

3.1.1.2 WSL (Windows Subsystem for Linux)

1. 安装 python/usb/odrivetool

如果用户安装了 WSL2，可进入 WSL2 的命令行，按照下面的步骤来安装（假设用户 WSL

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

安装的是 Ubuntu)：

第一行指令安装 python，第二行指令安装 usb 驱动，第三行指令安装 odrivetool 上位机。

2. 连接驱动器到 WSL

用 type-C 数据线插入 Windows，默认情况下 Windows 会加载此 USB 端口的驱动，而 WSL 并不会加载。需要将此 USB 端口加载到 WSL 中，请参照微软文档 (<https://learn.microsoft.com/zh-cn/windows/wsl/connect-usb>) 操作。

3.1.1.3 Ubuntu

Ubuntu 下的安装过程跟 WSL 下非常类似，请参见上一小节。

3.1.2 USB 和 CAN 兼容性

在本产品的早期版本（硬件版本小于等于 3.7，可通过下一小节 3.1.3 中的指令 odrv0.hw_version_major 和 odrv0.hw_version.minor 来获取）中，USB 与 CAN 不兼容，可通过下述方式在两种通信方式中切换（硬件版本大于 3.7 的可忽略本节内容）：

➤ USB 通信时切换到 CAN

当禁止 CAN 时，用户可使用 Type-C 接口通信（参见下一小节 3.1.3），此时，可通过下列

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

指令切换到 CAN:

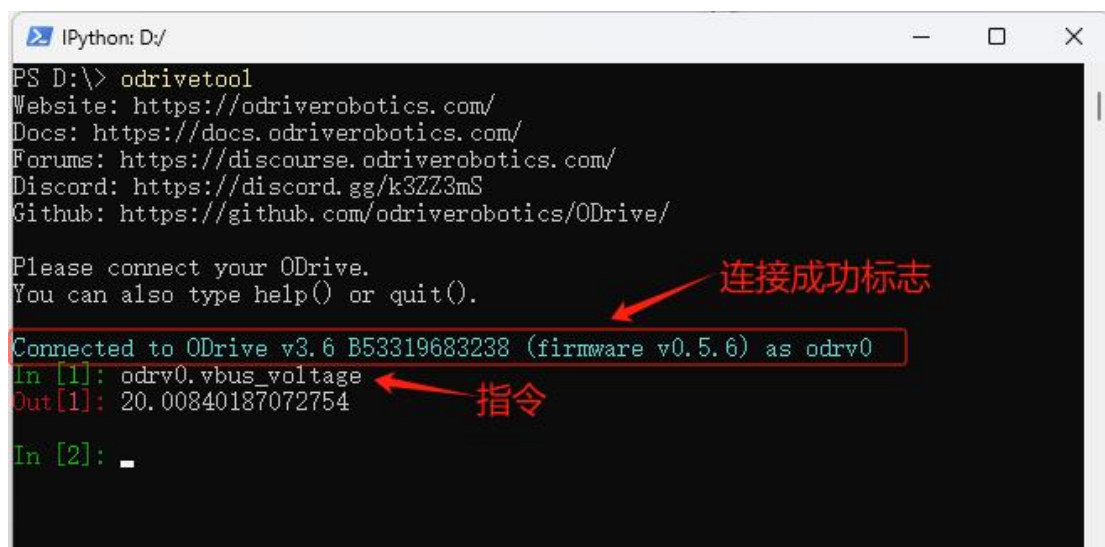
- CAN 通信时切换到 USB

当使能 CAN 时, 用户首先通过发送消息 Set_Axis_State (参数为 1, 表示空闲 (IDLE) 状态) 将电机切换到空闲状态, 再通过发送 Disable_Can 消息来切换到 USB (参见 4.1.2)。

请注意, 无论是从 USB 切换到 CAN, 还是 CAN 切换到 USB, 必须先让电机处于空闲 (IDLE) 状态, 否则会切换失败。

3.1.3 上位机调测

请连接电机主电源(可不连接 CAN 接口)和 Type-C 接口, 在电脑上进入 Windows PowerShell, 运行 odrivetool, 如果连接成功, 会以绿色字体显示如下:



```
IPython: D:/
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]: _
```

如果没有出现上述连接成功标志, 请重新拔插 USB, 并确认电源连接正常, 或参照 3.1.2 查看如何切换 USB 和 CAN 通信。

3.1.3.1 指令列表

在连接成功后, 用户可通过指令对电机进行控制, 并获取电机运行的参数。下表是常用指令和调测过程及说明:

类型	指令	说明
基础指令	dump_errors(odrv0)	打印所有的错误信息
	odrv0.clear_errors()	清除所有的错误信息
	odrv0.save_configuration()	在修改过参数, 或电机自动识别参数或校准过后, 请务必执行此指令存储修改, 否则断电后丢失所有修改。
	odrv0.reboot()	重启驱动器
	odrv0.vbus_voltage	获取电源电压 (V)
	odrv0.ibus	获取电源电流 (A)
	odrv0.hw_version_major	硬件主版本号, GIM6010-8 目前的主版本

