Interfacing TF03-CAN Single Point LiDAR with Raspberry Pi HAT

Written by: Ibrahim (FAE)

www.benewake.com
Benewake (Beijing) Co., Ltd.

This article explains how to interface Benewake TF03-CAN single point LiDAR with Raspberry Pi 485-CAN HAT. The HAT is compatible with Raspberry Pi Zero/Zero W/ZeroWH/2B/3B/3B+/4B. The following assumptions are made while writing this article:

1. The end-user has Raspberry Pi single board computer and 485-CAN HAT [1] (manufactured by Waveshare). I have added four different links from where 485-CAN HAT can be purchased
2. Benewake TF03-UART single point LiDAR in hands
3. Raspbian operating system. At the time of writing this article the version *2021-03-04-raspios-buster-armhf*[2] was tested
4. All the necessary cables and 5V power supply. New version of TF03-UART/CAN version supports wide voltage range 5-24V
5. The default interface mode of LiDAR is UART so we first need to switch the interface from UART to CAN by sending commands: 5A 05 45 02 A6 (**switch to CAN**) and 5A 04 11 6F (**save settings**). These commands need to be sent using TTL-USB adapter or using UART port of Raspberry Pi and writing Python script or some serial port tools.
6. Any editor for writing and modifying the script (VS Code is recommended)
7. **UART port of Raspberry Pi is enabled**:
   I. Start raspi-config(in terminal type): `sudo raspi-config`.
   II. Select option 3 - Interface Options.
   III. Select option P6 - Serial Port.
   IV. At the prompt "`Would you like a login shell to be accessible over serial"?` answer is '**No**'
   V. At the prompt "`Would you like the serial port hardware to be enabled"?` answer is '**Yes**'
   VI. Exit **raspi-config** and don't reboot at this moment
   VII. [**Configure Raspberry Pi system file**]: Open /boot/config.txt (using any text editor can be used like gedit, vim, nano etc.) in terminal and add the following line: : `enable_uart=1`
   For Raspberry Pi 3B comment out: `#dtoverlay=pi3-miniuart-bt` and reboot the Pi for changes to take effect.

[1] https://www.waveshare.com/product/raspberry-pi/rs485-can-hat.htm
https://www.ebay.com/itm/263753270299
https://www.cytron.io/p-rs485-can-hat-for-raspberry-pi
https://botland.store/raspberry-pi-hat-connection/12532-rs485-can-hat-overlay-for-raspberry-pi-waveshare-14882-5904422319502.html

[2] https://downloads.raspberrypi.org/raspios_armhf/images/raspios_armhf-2021-03-25/2021-03-04-raspios-buster-armhf.zip

Pin configuration of 485-CAN HAT:

We are only interested in two pins of HAT which are **H** and **L** as only these two pins are needed for CAN communication. The HAT can directly be inserted into 40-pin header of Raspberry Pi. In this tutorial we are not discussing other pins. The following image shows CAN pins marked in red:

Figure 1: Pinouts

Schematic diagram:

The connection diagram for interfacing TF03 LiDAR to HAT and using a separate power supply is given below:

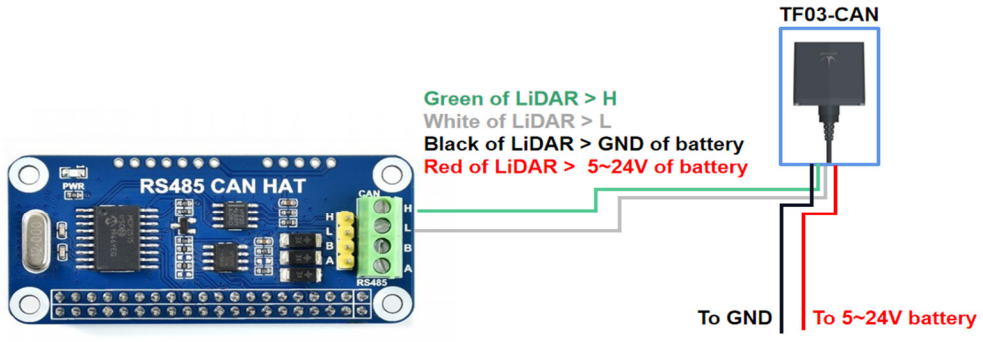


Figure 2: Connection Diagram

For exact details about LiDAR current rating and your supply capacity, please refer to their respective data-sheets. It should be noted that if you are using the same power supply for LiDAR and Raspberry Pi + HAT, then your supply should be able to provide enough current to all these devices otherwise some parts may behave abnormally.

