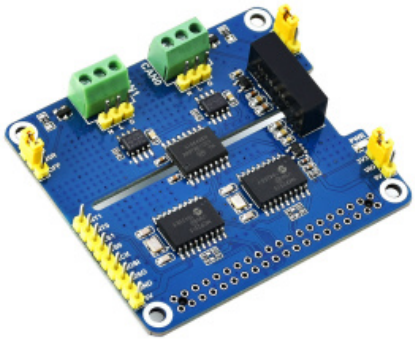
2-CH CAN HAT

来自Waveshare Wiki
跳转至： 导航、 搜索



功能简介		
树莓派传感器扩展		
	CAN	RPi

说明

产品概述

2-CH CAN HAT 是专为树莓派开发的一款隔离式的双通道CAN 通信功能的扩展板，具备有多重保护。

特点

- 基于树莓派设计，适用于Raspberry Pi Zero/Zero W/Zero WH/2B/3B/3B+/4B
- 采用MCP2515与 SI65HVD230DR(或SN65HVD230) 双芯片组合方案，支持2路CAN接口通信
- 板载一体式电源隔离，可提供稳定的隔离电压，隔离端无须额外供电
- 板载数字隔离芯片，信号隔离通信更安全、稳定性更好、抗干扰性更强
- 板载SM24CANB(瞬态电压抑制管)，防静电和瞬态尖峰电压
- 板载电平转换电路，可通过跳线帽切换3.3V/5V的工作电平
- 板载120Ω终端电阻，可通过跳线帽设置使能
- 引出SPI控制接口，方便接入STM32/Arduino等主控板
- 提供完善的配套资料手册及例程

产品参数

- 输入电压： 5V
- 逻辑电压： 3.3V/5V
- CAN控制芯片： MCP2515
- CAN收发器： SI65HVD230DR (或SN65HVD230)
- 产品尺寸： 65.0x56.5mm
- 固定孔通经： 3.0mm

接口说明

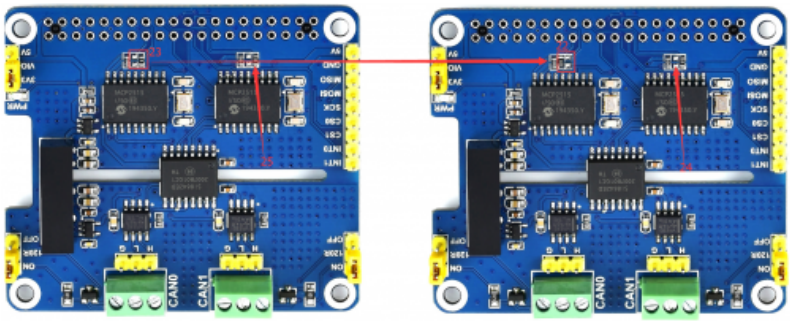
CAN接口总线

功能引脚	树莓派接口 (BCM)	树莓派接口 (WPI)	描述
5V	5V	5V	5V电源正
GND	GND	GND	电源地
MISO	9(MISO)	13(MISO)	SPI时钟输入
MOSI	10(MOSI)	12(MOSI)	SPI数据输入
SCK	11(SCLK)	14(SCLK)	SPI数据输出
CS_0	8(CE0)	10(CE0)	CAN_0片选
INT_0	23（默认）/22	6（默认）/5	CAN_0中断输出
CS_1	7(CE1)	11(CE1)	CAN_1片选
INT_1	25（默认）/24	4（默认）/3	CAN_1中断输出

说明:

INT_0默认焊接的是PIN23，INT_1默认焊接的是PIN25，如果需要修改引脚，需要对PCBA上对应的焊点做改动，并且把/boot/config.txt对应的引脚也修改下。

