RPi Zero Relay

From Waveshare Wiki
Jump to: navigation, search

RPi Zero Relay

(https://www.waveshare.com/rpi-zero-relay.htm)

Industrial 6-ch Relay Module for Raspberry Pi Zero

Overview

Introduction

C and Python demos are available.

Features

- Equipped with CAN functionality, using CAN controller MCP2515 with SPI interface, with the transceiver SN65HVD230
- Equipped with RS485 function, using UART control, half-duplex communication, transceiver is SP3485
- Onboard TVS (Transient Voltage Suppressor), effectively suppress surge voltage and transient spike voltage in the circuit for RS485 transceiving, lightningproof & anti-electrostatic

Specifications

- Supply voltage: 7V~36V
- CAN control chip: MCP2515
- CAN Transceiver: SN65HVD230
- 485 Transceiver: SP3485
- Product size: 123mmx87.92mm
- Fixed hole diameter: 3.5mm

Pinouts

- CAN bus

PIN	Raspberry Pi (BCM)	Description
GND	GND	Ground
SCK	SCK	SPI clock input
MOSI	MOSI	SPI data input
MISO	MISO	SPI data output
CS	CE0	Data/command selection
INT	25	Interrupt output

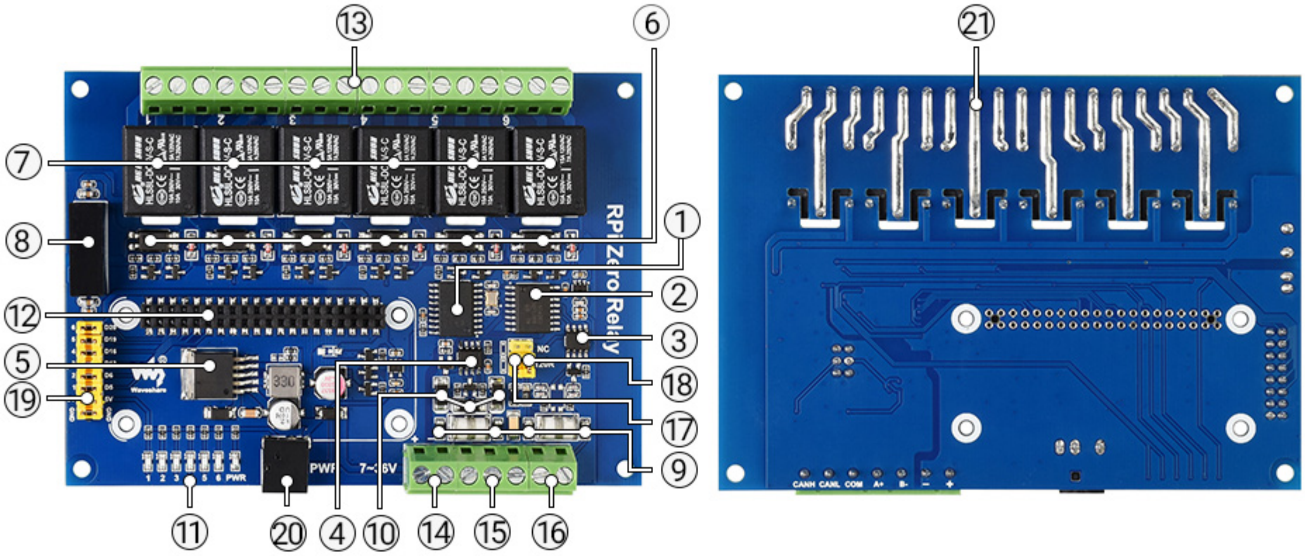
- RS485 bus

PIN	Raspberry Pi (BCM)	Description
GND	GND	Ground
RXD	RXD	UART receive
TXD	TXD	UART transimit

- Relay interfaces

PIN	Raspberry Pi (BCM)	Description
GND	GND	Ground
IN_CH1	GPIO5	Relay Channel 1
IN_CH2	GPIO6	Relay Channel 2
IN_CH3	GPIO13	Relay Channel 3
IN_CH4	GPIO16	Relay Channel 4
IN_CH5	GPIO19	Relay Channel 5
IN_CH6	GPIO20	Relay Channel 6

Onboard resources



(/wiki/File:RPi-Zero-Relay-details-intro.jpg)

No.	Name	Description
1	MCP2515	CAN controller
2	SI8642ED-B-IS	Digital isolator
3	SN65HVD230	CAN transceiver
4	SP3485EN	RS485 transceiver
5	LM2596 regulator chip	Provides stable 5V voltage
6	Optocoupler isolation	Avoids interference with the control chip by the external high-voltage circuit of the relay
7	|Contact rating per channel: ≤ 10A 250V AC or ≤ 10A 30V DC	

8	Power isolation	Provides a stable isolation voltage, and no additional power supply is required for the isolation terminal
9	Resettable fuse	Resettable fuse
10	TVS diode	Transient voltage protection
11	Relay and power LED indicator	The LED lights up to indicate that the corresponding relay is engaged
12	40PIN GPIO header	Applicable to Raspberry Pi Zero series motherboards with pre-soldered pinheaders
13	Relay screw terminal	Convenient for connection with user devices
14	Power supply screw terminal	Supports 7~36V DC power supply
15	RS485 bus interface	Connects RS485 devices
16	CAN bus interface	Connects CAN devices
17	RS485 120Ω terminal resistor switch	RS485 120Ω terminal resistor switch
18	CAN 120Ω terminal resistor switch	CAN 120Ω terminal resistor switch
19	Relay control pins	Can disconnect or connect relay control
20	External power supply jack	Round hole 7~36V DC power supply
21	Thickened tracks	Supports access to high-current devices