		号为 3
	odrv0.hw_version_minor	硬件次版本号, GIM6010-8 目前的次版本号为 7
	odrv0.hw_version_variant	同一硬件配置下的不同型号码, GIM6010-8 对应的型号码为 1
	odrv0.can.config.r120_gpio_num	控制 CAN 接口的 120R 匹配电阻开关的 GPIO 号
	odrv0.can.config.enable_r120	控制 CAN 接口的 120R 匹配电阻开关
	odrv0.can.config.baud_rate	CAN 的波特率设置
参数配置指令	odrv0.config.dc_bus_undervoltage_trip_level	低电压告警门限 (V)
	odrv0.config.dc_bus_overvoltage_trip_level	超电压告警门限 (V)
	odrv0.config.dc_max_positive_current	线电流最大值 (正值) (A)
	odrv0.config.dc_max_negative_current	线电流反向充电最大值 (负值) (A)
	odrv0.axis0.motor.config.resistance_calibration_max_voltage	电机参数识别时最大电压值, 一般此值稍小于电源电压一半, 如 24V 供电, 可设置为 10
	odrv0.axis0.motor.config.calibration_current	电机参数识别时最大电流值, 此值一般可设置为 2~5A, 不可过大, 也不可过小。
	odrv0.axis0.motor.config.torque_constant	电机的力矩常数 (Nm/A)
	odrv0.axis0.encoder.config.index_offset	用户设置的零点偏移, 这个值为用户零点相对于编码器零点的偏移值。在设置这个偏移值并保存设置后, 所有用户输入的位置控制目标值均是以此用户零点为基准。
校准指令	odrv0.axis0.requested_state=4	对电机进行参数识别, 包括识别相电阻, 相电感, 以及对三相电流平衡性做出校准。这个过程会历时 3~6 秒钟, 电机发出尖利的声音。在声音停止后, 或 6 秒后没有声音时, 均运行 dump_errors(odrv0)查看错误, 确认没有错误再进行其他操作。
	odrv0.axis0.requested_state=7	对编码器进行校准。执行此操作前, 请确认电机输出轴没有任何负载, 且用手或其他装置固定住电机。此操作执行后, 电机正向和反向旋转一定时间, 对编码器进行识别和校准。在电机停转后, 运行 dump_errors(odrv0)查看错误, 确认没有错误再进行其他后续步骤。
	odrv0.axis0.encoder.config.pre_calibrate=1	写入预校准成功, 表示不用每次上电进行校准。这个参数仅在上述校准成功过后才能写入, 否则写入会失败。
	odrv0.axis0.controller.config.load_encoder_axis=0	确保当前操作电机为第 0 个电机。此操作仅在 BETA 中必要, 量产版本中无效。

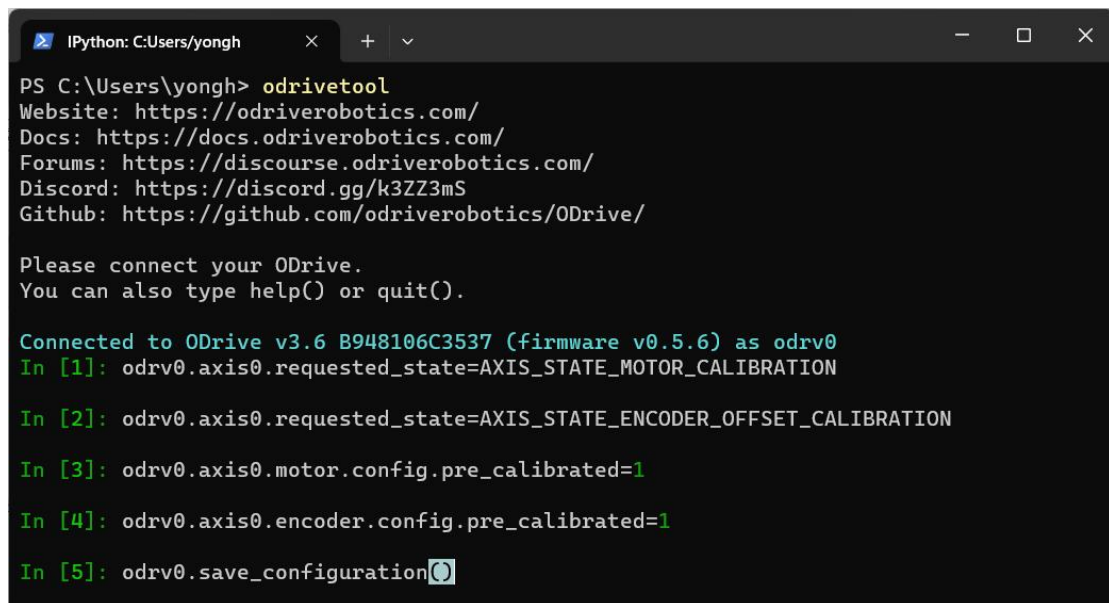
控制指令	odrv0.axis0.requested_state=1	停止电机，进入空闲状态
	odrv0.axis0.requested_state=8	启动电机，进入闭环状态
	odrv0.axis0.motor.config.current_lim	电机运行最大线电流 (A)，超过此值会报过流告警。 请注意，此值不得大于 100。
	odrv0.axis0.controller.config.vel_limit	电机运行最大速度 (turns/s)，电机转子速度超过此值会报超速告警。
	odrv0.axis0.controller.config.enable_vel_limit	速度限制开关，为 True 时上述 vel_limit 生效，为 False 时无效。
	odrv0.axis0.controller.config.control_mode	控制模式。 0: 电压控制 1: 力矩控制 2: 速度控制 3: 位置控制
	odrv0.axis0.controller.config.input_mode	输入模式。表示用户输入的控制值以什么方式去控制电机运转： 0: 闲置 1: 直接控制 2: 速度斜坡 3: 位置滤波 5: 梯形曲线 6: 力矩斜坡
	odrv0.axis0.controller.config.vel_gain	速度环 PID 控制的 P 值
	odrv0.axis0.controller.config.vel_integrator_gain	速度环 PID 控制的 I 值
	odrv0.axis0.controller.config.pos_gain	位置环 PID 控制的 P 值
	odrv0.axis0.controller.input_torque	力矩控制的目标，或速度控制/位置控制的力矩前馈 (Nm)
	odrv0.axis0.controller.input_vel	速度控制的目标，或位置控制的速度前馈 (turns/s)
	odrv0.axis0.controller.input_pos	位置控制的目标 (turns)
	odrv0.axis0.encoder.set_linear_count()	设置编码器的绝对位置，括号中输入 32 位整数，此整数绝对值需要小于 odrv0.axis0.encoder.config.cpr

