高擎机电

HTM5046 的关节电机驱动说明书

使用说明书 V1.3 2023.3.31

修订记录

序号	修订时间	版本号	文件名称	修订说明	
1	2022/10/27	V1.1	电机驱动板说明书	出版创建	
2	2022/11/24	V1.2	HTM5046 的关节电机驱动说明书		
3	2023/3/31	V1.3	Lively board 使用说明书	齿槽转矩板固件更新，ID 号设置	

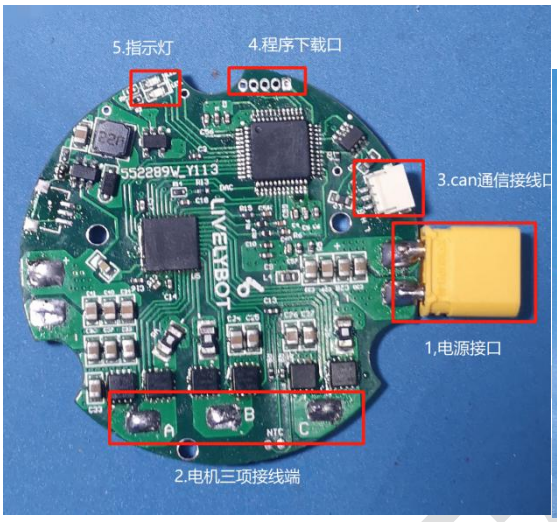
目录

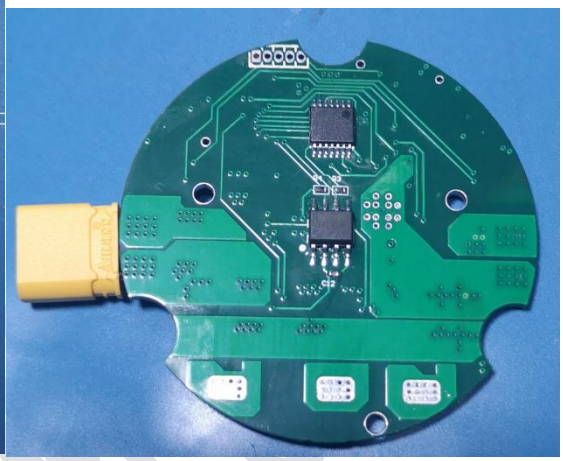
第一章 驱动使用须知

1.1 工作电压

工作电压范围为 24V-38V。

1.2 驱动板信息





- 工作电压 16V-38V
- 底部集成双编码器
- 电源指示灯
- 提供程序下载端供固件更新使用

1.3 电机与驱动板硬件接线

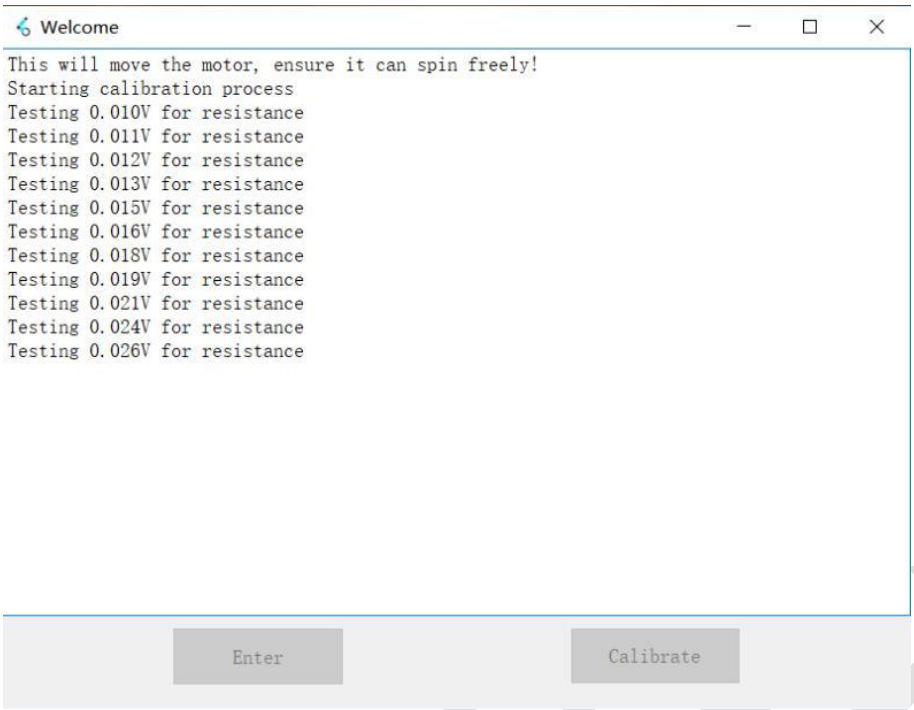
1、连接电机三根线
2、编码器与驱动板相连接，确保编码器对准电机磁铁。
3、CANFD 接口与 CAN 模块相连。
4、连接电源，电源上电后指示灯亮说明驱动板正常工作。