Install libraries

wiringPi

```
git clone https://github.com/WiringPi/WiringPi
cd WiringPi
./build
gpio -v
# Run gpio -v and corresponding version information will appear. If it does not appear, there is an installation error.
```

python

In the latest version of the system, some python libraries cannot be installed normally, you need to use a virtual environment, and then install the library and run the program

Non-virtual environment configuration

```
sudo apt-get update
sudo apt-get install python3-serial
sudo apt-get install python3-can
sudo apt-get install python3-gpiozero
```

Virtual environment configuration, execution, exit

```
# Install the libraries required for the virtual environment
sudo apt-get update
sudo apt-get install git python3-pip -y
sudo apt install python3-venv

# Create a new virtual environment (myenv is the name of the virtual environment, which can be modified)
python3 -m venv myenv

# Activate the virtual environment
source myenv/bin/activate

# Install the library
pip3 install pyserial
pip3 install python-can
pip3 install gpiozero

# Download demo (see the corresponding location in the wiki)

# Run demo (see the corresponding location in the wiki)

# Exit the virtual environment
deactivate
```

Download demo

Run on the Raspberry Pi terminal:

```
sudo apt-get install p7zip-full
wget https://www.waveshare.net/w/upload/2/2f/RPi_Zero_Relay_Code.7z
7z x RPi_Zero_Relay_Code.7z  -r -o./RPi_Zero_Relay_Code
sudo chmod 777 -R RPi_Zero_Relay_Code/
```

Relay usage

The program phenomenon is to close the relay sequentially and then disconnect the relay sequentially

C

```
cd RPi_Zero_Relay_Code/Relay/wiringPi/
make clean
make
sudo ./relay
```

python

```
cd RPi_Zero_Relay_Code/Relay/python/
python3 relay.py
```

CAN usage

This demo program uses two RPi-Zero-Relay modules

Providing Python and C language demos

Preliminary Work

Connect the modules, then modify the boot script config.txt

```
sudo nano /boot/config.txt
```

Add the following to the last line:

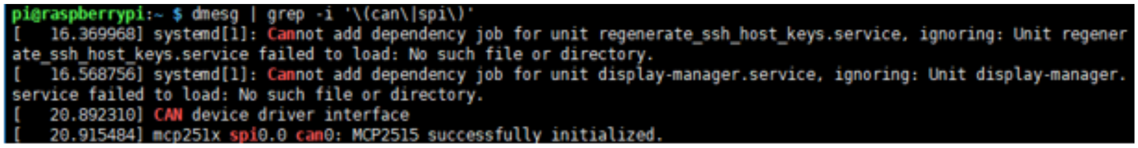
```
dtparam=spi=on
dtoverlay=mcp2515-can0,oscillator=16000000,interrupt=25,spimaxfrequency=3000000
```

After saving the exit, restart the Raspberry Pi:

```
sudo reboot
```

After the restart, run the command to check whether the initialization is successful:

```
dmesg | grep -i '\(can\|spi\)'
```

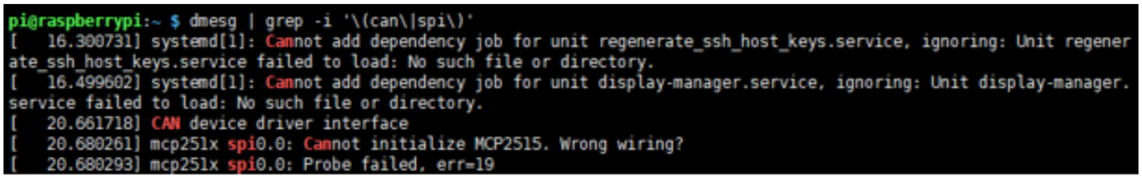


```
pi@raspberrypi:~ $ dmesg | grep -i '\(can\|spi\)'
[   16.369968] systemd[1]: Cannot add dependency job for unit regenerate_ssh_host_keys.service, ignoring: Unit regener
ate_ssh_host_keys.service failed to load: No such file or directory.
[   16.568756] systemd[1]: Cannot add dependency job for unit display-manager.service, ignoring: Unit display-manager.
service failed to load: No such file or directory.
[   20.892310] CAN device driver interface
[   20.915484] mcp251x spi0.0 can0: MCP2515 successfully initialized.
```

(/wiki/

File:RS485_CAN_HAT_CAN1.png)

If the module is not connected, the prompt may be as follows:



```
pi@raspberrypi:~ $ dmesg | grep -i '\(can\|spi\)'
[   16.300731] systemd[1]: Cannot add dependency job for unit regenerate_ssh_host_keys.service, ignoring: Unit regener
ate_ssh_host_keys.service failed to load: No such file or directory.
[   16.499602] systemd[1]: Cannot add dependency job for unit display-manager.service, ignoring: Unit display-manager.
service failed to load: No such file or directory.
[   20.661718] CAN device driver interface
[   20.680261] mcp251x spi0.0: Cannot initialize MCP2515. Wrong wiring?
[   20.680293] mcp251x spi0.0: Probe failed, err=19
```

(/wiki/

File:RS485_CAN_HAT_CAN2.png)

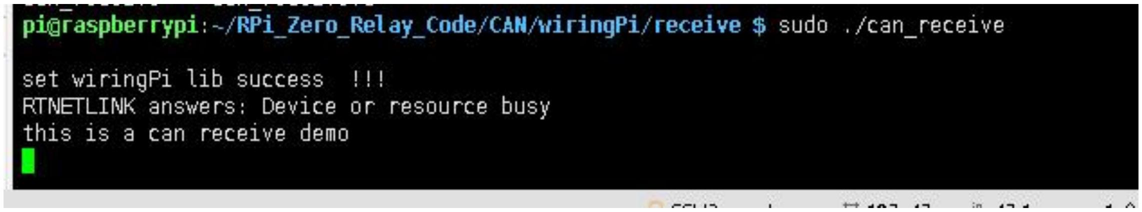
Please check whether the module is connected; whether SPI and MCP2515 kernel driver are enabled; whether it is restarted.
Make sure both RPi-Zero-Relay modules are handled in the same way, connect the H of one module to the L of the other module
If you are using other CAN devices, make sure to connect H-H, L-L

C

- Block reception, open the terminal on the Raspberry Pi and run:

```
cd RPi_Zero_Relay_Code/CAN/wiringPi/receive/
make clean
make
sudo ./can_receive
```

The receiving program is blocked.