控制指令	odrv0.axis0.traj_traj.config	包含三个参数： <ul style="list-style-type: none"> ➤ accel_limit: 最大加速度 (rev/s²) ➤ decel_limit: 最大减速度 (rev/s²) ➤ vel_limit: 最大速度 (rev/s) 这三个参数在 odrv0.axis0.controller.config.input_mode 为梯形曲线的时候起作用，调整位置控制的加减速效果。
	odrv0.axis0.controller.config.input_filter_bandwidth	位置滤波带宽，这个参数在 odrv0.axis0.controller.config.input_mode

		为位置滤波的时候起作用，调节位置控制的加减速效果。
--	--	---------------------------

3.1.3.2 实战：上电校准

用户第一次使用微电机时，需要对电机以及编码器进行校准。在校准之前，请固定好电机，或用手握紧，输出轴空载，校准过程如下：



```
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()
```

解释如下：

➤ 第一步：电机参数自识别

测量电机的相电阻和相电感，会听到尖利的“嘀”一声。相电阻和相电感的测量结果可通过下

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

述指令查看：

➤ 第二步：查看错误码

查看第一步过后的系统错误码，如果出现任何的红色错误码，则需要重启电机，然后重试，或者报告售后。

➤ 第三步：编码器校准

对编码器进行校准，包括编码器的安装角度与电机机械角度的校准，以及编码器自身的校准。在这个校准过程中，电机会缓慢正转一个角度，再反转一个角度。如果只正转后就停止，则说明有错误，请通过第四步查看错误码。

➤ 第四步：查看错误码

在第三步编码器校准后，查看系统的错误码。通常出现的错误是 ERROR_CPR_POLEPAIRS_MISMATCH，表示编码器的 CPR 设置错误，或者电机的极对数设置错误，

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

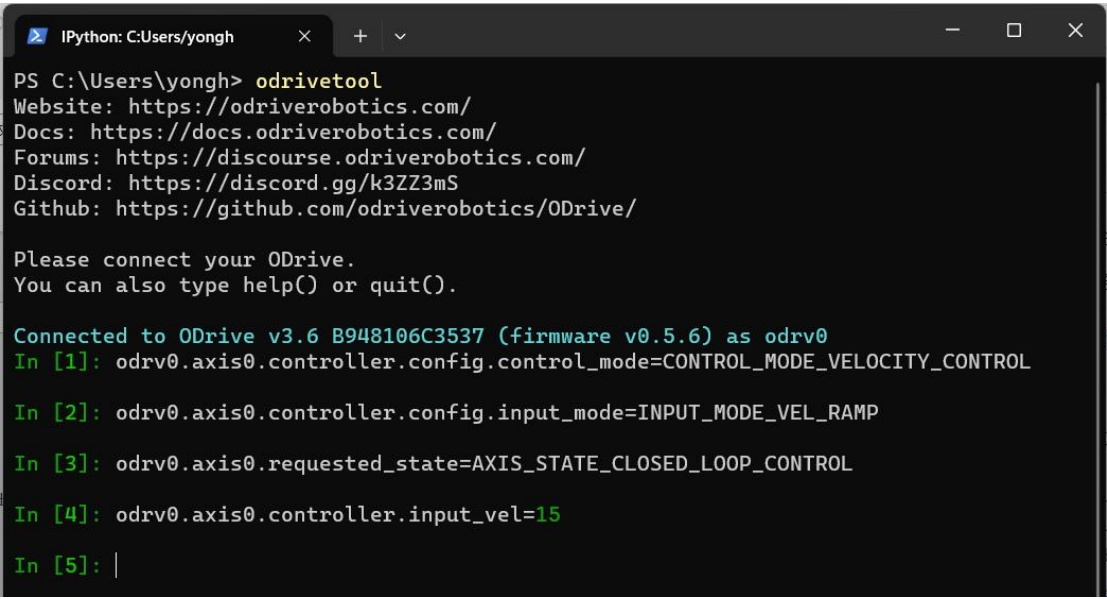
请通过下述指令查看/设置：

- 第五步：写入电机校准成功标志
- 第六步：写入编码器校准成功标志
- 第七步：存储校准结果并重启

3.1.3.3 实战：速度控制

下列指令是进行速度控制的实例：

```
odrv0.axis0.controller.config.control_mode=CONTROL_MODE_VELOCITY_CONTROL  
odrv0.axis0.controller.config.input_mode=INPUT_MODE_VEL_RAMP  
odrv0.axis0.requested_state=AXIS_STATE_CLOSED_LOOP_CONTROL  
odrv0.axis0.controller.input_vel=15
```



```
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.controller.config.control_mode=CONTROL_MODE_VELOCITY_CONTROL  
  
In [2]: odrv0.axis0.controller.config.input_mode=INPUT_MODE_VEL_RAMP  
  
In [3]: odrv0.axis0.requested_state=AXIS_STATE_CLOSED_LOOP_CONTROL  
  
In [4]: odrv0.axis0.controller.input_vel=15  
  
In [5]: |
```