以修改INT_0为例：如果由PIN23修改为PIN22，那么对应的下图左的板子红框的焊点，需要修改成下图右的板子的焊点



同样的，如果需要把INT_1从默认的PIN25，修改成PIN24，则需要把PIN25箭头处的0欧改焊接到PIN24处。

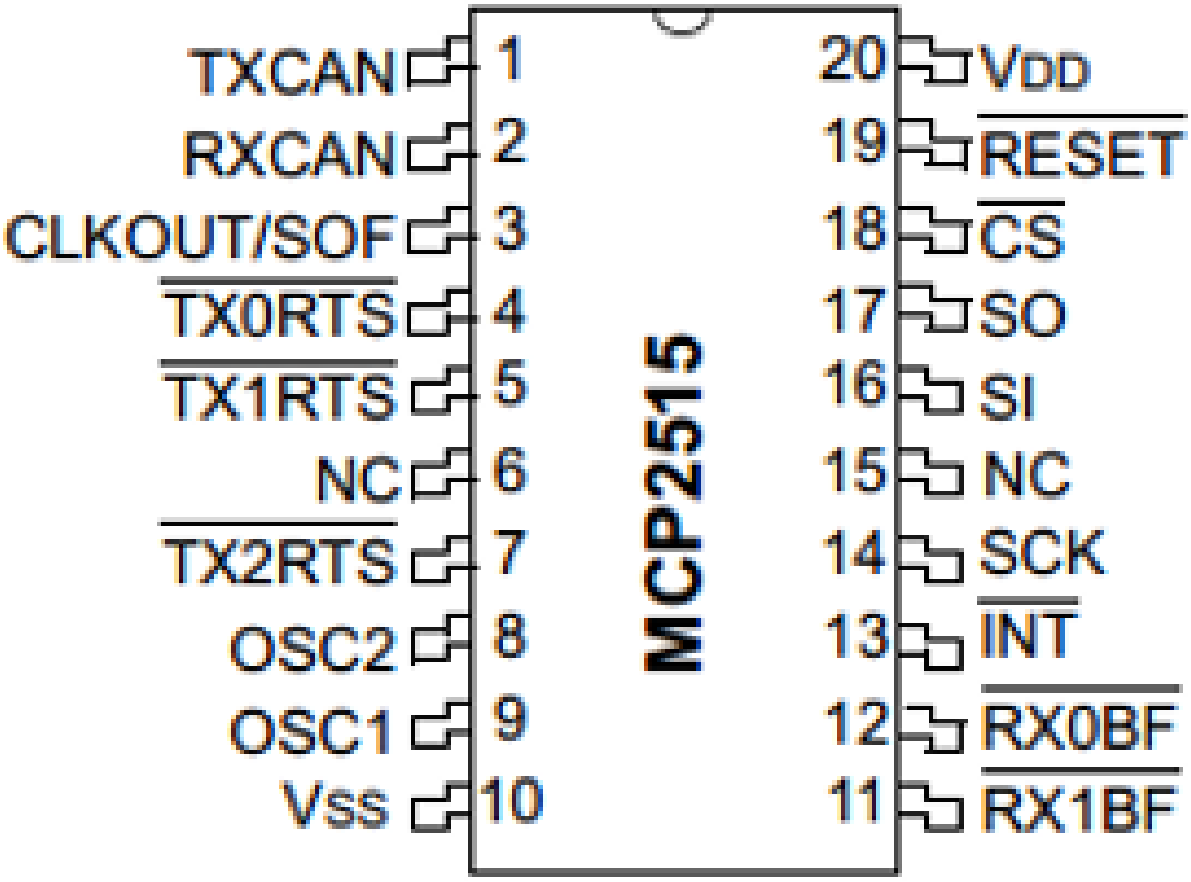
CAN总线

CAN 模块的功能是处理所有 CAN 总线上的报文接收和发送。报文发送时，首先将报文装载到正确的报文缓冲器和控制寄存器中。通过 SPI 接口设置控制寄存器中的相应位或使用发送使能引脚均可启动发送操作。通过读取相应的寄存器可以检查通讯状态和错误。 会对在 CAN总线上检测到的任何报文进行错误检查，然后与用户定义的滤波器进行匹配，以确定是否将报文移到两个接收缓冲器中的一个。

由于树莓派本身并不支持CAN总线，因此使用SPI接口的CAN控制器，搭配一个收发器完成CAN功能。

Microchip 的 MCP2515 是一款CAN协议控制器，完全支持 CAN V2.0B 技术规范。该器件能发送和接收标准和扩展数据帧以及远程帧。 MCP2515 自带的两个验收屏蔽寄存器和六个验收滤波寄存器可以过滤掉不想要的报文，因此减少了主单片机（MCU）的开销。MCU通过SPI接口与该器件连接,即树莓派通过SPI接口连接芯片，对于树莓派使用该芯片不需要编