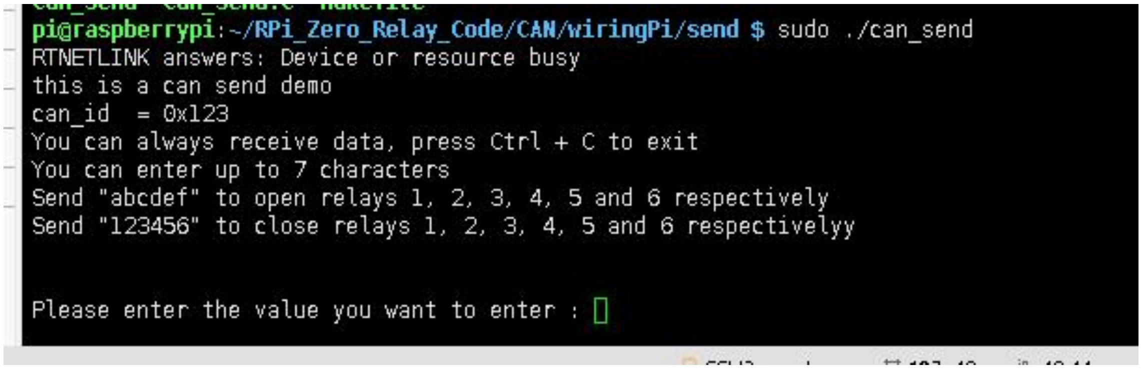


```
pi@raspberrypi:~/RPi_Zero_Relay_Code/CAN/wiringPi/receive $ sudo ./can_receive

set wiringPi lib success  !!!
RTNETLINK answers: Device or resource busy
this is a can receive demo
```

(/wiki/
File:RPi_Zero_Relay_CAN_2.jpg)

- Send, the Raspberry Pi opens the terminal and runs:

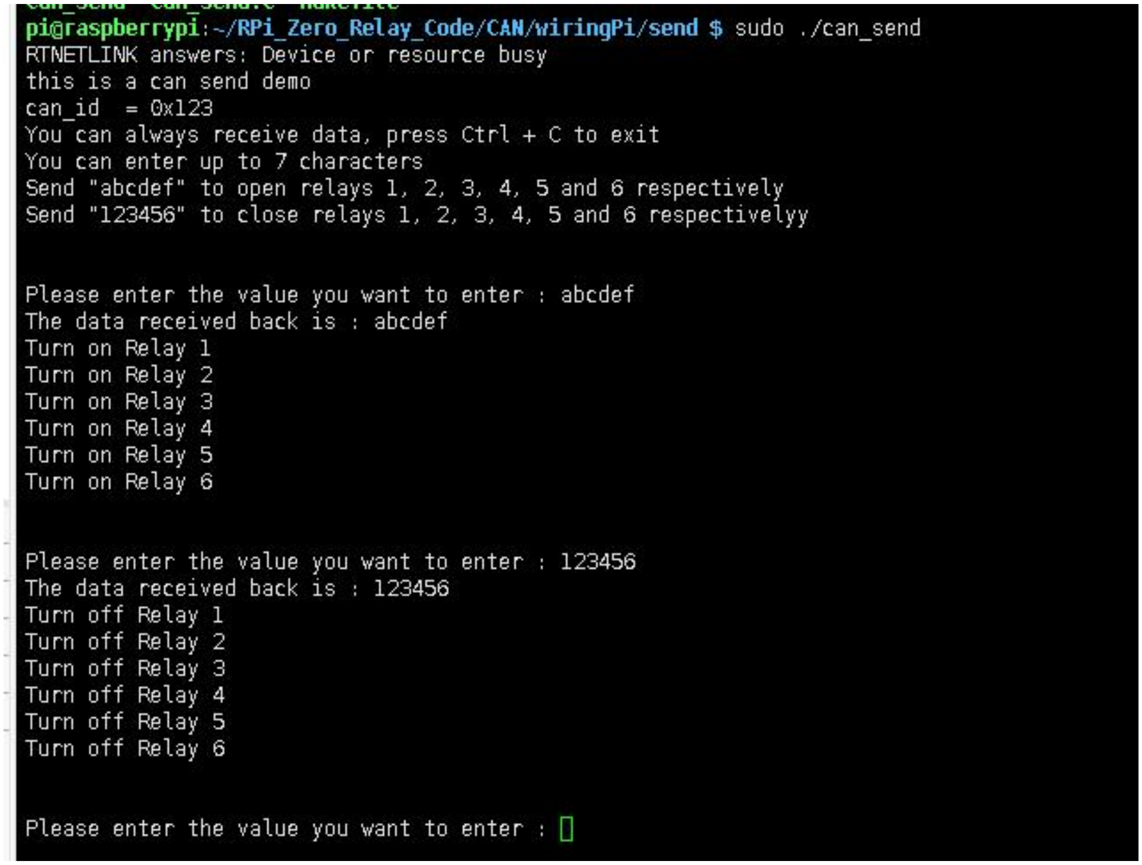
```
cd RPi_Zero_Relay_Code/CAN/wiringPi/send/
make clean
make
sudo ./can_send
```



```
pi@raspberrypi:~/RPi_Zero_Relay_Code/CAN/wiringPi/send $ sudo ./can_send
RTNETLINK answers: Device or resource busy
this is a can send demo
can_id  = 0x123
You can always receive data, press Ctrl + C to exit
You can enter up to 7 characters
Send "abcdef" to open relays 1, 2, 3, 4, 5 and 6 respectively
Send "123456" to close relays 1, 2, 3, 4, 5 and 6 respectivelyy


Please enter the value you want to enter :
```