3.1.3.4 实战：位置控制

下列指令控制电机转到 10 转的转子位置：


```
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.controller.config.control_mode=CONTROL_MODE_POSITION_CONTROL
In [2]: odrv0.axis0.controller.config.input_mode=INPUT_MODE_POS_FILTER
In [3]: odrv0.axis0.requested_state=AXIS_STATE_CLOSED_LOOP_CONTROL
In [4]: odrv0.axis0.controller.input_pos=10
In [5]: |
```

```
odrv0.axis0.controller.config.control_mode=CONTROL_MODE_POSITION_CONTROL
odrv0.axis0.controller.config.input_mode=INPUT_MODE_POS_FILTER
odrv0.axis0.requested_state=AXIS_STATE_CLOSED_LOOP_CONTROL
odrv0.axis0.controller.input_pos=10
```

3.1.4 图形化调测

在对电机进行调测时，如果需要实时监控某些运行参数，可以利用 python 强大的计算库和图形库，以及 Type-C 接口的高速吞吐能力实时输出电机参数。

1. 环境准备

```
pip install numpy matplotlib
```

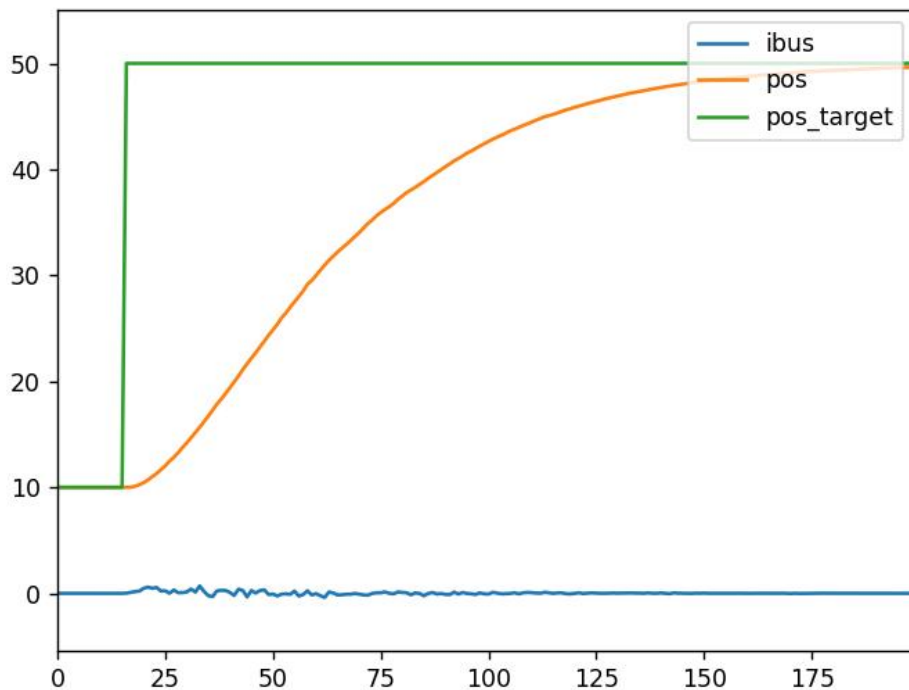
安装计算库和图形库：

2. 图形化参数输出

```
start_liveplotter(lambda:[odrv0.ibus,odrv0.axis0.encoder.pos_estimate,
odrv0.axis0.controller.input_pos],["ibus","pos","pos_target"])
```

在 odrivetool 命令行界面，调起图形库，读取任何电机运行指标，如：

这个指令将调起一个图形界面，实时输出下述三个指标：线电流、位置、目标位置。接下去，对电机进行位置控制，就会看到电机的实时位置控制曲线：



3.1.5 CAN 匹配电阻开关

在驱动器上，已经板载一个 120 欧阻抗匹配电阻，用户可根据需要打开或关闭，指令示例如下：

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

3.1.6 第二编码器

第二编码器的作用是在上电时，根据输出轴的位置来确定转子的初始位置，从而保障电机断电重启时，可正确识别输出轴的单圈位置。可通过下述指令判断是否存在第二编码器，以及第二编码器是否工作正常：`odrv0.axis0.encoder.poll_sec_enc()`，返回 True/False。

3.1.7 用户零点配置

默认情况下，用户从电机读取到的位置，以及做位置控制时的输入值，均是基于驱动器上的绝对值编码器的零点为基准。但是，在用户场景下，编码器的零点在大多数时候并不是用户零点，所以用户需要手动设置这个零点偏移。

一般来说，用户可以通过两种手段来定位这个零点，一种手段是通过限位开关，一种手段是手动设置零点偏移，即用户零点相对于编码器零点的偏移值：

```
# 用户先手动或者通过位置控制转动到期望的用户零点位置后：
odrv0.axis0.encoder.config.index_offset =
odrv0.axis0.encoder.pos_estimate
```

