

GIM6010-8 Micromotor User Manual

Table of contents

Version History.....	2
Notes.....	3
Legal Notice.....	3
After-sales Policy.....	3
1 Motor Specifications.....	5
1.1 Drawings and Dimensions.....	5
1.2 Electrical Characteristics.....	6
1.3 Mechanical properties.....	7
2 Drive Information.....	8
2.1 Appearance and 3D Dimensions.....	8
2.2 Interface Overview.....	9
2.3 Specifications.....	9
2.4 Interface Detailed Definition.....	10
2.5 Main components and specifications.....	14
3 Commissioning Instructions.....	15
3.1 Getting Started Guide.....	15
3.2 Firmware Update Download.....	33
4 Communication Protocol and Examples.....	35
4.1 CAN Protocol.....	35
4.2 PYTHONSDK.....	47
4.3 ARDUINOSDK.....	49
4.4 ROS SDK.....	56
5 Common Problems and Exception Codes (to be updated).....	57
5.1 Frequently Asked Questions (FAQ).....	57
5.2 Exception Code.....	57

Version History

Version Number	date	Revisions/Explanations
0.1	2023.12.5	Support for the first version of odrivetool
0.2	2023.12.15	Added command list and command classification
0.3	2023.12.19	Added Python, Arduino, ROS SDK
0.4	2023.12.23	Added switching description for USB and CAN compatibility
0.5	2023.12.29	Added practical cases for upper computer commissioning
0.6	2024.1.2	Added CAN protocol and Python debugging examples
0.7	2024.1.8	Add CAN interface 120R matching resistor switch option
0.8	2024.2.15	Added content related to the second encoder and user zero point setting
0.9	2024.2.23	Added CAN instructions that can modify parameters and call interface functions
0.91	2024.3.12	Added the driver download address for the national download software
0.92	2024.3.22	Added description of CAN default baud rate and description of rotor initial position range under the action of second encoder.
1.0	2024.3.26	Added motor temperature protection instructions
1.1	2024.5.20	Added commissioning guide
1.2	2024.5.24	Fixed CAN MIT protocol error
1.3	2024.7.2	Added error code table

Precautions

1. Please use the product according to the operating parameters in this manual, otherwise it will cause irreversible damage to the product!
2. During the operation of the motor, please take over-current and over-voltage protection measures for the power supply to avoid damaging the drive.
3. Please check that all parts are intact before use. If any parts are missing or damaged, please contact technical support in time.
4. The driver does not have reverse polarity protection capability. Please refer to Section 2.4.1 before connecting the power supply to ensure that the positive and negative poles of the power supply are correct.
5. Please do not touch the exposed parts of the drive with your hands to avoid damage due to static electricity!

Legal Notice

Before using this product, please read this manual carefully and operate this product according to the contents. If the user uses this product in violation of the contents of the manual, the company will not assume any responsibility for any property loss or personal injury caused. Since this product is composed of many parts, children should not be allowed to touch this product to avoid accidents. To extend the service life of the product, do not use this product in a high temperature or high pressure environment. This manual has been printed to include the introduction of various functions and instructions for use as much as possible. However, due to the continuous improvement of product functions and design changes, there may still be discrepancies with the products purchased by users.

This manual may differ from the actual product in terms of color, appearance, etc. Please refer to the actual product. The company may make necessary improvements and changes to the typographical errors, inaccurate latest information in this manual, or improve the program and/or equipment at any time without prior notice. Such changes will be uploaded to the new version of this manual, please contact technical support to obtain it. All pictures are for reference only for functional description, please refer to the actual product.

After-sales policy

The after-sales service of this product is strictly implemented in accordance with the "Consumer Protection Law of the People's Republic of China" and the "Product Quality Law of the People's Republic of China". The service content is as follows:

1. Warranty period and contents

- 1) Users who place an order for this product through online channels can enjoy the unconditional return service within seven days from the day after signing for it. When returning the goods, users must show a valid purchase receipt and return the invoice. Users must ensure that the returned goods maintain their original quality and function, the appearance is intact, and the trademarks and various logos of the goods themselves and accessories are intact. If there are any gifts, they must be returned together. If the goods are damaged by human factors, disassembled by human factors, the packaging box is missing, or the spare parts are missing, they will not be returned. The logistics costs incurred during the return shall be borne by the user. If the user has not settled the logistics costs, the actual amount will be deducted from the refund amount. The paid amount will be returned to the user within seven days from the date of receipt of the returned goods. The refund method is the same as the payment method. The specific arrival date may be affected by factors such as banks and payment institutions.
- 2) Within 7 days from the day after the user signs for the product, if there is any performance failure not caused by human factors, after inspection and confirmation by our after-sales service center, we will process the return for the user. When returning the product, the user must show a valid purchase receipt and return the invoice. Any gifts must also be returned.
- 3) Within 7 to 15 days after the user signs for the product, if there is a performance failure not caused by human factors, after inspection and confirmation by our after-sales service center, we will handle the exchange business for the user and replace the entire set of products. After the exchange, the three-year warranty period of the product itself will be recalculated.

- 4) Within 15 days to 365 days after the day the user signs for the product, if the product is found to be defective, free repair service will be provided. The defective product will be owned by the company. Non-defective products will be returned in their original condition. This product is shipped after various strict tests. If there is a defect that is not caused by the product itself, we have the right to refuse the user's return or exchange request.

If the after-sales policy in this manual is inconsistent with the after-sales policy of the store, the after-sales policy of the store shall prevail.

2. Non-warranty regulations

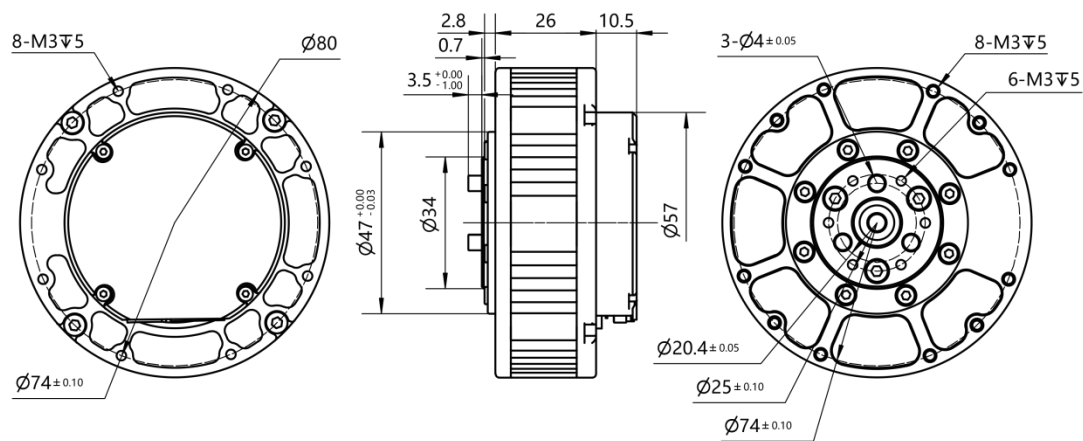
The following situations are not covered by the warranty:

- 1) Beyond the warranty period specified in the warranty terms.
- 2) Product damage caused by improper use without following the instructions.
- 3) Damage caused by improper operation, maintenance, installation, modification, testing, or other improper use.
- 4) Routine mechanical loss and wear not caused by quality failure.
- 5) Damage caused by abnormal operating conditions, including but not limited to falling, impact, liquid immersion, severe impact, etc.
- 6) Damage caused by natural disasters (such as floods, fires, lightning strikes, earthquakes, etc.) or force majeure.
- 7) Damage caused by exceeding the peak torque.
- 8) The product is not an original product of our company or the legal proof of purchase cannot be provided.
- 9) Failure or damage caused by other issues not related to product design, technology, manufacturing, quality, etc.
- 10) Damage caused by unauthorized disassembly of the product.

If the above situation occurs, the user will be responsible for paying the fees.

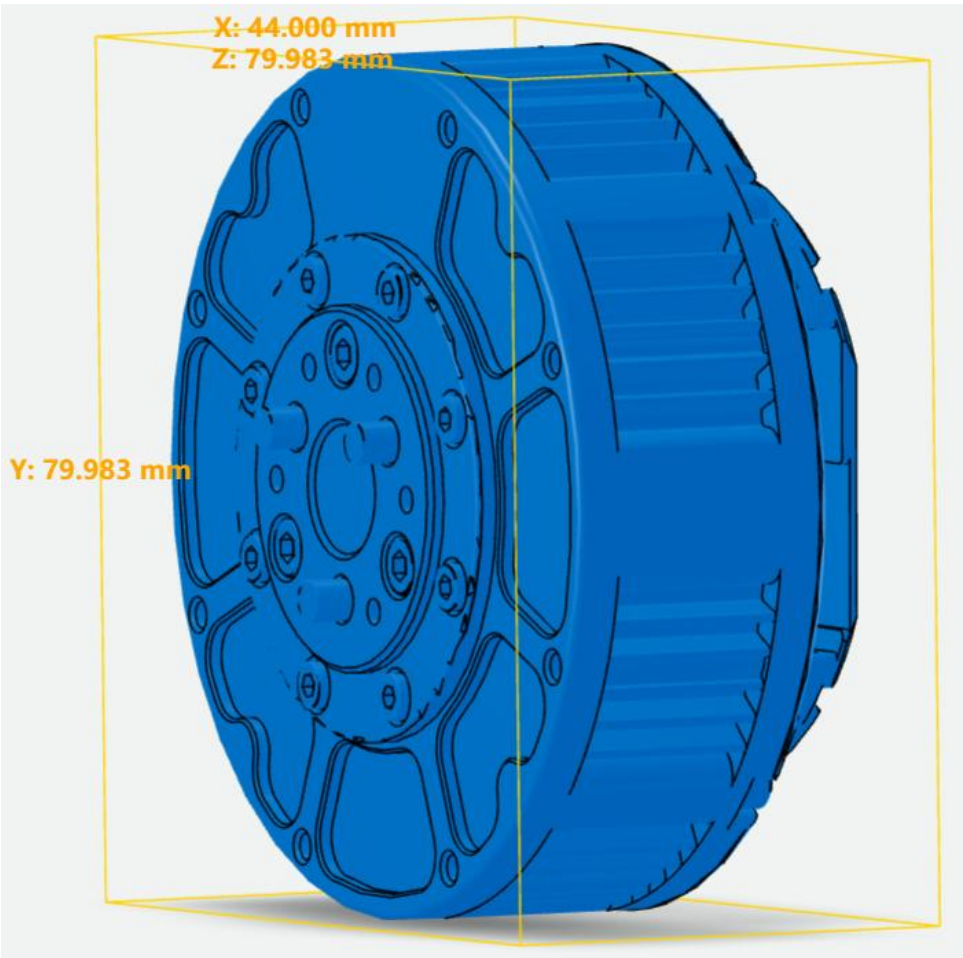
1 Motor Specifications

1.1 Drawings and dimensions



未注公差分段						
<=6	>6<=30	>30<=120	>120<=400	>400<=2000	比例	1 : 1
± 0.1	± 0.2	± 0.3	± 0.5	± 1.0	重量kg	无