(/wiki/
File:RPi_Zero_Relay_CAN_1.jpg)

Enter the corresponding value:

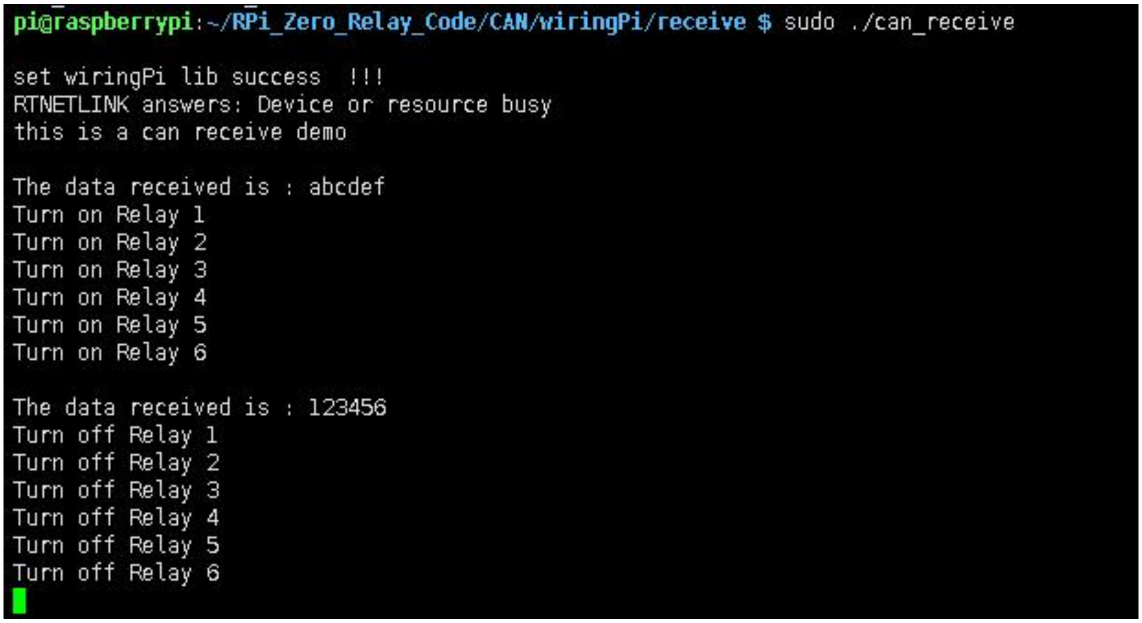


```
pi@raspberrypi:~/RPi_Zero_Relay_Code/CAN/wiringPi/send $ sudo ./can_send
RTNETLINK answers: Device or resource busy
this is a can send demo
can_id  = 0x123
You can always receive data, press Ctrl + C to exit
You can enter up to 7 characters
Send "abcdef" to open relays 1, 2, 3, 4, 5 and 6 respectively
Send "123456" to close relays 1, 2, 3, 4, 5 and 6 respectivelyy


Please enter the value you want to enter : abcdef
The data received back is : abcdef
Turn on Relay 1
Turn on Relay 2
Turn on Relay 3
Turn on Relay 4
Turn on Relay 5
Turn on Relay 6


Please enter the value you want to enter : 123456
The data received back is : 123456
Turn off Relay 1
Turn off Relay 2
Turn off Relay 3
Turn off Relay 4
Turn off Relay 5
Turn off Relay 6


Please enter the value you want to enter :
```

(/wiki/

File:RPi_Zero_Relay_CAN_3.jpg)

At this time, a packet with the corresponding ID is received:



```
pi@raspberrypi:~/RPi_Zero_Relay_Code/CAN/wiringPi/receive $ sudo ./can_receive

set wiringPi lib success  !!!
RTNETLINK answers: Device or resource busy
this is a can receive demo

The data received is : abcdef
Turn on Relay 1
Turn on Relay 2
Turn on Relay 3
Turn on Relay 4
Turn on Relay 5
Turn on Relay 6

The data received is : 123456
Turn off Relay 1
Turn off Relay 2
Turn off Relay 3
Turn off Relay 4
Turn off Relay 5
Turn off Relay 6
```

(/wiki/

File:RPi_Zero_Relay_CAN_4.jpg)

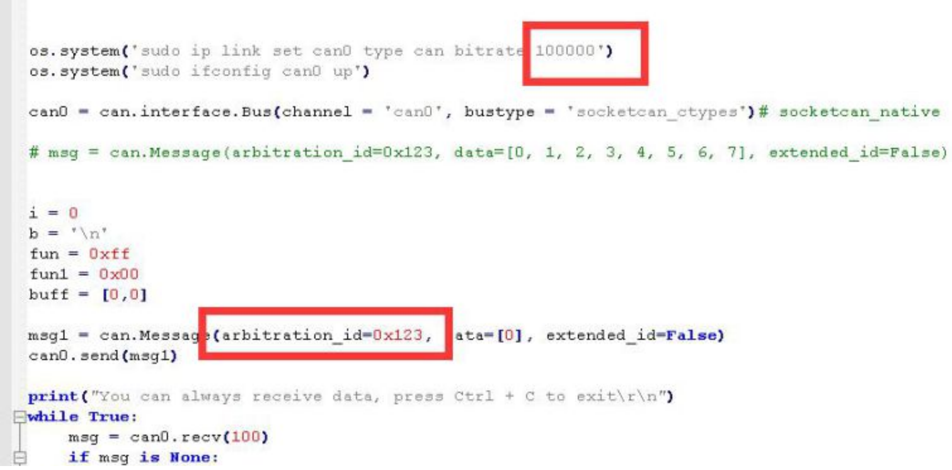
python

The Raspberry Pi opens the terminal and runs:

```
# Enter the corresponding directory
cd RPi_Zero_Relay_Code/CAN/python/

# The receiver runs the program first:
sudo python can_reveive.py

# After the receiver runs the program, the sender runs the program:
sudo python can_send.py
```