注：请保证 can 线线序与编码器线序正确无误。以防接反损坏芯片，当打开上位机通讯时，can 模块灯会闪烁。

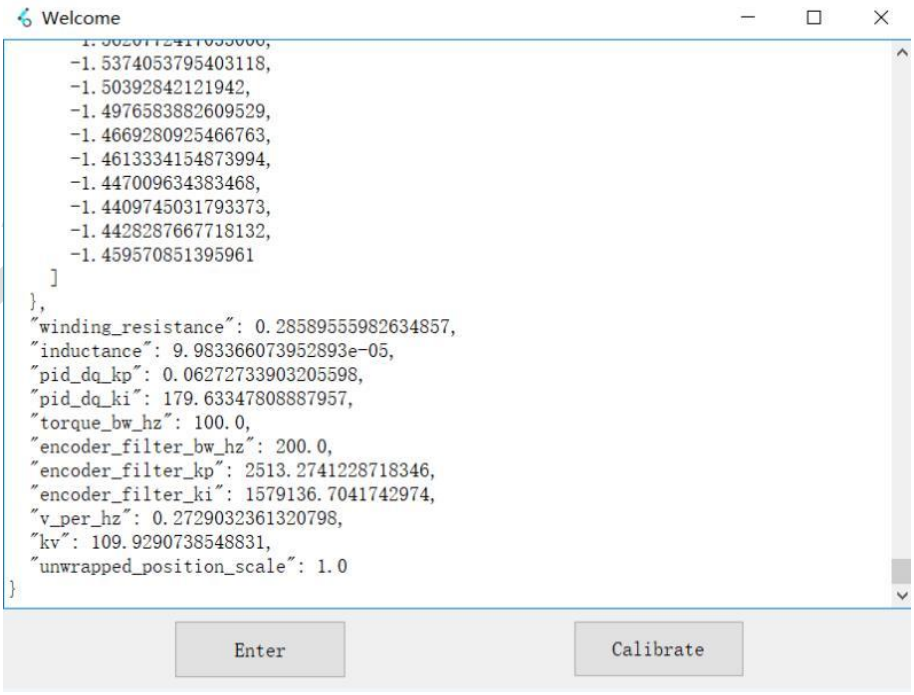
1.4 电机校准流程

1、硬件接线无误后，打开软件

2、点击 Calibrate



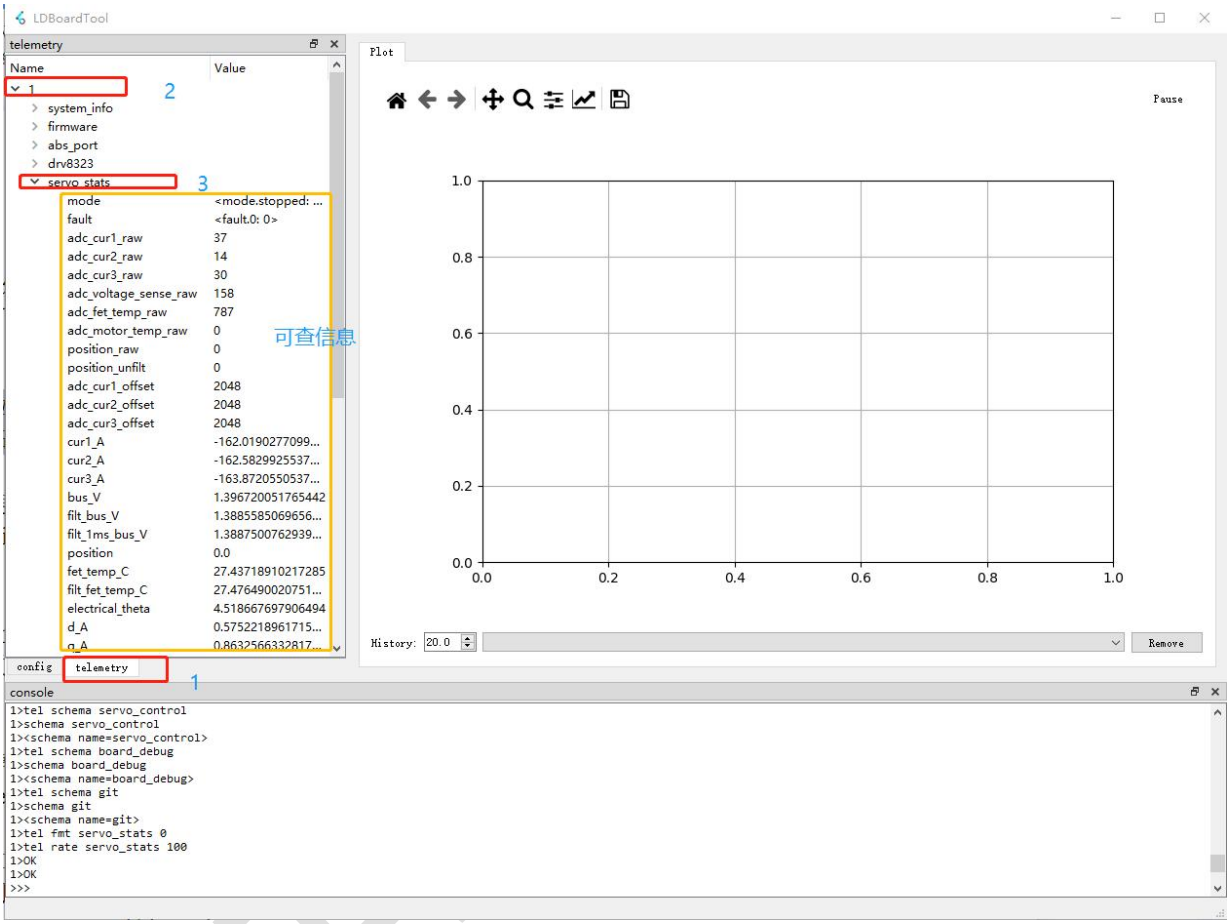
3、等待校准，校准成功界面