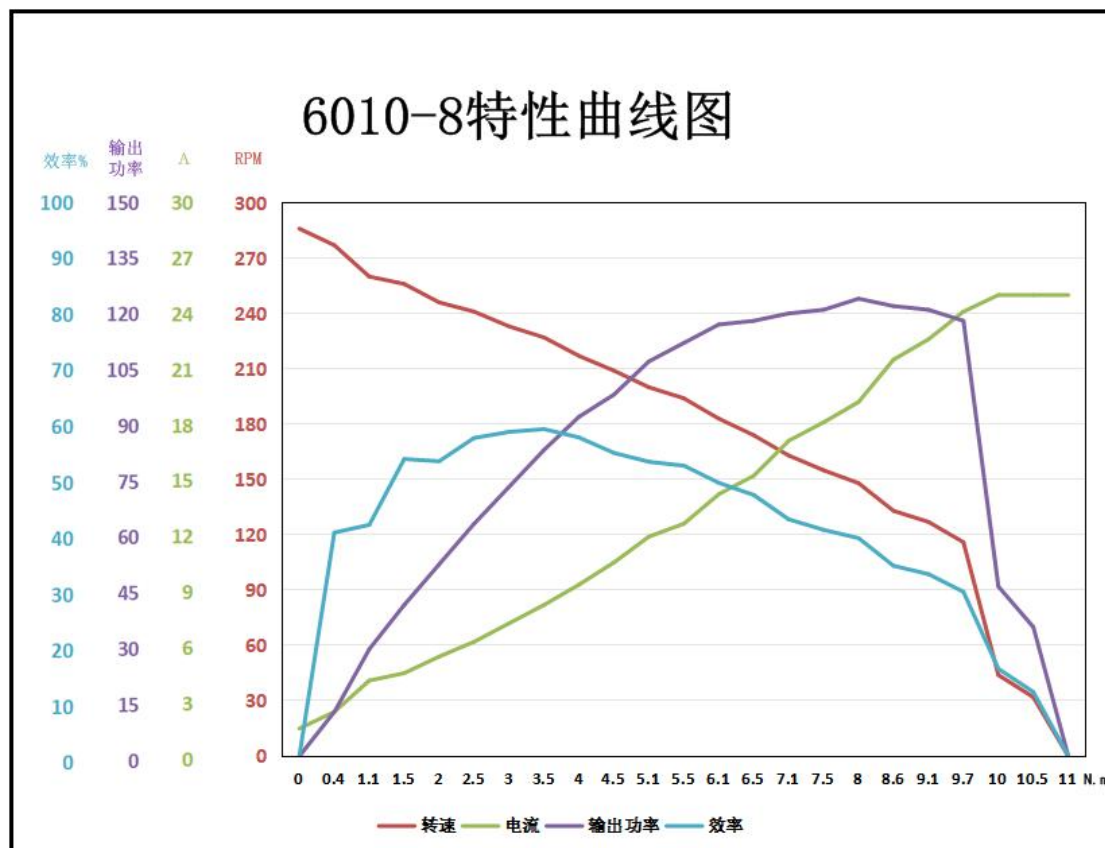
6010-8安装尺寸图



1.2 Electrical characteristics

Rated speed	120rpm±10%
Maximum speed	420rpm±10%
Rated torque	5N.m
Stall torque	11N.m
Rated current	10.5A
Stall current	25A
No-load current	0.4A
Insulation resistance/stator winding	DC 500VAC, 100M Ohms
High pressure resistance/stator and housing	600 VAC, 1s, 2mA
Motor back EMF	0.054~0.057Vrms/rpm
Phase resistance	0.48Ω±10%
Phase inductance	368μH±10%
Speed constant	12.3rpm/v
Torque constant	0.47NM/A

The characteristic curve is as follows:

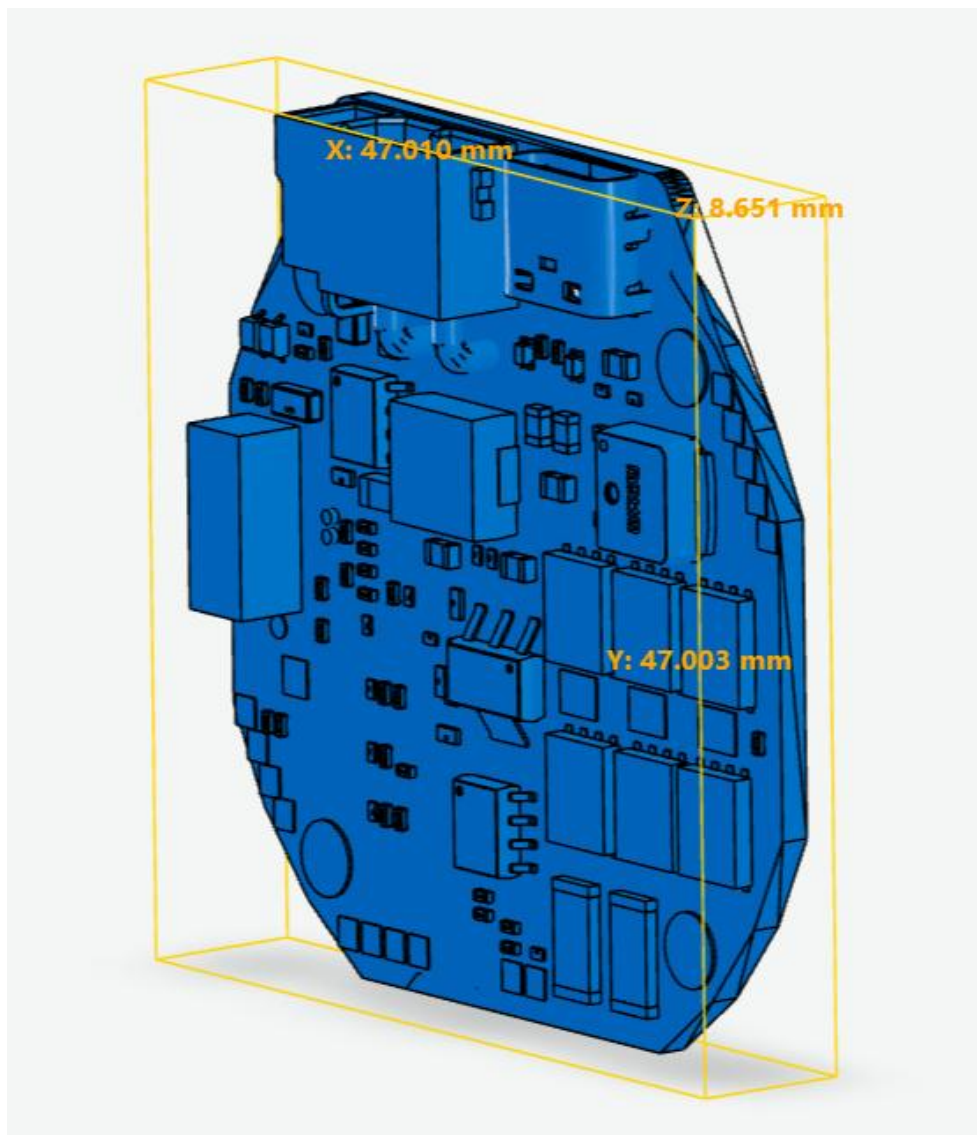


1.3 Mechanical properties

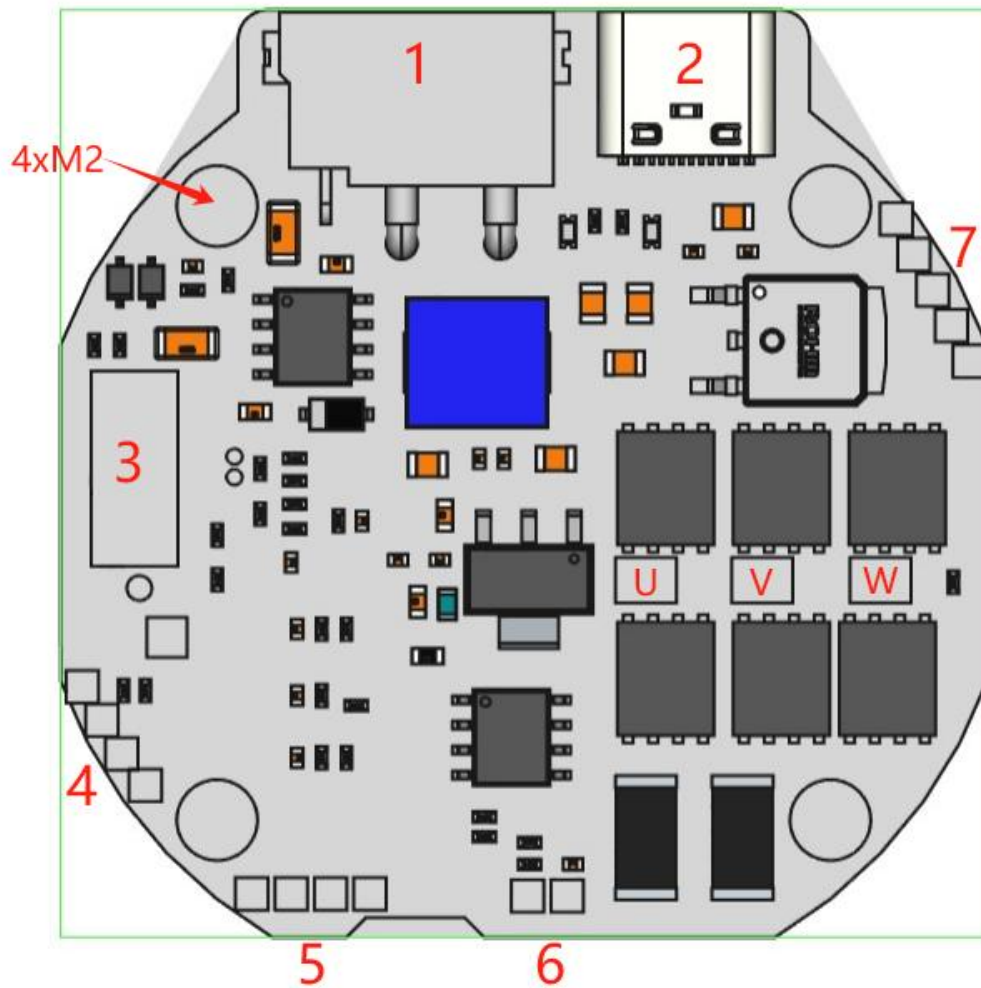
weight	388g±3
Pole pairs	14 pairs
Number of phases	3 Phase
Drive mode	FOC
Reduction ratio	8:1

2 Drive Information

2.1 Appearance and 3D Dimensions



2.2 Interface Overview



Interface number	definition
1	15~60V power supply and CAN communication integrated terminal
2	Type-C debugging interface and host computer communication interface
3	Interface expansion slot (can expand RS485, EtherCAT, aircraft model, pulse direction, throttle control and other interfaces/protocols)
4	SWD debugging and downloading interface
5	Second encoder interface (supports I2C and UART)
6	Motor temperature interface (NTC)
7	Brake/brake resistor interface, 12V power supply, minimum/maximum limit switch interface
U/V/W	Three-phase winding welding holes
4xM2	Mounting holes

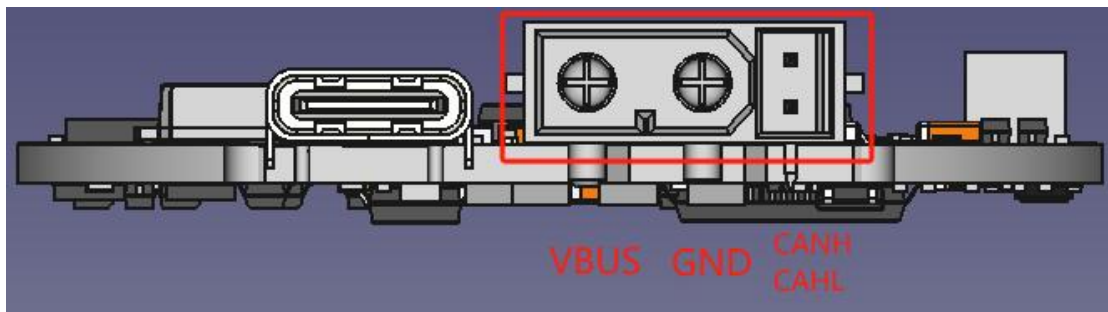
2.3 Specifications

Rated voltage	15~48V DC
---------------	-----------

Min./Max. Voltage	12/72V DC
Rated current	6A
Maximum line current	30A
Maximum phase current	120A
Standby power consumption	<10mA
CAN bus maximum baud rate	1Mbps
Type-C Speed	10Mbps
Encoder resolution	16bit (single-turn absolute value)
Operating temperature	-20°C to 70°C
Alarm motor temperature	90°C (adjustable)
Warning driver board temperature	90°C (adjustable)

2.4 Interface Detailed Definition

2.4.1 Power supply and CAN communication terminals



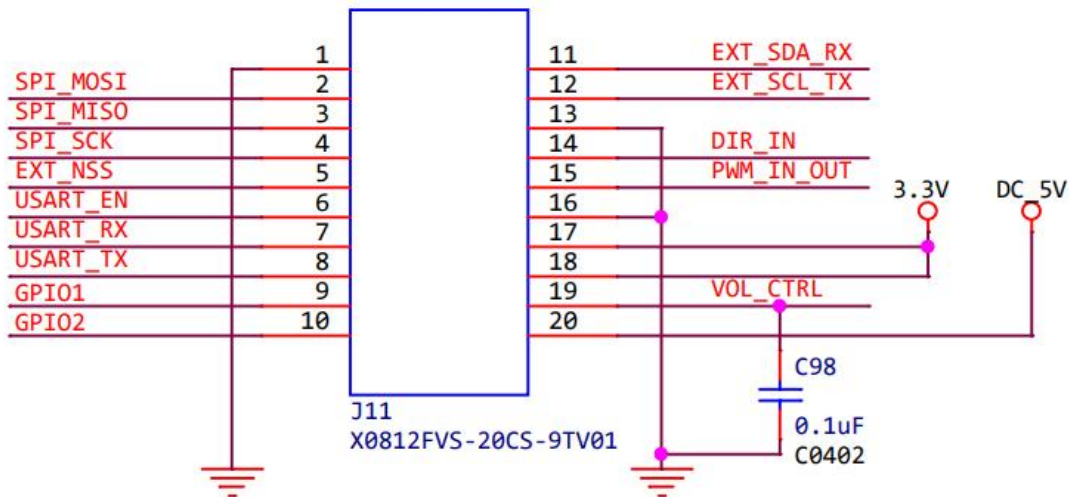
The onboard terminal model is XT30PB(2+2)-M, the line terminal model is XT30(2+2)-F, and the brand manufacturer is AMASS.