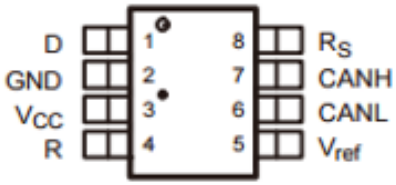
写驱动，只需要打开设备树中的内核驱动即可使用。



更多详细请参考数据手册；

说明：自2021年10月份后，因SN65HVD230芯片缺货涨价，芯片更换成 SI65HVD230，软硬件功能兼容。

SN65HVD230 是德州仪器公司生产的 3.3V CAN 收发器，该器件适用于较高通信速率、良好抗干扰 能力和高可靠性 CAN 总线的串行通信。SN65HVD230 具有高速、斜率和等待 3 种不同的工作模式。 其工作模式控制可通过 Rs 控制引脚来实现。CAN 控制器的输出引脚 Tx 接到 SN65HVD230 的数据 输入端 D，可将此 CAN 节点发送的数据传送到 CAN 网络中；而 CAN 控制器的接收引脚 Rx 和 SN65HVD230 的数据输出端 R 相连，用于接收数据。

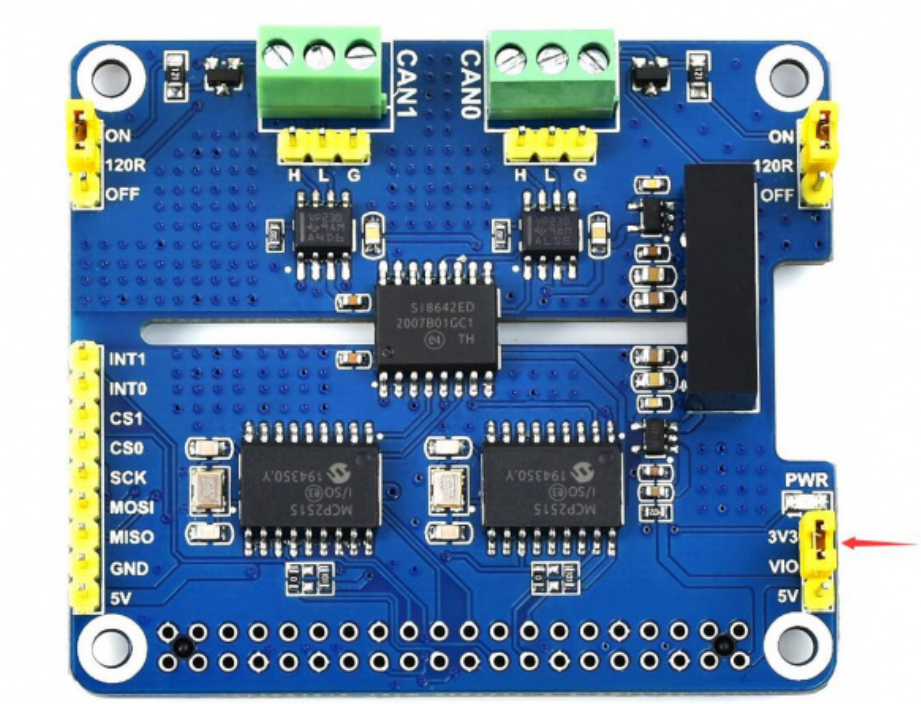


SN65HVD230 引脚
（顶视图）

引脚	名称	说明
1	D	驱动输入 Driver input
2	GND	电源地线 Ground
3	V_{CC}	电源线 Supply voltage
4	R	接收输出 Receiver output
5	V_{ref}	参考输出 Reference output
6	CANL	低总线输出 Low bus output
7	CANH	高总线输出 High bus output
8	RS	工作模式控制端 Standby/slope control