Default CAN Communication Parameters of TF03:

According to the data-sheet of TF03 LiDAR the parameters of CAN communication:

Item	Content
Communication protocol	CAN
Baud rate	1M
Receive ID	0x3003
Transmit ID	0x3
Frame format	The default transmit frame is standard frame. Receive frame supports standard and extended frame.

Figure 3: CAN Communication

So CAN communication in code script is established based on this information. The baud rate and other parameters must match on two sides (HAT and LiDAR) otherwise communication cannot be established.

Cable and Connector Selection:

All Benewake Single Points LiDARs have either 4-pin or 7pin (for TF03) Molex connector, so the required mating connector is JST socket 1.25mm. Please refer to the link[3]. You can choose either 4-pin or 7-pin depending upon the LiDAR that you have. On the HAT side will need male-DuPont connector or any other suitable male connector that can be inserted into HAT:

[3] https://www.aliexpress.com/i/4000580232835.html

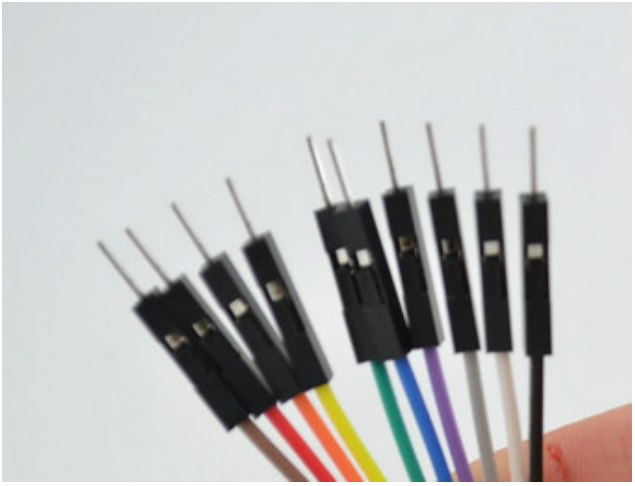


Figure 4: Male DuPont cable

Table III: Data format (for TF03 Series)

Data bit	Definition	Description
Byte0	DIST_L	DIST low 8-bits
Byte1	DIST_H	DIST high 8-bits
Byte2	Strength_L	Signal strength low 8-bits
Byte3	Strength_H	Signal strength high 8-bits
Byte4	Reserved bit	/
Byte5	Reserved bit	/

Setting Development Environment:

1. [Installing BCM2835]: in terminal type the following commands step by step:

```
$ 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
```

2. [Installing Python libraries]: Type the following commands.

```
$ sudo pip3 install pillow
$ sudo pip3 install numpy
$ sudo apt-get install libopenjp2-7
$ sudo apt install libtiff
$ sudo apt install libtiff5
$ sudo apt-get install libatlas-base-dev
```

I. [For Python2]:

```
$ sudo apt-get update
$ sudo apt-get install python-pip
$ sudo pip install RPi.GPIO
$ sudo pip install smbus
```

II. [For Python3]:

```
$ sudo apt-get update
$ sudo apt-get install python3-pip
$ sudo pip3 install RPi.GPIO
$ sudo pip3 install smbus
```

3. [Installing WiringPi Library]:

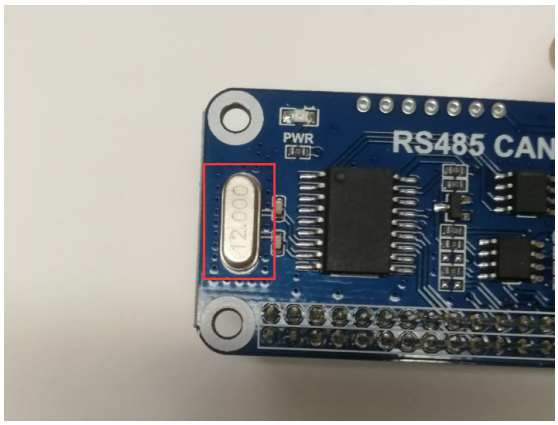
```
$ sudo apt-get install wiringpi
$ wget https://project-downloads.drogon.net/wiringpi-latest.deb
$ sudo dpkg -i wiringpi-latest.deb
$ gpio -v
```