2.4.2 Type-C Debug Interface

Type-C uses standard data cable specifications and is compatible with commonly used PC or mobile phone Type-C data cables.

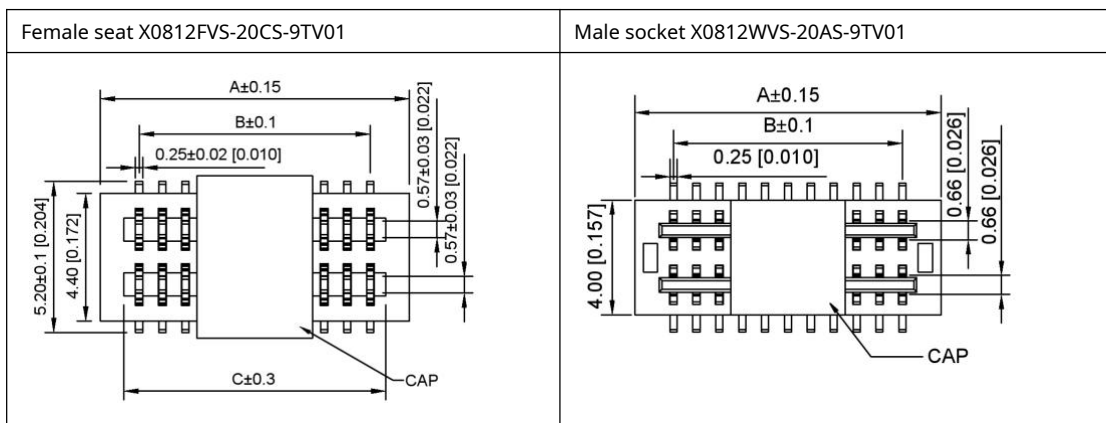
2.4.3 Interface expansion slot

This slot adopts the following design method, providing a variety of inter-board expansion interfaces, allowing any expansion board to be developed by a third party:



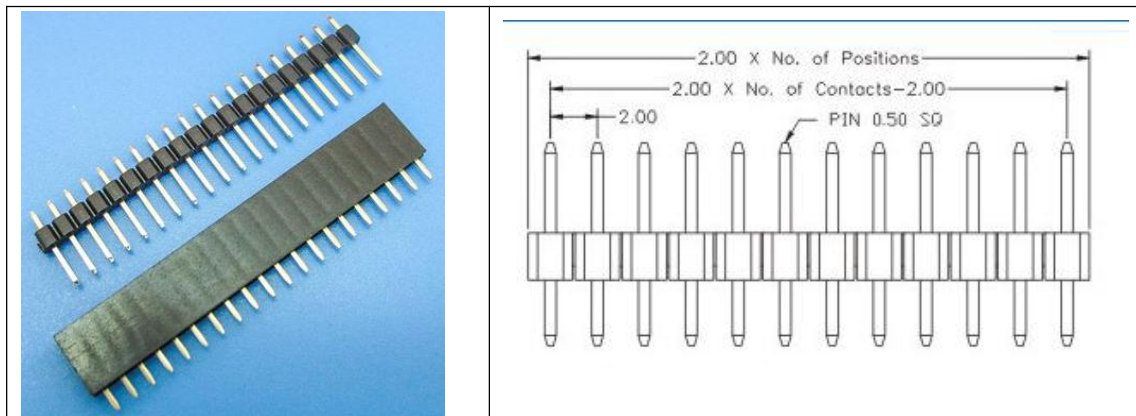
The third party can interact with the driver through SPI, USART, I2C, PWM, ADC, GPIO, etc. to achieve various extended functions.

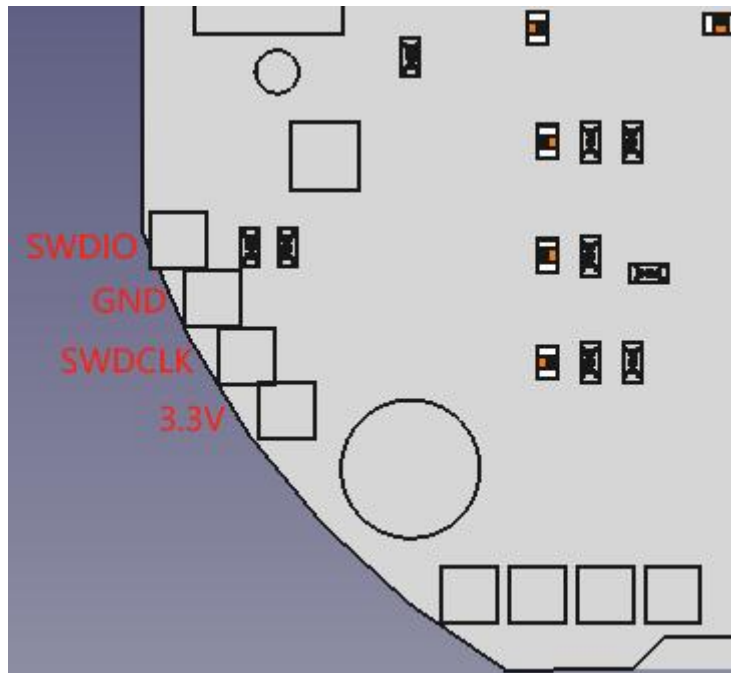
The onboard slot model is X0812FVS-20CS-9TV01 (female), the expansion board slot model is X0812WVS-20AS-9TV01 (male), and the brand manufacturer is Xingkun.



2.4.4 SWD Debug Interface

The pin holes are spaced 2mm apart, and users can solder 2mm straight-inserted single-row pins, as shown below:

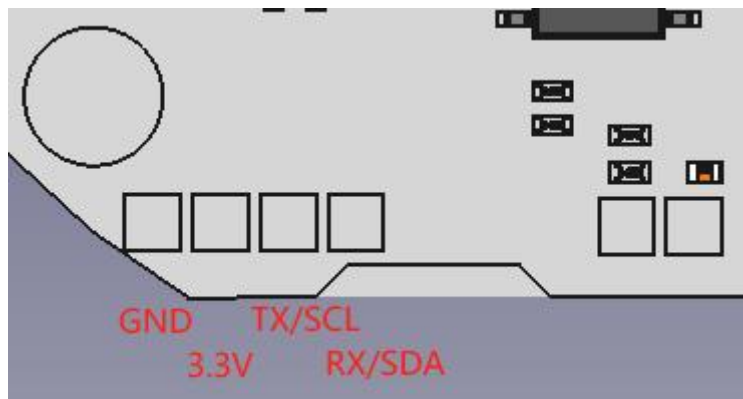




2.4.5 Second encoder interface

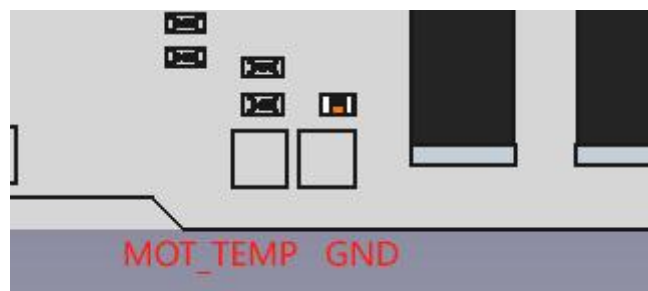
The pin holes are spaced 2mm apart, and users can solder 2mm straight-inserted single-row pins, see 2.4.4.

This interface can communicate with the second encoder via USART (TX/RX) or I2C (SCL/SDA).



2.4.6 Motor temperature interface

The motor has a built-in 10K NTC resistor, and the two leads are soldered to MOT_TEMP and GND, with no line sequence.



3 Commissioning Instructions

3.1 Getting Started Guide

3.1.1 Preparation

To make the motor work, you need:

- power supply

Please refer to Chapter 1 for the power supply voltage requirements. It is recommended to use a regulated power supply or battery. The question that users often wonder is, how should I choose a power supply? The following are some simple suggestions for reference only:

Several points to consider when choosing a power source:

-Current requirement

Generally speaking, it should be at least greater than 5 A. The specific value depends on the power requirements and voltage of the system.

-Voltage requirements

The voltage requirement depends on two factors: the Kv of the motor and the maximum speed RPMmax required by the system. The maximum value of the required power supply voltage can be found in the formula:

$$V_{max} = \frac{K_v \times RPM_{max}}{1.25}$$

The 1.25 is an empirical coefficient that gives the system a safe voltage threshold.

-Power requirements

As for power, simply put, it depends on the maximum current value I_{max} at the highest speed. Please refer to the formula:

$$P_{max} = V_{max} \times I_{max}$$

- Power supply + communication interface cable

6010-8 has 2+2 power communication sockets. Please contact after-sales service to recommend a suitable 2+2 cable. Please refer to 2.4.1, **Be sure to connect the positive and negative poles of the power supply correctly, otherwise there is a risk of burning the driver.** Because the driver does not have the ability to prevent reverse connection. **Please pay attention to the definition order of the two communication lines, such as the order of CANH/CANL, wrong connection will cause abnormal communication.**

warn

- Please be sure to avoid touching the communication bus with your hands to prevent static electricity from damaging the driver's interface chip, especially in dry areas and dry seasons!
- Please be sure not to plug or unplug the power terminal when power is on!
- Avoid using a circuit breaker as the power switch, as there is a risk of destroying the power chip on the driver! The
- power supply voltage should not exceed 72V!

- Type-C cable

In the early stages of commissioning, it is strongly recommended to use a USB Type-C data cable to test the motor. You can use the most commonly used mobile phone Type-C data cable, but do not use a charging-only Type-C cable.

Please note that the Type-C data cable cannot power the driver, let alone drive the motor!

- Power Up

Please turn off the power first, connect the power cable, and then turn on the power. Be sure not to unplug or plug while the power is on, or use a circuit breaker to control a single positive or negative cable to switch on and off, as this will cause excessive startup current to burn the driver.

After power is applied, the USB Type-C cable can be plugged in or unplugged at any time.

3.1.2 Starting with odrivetool

Drive compatible with odrive (<https://github.com/odriverobotics/odrive.git>), so odrivetool is used as the host computer for debugging.

Follow the steps below to install odrivetool:

- Windows

1. Install Python

Go to the Python official website <https://www.python.org> Download the latest Python installer and follow the prompts to install it. Please do not download Python versions from third-party websites or Microsoft Store.

2. Install the Visual C++ build tool

Install Visual C++ build tools <https://visualstudio.microsoft.com/visual-cpp-build-tools/> During the installation process, check "Use C++ desktop development", as shown in the figure below.