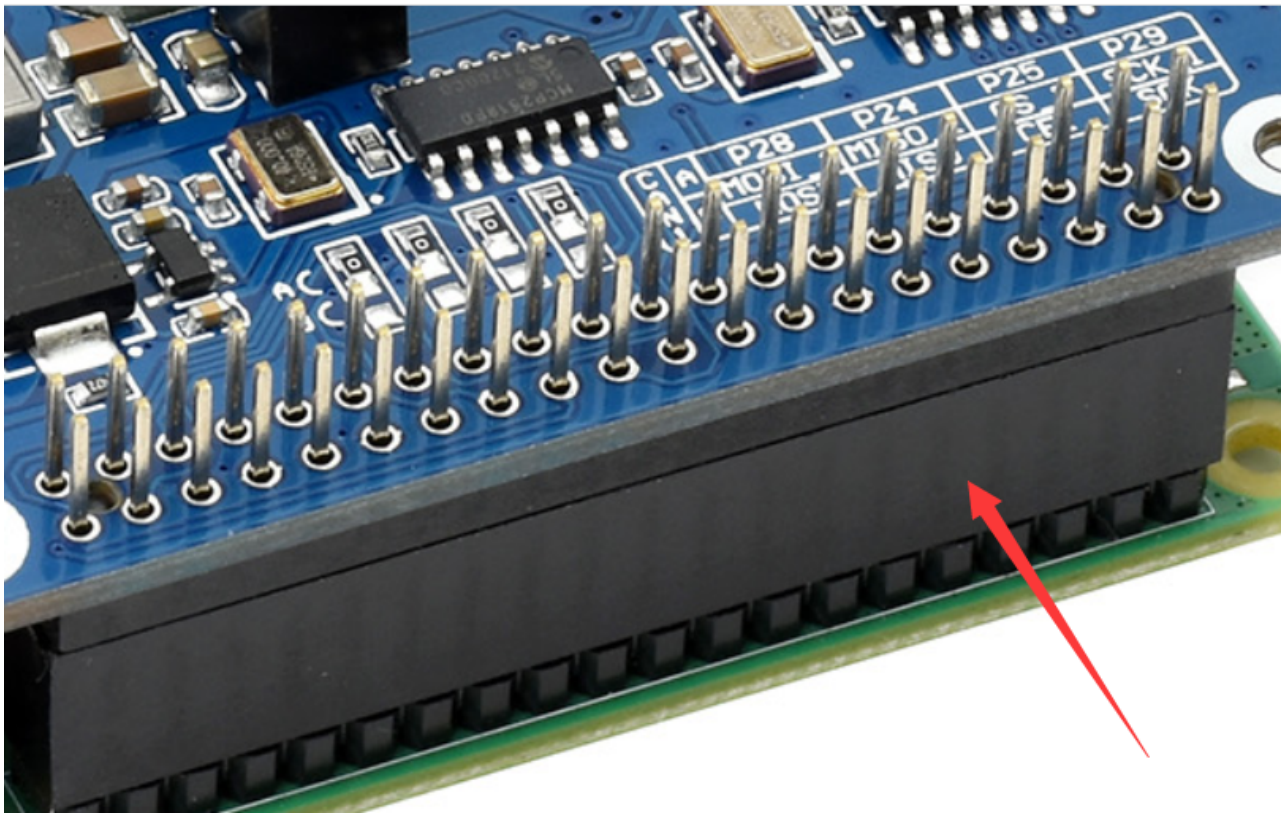
树莓派使用

在使用树莓派演示该例程时，需注意的是由于树莓派属于3.3V逻辑系统，需改变将2-CH CAN HAT的逻辑电压引脚跳帽到3.3V，如下图所示。



注意：接入树莓派2b，3b和4b板子使用时，请接上增高座和尼龙柱固定，避免CAN接线端子处背面触碰到HDMI接口导致短路，避免接错或接触不良：

接入树莓派使用时，必须加上增高座，然后排针穿过底板，效果如下：



安装必要的函数库

- 安装BCM2835， 打开树莓派终端，并运行一下指令

```
wget http://www.airspayce.com/mikem/bcm2835/bcm2835-1.60.tar.gz
tar zxvf bcm2835-1.60.tar.gz
cd bcm2835-1.60/
sudo ./configure
sudo make
sudo make check
sudo make install
# 更多的可以参考官网：http://www.airspayce.com/mikem/bcm2835/
```