How to use with other CAN devices

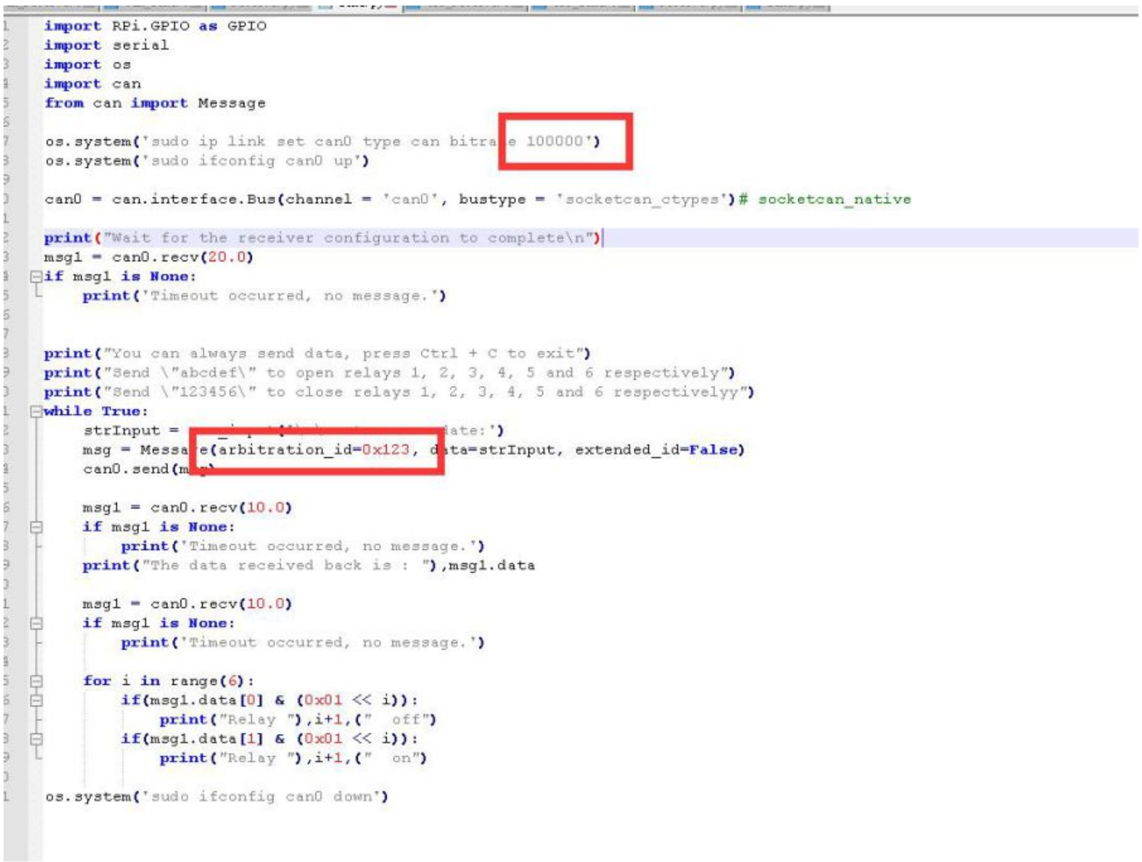
1. Make sure the hardware wiring is correct, i.e. H-H, L-L connection
2. Make sure that the baud rate settings on both sides are consistent, the demo baud rate is set to 100K by default
3. Ensure that the CAN IDs on both sides are consistent, otherwise reception will not be possible



```
os.system('sudo ip link set can0 type can bitrate 100000')
os.system('sudo ifconfig can0 up')

can0 = can.interface.Bus(channel = 'can0', bustype = 'socketcan_ctypes')# socketcan_native

# msg = can.Message(arbitration_id=0x123, data=[0, 1, 2, 3, 4, 5, 6, 7], extended_id=False)


i = 0
b = '\n'
fun = 0xff
fun1 = 0x00
buff = [0,0]

msg1 = can.Message(arbitration_id=0x123, ata=[0], extended_id=False)
can0.send(msg1)

print("You can always receive data, press Ctrl + C to exit\r\n")
while True:
    msg = can0.recv(100)
    if msg is None:
```

(/wiki/

File:RPi_Zero_Relay_CAN_10.jpg)



```
import RPi.GPIO as GPIO
import serial
import os
import can
from can import Message

os.system('sudo ip link set can0 type can bitrate 100000')
os.system('sudo ifconfig can0 up')

can0 = can.interface.Bus(channel = 'can0', bustype = 'socketcan_ctypes')# socketcan_native

print("Wait for the receiver configuration to complete\n")
msg1 = can0.recv(20.0)
if msg1 is None:
    print('Timeout occurred, no message.')


print("You can always send data, press Ctrl + C to exit")
print("Send \"abcdef\" to open relays 1, 2, 3, 4, 5 and 6 respectively")
print("Send \"123456\" to close relays 1, 2, 3, 4, 5 and 6 respectivelyy")
while True:
    strInput = [illegible]ate:')
    msg = Message(arbitration_id=0x123, data=strInput, extended_id=False)
    can0.send(msg)

    msg1 = can0.recv(10.0)
    if msg1 is None:
        print('Timeout occurred, no message.')
    print("The data received back is : "),msg1.data

    msg1 = can0.recv(10.0)
    if msg1 is None:
        print('Timeout occurred, no message.')

    for i in range(6):
        if(msg1.data[0] & (0x01 << i)):
            print("Relay "),i+1,("  off")
        if(msg1.data[1] & (0x01 << i)):
            print("Relay "),i+1,("  on")

os.system('sudo ifconfig can0 down')
```

