Interfacing CE30-A-CAN LiDAR with Raspberry Pi HAT

Written by: Ibrahim (FAE)

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

This article explains how to interface Benewake CE30-A-CAN 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 CE30-A 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 for Raspberry and 12V for LiDAR.
5. The default interface mode of LiDAR is CAN so we don't need to switch the interface.
6. SPI interface of Raspberry Pi is enabled.
7. Any editor for writing and modifying the script (VS Code is recommended).

[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 CE30-A-CAN LiDAR to HAT and using a separate power supply is given below:

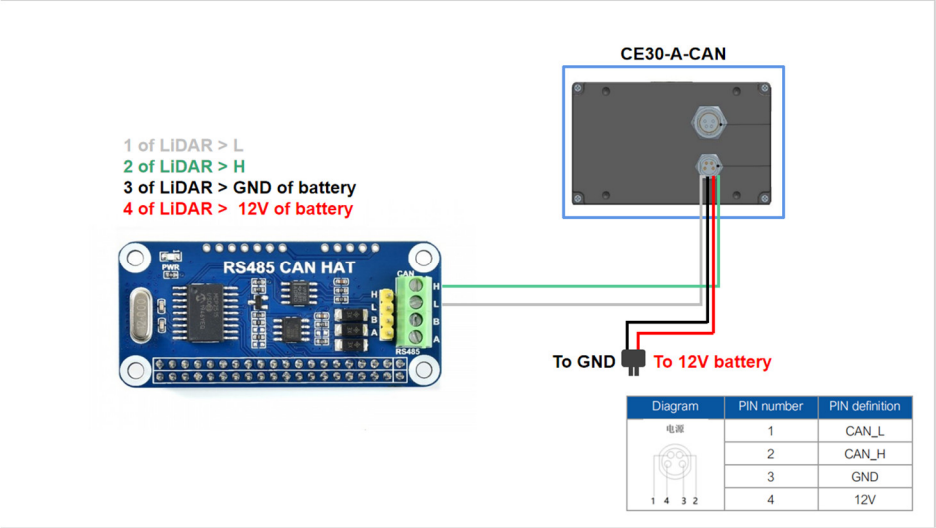


Figure 2: Connection Diagram

For exact details about LiDAR current rating and your supply capacity, please refer to their respective data-sheets. It should be noted that your supply should be able to provide enough current to Raspberry Pi and HAT otherwise you may experience abnormal behavior.

Default CAN Communication Parameters of CE30-A:

According to the data-sheet of CE30-A LiDAR the parameters of CAN communication:

Item	Content
Communication protocol	CAN
Baud rate	250K
Receive ID	0x586
Transmit ID	0x607
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:

Benewake CE30-A LiDAR needs four wires for CAN communication, on HAT side will need male-DuPont connector or any other suitable male connector that can be inserted into HAT:

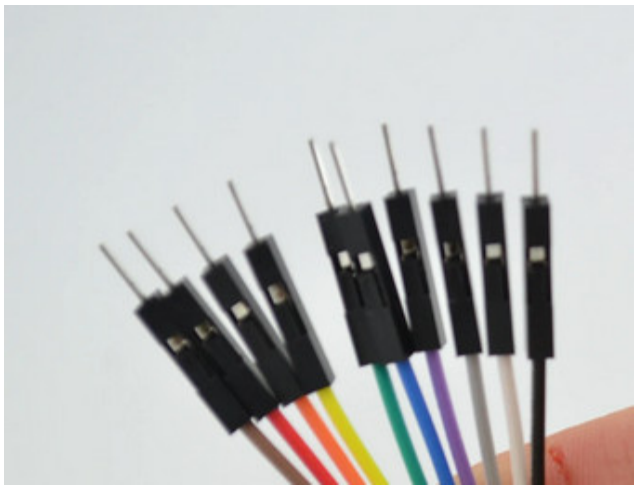


Figure 4: Male DuPont cable

Table III: Data format

CE30-->MPU ID:0x586	Data length	Description
The projected distance of Nearest obstacle point	2	Little-endian. Unit: cm
Reserve	1	
The azimuth of the nearest obstacle	1	Degree
Reserve	3	
State	1	bit0: 1-obstacle detected, 0-no obstacle

1) The data provided by the LiDAR is the projected distance of the obstacle to the front surface of the machine, and the azimuth of the nearest obstacle, i.e. (z, θ). The central line of the FOV is 0 degree, the left is negative and the right is the positive, see the data-sheet.
2) Distance data is 2byte, and transmitted by the means of little-endian, in cm.
3) All azimuth values are signed 8-bit angle values, in degree.