- 安装wiringPi

```
sudo apt-get install wiringpi
#对于树莓派4B可能需要进行升级：
wget https://project-downloads.drogon.net/wiringpi-latest.deb
sudo dpkg -i wiringpi-latest.deb
gpio -v
# 运行gpio -v会出现2.52版本，如果没有出现说明安装出错
```

- 安装Python函数库

```
#python2
sudo apt-get update
sudo apt-get install python-pip
sudo apt-get install python-pil
sudo apt-get install python-numpy
sudo pip install RPi.GPIO
sudo pip install spidev
sudo pip install python-can
#python3
sudo apt-get update
sudo apt-get install python3-pip
sudo apt-get install python3-pil
sudo apt-get install python3-numpy
sudo pip3 install RPi.GPIO
sudo pip3 install spidev
sudo pip3 install python-can
```

- 在官网上找到对应产品，在产品资料打开下载路径，在wiki中下载示例程序：
- 得到解压包，解压并将程序复制到树莓派中。

前置工作

开启SPI接口

- 打开树莓派终端，输入以下指令进入配置界面

```
sudo raspi-config
选择Interfacing Options -> SPI -> Yes 开启SPI接口
```

```
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
```

```
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
```

```
Would you like the SPI interface to be enabled?

                <Yes>                 <No>
```

然后重启树莓派：

```
sudo reboot
```

请确保SPI没有被其他的设备占用，你可以在/boot/config.txt中间检查

修改脚本config.txt

将模块插在树莓派上，然后修改开机脚本config.txt

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

在最后一行加入如下：

```
dtparam=spi=on
dtoverlay=mcp2515-can1,oscillator=16000000,interrupt=25
dtoverlay=mcp2515-can0,oscillator=16000000,interrupt=23
```

- 重启树莓派以应用所有设置：

```
sudo reboot
```

- 待树莓派重启后，查看SPI信息：

```
dmesg | grep spi0
```

```
pi@raspberrypi:~ $ dmesg | grep spi
[    5.352796] mcp251x spi0.0 can0: MCP2515 successfully initialized.
[    5.371104] mcp251x spi0.1 can1: MCP2515 successfully initialized.
```

- 开启CAN：

```
sudo ip link set can0 up type can bitrate 1000000
sudo ip link set can1 up type can bitrate 1000000
sudo ifconfig can0 txqueuelen 65536
sudo ifconfig can1 txqueuelen 65536
```

- 更多相关CAN内核指令可以查看：

```
https://www.kernel.org/doc/Documentation/networking/can.txt
```