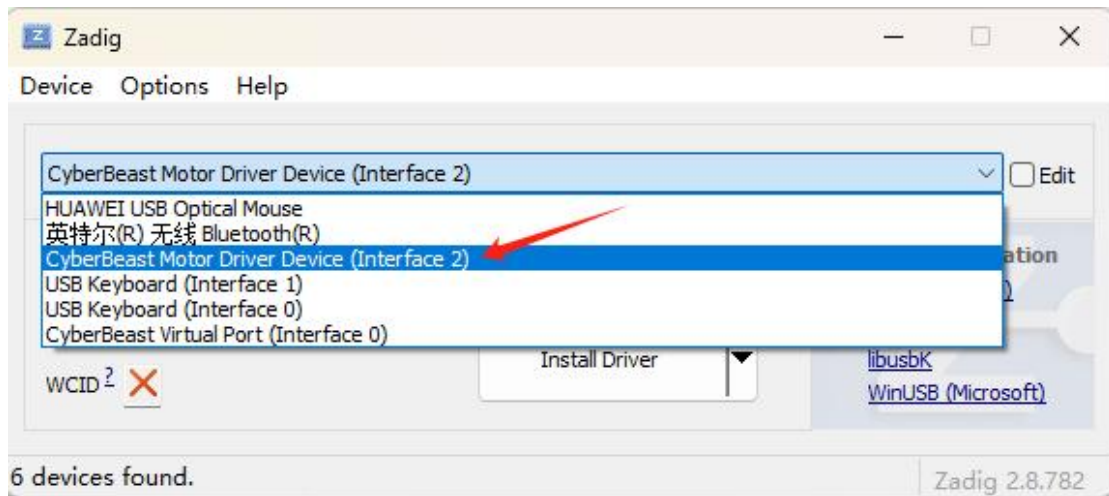


3. Install odrivetool

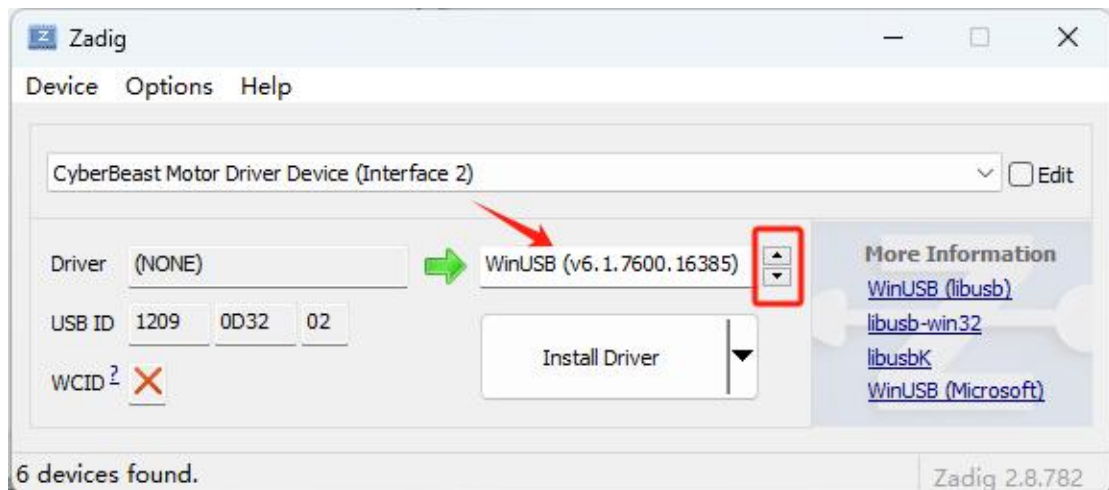
Run Windows PowerShell as an administrator, run `pip install odrive` in it and press Enter to install. If an error occurs during the process, please try again. If an error occurs again and again, please restart the computer and try again.

4. Install USB driver

Enter the website <https://zadig.akeo.ie> Download the USB driver tool Zadig, connect the drive to the computer with a Type-C data cable, the drive power light is on, open Zadig, and select "CyberBeast Motor Driver Device (Interface 2)" from the drop-down box:



Select different USB drivers by clicking the up and down keys. Please select the "WinUSB" driver version for this interface and click "Install Driver" to install the driver for this interface:



- WSL (Windows Subsystem for Linux)

1) Install python/usb/odrivetool

If the user has installed WSL2, enter the WSL2 command line and follow the steps below to install it (assuming the user has installed WSL2)

```
sudo apt install python3 python3-pip sudo
apt install libusb-1.0-0
sudo pip install odrive numpy matplotlib
```

Installed on Ubuntu):

The first line of instructions installs Python, the second line of instructions installs the USB driver, and the third line of instructions installs the odrivetool host computer.

2) Connect the drive to WSL

Plug the USB port into Windows with a type-C cable. By default, Windows will load the driver for this USB port, but WSL will not load it. To load this USB port into WSL, please refer to the Microsoft documentation.

(<https://learn.microsoft.com/zh-cn/windows/wsl/connect-usb>) operate.

- Ubuntu

The installation process under Ubuntu is very similar to that under WSL, please refer to the previous section.

3.1.3 Reasonable configuration of motor parameters

warn

It is recommended to read this section carefully, as it is the key to successful operation and to avoid burning the motor!

After successfully installing odrivetool as described in the previous section, power on the motor and connect the USB Type-C cable, run odrivetool in the shell (Windows PowerShell or Linux Terminal) (type odrivetool and press Enter). The following figure shows a successful connection under Windows (connection information is displayed in green font):

```
PS D:\> odrivetool
Website: https://odriverobotics.com/
Docs: https://docs.odriverobotics.com/
Forums: https://discourse.odriverobotics.com/
Discord: https://discord.gg/k3ZZ3mS
Github: https://github.com/odriverobotics/ODrive/

Please connect your ODrive.
You can also type help() or quit().

Connected to ODrive v3.6 B53319683238 (firmware v0.5.6) as odrv0
In [1]: odrv0.vbus_voltage
Out[1]: 20.00840187072754
In [2]:
```

Command example: `odrv0.axis0.controller.input_vel`, `odrv0` represents the currently connected motor. By default, the first connected motor is called `odrv0`, the second is called `odrv1`, and so on; `axis0` represents the first motor connected to the driver. The current version only supports connecting one motor. This command means to query the current speed control target value of the driver.

Operation Tips

- If you frequently use the TAB key, there will be a command prompt, similar to the command prompt under Linux. You can use the up, down, left, and right keys to select a command.
- The up and down keys will display the history commands
- When entering a command, a history of similar commands will be displayed, and right-clicking will directly complete the command.

- Setting critical limits


```
odrv0.axis0.motor.config.current_lim = 30
```

- Current Limit

The above command sets the current limit to 30A. Please note that this current limit refers to the Q-axis current, not the supply current. This threshold directly limits the output torque. **For 6010-8, please do not set this threshold above 50A!**

Other factors affecting current limit

-Motor temperature

If motor temperature protection is enabled (odrv0.axis0.motor.motor_thermistor.config.enabled=1), the current temperature of the motor will also affect the Q-axis current.

-Driver board temperature

If the driver board temperature protection is enabled (odrv0.axis0.motor.fet_thermistor.config.enabled=1), the current temperature of the driver board will also affect the Q axis current.

The effects of the two temperatures mentioned above are very similar and can be expressed by the following formula:

$$I_{eff} = \frac{I_{lim} - (T - T_{lower})}{T_{upper} - T_{lower}}$$

in the formula, I_{eff} is the final effective current threshold, I_{lim} is the configured current limit, T is the current temperature (motor temperature or drive Board temperature), T_{lower} is the lower temperature limit set (motor_thermistor.config.temp_limit_lower and fet_thermistor.config.temp_limit_lower), T_{upper} is the upper temperature limit set (motor_thermistor.temp_limit_upper and fet_thermistor.config.temp_limit_upper).

```
odrv0.axis0.controller.config.vel_limit = 30
```

- Speed Threshold

The system global speed limit, the above command limits it to 30 turns/s (turns/second). Please note that this speed limit does not work in pure torque mode by default, but can be enabled by turning on the following switch:

```
odrv0.axis0.controller.config.enable_torque_mode_vel_limit = 1
```

- Calibration current

The default current is 5A. You do not need to modify it by default. However, if the user's power supply current is small, you can reduce this value.

```
odrv0.axis0.motor.config.calibration_current = 2
```

A low voltage alarm will appear during calibration.

- Set key hardware parameters

- Maximum discharge/charge current

The discharge current refers to the forward current supplied by the power supply to the driver and motor, and the charging current refers to the current flowing into the power supply in the reverse direction.

These two values are related to the power supply. Please set them to a suitable value to avoid the power supply failing to discharge, causing the voltage to be pulled down, or being reversed by the electromotive force.

```
odrv0.config.dc_max_negative_current
odrv0.config.dc_max_positive_current
```

However, please note that if these two values are set to a value that is absolutely small, it is easy to generate an alarm.

- Pole pairs

The number of pole pairs is the number of magnetic poles in the motor rotor divided by 2. The user must set this value correctly for the calibration to succeed, otherwise

```
odrv0.axis0.motor.config.pole_pairs
```

There is a calibration alarm.

- Torque constant

The torque constant is the torque generated by the motor divided by the Q axis current. It is related to the motor Kv value as follows: $\frac{1}{K_t} = \frac{K_v}{9.549}$

```
odrv0.axis0.motor.config.torque_constant
```

8.27

Whether the torque constant is correct does not affect the operation of the motor, but it will affect the unit conversion of the value entered by the user when performing torque control. If the user wants to use the unit A instead of Nm for torque control, just set this value to 1.

- Temperature Sensing

The driver board and the motor are equipped with NTC temperature sensors. If enabled, the driver will control the output according to the temperature.

```
# Motor temperature protection
odrv0.axis0.motor.motor_thermistor.config.enabled = 1
odrv0.axis0.motor.motor_thermistor.config.temp_limit_lower = 20
odrv0.axis0.motor.motor_thermistor.config.temp_limit_upper = 100

# Driver board temperature protection
odrv0.axis0.motor.fet_thermistor.config.enabled = 1
odrv0.axis0.motor.fet_thermistor.config.temp_limit_lower = 20
odrv0.axis0.motor.fet_thermistor.config.temp_limit_upper = 100

# Get temperature
odrv0.axis0.motor.motor_thermistor.temperature #Motor temperature
odrv0.axis0.motor.fet_thermistor.temperature #Driver board temperature
```

Current (torque), thereby protecting the drive and motor.

- PID Tuning

```
odrv0.axis0.controller.config.pos_gain=20.0  
odrv0.axis0.controller.config.vel_gain=0.16  
odrv0.axis0.controller.config.vel_integrator_gain=0.32
```

The following process can provide a reference for users to adjust PID parameters:

1. Set the PID initial value
2. Set vel_integrator_gain to 0

```
odrv0.axis0.controller.config.vel_integrator_gain=0
```

3. How to adjust vel_gain:

- 1) Use speed control mode to rotate the motor. If the rotation is not smooth, with jitter or vibration, reduce vel_gain until the rotation is smooth.
- 2) Next, increase vel_gain by about 30% each time until noticeable jitter occurs.
- 3) At this point, reduce vel_gain by about 50% to stabilize

4. How to adjust pos_gain:

- 1) Try to rotate the motor in position mode. If the rotation is not smooth, with pulling or vibration, reduce pos_gain until the rotation is smooth.
- 2) Next, increase pos_gain by about 30% each time until the position control shows obvious overshoot (i.e. each time the position control motor will exceed the target position and then oscillate back to the target position)
- 3) Then, gradually reduce pos_gain until the overshoot disappears.

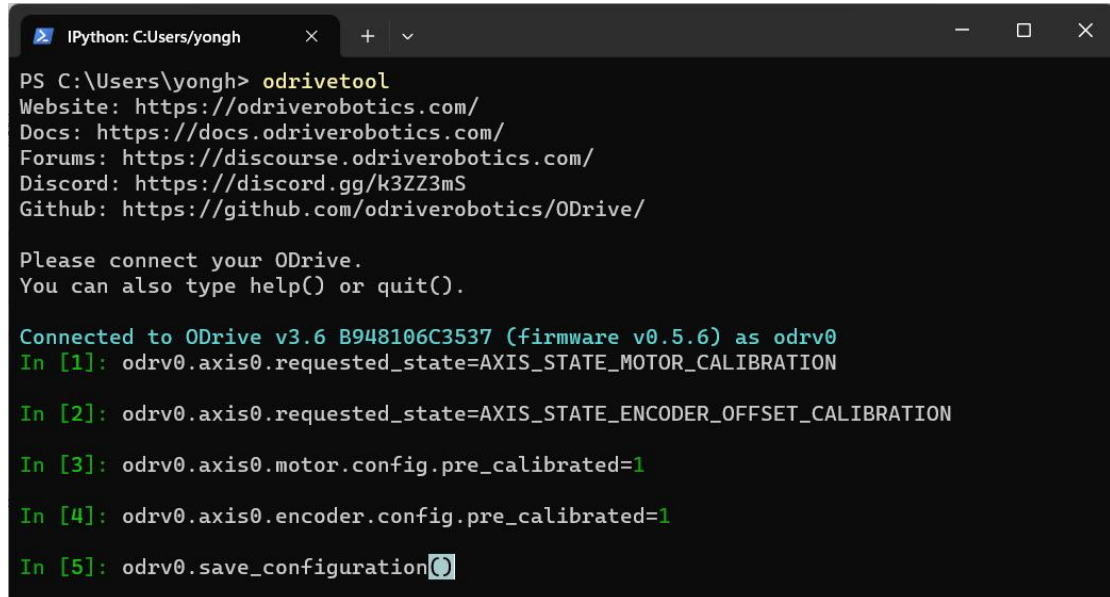
5. After the above 4 steps, vel_integrator_gain can be set to $0.5 \times \text{bandwidth} \times \text{vel_gain}$, where bandwidth is the system control bandwidth. What is control bandwidth? For example, if the time from the user setting the target position to the motor actually reaching the target position is 10ms, then the control bandwidth is 100Hz.
 $\text{vel_integrator_gain} = 0.5 \times 100 \times \text{vel_gain}$.

During the above parameter adjustment process, it is recommended to use the graphical method in 3.1.7 to view the parameter adjustment effect in real time to avoid misunderstanding by naked eye.

Difference.