(/wiki/

File:RPi_Zero_Relay_CAN_9.jpg)

4. If there is frame loss when sending data for a long time, you can try to reduce the baud rate to solve it

RS485 usage

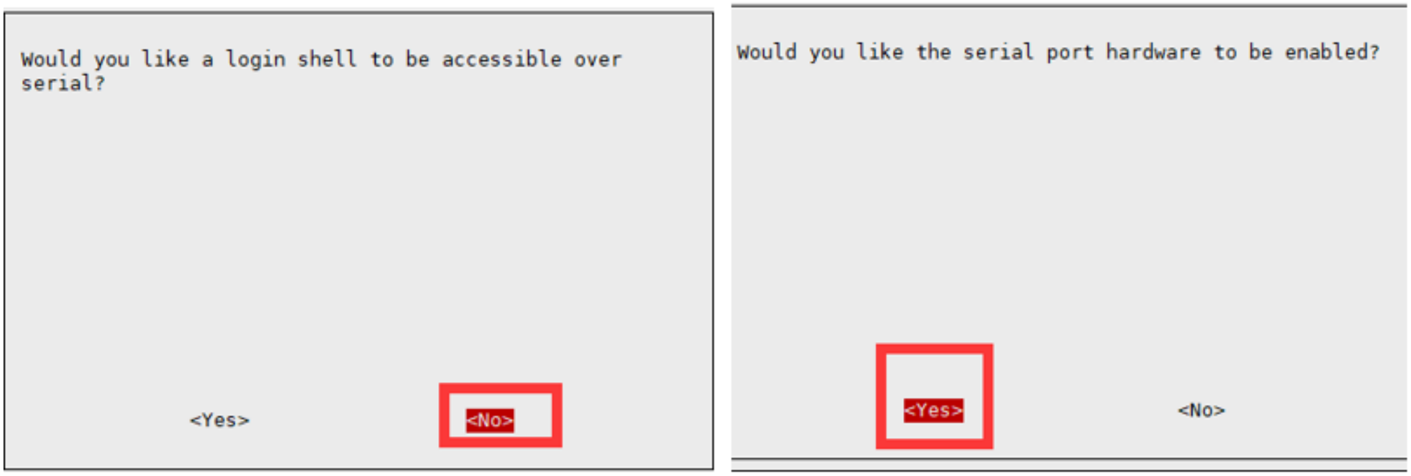
This demo program uses two RPi-Zero-Relay modules
Providing Python and wiringPi language demos

Preliminary Work

Enable UART Interface

Open the Raspberry Pi terminal and enter the following command to enter the configuration interface

```
sudo raspi-config
Select Interfacing Options -> Serial, disable shell access, and enable the hardware seria
l port
```



Would you like a login shell to be accessible over serial?

Would you like the serial port hardware to be enabled?

(/wiki/File:L76X_GPS_Module_rpi_serial.png)

Open the /boot/config.txt file, find the following configuration statement to enable the serial port, if not, you can add it to the end of the file.

```
enable_uart=1
```

Then restart Raspberry Pi:

```
sudo reboot
```

Make sure both Raspberry Pis are processed in the same way, then connect the A of one module to the B of the other module

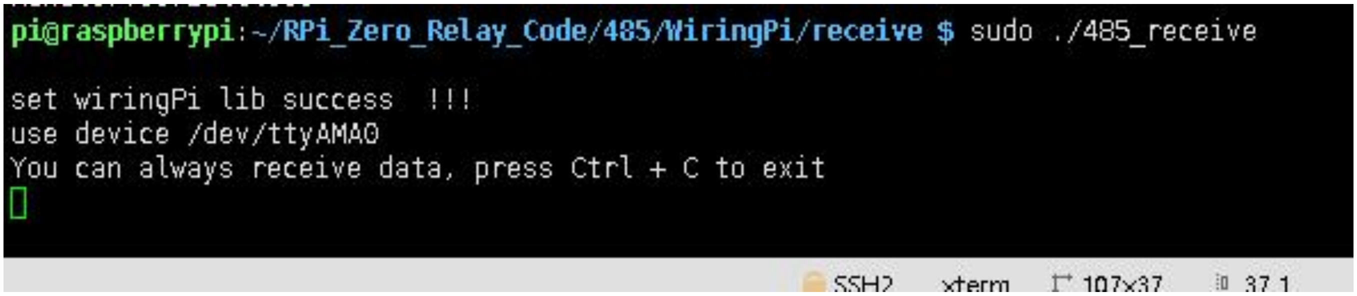
If you are using other 485 devices, make sure to connect A-A, B-B

C

- Block reception, open the terminal on the Raspberry Pi and run:

```
cd RPi_Zero_Relay_Code/485/WiringPi/receive
make clean
make
sudo ./485_receive
```

The receiving program is blocked.