3.1.8 PID 调整

```
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
```

下述过程可以为用户调节 PID 参数提供一个参考：

1. 设置 PID 初始值
2. 将 vel_integrator_gain 调为 0

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

3. 调节 vel_gain 方法：
 - 1) 用速度控制模式转动电机，如果转动不平稳，有抖动或震动，减小 vel_gain，直到转动平稳
 - 2) 接着，每次把 vel_gain 增大 30%左右，直到出现明显的抖动
 - 3) 此时，将 vel_gain 减小 50%左右，即可稳定
4. 调节 pos_gain 方法：
 - 1) 用位置模式尝试转动电机，如果转动不平稳，有拉动或震动，减小 pos_gain，直到转动平稳
 - 2) 接着，每次把 pos_gain 增大 30%左右，直到位置控制出现明显的过调（即每次位置控制电机超出目标位置，然后振荡回到目标位置）
 - 3) 然后，逐渐减小 pos_gain，直到过调现象消失
5. 在上述 4 步调整过后，可将 vel_integrator_gain 设置为 $0.5 \times \text{bandwidth} \times \text{vel_gain}$ ，其中 bandwidth 是系统控制带宽。何为控制带宽？比如，从用户设置目标位置，到电机真正到达目标位置的时间为 10ms，则控制带宽就是 100Hz，那么 $\text{vel_integrator_gain} = 0.5 \times 100 \times \text{vel_gain}$ 。

在上述调参过程中，建议使用 3.1.4 中的图形化手段实时查看调参效果，避免肉眼感知的误差。

3.1.9 电机参数备份与恢复

利用 odrivetool，可将调测完备的电机进行参数备份：

```
odrivetool backup-config d:\test.json
```

其中“d:\test.json”是用户可自由修改的保存路径和文件名。

参数恢复的指令为：

```
odrivetool restore-config d:\test.json
```

3.2 固件更新下载

可以通过 SWD 接口（2.4.4）或 Type-C 接口（2.4.2）烧录固件，提供下述三种方式：

3.2.1 pyocd

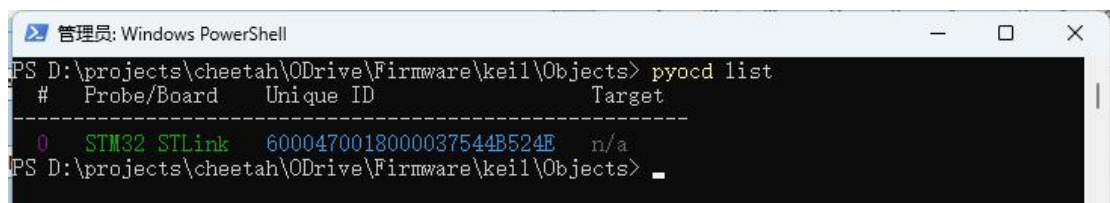
pyocd 是 openOCD 的 python 版本，可支持 STLink, JLink, DAP 等通用调试工具进行擦除、烧写、重置等操作。**请注意，必须用 SWD 接口连接驱动器。**关于 SWD 的线序，请参见 2.4.4。**SWD 接口中有 3.3V 电源，请不要接错线序，以免损毁驱动器！**

```
pip install pyocd
```

1. 安装
2. 烧写

```
pyocd list
```

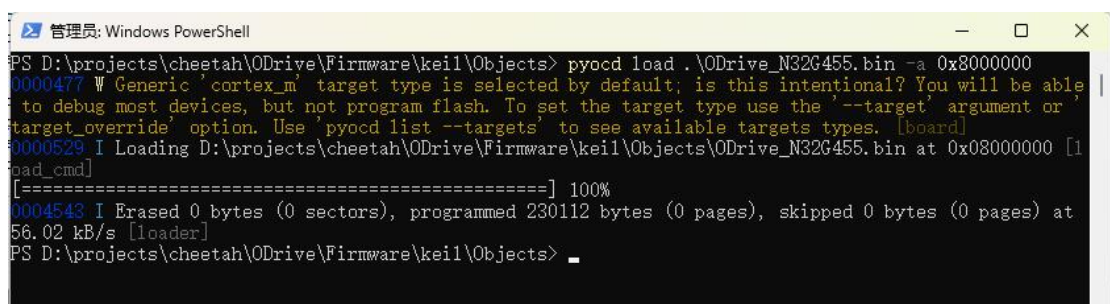
首先，列出连接的调试工具：



```
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> _
```