3.1.4 Power-on calibration

When the user uses the micro motor for the first time, the motor and encoder need to be calibrated. Before calibration, please fix the motor or hold it tightly with your hand, and the output shaft is unloaded. The calibration process is as follows:



```
IPython: C:\Users\yongh
PS C:\Users\yongh> odrivetool
Website: https://odriverobotics.com/
Docs: https://docs.odriverobotics.com/
Forums: https://discourse.odriverobotics.com/
Discord: https://discord.gg/k3ZZ3mS
Github: https://github.com/odriverobotics/ODrive/

Please connect your ODrive.
You can also type help() or quit().

Connected to ODrive v3.6 B948106C3537 (firmware v0.5.6) as odrv0
In [1]: odrv0.axis0.requested_state=AXIS_STATE_MOTOR_CALIBRATION

In [2]: odrv0.axis0.requested_state=AXIS_STATE_ENCODER_OFFSET_CALIBRATION

In [3]: odrv0.axis0.motor.config.pre_calibrated=1

In [4]: odrv0.axis0.encoder.config.pre_calibrated=1

In [5]: odrv0.save_configuration
```

```
odrv0.axis0.requested_state = AXIS_STATE_MOTOR_CALIBRATION
dump_errors(odrv0)
odrv0.axis0.requested_state = AXIS_STATE_ENCODER_OFFSET_CALIBRATION
dump_errors(odrv0)
odrv0.axis0.motor.config.pre_calibrated = 1
odrv0.axis0.encoder.config.pre_calibrated = 1
odrv0.save_configuration()
```

The step-by-step explanation is as follows:

- Step 1: Motor parameter self-identification

When measuring the phase resistance and phase inductance of the motor, a sharp "beep" sound will be heard. The measurement results of phase resistance and phase inductance can be obtained by the following

```
odrv0.axis0.motor.config.phase_resistance
odrv0.axis0.motor.config.phase_inductance
```

The above instructions can be viewed:

- Step 2: Check the error code

Check the system error code after the first step. If any red error code appears, you need to restart the motor and try again, or report it to after-sales service.

- Step 3: Encoder Calibration

Calibrate the encoder, including the calibration of the encoder installation angle and the motor mechanical angle, as well as the calibration of the encoder itself. During this calibration process, the motor will slowly rotate forward an angle and then reverse an angle. If it stops after rotating forward only, it means there is an error. Please check the error code in step 4.

- Step 4: Check the error code

After the third step of encoder calibration, check the system error code. The common error is ERROR_CPR_POLEPAIRS_MISMATCH, which means that the encoder CPR setting is wrong, or the motor pole pair setting is wrong.

```
odrv0.axis0.encoder.config.cpr  
odrv0.axis0.motor.config.pole_pairs
```

Please view/set via the following command:

- Step 5: Write the motor calibration success flag
- Step 6: Write the encoder calibration success flag

```
odrv0.axis0.encoder.config.pre_calibrated = 1  
odrv0.axis0.motor.config.pre_calibrated = 1  
odrv0.save_configuration()
```

- Step 7: Save the calibration results and restart

3.1.5 Storage and backup parameters

After any parameter modification, be sure to save it, otherwise the changes will be invalid after power failure or restart.

```
odrv0.save_configuration()
```

The device will restart.

Parameter backup:

```
odrivetool backup-config"d:/test.json"
```

"d:\test.json" is the save path and file name that can be modified freely by the user.

```
odrivetool restore-config"d:/test.json"
```

The command for parameter recovery is:

3.1.6 Four control modes

After the above preparations and parameter configuration, you can try to control the motor to rotate in different modes. 6010-8 supports position control, speed control, torque control, and motion control modes.

In the position control mode, it supports filtered position control, trapezoidal curve position control, and circular position control.

In the speed control mode, it supports direct speed control (Velocity Control) and ramp speed control (Ramped Velocity Control);

In the torque control mode, direct torque control (Torque Control) and ramped torque control (Ramped Torque Control) are supported.

The motion control mode is a control mode that integrates position, speed and torque. It is usually used in scenarios that require strong instantaneous explosive force, such as robot knee joints. Some users in the industry also call it the MIT control mode, which comes from the MIT open source robot dog because it uses this motion control mode to control the motor.

In the subsequent detailed description of each control mode, this document uses USB control instructions as examples, but the same control can also be done using communication protocols (such as CAN), and the logic is consistent.

How to make the motor turn?

-Start the motor (enter the closed-loop control state)

In all subsequent control operations, to make the motor rotate, it is necessary to put the motor into closed-loop control state. The instructions are as follows:

```
odrv0.axis0.requested_state = 8
```

-Stop the motor (enter idle state)

If the user needs to stop the motor, or wants to modify or save parameters, the user needs to put the motor into idle state first. The instructions are as follows:

```
odrv0.axis0.requested_state = 1
```

- Filtered Position Control

If the user wants to generate the position curve by himself and then send the position control command at a certain frequency, it is recommended to use the filter position control, because this mode smoothly connects these commands together for execution. If the trapezoidal curve position control is used in this case, the motor rotation may feel jerky or grainy.

In this mode, you need to adjust the filter bandwidth according to the frequency of sending commands. A good experience is to set the bandwidth

```
odrv0.axis0.controller.config.input_filter_bandwidth = 25
```

It is half of the command frequency (unit: Hz). If the command is sent at a frequency of 50Hz, then:

Enable filtered position control:

Then perform position control:

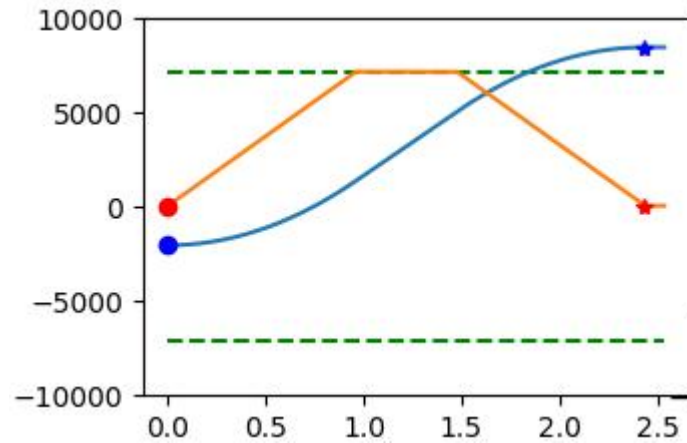
```
odrv0.axis0.controller.config.control_mode = 3
```

```
odrv0.axis0.controller.config.input_mode = 3  
twenty four
```

```
odrv0.axis0.controller.input_pos = 10 #unit turns
```

- Trajectory Control

This mode allows the user to set acceleration, glide speed and deceleration to control the motor to move smoothly from one position to another. The so-called "trapezoidal" means that its speed curve looks like a trapezoid, as shown in the figure below, orange is speed, and blue is position:



```
odrv0.axis0.traj_traj.config.vel_limit # Maximum glide speed, unit turn/s
odrv0.axis0.traj_traj.config.accel_limit # Maximum acceleration, unit turn/s^2
odrv0.axis0.traj_traj.config.decel_limit # Maximum deceleration, unit turn/s^2
odrv0.axis0.controller.config.inertia # Inertia, unit Nm/(turn/s^2)
```

Adjustable control parameters:

Please note that inertia x acceleration = torque, and this value defaults to 0. This value can improve system response, but it is directly related to the load of the motor. The above four values are all greater than or equal to 0. It should also be noted that the current threshold and speed threshold mentioned above will still work globally. For example, if the above maximum glide speed is set higher than the system-level vel_limit, the global vel_limit will work.

```
odrv0.axis0.controller.config.control_mode = 3
odrv0.axis0.controller.config.input_mode = 5
```

To enable trapezoidal curve control mode, first:

Then perform position control:

- Circular Position Control

```
odrv0.axis0.controller.input_pos = 10 #unit turns
```

This mode is suitable for continuous incremental position control, such as when the robot wheel rotates in one direction for a period of time, or the conveyor belt keeps running. If the usual position control mode is used, the target position will gradually increase to a large value, resulting in inaccurate positioning errors due to floating point precision problems.

```
odrv0.axis0.controller.config.circular_setpoints = 1
```

Enable:

In this mode, each small step is within a single turn, and the range of input_pos is [0, 1). If input_pos increases beyond this range, it will be converted to a value within a single turn. If the user wants a single step to exceed a single turn, the following parameters can be set to be greater than

```
odrv0.axis0.controller.config.circular_setpoint_range = <N>
```

Number of 1s:

- **Direct velocity control (Velocity Control)**

```
odrv0.axis0.controller.config.control_mode = 2  
odrv0.axis0.controller.config.input_mode = 1
```

This mode is the simplest speed control and is enabled as follows:

Then enter the target speed for control:

```
odrv0.axis0.controller.input_vel = 10 #Unit turn/s
```

- **Ramped Velocity Control**

The ramp speed control mode is to gradually increase the speed to the target value according to a certain slope, which is more efficient than the direct speed control mentioned above.

```
odrv0.axis0.controller.config.control_mode = 2  
odrv0.axis0.controller.config.input_mode = 2
```

Add mitigation to enable the following:

Control the acceleration by adjusting the slope:

Then enter the target speed for control:

```
odrv0.axis0.controller.config.vel_ramp_rate = 0.5 #slope unit is turn/s^2  
odrv0.axis0.controller.input_vel = 10 #unit is turn/s
```

- **Direct torque control (Torque Control)**


```
odrv0.axis0.controller.config.control_mode = 1
odrv0.axis0.controller.config.input_mode = 1
```

This is the simplest torque (current) control mode and is enabled as follows:

The unit of torque control is Nm, and the current unit in the driver firmware is A, so the torque constant also needs to be set to enable the driver to convert Nm into current, thereby driving the motor to output torque as required.

```
# The torque constant is approximately equal to 8.23/12.3
odrv0.axis0.motor.config.torque_constant = 8.23/12.3
```

```
odrv0.axis0.controller.input_torque = 1.2 #Unit: Nm
```

Then enter the target speed for control:

It should also be noted that if the user wants to limit the maximum speed in torque mode, he can turn on enable_torque_mode_vel_limit and set vel_limit, such as:

```
odrv0.axis0.controller.config.enable_torque_mode_vel_limit = 1
odrv0.axis0.controller.config.vel_limit = 30 #Unit turn/s
```

- Ramped Torque Control

```
odrv0.axis0.controller.config.control_mode = 1
odrv0.axis0.controller.config.input_mode = 6
```

Ramp torque control is very similar to ramp speed control and is enabled as follows:

Adjust the slope as follows:

```
odrv0.axis0.controller.config.torque_ramp_rate = 0.1 #The slope unit is Nm/s
```

- Motion Control (MIT Control)

The motion control mode controls the motor to move to the target position by comprehensively controlling position, speed and torque, which can be expressed by the following formula:

$$\text{Torque} = \text{Position} \times \text{Torque_Ramp_Rate} + \text{Velocity} \times \text{Velocity_Limit} + \text{Torque_Limit}$$

$$\text{Position} = \text{Position_Limit} - \text{Position}$$

$$\text{Velocity} = \text{Velocity_Limit} - \text{Velocity}$$

in τ_{target} is the target torque, e_{pos} is the position error, e_{vel} is the speed error, K_{p} is the position control gain, K_{v} is the speed control gain (or damping coefficient), $\tau_{\text{feedforward}}$ is the feedforward torque.

```
odrv0.axis0.controller.config.control_mode = 3
odrv0.axis0.controller.config.input_mode = 9
```

The motion control mode is enabled as follows:

Adjust the gain:

```
odrv0.axis0.controller.input_pos = 5 #unit turns
odrv0.axis0.controller.config.kp = 10 #set gain, unit Nm/turn
odrv0.axis0.controller.config.kv = 2 #set gain, unit Nm/turn/s
odrv0.axis0.controller.config.kd = 0.1 #set gain, unit Nm/turn/s
```

Then perform motion control by inputting input_pos, input_vel, input_torque:

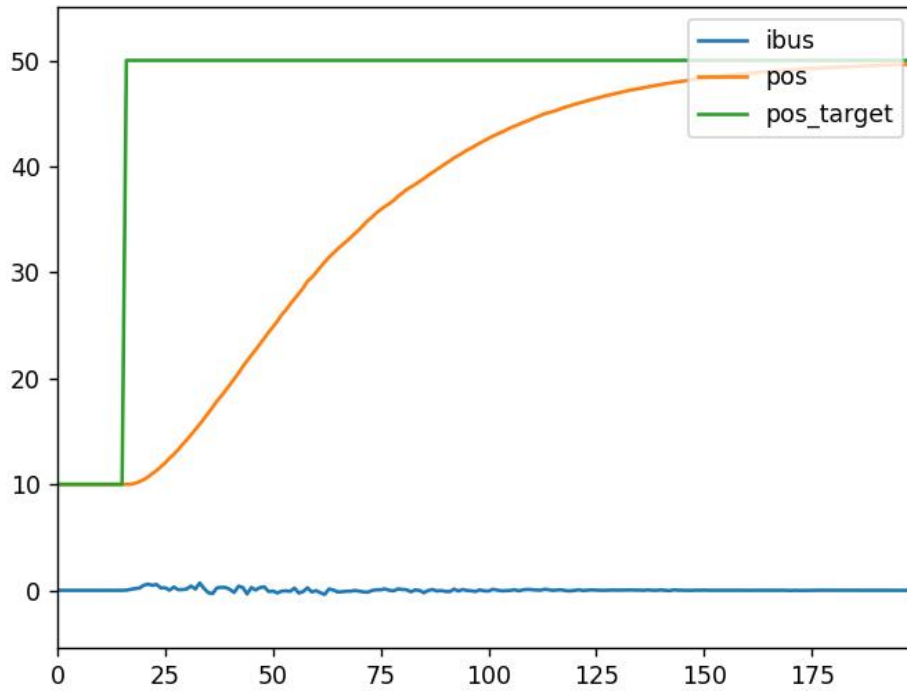
Please note that the position, speed and torque input during USB control refer to the rotor side, while when using CAN for MIT control, the position, speed and torque in the protocol refer to the output shaft side, in order to be consistent with the MIT open source protocol!