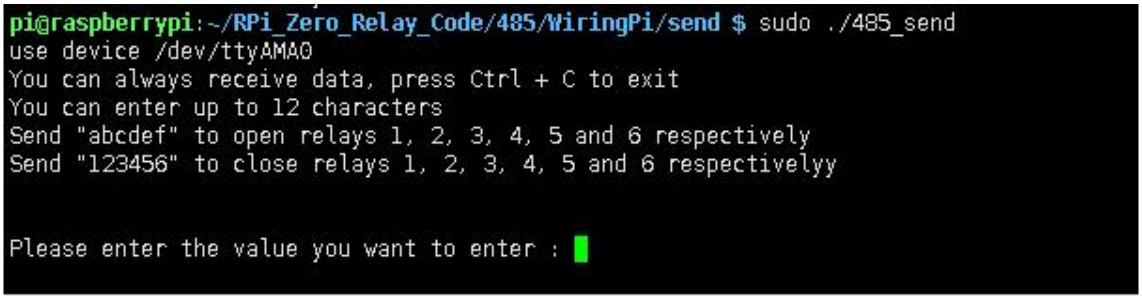


```
pi@raspberrypi:~/RPi_Zero_Relay_Code/485/WiringPi/receive $ sudo ./485_receive

set wiringPi lib success  !!!
use device /dev/ttyAMA0
You can always receive data, press Ctrl + C to exit
```

(/wiki/File:RPi_Zero_Relay_485_2.jpg)

- Send, the Raspberry Pi opens the terminal and runs:

```
cd RPi_Zero_Relay_Code/485/WiringPi/send
make clean
make
sudo ./485_send
```

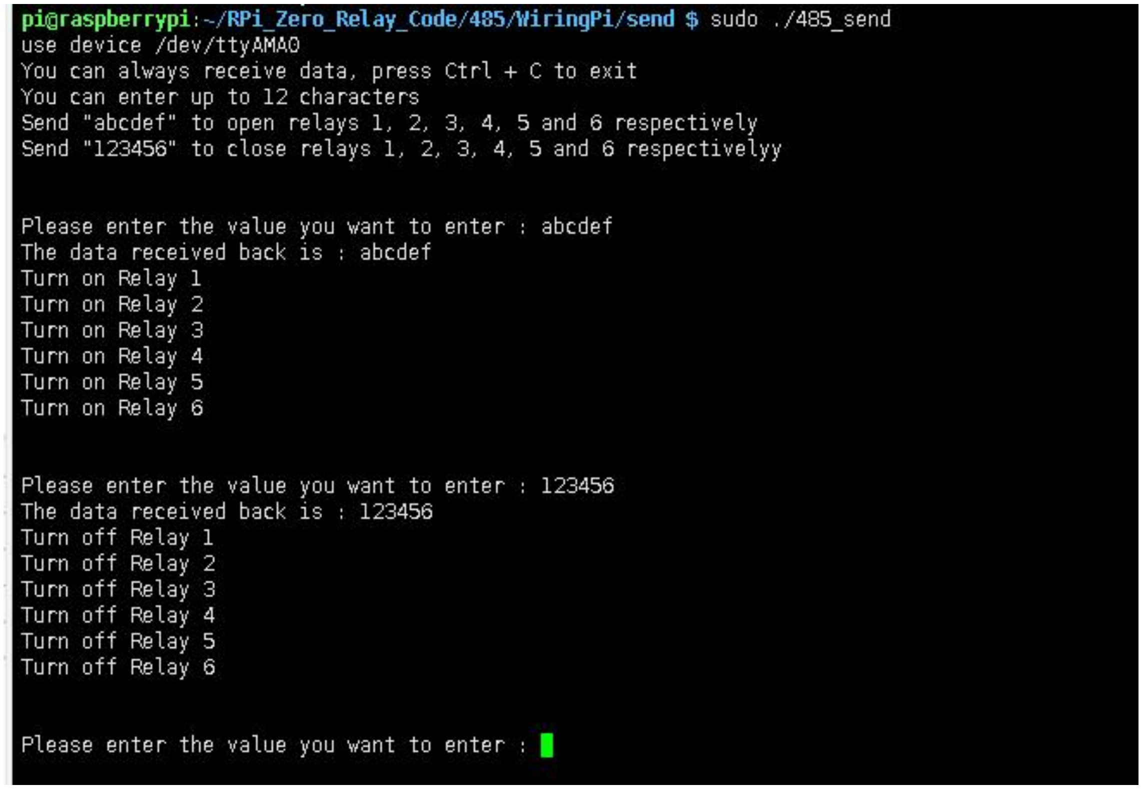


```
pi@raspberrypi:~/RPi_Zero_Relay_Code/485/WiringPi/send $ sudo ./485_send
use device /dev/ttyAMA0
You can always receive data, press Ctrl + C to exit
You can enter up to 12 characters
Send "abcdef" to open relays 1, 2, 3, 4, 5 and 6 respectively
Send "123456" to close relays 1, 2, 3, 4, 5 and 6 respectivelyy


Please enter the value you want to enter :
```

(/wiki/

File:RPi_Zero_Relay_485_1.jpg)

Enter the corresponding value:

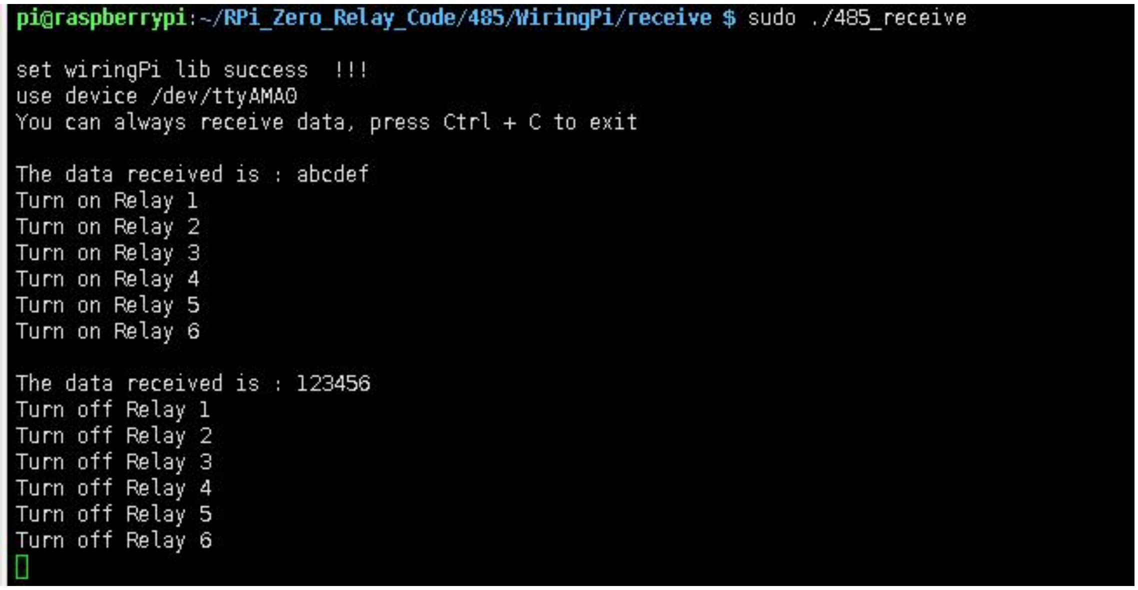


```
pi@raspberrypi:~/RPi_Zero_Relay_Code/485/WiringPi/send $ sudo ./485_send
use device /dev/ttyAMA0
You can always receive data, press Ctrl + C to exit
You can enter up to 12 characters
Send "abcdef" to open relays 1, 2, 3, 4, 5 and 6 respectively
Send "123456" to close relays 1, 2, 3, 4, 5 and 6 respectivelyy


Please enter the value you want to enter : abcdef
The data received back is : abcdef
Turn on Relay 1
Turn on Relay 2
Turn on Relay 3
Turn on Relay 4
Turn on Relay 5
Turn on Relay 6


Please enter the value you want to enter : 123456
The data received back is : 123456
Turn off Relay 1
Turn off Relay 2
Turn off Relay 3
Turn off Relay 4
Turn off Relay 5
Turn off Relay 6


Please enter the value you want to enter :
```

(/wiki/

File:RPi_Zero_Relay_485_3.jpg)

At this time, the receiver receives the program



```
pi@raspberrypi:~/RPi_Zero_Relay_Code/485/WiringPi/receive $ sudo ./485_receive

set wiringPi lib success  !!!
use device /dev/ttyAMA0
You can always receive data, press Ctrl + C to exit

The data received is : abcdef
Turn on Relay 1
Turn on Relay 2
Turn on Relay 3
Turn on Relay 4
Turn on Relay 5
Turn on Relay 6

The data received is : 123456
Turn off Relay 1
Turn off Relay 2
Turn off Relay 3
Turn off Relay 4
Turn off Relay 5
Turn off Relay 6
```

(/wiki/File:RPi_Zero_Relay_485_4.jpg)

python demo

```
# Enter the corresponding directory
cd RPi_Zero_Relay_Code/485/python/

# The receiver runs the program first
sudo python receive.py

# After the receiver runs the program, the sender runs the program
sudo python send.py
```

Troubleshooting

If the 485 communication is not normal, debug step by step:

1. Confirm if serial login to the Raspberry Pi shell is blocked;
2. Determine whether A,B of 485 correspond to A,B of controlled 485 equipment one by one;
3. You can start by using a USB to 485 device to communicate with the RPi Zero Relay, ensuring that the Raspberry Pi's setup is correct;

Resources

Document

- RPi Zero Relay schematic diagram (https://files.waveshare.com/wiki/RPi%20Zero%20Relay/RPi_Zero_Relay%20schematic%20diagram.pdf)

Demo

- Demo (https://files.waveshare.com/wiki/RPi%20Zero%20Relay/RPi_Zero_Relay_Code.7z)

Datasheets

- SP3485 Datasheet (https://files.waveshare.com/wiki/RPi%20Zero%20Relay/SP3481_SP3485%20Datasheet.pdf)
- SN65HVD230 Datasheet (https://files.waveshare.com/wiki/RPi%20Zero%20Relay/SN65HVD230%20Datasheet.pdf)
- MCP2515 Datasheet (https://files.waveshare.com/wiki/RPi%20Zero%20Relay/MCP2515%20Datasheet.pdf)

FAQ

Question: In the python virtual environment, it is prompted that there is no corresponding library

Answer:

Refer to the Install libraries (https://www.waveshare.com/wiki/RPi_Zero_Relay#Install_libraries) section to install the corresponding library

Question: The corresponding library is installed in the Python virtual environment, but the execution error indicates that there is no corresponding function

Answer:

There is a problem with the corresponding library, such as incomplete installation, etc.;
Uninstall the corresponding library file and reinstall it
For example:

```
# Uninstall pyserial
pip3 uninstall pyserial
# Install pyserial
pip3 install pyserial
```

Question: When running the program in a Python virtual environment, a

warning appears that there is no corresponding library file

Answer:

This warning does not affect the execution of the program, but the corresponding function in the library does not find the corresponding dependency, and our demo does not actually use this part;
If you want to dismiss these warnings, just install the corresponding library
For example:

```
# Install lgpio
pip3 install lgpio
# Install RPi.GPIO
pip3 install RPi.GPIO
```

Support

Technical Support

If you need technical support or have any feedback/review, please click the **Submit Now** button to submit a ticket, Our support team will check and reply to you within 1 to 2 working days. Please be patient as we make every effort to help you to resolve the issue.
Working Time: 9 AM - 6 PM GMT+8 (Monday to Friday)

Submit Now (https://service.waveshare.com/)

Retrieved from "https://www.waveshare.com/w/index.php?title=RPi_Zero_Relay&oldid=94074 (https://www.waveshare.com/w/index.php?title=RPi_Zero_Relay&oldid=94074)"