Table IV: Data format (Heart Beat Frame Format)

CE30-A-->CPU ID:0x587	Byte length	Description
Heartbeat packet	1	bit0: 1 – Running state; 0 – Stop state bit1: 1 – Valid version number; 0 - Invalid version number bit2: 1 – Error; 0 – Normal; bit3: reserve bit4: 1 – Center calibrated; 0 – Without center calibration; bit5_7: Heartbeat value (From 0 to 7 in turn)
Version number	2	Version number

The time interval of the heartbeat packet is 150ms. If there is a version inquiry, the heartbeat packet (including the version number) will respond immediately.

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 CAN Library]:

$ `sudo apt-get install python-can`

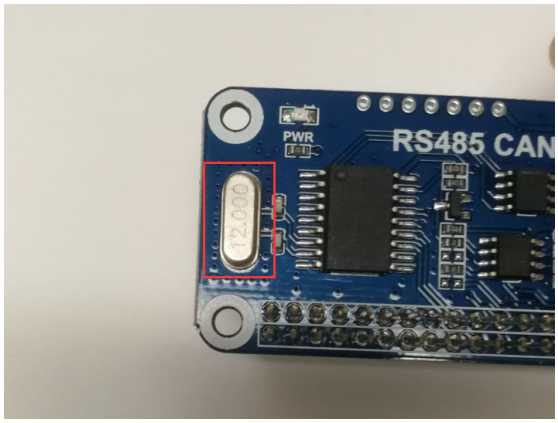
5. [Configure Raspberry Pi system files]:

$ `sudo nano /boot/config.txt` (any other text editor can be used like gedit, vim etc. instead of nano)

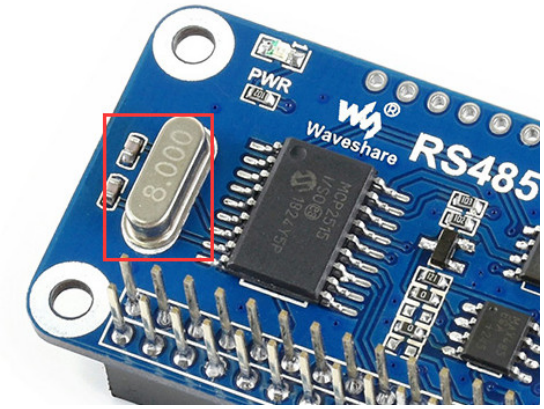
Enter the following contents at the end of config file:

```
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
```

6. After saving and exiting, restart the Raspberry Pi once again: `sudo reboot`
7. 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
```

8. [Downloading example code[3] 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.

[3] https://github.com/ibrahimqazi/CE30-A-Python-code-for-485-CAN-HAT

For Python2/3:

```
$ sudo python3 CE30-A-CAN.py
```

```
pi@raspberrypi:~/485-CAN-HAT/CAN/python $ python3 CE30-A-CAN.py
[INFO] Python code for CAN communication
[INFO] You will continuously receive data, press Ctrl + C to exit
[INFO] This code works with Python2 and Python3
Python version is:3.7.3 (default, Jul 25 2020, 13:03:44)
[GCC 8.3.0]
Time stamp:1627625032.21451
LiDAR id:0x587
Size of data-packet in bytes:3
Heart beat packet:134
Heart beat binary:0b10000110
Version:3

Time stamp:1627625035.21852
LiDAR id:0x587
Size of data-packet in bytes:3
Heart beat packet:162
Heart beat binary:0b10100010
Version:3

Time stamp:1627625038.222314
LiDAR id:0x587
Size of data-packet in bytes:3
Heart beat packet:194
Heart beat binary:0b11000010
Version:3

Time stamp:1627625041.226067
LiDAR id:0x587
Size of data-packet in bytes:3
Heart beat packet:226
Heart beat binary:0b11100010
Version:3
```

Europe

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 250K
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