3.1.7 Advanced

1) List of commonly used commands

After the connection is successful, the user can control the motor through instructions and obtain the motor operation parameters. The following table shows the common instructions and debugging process and description:

type	instruction	illustrate
Base instruction	dump_errors(odrv0)	Print all error messages
	odrv0.clear_errors()	Clear all error messages
	odrv0.save_configuration()	After modifying the parameters, or the motor automatically recognizes the parameters or calibrates After approval, please be sure to execute this command to save the changes. All changes will be lost after a power outage.
	odrv0.reboot()	Restart the drive
	odrv0.vbus_voltage	Get the power supply voltage (V)
	odrv0.ibus	Get the power supply current (A)
	odrv0.hw_version_major	Hardware major version number, 6010-8 The current major version number is 3
	odrv0.hw_version_minor	Hardware minor version number, 6010-8 The current minor version number is 8
	odrv0.hw_version_variant	Different model numbers under the same hardware configuration, 6010-8 The corresponding model number is 1
	odrv0.can.config.r120_gpio_num	GPIO number to control the 120R matching resistor switch of the CAN interface
	odrv0.can.config.enable_r120	Control the 120R matching resistor switch of the CAN interface



3) USB and CAN compatibility

In the earlier versions of this product (hardware version is less than or equal to 3.7, which can be obtained by the instructions `odrv0.hw_version_major` and `odrv0.hw_version_minor` in the next section 3.1.7), USB and CAN are incompatible. You can switch between the two communication modes in the following way (those with hardware version greater than 3.7 can ignore this section):

- Switch to CAN during USB communication

When CAN is disabled, users can use the Type-C interface to communicate. At this time, you can switch to CAN through the following instructions:

```
odrv0.config.enable_can_a = True
odrv0.axis0.requested_state = AXIS_STATE_IDLE
odrv0.save_configuration()
```

- Switch to USB during CAN communication

When CAN is enabled, the user first switches the motor to the idle state by sending the message `Set_Axis_State` (parameter 1, indicating the IDLE state), and then switches to USB by sending the `Disable_Can` message (see 4.1.2).

Please note that no matter you switch from USB to CAN or from CAN to USB, the motor must be in the IDLE state first, otherwise the switch will fail.

4) CAN matching resistor switch

The driver has a 120 ohm impedance matching resistor on board, which can be turned on or off as needed. Example:

```
odrv0.can.config.r120_gpio_num = 5
odrv0.can.config.enable_r120 = True
```

as follows:

5) User zero configuration

By default, the position read from the motor and the input value for position control are based on the zero point of the absolute encoder on the drive. However, in user scenarios, the zero point of the encoder is not the user zero point in most cases, so the user needs to manually set the zero point offset.

Generally speaking, users can locate the zero point in two ways. One is through the limit switch, and the other is to manually set the zero offset, that is, the offset value of the user's zero point relative to the encoder zero point:

```
# After the user rotates to the desired user zero position manually or  
through position control: odrv0.axis0.encoder.config.index_offset =  
odrv0.axis0.encoder.pos_estimate
```

6) Limit switch

The drive supports two limit switches (LW1 and LW2), where LW1 is the minimum position and also the zero position, and LW2 is

```
odrv0.axis0.min_endstop.config.enabled = True  
odrv0.axis0.min_endstop.config.gpio_num = 1  
odrv0.axis0.max_endstop.config.enabled = True  
odrv0.axis0.max_endstop.config.gpio_num = 2
```

Maximum position. To use two limit switches, use the following configuration:

When the limit switch is triggered, the system will report a MIN_ENDSTOP_PRESSED or MAX_ENDSTOP_PRESSED error, and the host computer can perform related operations at this time.

Please note that hardware version 3.7 does not support the limit switch function.

3.2 Firmware Update Download

The firmware can be burned through the SWD interface (2.4.4) or Type-C interface (2.4.2), providing the following three methods:

3.2.1 National Download Software

1. USB (DFU) flashing

Please note that the national download software can be burned through the Type-C interface, and can also be burned through the SWD interface (only supports JLink and DAP). This section mainly takes the Type-C interface burning as an example.

First, download the USB driver of the national burning software (<https://www.cyberbeast.cn/filedownload/789489>) and install the corresponding system driver; then, download the national burning software (<https://cyberbeast.cn/filedownload/766844>), unzip it to any directory, and run it.

Then, connect the Type-C port, enter odrivetool, and execute the following command to put the drive into DFU mode:

```
odrv0.enter_dfu_mode()
```

Finally, use the national burning software to burn, as shown in the figure below. Please note that after the burning is completed, please click "Common Operations" and then click "Reset" to restart the drive and connect and debug normally through odrivetool.



2. SWD (JLink or DAP) programming

Downloading using SWD is similar to DFU mode, but requires connecting via the SWD debug interface (2.4.4) and selecting the corresponding debugging tool (JLink or DAP) in the figure above.

3.2.2 pyocd

pyocd is the python version of openOCD, which can support STLink, JLink, DAP and other general debugging tools for erase, burn, reset and other operations. **Please note that the drive must be connected using the SWD interface.** For the SWD line sequence, please refer to 2.4.4. **There is a 3.3V power supply in the SWD interface. Please do not connect the wires in the wrong order to avoid damaging the drive!**

```
pip install pyocd
```

1. Installation

2. Burn

```
pyocd list
```

First, list the connected debugging tools:

```

管理员: Windows PowerShell
PS D:\projects\cheetah\ODrive\Firmware\keil\Objects> pyocd list
#   Probe/Board   Unique ID           Target
-----
0   STM32 STLink   6000470018000037544B524E n/a
PS D:\projects\cheetah\ODrive\Firmware\keil\Objects>

```

Then, execute the following command to burn the bin file:

```
pyocd load .\ODrive_N32G455.bin -a 0x8000000
```

```

管理员: Windows PowerShell
PS D:\projects\cheetah\ODrive\Firmware\keil\Objects> pyocd load .\ODrive_N32G455.bin -a 0x8000000
0000477 W Generic 'cortex_m' target type is selected by default; is this intentional? You will be able
to debug most devices, but not program flash. To set the target type use the '--target' argument or '
target_override' option. Use 'pyocd list --targets' to see available targets types. [board]
0000529 I Loading D:\projects\cheetah\ODrive\Firmware\keil\Objects\ODrive_N32G455.bin at 0x08000000 [l
oad_cmd]
[=====] 100%
0004543 I Erased 0 bytes (0 sectors), programmed 230112 bytes (0 pages), skipped 0 bytes (0 pages) at
56.02 kB/s [loader]
PS D:\projects\cheetah\ODrive\Firmware\keil\Objects>

```

3.2.3 Motor Wizard (coming soon)

4 Communication Protocol and Examples

4.1 CAN Protocol

The default communication interface is CAN, with a maximum communication rate of 1Mbps (can be read and set via

odrv0.can.config.baud_rate), and a factory default rate of 500Kbps. **Please note: USB and CAN are not compatible in earlier hardware versions (less than or equal to 3.7). Please refer to 3.1.7, item 3) for how to switch from USB to CAN.**

4.1.1 Protocol frame format

CAN communication uses a standard frame format, data frame, 11-bit ID, 8-byte data, as shown in the following table (MSB on the left and LSB on the right):

Data Domain	CAN ID (11 bits)		Data (8 bytes)
Segmentation	Bit10 ~ Bit5	Bit4 ~ Bit0	Byte0 ~ Byte7
describe	node_id	cmd_id	Communication data

- node_id: represents the unique ID of this motor on the bus, which can be read and set in odrivetool using `odrv0.axis0.config.can.node_id`.
- cmd_id: command code, indicating the message type of the protocol, see the rest of this section.

- Communication data: 8 bytes. The parameters carried in each message are encoded as integers or floating point numbers. The byte order is small endian. Floating point numbers are encoded according to the IEEE 754 standard (available on the website <https://www.h-schmidt.net/FloatConverter/IEEE754.html> Test encoding).

Take the Set_Input_Pos message described in 4.1.2 as an example. Assume that its three parameters are: Input_Pos=3.14, Vel_FF=1000 (indicating 1rev/s), Torque_FF=5000 (indicating 5Nm), and the CMD ID of the Set_Input_Pos message is 0x00C. Assume that the node (node_id) of the drive is set to 0x05, then:

- 11-bit CAN ID=(0x05<<5)+0x0C=0xAC
- According to the description of Set_Input_Pos in 4.1.2, Input_Pos is encoded as C3 F5 48 40 (the floating point number 3.14 is encoded as 32-bit number 0x4048f5c3 using the IEEE 754 standard) in the 4th byte, Vel_FF is encoded as E8 03 (1000=0x03E8) in the 2 bytes starting from the 6th byte, Torque_FF is encoded as 88 13 (5000=0x1388) in the 2 bytes starting from the 6th byte, so the 8-byte communication data is:

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
C3	F5	48	40	E8	03	88	13

4.1.2 Frame Message

The following table lists all available messages:

CMD ID	name	Direction	parameter
0x001	Heartbeat	Motor-Host	Axis_Error Axis_State Motor_Flag Encoder_Flag Controller_Flag Traj_Done Life
0x002	Estop	Host-Motor	
0x003	Get_Error	Motor-Host	Error_Type
0x004	RxF	Motor-Host	
0x005	TxD	Motor-Host	
0x006	Set_Axis_Node_ID	Host-Motor	Axis_Node_ID
0x007	Set_Axis_State	Host-Motor	Axis_Requested_State
0x008	Mit_Control	Host-Motor	
0x009	Get_Encoder_Estimates	Motor-Host	Pos_Estimate Vel_Estimate
0x00A	Get_Encoder_Count	Motor-Host	Shadow_Count Count_In_Cpr
0x00B	Set_Controller_Mode	Host-Motor	Control_Mode Input_Mode
0x00C	Set_Input_Pos	Host-Motor	Input_Pos Vel_FF

			Torque_FF
0x00D	Set_Input_Vel	Host-Motor	Input_Vel Torque_FF
0x00E	Set_Input_Torque	Host-Motor	Input_Torque
0x00F	Set_Limits	Host-Motor	Velocity_Limit Current_Limit
0x010	Start_Anticogging	Host-Motor	
0x011	Set_Traj_Vel_Limit	Host-Motor	Traj_Vel_Limit
0x012	Set_Traj_Accel_Limits	Host-Motor	Traj_Accel_Limit Traj_Decel_Limit
0x013	Set_Traj_Inertia	Host-Motor	Traj_Inertia
0x014	Get_Iq	Motor-Host	Iq_Setpoint Iq_Measured
0x015	Get_Sensorless_Estimates	Motor-Host	Pos_Estimate Vel_Estimate
0x016	Reboot	Host-Motor	
0x017	Get_Bus_Voltage_Current	Motor-Host	Bus_Voltage Bus_Current
0x018	Clear_Errors	Host-Motor	
0x019	Set_Linear_Count	Host-Motor	Linear_Count
0x01A	Set_Pos_Gain	Host-Motor	Pos_Gain
0x01B	Set_Vel_Gains	Host-Motor	Vel_Gain Vel_Integrator_Gain
0x01C	Get_Torques	Motor-Host	Torque_Setpoint Torque
0x01D	Get_Powers	Motor-Host	Electrical_Power Mechanical_Power
0x01E	Disable_Can	Host-Motor	
0x01F	Save_Configuration	Host-Motor	

A detailed description of all messages is as follows:

- Heartbeat

CMD ID: 0x001 (motor-host)

The heartbeat format for firmware versions less than (including) 0.5.11 is as follows:

Start Byte	name	type	odrivetool access
0	Axis_Error	uint32	odrv0.axis0.error
4	Axis_State	uint8	odrv0.axis0.current_state
5	Motor_Flag	uint8	1: odrv0.axis0.motor.error is not 0 0: odrv0.axis0.motor.error is 0
6	Encoder_Flag	uint8	1: odrv0.axis0.encoder.error is not 0 0: odrv0.axis0.encoder.error is 0

7	Controller_Flag	uint8	bit7: odrv0.axis0.controller.trajectory_done bit0: 1: odrv0.axis0.controller.error is not 0 0: odrv0.axis0.controller.error is 0
---	-----------------	-------	---

The heartbeat format for firmware versions greater than (including) 0.5.12 is as follows:

Start Byte	name	type	odrivetool access
0	Axis_Error	uint32	odrv0.axis0.error
4	Axis_State	uint8	odrv0.axis0.current_state
5	Flags	uint8	bit0: Is odrv0.axis0.motor.error 0? bit1: Is odrv0.axis0.encoder.error 0? bit2: Is odrv0.axis0.controller.error 0? bit7: odrv0.axis0.controller.trajectory_done, that is, whether the position curve has been executed.
6	Reserved	uint8	reserve
7	Life	uint8	The life value of the periodic message is increased by 1 for each heartbeat message. The range is 0-255. If this life value is not continuous, it means that the heartbeat message is lost, that is, the communication is unstable.