- 查看ifconfig：

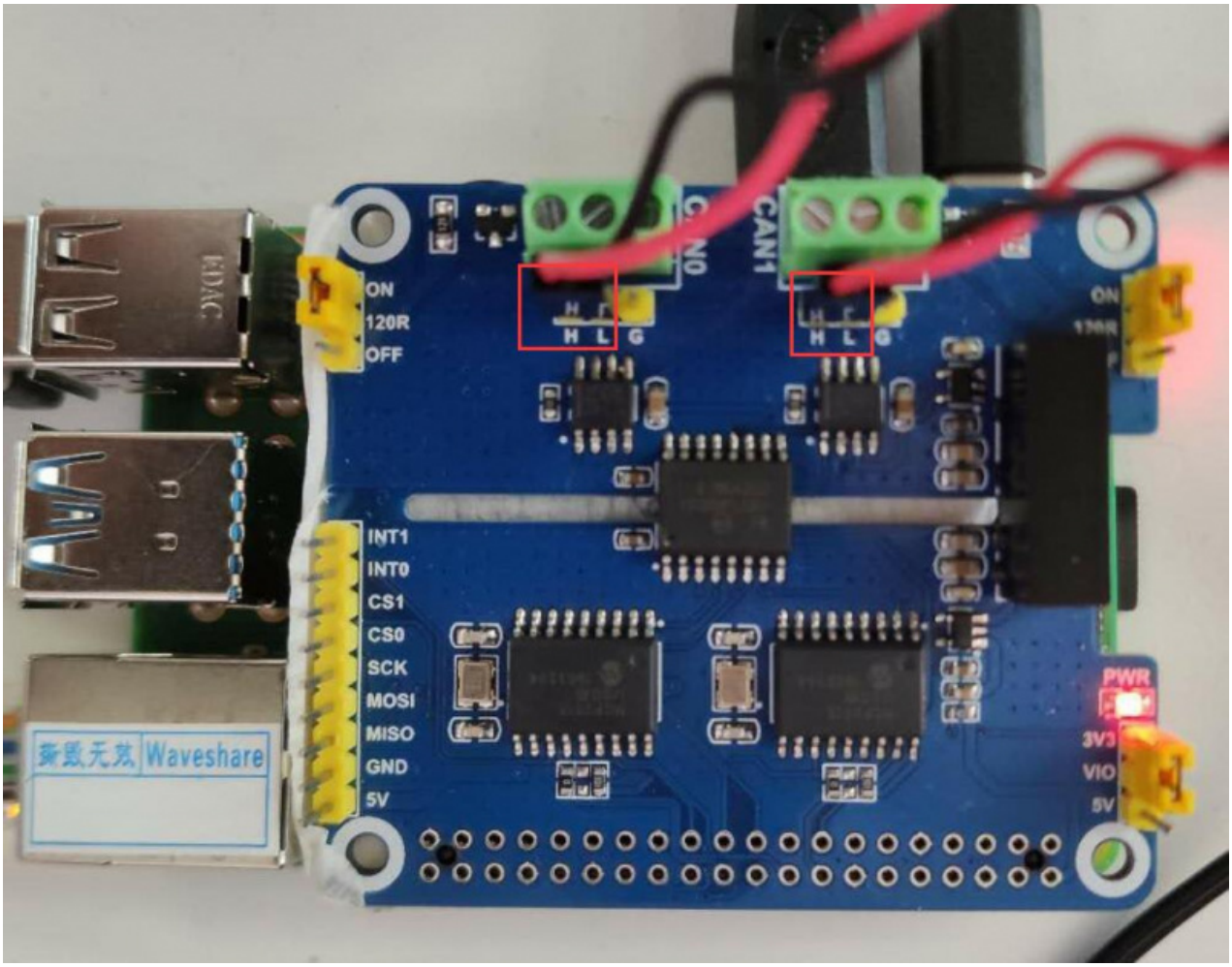
```
ifconfig
```

```
pi@raspberrypi:~ $ ifconfig
can0: flags=193<UP,RUNNING,NOARP>  mtu 72
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 65536  (UNSPEC)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

can1: flags=193<UP,RUNNING,NOARP>  mtu 72
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 65536  (UNSPEC)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

开始测试

若手上只有一个2-CH CAN HAT，可以将可通过将模块的CAN0_H与CAN1_H，CAN0_L与CAN1_L相连，如下图所示：



- 安装can-utils:

```
sudo apt-get install can-utils
```

- 打开两个终端窗口：

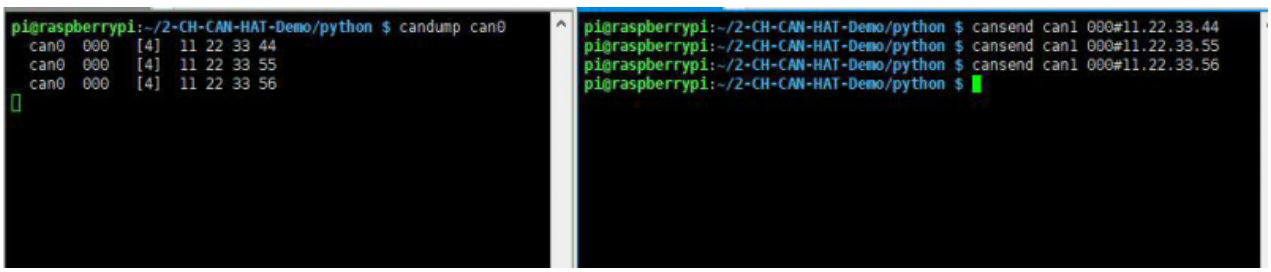
其中一个终端输入接收CAN0数据指令：