第二章 相关使用方法

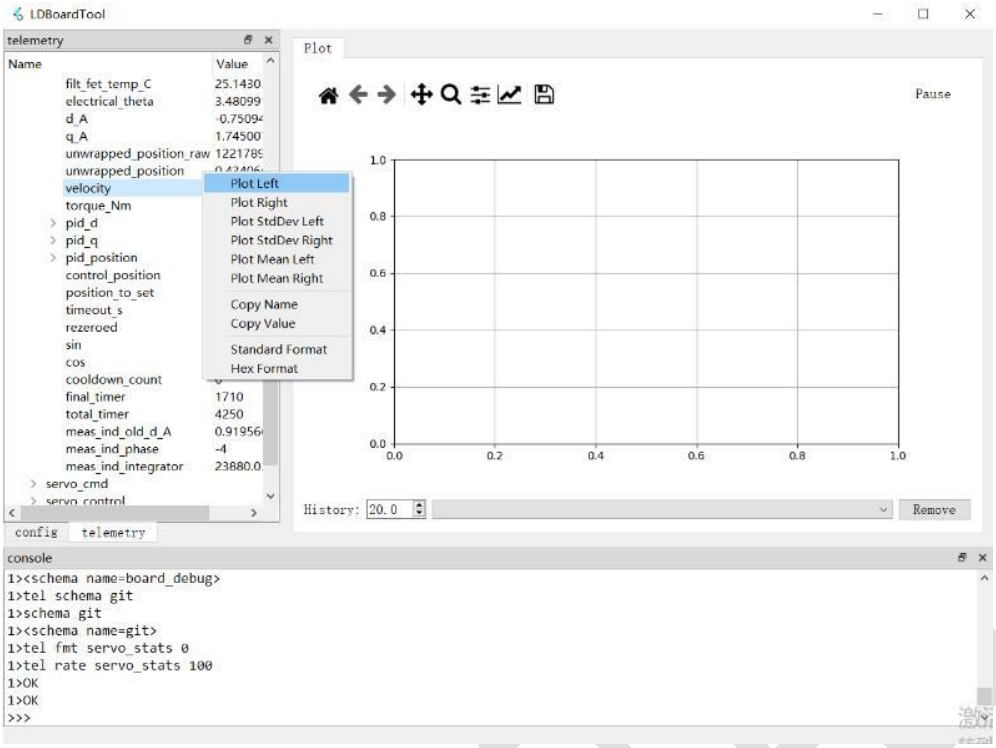
2.1 软件使用方法

信息查看：校准成功后，在 telemetry 里面，点开 servo_stats

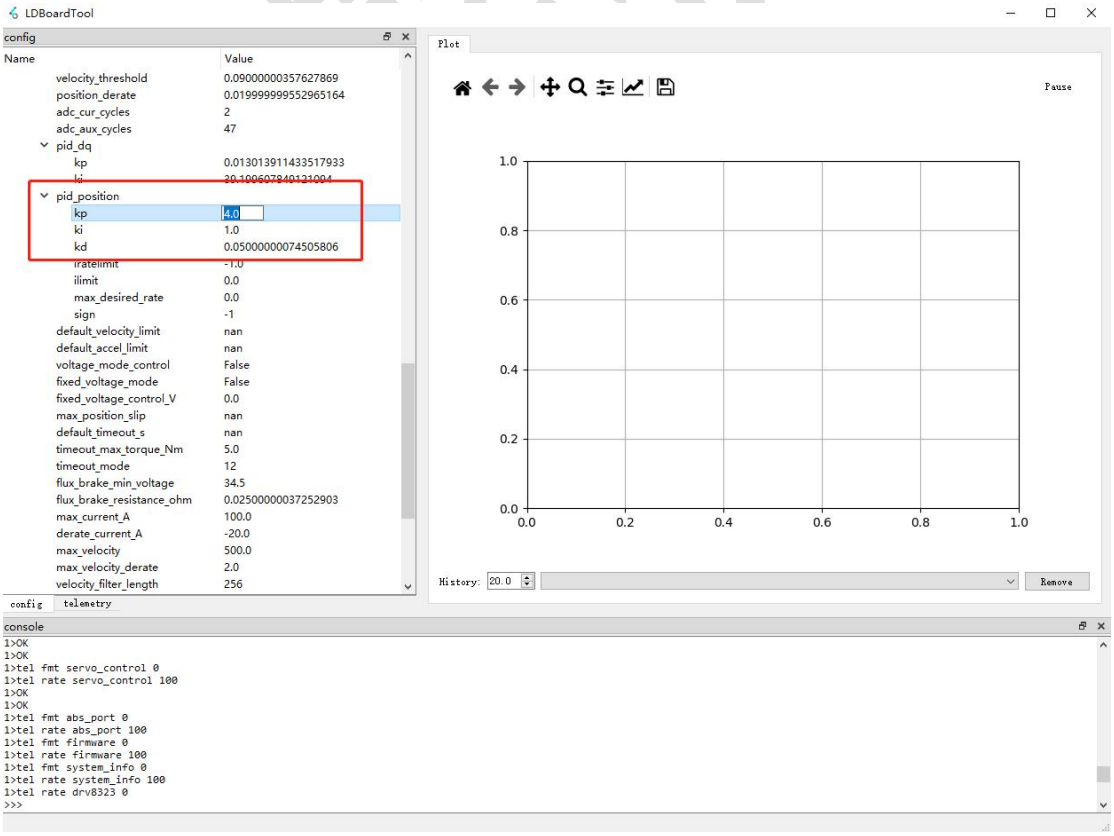


位置查看：在 telemetry 里面，点开 servo_stats，找 