- Estop

CMD ID: 0x002 (host-motor) No parameters and no data.

This command will cause the motor to stop urgently and report the ESTOP_REQUESTED exception.

- Get_Error

CMD ID: 0x003 (motor-host)

Input (host-motor):

Start Byte	name	type	illustrate
0	Error_Type	uint8	0: Get motor abnormality 1: Get encoder exception 2: Get no sense of abnormality 3: Get controller exception

Output (motor-host):

Start Byte	name	type	odrivetool access
0	Error	uint32	Different input Error_Type: 0: odrv0.axis0.motor.error 1: odrv0.axis0.encoder.error 2: odrv0.axis0. 3: odrv0.axis0.controller.error

- Set_Axis_State

CMD ID: 0x007 (Host-Motor)

Start Byte	name	type	odrivetool access
0	Axis_Requested_State	uint32	odrv0.axis0.requested_state

- Mit_Control

CMD ID: 0x008

This is an implementation that simulates the MIT open source motion control protocol (<https://github.com/mit-biomimetics/Cheetah-Software>).

Please note that the position, speed and torque input during USB control refer to the rotor side, while when using CAN for MIT control, the position, speed and torque in the protocol refer to the output shaft side, in order to be consistent with the MIT open source protocol!

- Host-Motor

CAN number	meaning	illustrate
Frame bit		
BYTE0	Location : A total of 16 bit\$BYTE0 is the high 8 bits, BYTE1	actual Location It is of double type and needs to be converted to 16
BYTE1	The lower 8 bits Multi-turn position of the output shaft in radians (RAD)	The bit is int type, and the conversion process is: $\text{pos_int} = (\text{pos_double} + 12.5) * 65535 / 25$
BYTE2	speed : A total of 12 bits, BYTE2 is the highest 8 bits,	actual speed It is of double type and needs to be converted to 12
BYTE3	BYTE3[7-4] (high 4 bits) is its low 4 bits.	The bit is int type, and the conversion process is:
BYTE4	Angular velocity of the output shaft, in RAD/s KP Value : A total of 12 bits, BYTE3[3-0] (lower 4 Byte) is its upper 4 bits, and BYTE4 is its lower 8 bits.	$\text{vel_int} = (\text{vel_double} + 65) * 4095 / 130$ KP Value The actual type is double and needs to be converted to 12 The bit is int type, and the conversion process is: $\text{kp_int} = \text{kp_double} * 4095 / 500$
BYTE5	KD value : A total of 12 bits, BYTE5 is the highest 8 bits,	KD value The actual type is double and needs to be converted to 12
BYTE6	BYTE6[7-4] (higher 4 bits) is the lower 4 bits.	The bit is int type, and the conversion process is:
BYTE7	Torque : A total of 12 bits, BYTE6[3-0] (lower 4 bits) The upper 4 bits are BYTE7, and the lower 8 bits are BYTE7. It is Nm.	$\text{kd_int} = \text{kd_double} * 4095 / 5$ actual Torque It is of double type and needs to be converted to 12 The bit is int type, and the conversion process is: $\text{t_int} = (\text{t_double} + 50) * 4095 / 100$

		The unit of torque constant is Nm/A
--	--	-------------------------------------

- Motor-Host

CAN number	meaning	illustrate
Frame bit		
BYTE0	node id	Driver node id
BYTE1	Location : A total of 16 bits, BYTE1 is the high 8 bits, BYTE2 is the lower 8 bits	actual Location Double type, need to be converted from 16-bit int The conversion process is as follows: $\text{pos_double} = \text{pos_int} * 25 / 65535 - 12.5$
BYTE2	Multi-turn position of the output shaft in radians (RAD)	
BYTE3	speed : A total of 12 bits, BYTE3 is the highest 8 bits, BYTE4[7-4] (higher 4 bits) is the lower 4 bits.	actual speed Double type, need to be converted from 12-bit int The conversion process is as follows: $\text{vel_double} = \text{vel_int} * 130 / 4095 - 65$ actual Torque Double type, need to be converted from 12-bit int The conversion process is as follows: $\text{t_double} = \text{t_int} * 100 / 4095 - 50$ The unit of torque constant is Nm/A
BYTE4	Angular velocity of the output shaft, in RAD/s	
BYTE5	Torque : A total of 12 bits, BYTE4[3-0] (lower 4 bits) The upper 4 bits are BYTE5, and the lower 8 bits are BYTE5. It is Nm.	

- Get_Encoder_Estimates

CMD ID: 0x009 (motor-host)

Start Byte	name	type	unit	odrivetool access
0	Pos_Estimate	float32	rev	odrv0.axis0.encoder.pos_estimate
4	Vel_Estimate	float32	rev/s	odrv0.axis0.encoder.vel_estimate

- Get_Encoder_Count

CMD ID: 0x00A (motor-host)

Start Byte	name	type	odrivetool access
0	Shadow_Count	int32	odrv0.axis0.encoder.shadow_count
4	Count_In_Cpr	int32	odrv0.axis0.encoder.count_in_cpr

- Set_Controller_Mode

CMD ID: 0x00B (host-motor)

Start Byte	name	type	odrivetool access
0	Control_Mode	uint32	odrv0.axis0.controller.config.control_mode
4	Input_Mode	uint32	odrv0.axis0.controller.config.input_mode

- Set_Input_Pos

CMD ID: 0x00C (Host-Motor)

Start Byte	name	type	unit	odrivetool access
0	Input_Pos	float32	rev	odrv0.axis0.controller.input_pos
4	Vel_FF	int16	0.001rev/s	odrv0.axis0.controller.input_vel
6	Torque_FF	int16	0.001Nm	odrv0.axis0.controller.input_torque

- Set_Input_Vel

CMD ID: 0x00D (host-motor)

Start Byte	name	type	unit	odrivetool access
0	Input_Vel	float32	rev/s	odrv0.axis0.controller.input_vel
4	Torque_FF	float32	Nm	odrv0.axis0.controller.input_torque

- Set_Input_Torque

CMD ID: 0x00E (Host-Motor)

Start Byte	name	type	unit	odrivetool access
0	Input_Torque	float32	Nm	odrv0.axis0.controller.input_torque

- Set_Limits

CMD ID: 0x00F (Host-Motor)

Start Byte	name	type	unit	odrivetool access
0	Velocity_Limit	float32	rev/s	odrv0.axis0.controller.config.vel_limit
4	Current_Limit	float32	A	odrv0.axis0.motor.config.current_lim

- Start_Anticogging

CMD ID: 0x010 (Host-Motor)

Perform torque ripple calibration.

- Set_Traj_Vel_Limit

CMD ID: 0x011 (Host-Motor)

CAN ID	Frame Type	Frame data	illustrate
0x00B	Data Frame	03 00 00 00 03 00 00 00	Message: Set_Controller_Mode Parameters: 3/3 Set the control mode to position control, input mode Filter for position
0x007	Data Frame	08 00 00 00 00 00 00 00	Message: Set_Axis_State Parameters: 8 Entering closed-loop control state
0x0C	Data Frame	CD CC 0C 40 00 00 00 00	Message: Set_Input_Pos Parameters: 2.2/0/0 Set the target position, velocity feedforward and torque feedforward Feedback, where the target position is 2.2 (floating point number: 0x400CCCCD), torque feedforward and speed feedforward Feedback is 0

4.1.4 CANOpen compatibility

If the node ID is properly assigned, it can interoperate with CANOpen. The following table lists the valid node ID combinations of CANopen and this protocol:

CANOpen node IDs	Node IDs for this protocol
32 ... 127	0x10, 0x18, 0x20, 0x28
64 ... 127	0x10, 0x11, 0x18, 0x19, 0x20, 0x21, 0x28, 0x29
96 ... 127	0x10, 0x11, 0x12, 0x18, 0x19, 0x1A, 0x20, 0x21, 0x22, 0x28, 0x29, 0x2A

4.1.5 Periodic Messages

The user can configure the motor to send messages to the host computer periodically, without the host computer sending request messages to the motor. The periodic message can be turned on/off through a series of configurations under `odrv0.axis0.config.can` (the value 0 means off, and other values indicate the cycle time, in ms), as shown in the following table:

information	odrivetool configuration	default value
Heartbeat	<code>odrv0.axis0.config.can.heartbeat_rate_ms</code>	100
Get_Encoder_Estimates	<code>odrv0.axis0.config.can.encoder_rate_ms</code>	10
Get_Motor_Error	<code>odrv0.axis0.config.can.motor_error_rate_ms</code>	0
Get_Encoder_Error	<code>odrv0.axis0.config.can.encoder_error_rate_ms</code>	0
Get_Controller_Error	<code>odrv0.axis0.config.can.controller_error_rate_ms</code>	0
Get_Sensorless_Error	<code>odrv0.axis0.config.can.sensorless_error_rate_ms</code>	0
Get_Encoder_Count	<code>odrv0.axis0.config.can.encoder_count_rate_ms</code>	0
Get_Iq	<code>odrv0.axis0.config.can.iq_rate_ms</code>	0
Get_Sensorless_Estimates	<code>odrv0.axis0.config.can.sensorless_rate_ms</code>	0
Get_Bus_Voltage_Current	<code>odrv0.axis0.config.can.bus_vi_rate_ms</code>	0

By default, the first two periodic messages are turned on at the factory, so when the user monitors the CAN bus, they will see two messages.

```
odrv0.axis0.config.can.heartbeat_rate_ms = 0  
odrv0.axis0.config.can.encoder_rate_ms = 0
```

The information is broadcast at a set period. Users can turn them off with the following command:

For details of each message, see 4.1.2.

4.2 Python SDK

Please first follow the steps in section 3.1 to install odrivetool (pip install -upgrade odrive). Please refer to 3.1.7, and you can use all the instructions described in this section to perform Python development.

Here are three examples:

```
import odrive  
import time  
  
odrv0 = odrive.find_any()  
odrive.utils.dump_errors(odrv0)  
odrv0.clear_errors()  
odrv0.axis0.requested_state=odrive.utils.AxisState.MOTOR_CALIBRATION  
time.sleep(5)  
while (odrv0.axis0.current_state!=1):  
    time.sleep(0.5)  
odrive.utils.dump_errors(odrv0)  
odrv0.axis0.requested_state=odrive.utils.AxisState.ENCODER_OFFSET_CALIBRATION  
  
time.sleep(6)  
while (odrv0.axis0.current_state!=1):  
    time.sleep(0.5)  
odrive.utils.dump_errors(odrv0)  
odrv0.axis0.motor.config.pre_calibrated=1  
odrv0.axis0.encoder.config.pre_calibrated=1  
odrv0.save_configuration()
```

4.2.1 Practice: Power-on calibration

```
import odrive
import time

odrv0 = odrive.find_any()
odrv0.axis0.controller.config.control_mode=odrive.utils.ControlMode.VELOCITY_CONTROL
odrv0.axis0.controller.config.input_mode=odrive.utils.InputMode.VEL_RAMP

odrv0.axis0.controller.config.vel_ramp_rate=50
odrv0.axis0.requested_state=odrive.utils.AxisState.CLOSED_LOOP_CONTROL
odrv0.axis0.controller.input_vel=15
odrive.utils.dump_errors(odrv0)
time.sleep(5)
odrv0.axis0.controller.input_vel=0
```

4.2.2 Practice: Speed Control

```
import odrive

odrv0 = odrive.find_any()
odrv0.axis0.controller.config.control_mode=odrive.utils.ControlMode.POSITION_CONTROL
odrv0.axis0.controller.config.input_mode=odrive.utils.InputMode.POS_FILTER

odrv0.axis0.requested_state=odrive.utils.AxisState.CLOSED_LOOP_CONTROL
odrv0.axis0.controller.input_pos=10
```

4.2.3 Practice: Position Control

4.2.4 Practice: Data Collection

During the R&D and integration process, users often need to collect motor operation data, such as voltage and current changes, position and speed changes, etc. The Python SDK integrates powerful data capture capabilities and can use simple scripts to capture massive operating data, making R&D and integration easier.