Note: `The 2.52 version will appear (in my case it appeared). If it does not appear, it means there is an installation error`

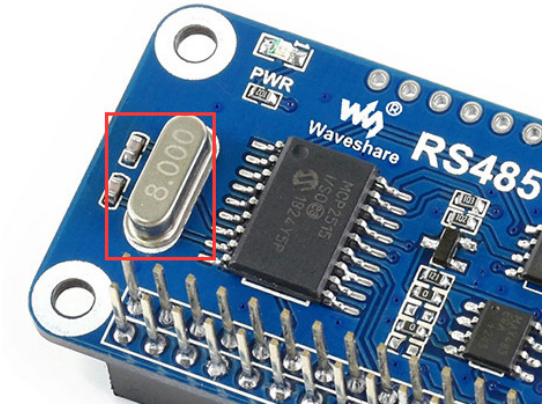
4. [Installing Serial Library]:
   $ `sudo apt-get install python-serial`
5. [Installing CAN Library]:
   $ `sudo apt-get install python-can`
6. [Configure Raspberry Pi system files]:
   $ `sudo nano /boot/config.txt` (any other text editor can be used like gedit, vim etc. instead of nano)
   Enter the following contents at the end of config file:

```
dtparam=spi=on
dtoverlay=mcp2515-
can0,oscillator=12000000,interrupt=25,spimaxfrequency=2000000
```

Among them, **oscillator=12000000**, is the on-board crystal oscillator size of 12M, as shown in the figure below (my board oscillator has also 12M value):



Older version of HAT has different oscillator installed:



So the settings will be as follows:

```
dtparam=spi=on
dtoverlay=mcp2515-can0,oscillator=8000000,interrupt=25,spimaxfrequency=1000000
```

7. After saving and exiting, restart the Raspberry Pi once again: `sudo reboot`
8. Run the command to check whether the initialization is successful: `dmesg | grep -i '\(can\|spi\)'`

   If everything go successful then it should print the following:

```
pi@raspberrypi:~/485-CAN-HAT/CAN/python $ dmesg | grep -i '\(can\|spi\)'
[    5.940145] CAN device driver interface
[    5.958689] mcp251x spi0.0 can0: MCP2515 successfully initialized.
[ 1389.605424] IPv6: ADDRCONF(NETDEV_CHANGE): can0: link becomes ready
[ 1389.620017] can: controller area network core
[ 1389.629938] can: raw protocol
```

   If the module is not connected, it may prompt as follows: Please check whether the module is connected. Whether to enable SPI and enable MCP2515 kernel driver. Whether to restart.

```
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
```

9. [Downloading example code[4] and grant permission]:

```
$ sudo chmod 777 -R Python-code-folder/
```

If everything is successful and there is no issue with connection, and system settings etc. then you should be able to run the code and see the data being printed to the terminal.

For Python2/3:

```
$ sudo python TF03-CAN.py
```

```
LiDAR id:00000003
Size of data-packet in bytes:8
Distance:45
Strength:4214

Time stamp:1626258206.64
LiDAR id:00000003
Size of data-packet in bytes:8
Distance:45
Strength:4211

Time stamp:1626258206.64
LiDAR id:00000003
Size of data-packet in bytes:8
Distance:45
Strength:4214

Time stamp:1626258206.64
LiDAR id:00000003
Size of data-packet in bytes:8
Distance:45
Strength:4209
```

[4] https://github.com/ibrahimqazi/TF03-CAN-Python-code-for-485-CAN-HAT

CAN Interface Troubleshooting:

1. Make sure that the hardware wiring is correct, that is, HH, LL connection
2. Make sure that the baud rate settings on both sides are the same, and the default routine set the baud rate to 100K
3. Make sure that the CAN IDs on both sides are the same (in case you are sending commands to LiDAR), otherwise it won't received
4. If there is frame loss in sending data for a long time, you can try to reduce the baud rate to solve it