然后，执行下述指令烧写 bin 文件：

```
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 电机精灵（即将推出）

4 通信协议及示例

4.1 CAN 协议

默认通信接口是 CAN，最大通信速率 1Mbps（可通过 `odrv0.can.config.baud_rate` 读取和设置）。**请注意：早期硬件版本（小于等于 3.7）中 USB 与 CAN 不兼容，请参见 3.1.2 如何从 USB 切换到 CAN。**

4.1.1 协议帧格式

CAN 通信采用标准帧格式，数据帧，11 位 ID，8 字节数据，如下表所示（左为 MSB，右为 LSB）：

数据域	CAN ID (11bits)		Data (8 bytes)
分段	Bit10 ~ Bit5	Bit4 ~ Bit0	Byte0 ~ Byte7
描述	node_id	cmd_id	通信数据

- node_id：代表这个电机在总线上的唯一 ID，可在 `odrivetool` 中用 `odrv0.axis0.config.can.node_id` 来读取和设置。
- cmd_id：指令编码，表示协议的消息类型，请参见本节余下内容。
- 通信数据：8 个字节，每一个消息中携带的参数会被编码成整数或浮点数，字节序为小端（small endian），其中浮点数是按照 IEEE 754 标准进行编码（可通过网站 <https://www.h-schmidt.net/FloatConverter/IEEE754.html> 测试编码）。

以 4.1.2 中描述的 `Set_Input_Pos` 消息为例，假设其三个参数分别为：`Input_Pos=3.14`，`Vel_FF=1000`（表示 1rev/s），`Torque_FF=5000`（表示 5Nm），而 `Set_Input_Pos` 消息的 CMD ID=0x00C，假设驱动器的节点（node_id）被设置成 0x05，则：

- 11 位 CAN ID=(0x05<<5)+0x0C=0xAC
- 根据 4.1.2 中对于 `Set_Input_Pos` 的描述可知，`Input_Pos` 在第 0 个字节开始的 4 个字节，编码为 C3 F5 48 40（浮点数 3.14 用 IEEE 754 标准编码为 32 位数 0x4048f5c3），`Vel_FF` 在第 4 个字节开始的 2 个字节，编码为 E8 03（1000=0x03E8），`Torque_FF` 在第 6 个字节开始的 2 个字节，编码为 88 13（5000=0x1388），则 8 个字节的通信数据为：

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

4.1.2 帧消息

下表列出了所有的可用消息：

CMD ID	名称	方向	参数
0x001	Heartbeat	电机→主机	Axis_Error Axis_State Motor_Flag Encoder_Flag Controller_Flag
0x002	Estop	主机→电机	
0x003	Get_Error	电机→主机	Error_Type
0x004	RxSdo	电机→主机	
0x005	TxSdo	电机→主机	
0x006	Set_Axis_Node_ID	主机→电机	Axis_Node_ID
0x007	Set_Axis_State	主机→电机	Axis_Requested_State
0x009	Get_Encoder_Estimates	电机→主机	Pos_Estimate Vel_Estimate
0x00B	Set_Controller_Mode	主机→电机	Control_Mode Input_Mode
0x00C	Set_Input_Pos	主机→电机	Input_Pos Vel_FF Torque_FF
0x00D	Set_Input_Vel	主机→电机	Input_Vel Torque_FF
0x00E	Set_Input_Torque	主机→电机	Input_Torque
0x00F	Set_Limits	主机→电机	Velocity_Limit Current_Limit
0x010	Start_Anticogging	主机→电机	
0x011	Set_Traj_Vel_Limit	主机→电机	Traj_Vel_Limit
0x012	Set_Traj_Accel_Limits	主机→电机	Traj_Accel_Limit Traj_Decel_Limit
0x013	Set_Traj_Inertia	主机→电机	Traj_Inertia
0x014	Get_Iq	电机→主机	Iq_Setpoint Iq_Measured
0x015	Get_Sensorless_Estimates	电机→主机	Pos_Estimate Vel_Estimate
0x016	Reboot	主机→电机	
0x017	Get_Bus_Voltage_Current	电机→主机	Bus_Voltage Bus_Current
0x018	Clear_Errors	主机→电机	
0x019	Set_Linear_Count	主机→电机	Linear_Count
0x01A	Set_Pos_Gain	主机→电机	Pos_Gain
0x01B	Set_Vel_Gains	主机→电机	Vel_Gain Vel_Integrator_Gain
0x01C	Get_Torques	电机→主机	Torque_Setpoint Torque

0x01D	Get_Powers	电机→主机	Electrical_Power Mechanical_Power
0x01E	Disable_Can	主机→电机	
0x01F	Save_Configuration	主机→电机	

所有消息的详细描述如下：

➤ Heartbeat

CMD ID: 0x001 (电机→主机)

起始字节	名称	类型	odrivetool 访问
0	Axis_Error	uint32	odrv0.axis0.error
4	Axis_State	uint8	odrv0.axis0.current_state
5	Motor_Flag	uint8	1: odrv0.axis0.motor.error 不为 0 0: odrv0.axis0.motor.error 为 0
6	Encoder_Flag	uint8	1: odrv0.axis0.encoder.error 不为 0 0: odrv0.axis0.encoder.error 为 0
7	Controller_Flag	uint8	bit7: odrv0.axis0.controller.trajectory_done bit0: 1: odrv0.axis0.controller.error 不为 0 0: odrv0.axis0.controller.error 为 0

➤ Estop

CMD ID: 0x002 (主机→电机) 无参数无数据。

此指令会导致电机紧急停机，并报 ESTOP_REQUESTED 异常。

➤ Get_Error

CMD ID: 0x003 (电机→主机)

输入 (主机→电机)：

起始字节	名称	类型	说明
0	Error_Type	uint8	0: 获取电机异常 1: 获取编码器异常 2: 获取无感异常 3: 获取控制器异常

输出 (电机→主机)：

起始字节	名称	类型	odrivetool 访问
0	Error	uint32	不同输入 Error_Type: 0: odrv0.axis0.motor.error 1: odrv0.axis0.encoder.error

			2: odrv0.axis0. 3: odrv0.axis0.controller.error
--	--	--	--

➤ RxSdo

CMD ID: 0x004 (主机→电机)

输入:

起始字节	名称	类型	说明
0	opcode	uint8	0: 读 1: 写
1	Endpoint_ID	uint16	请下载所有参数和接口函数对应的 ID 的 JSON 文件: https://cyberbeast.cn/filedownload/784822
3	预留	uint8	
4	Value	uint8[4]	根据 Endpoint_ID 不同而不同, 可参见上述 JSON 中的描述。如 Endpoint_ID 对应一个可读写的 float 值, 则此处 4 个字节为 IEEE 编码而成的 float 值, opcode=1 时将此值写入这个 float 值。

输出 (当上述 opcode=0 时) :

起始字节	名称	类型	说明
0	opcode	uint8	固定为 0
1	Endpoint_ID	uint16	请下载所有参数和接口函数对应的 ID 的 JSON 文件: https://cyberbeast.cn/filedownload/784822
3	预留	uint8	
4	Value	uint8[4]	根据 Endpoint_ID 不同而不同, 可参见上述 JSON 中的描述。如 Endpoint_ID 对应一个可读的 uint32 值, 则此处 4 个字节为小端字节序的 uint32。

➤ TxSdo

CMD ID: 0x005 (电机→主机)

用法跟 opcode=1 时的 RxSdo 一样。

➤ Set_Axis_Node_ID

CMD ID: 0x006 (主机→电机)

起始字节	名称	类型	odrivetool 访问
0	Axis_Node_ID	uint32	odrv0.axis0.config.can.node_id

➤ Set_Axis_State

CMD ID: 0x007 (主机→电机)

起始字节	名称	类型	odrivetool 访问
0	Axis_Requested_State	uint32	odrv0.axis0.requested_state

➤ Get_Encoder_Estimates

CMD ID: 0x009 (电机→主机)

起始字节	名称	类型	单位	odrivetool 访问
