

             <Select>                                   <Finish>
```



```
Raspberry Pi Software Configuration Tool (raspi-config)

P1 Camera      Enable/Disable connection to the Raspberry Pi Camera
P2 SSH         Enable/Disable remote command line access to your Pi using SSH
P3 VNC         Enable/Disable graphical remote access to your Pi using RealVNC
P4 SPI         Enable/Disable automatic loading of SPI kernel module
P5 I2C         Enable/Disable automatic loading of I2C kernel module
P6 Serial      Enable/Disable shell and kernel messages on the serial connection
P7 1-Wire      Enable/Disable one-wire interface
P8 Remote GPIO Enable/Disable remote access to GPIO pins

             <Select>                                   <Back>
```



```
Would you like the ARM I2C interface to be enabled?

             <Yes>                <No>
```

【Note】If you get errors information after running demo code, please modify the modules file as below:

sudo nano /etc/mdoules

Append these statements to the end and save:

i2c-dev

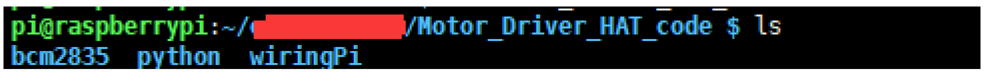
i2c-bcm2708

DOWNLOAD DEMO CODE

Demo code can be downloaded from Wiki:

- https://www.waveshare.com/wiki/Motor_Driver_HAT

Extract on your PC and copy to Raspberry Pi



```
pi@raspberrypi:~/[redacted]/Motor_Driver_HAT_code $ ls
bcm2835  python  wiringPi
```

LIBRARIES INSTALLATION

Before running the demo code, you should install necessary libraries (WiringPi, BCM2835 and python). About how to install libraries, you can refer to this page:

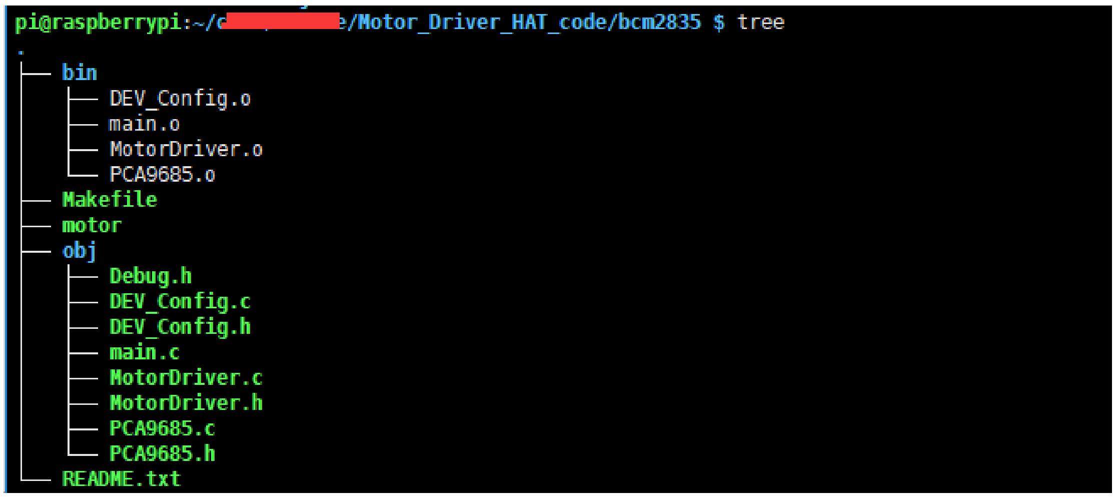
- Libraries_Installation_for_RPi

CODE ANALYSIS

We provide three demo codes for Raspberry Pi, which are based on BCM2835, WiringPi and python libraries separately.

BCM2835

PROJECT DIRECTORY AND FILES



```
pi@raspberrypi:~/c[redacted]e/Motor_Driver_HAT_code/bcm2835 $ tree
.
├── bin
│   ├── DEV_Config.o
│   ├── main.o
│   ├── MotorDriver.o
│   └── PCA9685.o
├── Makefile
├── motor
├── obj
│   ├── Debug.h
│   ├── DEV_Config.c
│   ├── DEV_Config.h
│   ├── main.c
│   ├── MotorDriver.c
│   ├── MotorDriver.h
│   ├── PCA9685.c
│   └── PCA9685.h
└── README.txt
```

/bin:

.o files generated by makefile

Makefile: makefile is used to tell the compiler how to compile your project/code.

motor: executable file, which you can execute to run the code

/Obj: workspace of project

Debug.h: heard files of debug function, you can use DEBUG() to print debug information like printf() by change USE_DEBUG to 1

DEV_Config.c(h): Defines PINS used and communication type, different between BCM2835 and WiringPi.

PCA9685.c(h): Drive code of PCA9685 chip. Could output 16-channel PWM signal via I2C interface

MotorDriver.c(.h): Drive code of TB6612FNG chip, control two motors.

main.c: main function

DEMO CODES

1. initialize BCM2835 libraries

```
if(DEV_ModuleInit())
        exit(0);
```

2. Initialize motor and PCA9685

```
Motor_Init();
```

3. Control motors

```
Motor_Run(MOTORA, FORWARD, 100);
```

Parameter 1: MOTORA or MOTORB

Parameter 2: FORWARD or BACKWARD

Parameter 3: 0~100

If you use this module to control robot, with this function you can control the whole motion of robot.

This project has Exception Handling. Generally, motor keeps moving even you stop project. value of control register doesn' t be clean even you use CTRL+c to stop code.