The following code is based on the previous position control example, adding the function of real-time position and current data capture.

```
import odrive
import numpy as np

odrv0 = odrive.find_any() cap =

odrive.utils.BulkCapture(lambda:[odrv0.axis0.motor.current_control.Iq_
measured,odrv0.axis0.encoder.pos_estimate],data_rate=500,duration=2.5)
odrv0.axis0.controller.config.control_mode=odrive.utils.ControlMode.POSI
TION_CONTROL
odrv0.axis0.controller.config.input_mode=odrive.utils.InputMode.POS_FIL
TER

odrv0.axis0.requested_state=odrive.utils.AxisState.CLOSED_LOOP_CONTROL
odrv0.axis0.controller.input_pos=10
np.savetxt("d:/test.csv",cap.data,delimiter=',')
```

Save the data into a csv file.

In the BulkCapture statement, data_rate represents the sampling frequency in hz, duration represents the sampling time in seconds, and the lambda expression can be inserted into any mathematical operation to facilitate data analysis.

4.3 Arduino SDK

Users can use Arduino to control the motor via the CAN bus. The underlying protocol is described in 4.1. Compatible hardware/library:

- Arduino with built-in CAN interface, such as Arduino UNO R4 Minima, Arduino UNO R4 WIFI, etc.
- Teensy development boards with built-in CAN interface can be accessed using the adapted FlexCAN_T4 library (Teensy 4.0 and Teensy 4.1)
- Other Arduino compatible boards can be accessed using the MCP2515 based CAN expansion board

Below is an example showing how to configure a motor to respond to position control commands from an Arduino:

- Configure the motor

In addition to the basic configuration of 3.1.3, configure the control to a control bandwidth of 20rad/s (Arduino Uno's transmission speed is limited, so the control bandwidth does not need to be too high. If you use a faster Arduino, you can increase this bandwidth value):

- Configuring CAN

Configure CAN as follows:

- Installing the ODriveArduino Library

Follow the steps below to install the OdriveArduino library (assuming the user has already installed the Arduino IDE):

- 1) Open Arduino IDE
- 2) Sketch -> Include Library -> Manage Libraries
- 3) Enter "ODriveArduino" to search
- 4) Click the ODriveArduino library found to install it

- Arduino source code

```
# include <Arduino.h>
#include "ODriveCAN.h"

// Documentation for this example can be found here:
// https://docs.odriverobotics.com/v/latest/guides/arduino-can-guide.html

/* Configuration of example sketch -----*/

// CAN bus baudrate. Make sure this matches for every device on the bus
#define CAN_BAUDRATE 500000

// ODrive node_id for odrv0
#define ODRV0_NODE_ID 0

// Uncomment below the line that corresponds to your hardware.
// See also "Board-specific settings" to adapt the details for your hardware setup.

// #define IS_TEENSY_BUILTIN // Teensy boards with built-in CAN interface (eg Teensy 4.1). See below
// to select which interface to use.
// #define IS_ARDUINO_BUILTIN // Arduino boards with built-in CAN interface (eg Arduino Uno R4
// Minima)
// #define IS_MCP2515 // Any board with external MCP2515 based extension module. See below to
// configure the module.

/* Board-specific includes -----*/

# if defined(IS_TEENSY_BUILTIN) + defined(IS_ARDUINO_BUILTIN) +
defined(IS_MCP2515) != 1
# warning "Select exactly one hardware option at the top of this file."

# if CAN_HOWMANY > 0 || CANFD_HOWMANY > 0
#define IS_ARDUINO_BUILTIN
# warning "guessing that this uses HardwareCAN"
# else
# error "cannot guess hardware version"
```

```
# endif

# endif

# ifdef IS_ARDUINO_BUILTIN
// See https://github.com/arduino/ArduinoCore-API/blob/master/api/HardwareCAN.h // and
https://github.com/arduino/ArduinoCore-renesas/tree/main/libraries/Arduino\_CAN

# include <Arduino_CAN.h>
# include <ODriveHardwareCAN.hpp>
# endif // IS_ARDUINO_BUILTIN

#ifdef IS_MCP2515
// See https://github.com/sandeepmistry/arduino-CAN/
#include "MCP2515.h"
#include "ODriveMCPCAN.hpp"
# endif // IS_MCP2515

# ifdef IS_TEENSY_BUILTIN
// See https://github.com/tonton81/FlexCAN\_T4
// clone https://github.com/tonton81/FlexCAN\_T4.git into /src
#include <FlexCAN_T4.h>
#include "ODriveFlexCAN.hpp"
struct ODriveStatus; // hack to prevent teensy compile error
# endif // IS_TEENSY_BUILTIN

/* Board-specific settings -----*/

/* Teensy */

# ifdef IS_TEENSY_BUILTIN

FlexCAN_T4<CAN1, RX_SIZE_256, TX_SIZE_16> can_intf;

bool setupCan() {
    can_intf.begin();
    can_intf.setBaudRate(CAN_BAUDRATE);
    can_intf.setMaxMB(16);
    can_intf.enableFIFO();
}
```

```
    can_intf.enableFIFOInterrupt();
    can_intf.onReceive(onCanMessage);
    return true;
}

#endif // IS_TEENSY_BUILTIN

/* MCP2515-based extension modules -*/

#ifdef IS_MCP2515

MCP2515Class& can_intf = CAN;

// chip select pin used for the MCP2515
#define MCP2515_CS 10

// interrupt pin used for the MCP2515
// NOTE: not all Arduino pins are interruptable, check the documentation for your board!
#define MCP2515_INT 2

// frequency of the crystal oscillator on the MCP2515 breakout board. // common
values are: 16 MHz, 12 MHz, 8 MHz
#define MCP2515_CLK_HZ 8000000

static inline void receiveCallback(int packet_size) {
    if (packet_size > 8) {
        return; // not supported
    }
    CanMsg msg = {.id = (unsigned int)CAN.packetId(), .len = (uint8_t)packet_size};
    CAN.readBytes(msg.buffer, packet_size);
    onCanMessage(msg);
}

bool setupCan() {
    // configure and initialize the CAN bus interface
    CAN.setPins(MCP2515_CS, MCP2515_INT);
    CAN.setClockFrequency(MCP2515_CLK_HZ); if (!
    CAN.begin(CAN_BAUDRATE)) {
        return false;
    }

    CAN.onReceive(receiveCallback);
```

```
    return true;
}

# endif // IS_MCP2515


/* Arduinos with built-in CAN */

# ifdef IS_ARDUINO_BUILTIN

HardwareCAN& can_intf = CAN;

bool setupCan() {
    return can_intf.begin((CanBitRate)CAN_BAUDRATE);
}

# endif


/* Example sketch -----*/

// Instantiate ODrive objects
ODriveCAN odrv0(wrap_can_intf(can_intf), ODRV0_NODE_ID); // Standard CAN message ID
ODriveCAN* odrives[] = {&odrv0}; // Make sure all ODriveCAN instances are accounted for here


struct ODriveUserData {
    Heartbeat_msg_t last_heartbeat; bool
    received_heartbeat = false;
    Get_Encoder_Estimates_msg_t last_feedback; bool
    received_feedback = false;
};

// Keep some application-specific user data for every ODrive.
ODriveUserData odrv0_user_data;

// Called every time a Heartbeat message arrives from the ODrive void
onHeartbeat(Heartbeat_msg_t& msg, void* user_data) {
    ODriveUserData* odrv_user_data = static_cast<ODriveUserData*>(user_data);
    odrv_user_data->last_heartbeat = msg;
    odrv_user_data->received_heartbeat = true;
}

// Called every time a feedback message arrives from the ODrive
```

```
void onFeedback(Get_Encoder_Estimates_msg_t& msg, void* user_data) {
    ODriveUserData* odrv_user_data = static_cast<ODriveUserData*>(user_data);
    odrv_user_data->last_feedback = msg;
    odrv_user_data->received_feedback = true;
}

// Called for every message that arrives on the CAN bus void
onCanMessage(const CanMsg& msg) {
    for (auto odrive: odrives) {
        onReceive(msg, *odrive);
    }
}

void setup() {
    Serial.begin(115200);

    // Wait for up to 3 seconds for the serial port to be opened on the PC side. // If no PC connects,
    continue anyway.
    for (int i = 0; i < 30 && !Serial; ++i) {
        delay(100);
    }
    delay(200);

    Serial.println("Starting ODriveCAN demo");

    // Register callbacks for the heartbeat and encoder feedback messages
    odrv0.onFeedback(onFeedback, &odrv0_user_data);
    odrv0.onStatus(onHeartbeat, &odrv0_user_data);

    // Configure and initialize the CAN bus interface. This function depends on // your hardware
    and the CAN stack that you're using.
    if (!setupCan()) {
        Serial.println("CAN failed to initialize: reset required"); while (true); // spin
        indefinitely
    }

    Serial.println("Waiting for ODrive..."); while (!
    odrv0_user_data.received_heartbeat) {
        pumpEvents(can_intf);
        delay(100);
    }

    Serial.println("found ODrive");
```

```
// request bus voltage and current (1sec timeout) Serial.println("attempting
to read bus voltage and current"); Get_Bus_Voltage_Current_msg_t vbus;

if (!odrv0.request(vbus, 1)) {
    Serial.println("vbus request failed!"); while
    (true); // spin indefinitely
}

Serial.print("DC voltage [V]: ");
Serial.println(vbus.Bus_Voltage);
Serial.print("DC current [A]: ");
Serial.println(vbus.Bus_Current);

Serial.println("Enabling closed loop control..."); while
(odrv0_user_data.last_heartbeat.Axis_State !=
ODriveAxisState::AXIS_STATE_CLOSED_LOOP_CONTROL) {
    odrv0.clearErrors();
    delay(1);
    odrv0.setState(ODriveAxisState::AXIS_STATE_CLOSED_LOOP_CONTROL);

    // Pump events for 150ms. This delay is needed for two reasons;
    // 1. If there is an error condition, such as missing DC power, the ODrive might //
        briefly attempt to enter CLOSED_LOOP_CONTROL state, so we can't rely on the first
    //     heartbeat response, so we want to receive at least two heartbeats (100ms default
    //     interval).
    // 2. If the bus is congested, the setState command won't get through // immediately
        but can be delayed.
    for (int i = 0; i < 15; ++i) {
        delay(10);
        pumpEvents(can_intf);
    }
}

Serial.println("ODrive running!");
}

void loop() {
    pumpEvents(can_intf); // This is required on some platforms to handle incoming feedback CAN
    messages

    float SINE_PERIOD = 2.0f; // Period of the position command sine wave in seconds

    float t = 0.001 * millis();
```

```
float phase = t * (TWO_PI / SINE_PERIOD);

odrv0.setPosition(
    sin(phase), // position
    cos(phase) * (TWO_PI / SINE_PERIOD) // velocity feedforward (optional)
);

// print position and velocity for Serial Plotter if
(odrv0_user_data.received_feedback) {
    Get_Encoder_Estimates_msg_t feedback = odrv0_user_data.last_feedback;
    odrv0_user_data.received_feedback = false;
    Serial.print("odrv0-pos:");
    Serial.print(feedback.Pos_Estimate);
    Serial.print(",");
    Serial.print("odrv0-vel:");
    Serial.println(feedback.Vel_Estimate);
}
}
```

4.4 ROS SDK

The following steps have been tested and verified on Ubuntu 23.04 and ROS2 Iron, but are not supported on MAC and Windows platforms, and have not been verified on other ROS2 versions and need to be corrected before use.

4.4.1 Install the odrive_can package

1. Create a new ROS2 workspace (see <https://docs.ros.org/en/iron/index.html>)
2. Use git clone https://github.com/odriverobotics/odrive_can Download the code to the src directory in the above workspace directory

```
colcon build --packages-select odrive_can
```

3. Go to the root directory of the workspace in the terminal and run:

4. Environment preparation before running:

5. Run the sample node:

```
source ./install/setup.bash
```

```
ros2 launch odrive_can example_launch.yaml
```