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 (电机→主机)

起始字节	名称	类型	odrivetool 访问
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 (主机→电机)

起始字节	名称	类型	odrivetool 访问
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 (主机→电机)

起始字节	名称	类型	单位	odrivetool 访问
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 (主机→电机)

起始字节	名称	类型	单位	odrivetool 访问
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 (主机→电机)

起始字节	名称	类型	单位	odrivetool 访问
0	Input_Torque	float32	Nm	odrv0.axis0.controller.input_torque

➤ Set_Limits

CMD ID: 0x00F (主机→电机)

起始字节	名称	类型	单位	odrivetool 访问
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 (主机→电机)

进行力矩纹波校准。

➤ Set_Traj_Vel_Limit

CMD ID: 0x011 (主机→电机)

起始字节	名称	类型	单位	odrivetool 访问
0	Traj_Vel_Limit	float32	rev/s	odrv0.axis0.traj_traj.config.vel_limit

➤ Set_Traj_Accel_Limits

CMD ID: 0x012 (主机→电机)

起始字节	名称	类型	单位	odrivetool 访问
0	Traj_Accel_Limit	float32	rev/s^2	odrv0.axis0.traj_traj.config.accel_limit
4	Traj_Decel_Limit	float32	rev/s^2	odrv0.axis0.traj_traj.config.decel_limit

➤ Set_Traj_Inertia

CMD ID: 0x013 (主机→电机)

起始字节	名称	类型	单位	odrivetool 访问
0	Traj_Inertia	float32	Nm/(rev/s^2)	odrv0.axis0.controller.config.inertia

➤ Get_Iq

CMD ID: 0x014 (电机→主机)

起始字节	名称	类型	单位	odrivetool 访问
0	Idq_Setpoint	float32	A	odrv0.axis0.motor.current_control.Idq_setpoint
4	Idq_Measured	float32	A	odrv0.axis0.motor.current_control.Idq_measured

➤ Get_Sensorless_Estimates

CMD ID: 0x015 (电机→主机)

起始字节	名称	类型	单位	odrivetool 访问
0	Pos_Estimate	float32	rev	odrv0.axis0.sensorless_estimator.pll_pos
4	Vel_Estimate	float32	rev/s	odrv0.axis0.sensorless_estimator.vel_estimate

➤ Reboot

CMD ID: 0x016 (主机→电机)

➤ Get_Bus_Voltage_Current

CMD ID: 0x017 (电机→主机)

起始字节	名称	类型	单位	odrivetool 访问
0	Bus_Voltage	float32	V	odrv0.vbus_voltage
4	Bus_Current	float32	A	odrv0.ibus

➤ Clear_Errors

CMD ID: 0x018 (主机→电机)

清除所有错误和异常。

➤ Set_Linear_Count

CMD ID: 0x019 (主机→电机)

设置编码器绝对位置。

起始字节	名称	类型	odrivetool 访问
0	Linear_Count	int32	odrv0.axis0.encoder.set_linear_count()

➤ Set_Pos_Gain

CMD ID: 0x01A (主机→电机)

起始字节	名称	类型	单位	odrivetool 访问
0	Pos_Gain	float32	(rev/s)/rev	odrv0.axis0.controller.config.pos_gain

➤ Set_Vel_Gains

CMD ID: 0x01B (主机→电机)

起始字节	名称	类型	单位	odrivetool 访问
0	Vel_Gain	float32	Nm/(rev/s)	odrv0.axis0.controller.config.vel_gain
4	Vel_Integrator_Gain	float32	Nm/rev	odrv0.axis0.controller.config.vel_integrator_gain

➤ Get_Torques

CMD ID: 0x01C (电机→主机)

起始字节	名称	类型	odrivetool 访问
0	Torque_Setpoint	float32	odrv0.axis0.controller.torque_setpoint
4	Torque	float32	无。表示当前力矩值。

➤ Get_Powers

CMD ID: 0x01D (电机→主机)

起始字节	名称	类型	odrivetool 访问
0	Electrical_Power	float32	odrv0.axis0.controller.electrical_power
4	Mechanical_Power	float32	odrv0.axis0.controller.mechanical_power

➤ Disable_Can

CMD ID: 0x01E (主机→电机)

禁用 CAN，并重启驱动器。

➤ Save_Configuration

CMD ID: 0x01F (主机→电机)

存储当前的配置，生效并重启。

4.1.3 CAN 协议实战

4.1.3.1 实战：上电校准

发送 CAN 消息的序列如下：

CAN ID	帧类型	帧数据	说明
0x007	数据帧	04 00 00 00 00 00 00 00	消息：Set_Axis_State 参数：4 对电机进行校准

0x007	数据帧	07 00 00 00 00 00 00 00	消息: Set_Axis_State 参数: 7 对编码器进行校准
-------	-----	-------------------------	---

4.1.3.2 实战：速度控制

发送 CAN 消息的序列如下：

CAN ID	帧类型	帧数据	说明
0x00B	数据帧	02 00 00 00 02 00 00 00	消息: Set_Controller_Mode 参数: 2/2 设置控制模式为速度控制, 输入模式为速度斜坡
0x007	数据帧	08 00 00 00 00 00 00 00	消息: Set_Axis_State 参数: 8 进入闭环控制状态
0x0D	数据帧	00 00 20 41 00 00 00 00	消息: Set_Input_Vel 参数: 10/0 设置目标速度和力矩前馈, 其中目标速度为 10 (浮点数: 0x41200000), 力矩前馈为 0 (浮点数: 0x00000000)

4.1.3.3 实战：位置控制

发送 CAN 消息的序列如下：

CAN ID	帧类型	帧数据	说明
0x00B	数据帧	03 00 00 00 03 00 00 00	消息: Set_Controller_Mode 参数: 3/3 设置控制模式为位置控制, 输入模式为位置滤波
0x007	数据帧	08 00 00 00 00 00 00 00	消息: Set_Axis_State 参数: 8 进入闭环控制状态
0x0C	数据帧	CD CC 0C 40 00 00 00 00	消息: Set_Input_Pos 参数: 2.2/0/0 设置目标位置, 速度前馈和力矩前馈, 其中目标位置为 2.2 (浮点数: 0x400CCCCD), 力矩前馈和速度前馈为 0

4.1.4 CANOpen 兼容性

如果 node ID 分配得当, 可与 CANOpen 互通。下表列出了 CANopen 和本协议的有效 node ID 组合：

CANOpen node IDs	本协议 node IDs
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 周期消息

用户可配置电机向上位机周期性发送消息，而不用上位机向电机发送请求消息。可通过 `odrv0.axis0.config.can` 下的一系列配置来打开/关闭周期消息（值为 0 表示关闭，为其他值表示周期时间，单位为 ms），如下表所示：

消息	odrivetool 配置	默认值
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

默认情况下，前两种周期消息在出厂时打开，所以当用户监控 CAN 总线时，会看到两种消

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

息以设定周期进行广播。用户可通过下述指令关闭它们：

对于各个消息的详细内容，请参见 4.1.2。

4.2 Python SDK

请首先参照 3.1 节步骤安装 `odrivetool`（`pip install --upgrade odrive`）。请参见 3.1.3，可利用该小节描述的所有指令，来进行 python 开发。

下面是三个示例：