```
signal(SIGINT, Handler);
```

single is signal handling function of linux system. SIGINT signal generates when CTRL+c is executed to stop process, then Handler() function will run. In this function, motors are stopped

```
Motor_Stop(MOTORA);

Motor_Stop(MOTORB);

DEV_ModuleExit();

    exit(0);
```

USING

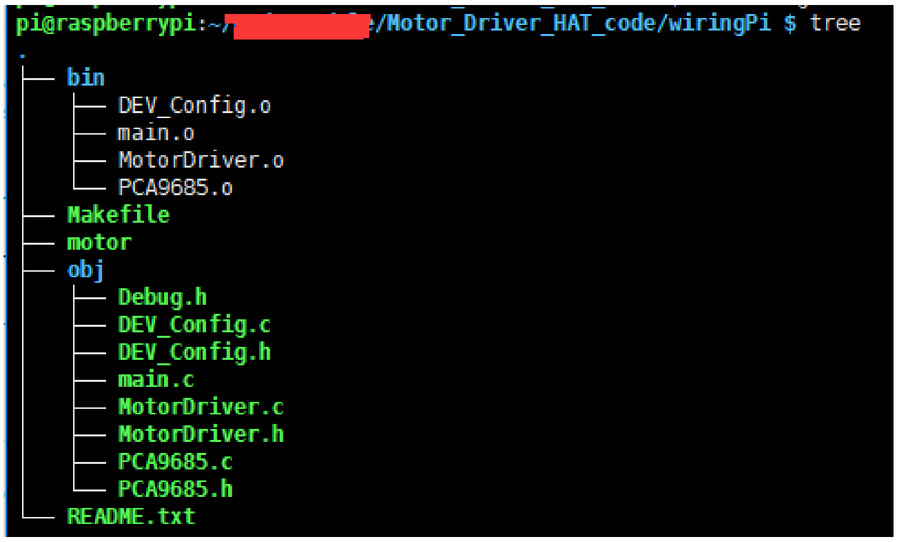
Demo code you download has no executable file moto, you need to run make to generate it then execute command ./motor to run the demo code.

If you modify or add any functions, you need to run make again to update

If you change values of heard files, command make clear is also need before make.

WIRINGPI

PROJECT DIRECTORY AND FILES



```
pi@raspberrypi:~/[redacted]/Motor_Driver_HAT_code/wiringPi $ tree
.
├── bin
│   ├── DEV_Config.o
│   ├── main.o
│   ├── MotorDriver.o
│   └── PCA9685.o
├── Makefile
├── motor
├── obj
│   ├── Debug.h
│   ├── DEV_Config.c
│   ├── DEV_Config.h
│   ├── main.c
│   ├── MotorDriver.c
│   ├── MotorDriver.h
│   ├── PCA9685.c
│   └── PCA9685.h
└── README.txt
```

The files on WiringPi project directory are similar to BCM2835. Their only difference are:

DEV_Config.c(h): functions called are different

Makefile: linker is different.

USING

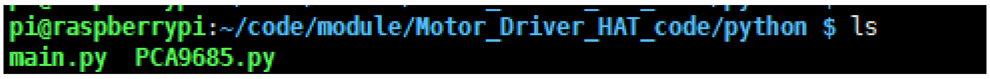
Demo code you download has no executable file moto, you need to run make to generate it then execute command ./motor to run the demo code.

If you modify or add any functions, you need to run make again to update

If you change values of heard files, command make clear is also need before make.

PYTHON

PROJECT DIRECTORY AND FILES



```
pi@raspberrypi:~/code/module/Motor_Driver_HAT_code/python $ ls
main.py  PCA9685.py
```

PCA9685.py is driver code, use I2C interface to output 16-channle PWM signals.

mian.py: Motor driver code

DEMO CODE

1. Instantiate PCA9685 library

```
pwm = PCA9685(0x40, debug=True)
```

Parameter 1: slave address of PCA9685, hardware configurable

Parameter 2: enable/disable debug information

```
pwm.setPWMFreq(50)
```

Set PWM frequency in range 40~1000

2. Initialize motors

```
class MotorDriver():
    def __init__(self):
        self.PWMA = 0
        self.AIN1 = 1
        self.AIN2 = 2
        self.PWMB = 5

        self.BIN1 = 3

        self.BIN2 = 4

Motor = MotorDriver()
```

Output PWM to channel 0~5

3. Control PWM

```
pwm.setDutycycle(self.PWMA, speed)
```

Parameter 1: output channel

Parameter 2: Duty ratio, range 0~100

4. Control level

```
pwm.setLevel(self.AIN1, 1)
```

Parameter 1: output channel

Parameter 2: level, 1 is high and 0 is low

5. Control motor

```
Motor.MotorRun(0, 'forward', 0)
```

Parameter 1: 0 and 1, stands for two motor separately

Parameter 2: forward and backward

Parameter 3: integers in 0~100, control speed

MORE

You can also control it via Bluetooth or WiFi. About how to use you can refer to Servo Driver HAT:

https://www.waveshare.com/w/upload/1/1b/Servo_Driver_HAT_User_Manual_EN.pdf