```
candump can0
```

另外一个终端输入发送CAN1数据指令：

```
cansend can1 000#11.22.33.44
```

- 演示效果如下：（左边为接收，右边为发送）



若手上有两个2-CH CAN HAT，可以直接将CAN_H，CAN_L两两相连。效果跟上述一样，需注意匹配好通信速率，识别ID，输出接口序号。

Python例程

- 浏览目录：
- 接收端运行receive.py：

```
sudo python reveive.py
```

- 发送端运行send.py：

```
sudo python send.py
```

需注意的是这里的发送端是使用CAN1发送，接收端则是CAN0，具体修改可以参照下面的代码分析。

代码分析

【Python例程】

本例程是基于python平台，确保以及安装了python-can库

在发送之前要先创建一个can设备，因为前面只是启用MCP2515内核：

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

上面这条则是通过对CAN0的配置初始化并启用，且指定CAN0作为发送/接收接口。如需改成CAN1，则代码如下：

```
os.system('sudo ip link set can1 type can bitrate 100000')
os.system('sudo ifconfig can1 up')
```

- 第一步：连接到CAN总线

```
can0 = can.interface.Bus(channel = 'can0', bustyp = 'socketcan_ctypes')
```

- 需改成CAN1，则代码如下：

```
can0 = can.interface.Bus(channel = 'can1', bustyp = 'socketcan_ctypes')
```

- 第二步：创建信息

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

- 第三步：发送信息

```
can0.send(msg)
```

- 需改成CAN1，则代码如下：

```
can1.send(msg)
```

- 最后同样要关闭can设备

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

- 需改成CAN1，则代码如下：

```
os.system('sudo ifconfig can1 down')
```

- 接收数据：

```
msg = can0.recv(10.0)
```

recv()中定义超时接收时间。

更多请参考：https://python-can.readthedocs.io/en/stable/interfaces/socketcan.html

【WringPi例程】

- 阻塞接收，树莓派打开终端，运行：

```
cd 2-CH_CAN_HAT_Code/wiringPi/receive/
make clean
sudo make
sudo ./can_receive
```

- 发送，树莓派打开终端，运行：

```
cd 2-CH_CAN_HAT_Code/ wiringPi/receive/
make clean
sudo make
sudo ./can_send
```

资料

文档

- 原理图

程序

- 示例程序

3D图纸

- 2-CH CAN HAT 3D图纸

相关资料

- 2-CH CAN HAT简单测试演示视频
- SN65HVD230
- MCP2515

FAQ

问题： 是否支持用在树莓派Ubuntu系统上？？

本产品上述的使用说明和驱动安装目前只支持Raspbian系统，暂不支持其他系统

问题： CAN 驱动安装配置失败?

试试更新下内核版本看看，指令如下：

```
sudo apt update
sudo apt upgrade
uname -a
```

问题： CAN初始化失败？？

如果出现初始化失败，可以重启树莓派或者其他主控平台开发板，确保连线是否正确，参考wiki仔细检查下配置是否有漏操作的，及逻辑电压跳帽引脚是否选择正确。

问题： 检查下硬件连接，是否存在接触不良。检查下软件配置，是否有开启SPI等等。

问题：CAN接口的G引脚是否连接?

G为信号地，也称隔离地。在工业环境当中，CAN受周围环境多变，为确保稳定性，两通信模块的G需要相连接。

问题：CAN通信速度达不到最高值?

芯片受周围所测环境、通信距离、线材、软件等多因素的影响。在高速通信时，数据波特率可能达不到所所标称的最高熟虑，用户需根据实际测量，确保稳定，选择适合的通信速度。

问题：CAN不能正常通信?

若初始化成功，且速度设置合适后，仍不可以正常通信。请检查下两块2-CH CAN HAT 的CAN_H是否与另外一块CAN_H相连，CAN_L是否与另外一块CAN_L相连，需注意的是这两条线不能接反，H对应H，L对应L。若没问题，请检查程序配置的CAN口是否正确，比如需要用到CAN0，程序代码却配置CAN1。同时也需要检查硬件是否连接正确且CAN接口存不存在短

路问题，如下图所示，CAN接口与树莓派的HDMI接口误触引起短路，使得CAN不能正常通信，这时需要使用购买2-CH CAN HAT时配送的配件2×20PIN长排座进行增高处理。800px