```
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 实战：上电校准


```
import odrive
import time

odrv0 = odrive.find_any()
odrv0.axis0.controller.config.control_mode=odrive.utils.ControlMode.VELLOCITY_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 实战：速度控制

```
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 实战：位置控制

4.2.4 实战：数据采集

用户在研发集成过程中，经常需要采集电机运行的数据，比如采集电压电流变化，位置速度变化等等，Python SDK 集成了强大的数据抓取能力，可以用简单的脚本实现海量运行数据抓取，让研发和集成变得更加简单。

下面的代码是在上一个位置控制实例的基础上，增加了实时位置和电流数据抓取的功能，并

```
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.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
np.savetxt("d:/test.csv",cap.data,delimiter=',')
```

将数据保存成一个 csv 文件。

其中进行 BulkCapture 的语句中，data_rate 代表采样频率，单位是 hz，duration 代表采样时间，单位是秒，其中的 lambda 表达式可以插入任何的数学运算，以方便数据分析。

4.3 Arduino SDK

用户使用 Arduino 可通过 CAN 总线来控制电机，底层协议如 4.1 中描述。可兼容硬件/库：

- ✓ 内置有 CAN 接口的 Arduino，如 Arduino UNO R4 Minima，Arduino UNO R4 WIFI 等
- ✓ 内置 CAN 接口的 Teensy 开发板可使用适配的 FlexCAN_T4 库 (Teensy 4.0 和 Teensy 4.1) 进行访问
- ✓ 其他 Arduino 兼容的板可使用基于 MCP2515 的 CAN 扩展板进行访问

下面是一个示例，展示如何配置电机响应 Arduino 的位置控制指令：

➤ 配置电机

除了 3.1.2 的基础配置外，将控制配置成控制带宽为 20rad/s (Arduino Uno 的发送速度受限，所以控制带宽不用太高，如果用更快的 Arduino，可提高此带宽值)：

➤ 配置 CAN

按照下述方法配置 CAN：

➤ 安装 ODriveArduino 库

按照下述步骤安装 OdriveArduino 库（假设用户已经安装 Arduino IDE）：

- 1) 打开 Arduino IDE
- 2) Sketch -> Include Library -> Manage Libraries
- 3) 输入"ODriveArduino"进行搜索
- 4) 点击搜索到的 ODriveArduino 库进行安装

➤ Arduino 源码

```
#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 (e.g. Teensy
4.1). See below to select which interface to use.
// #define IS_ARDUINO_BUILTIN // Arduino boards with built-in CAN interface (e.g.
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

以下步骤在 Ubuntu 23.04 和 ROS2 Iron 上经过测试验证，但不支持 MAC 和 Windows 平台，且在其他 ROS2 版本上未验证，需要经过修正后才可使用。

4.4.1 安装 odrive_can 包

1. 新建一个 ROS2 workspace（请参见 <https://docs.ros.org/en/iron/index.html>）
2. 用 git clone https://github.com/odriverobotics/odrive_can 将代码下载到上述 workspace 目录中的 src 目录

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

3. 在 terminal 中转到 workspace 的根目录，并运行：
4. 运行之前的环境准备：
5. 运行例程节点：

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

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