unwrapped_posistion,右键选择 plot right/left。

速度查看：在 telemetry 里面，点开 servo_stats，找 velocity,右键选择 plot right/left。



2.2 PID 调参（配置指令后需要保存才能生效 ：“conf write”）



2.2.1 PID 速度调参过程

1. 调出速度曲线。

2. d pos （d pos nan 5 0.5）指令，让电机先动起来,建议速度先设置小一点可以先给5，观察电机速度曲线与目标曲线的差距。

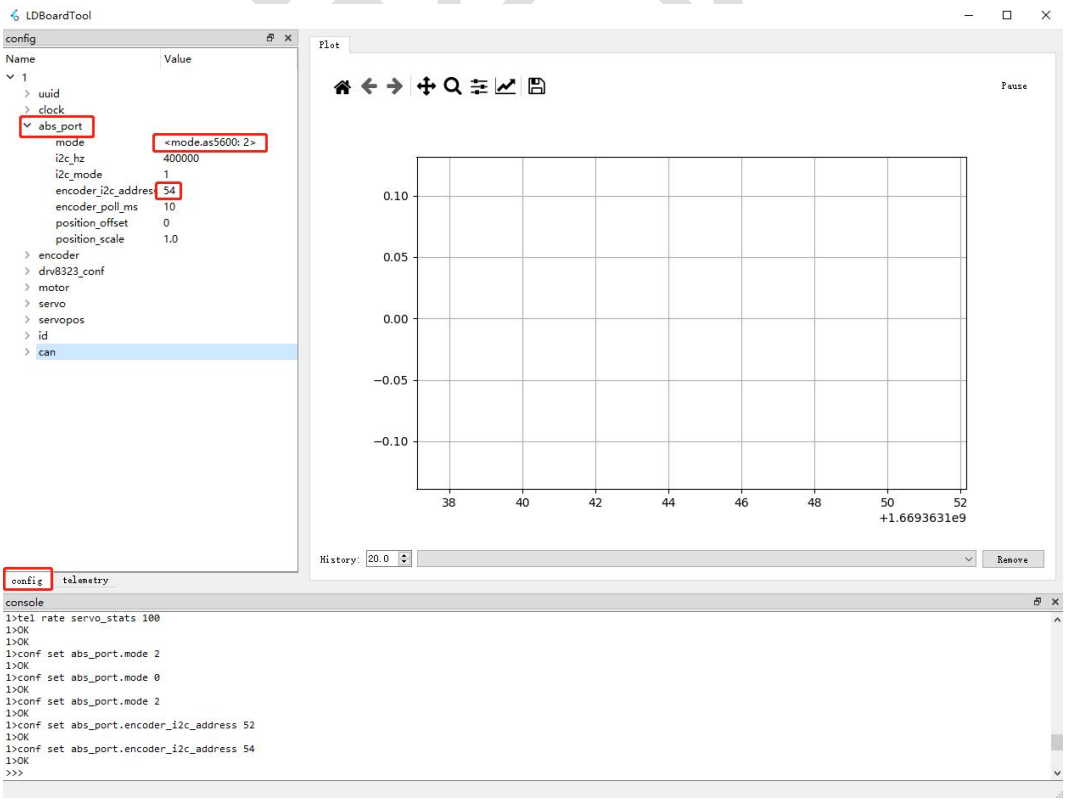
3. 开始调 PID，观察速度曲线的变化。

4. 输入指令 d stop 让电机停下来，然后重新输入指令 d pos 让电机旋转，观察电机速度曲线和目标曲线的差距，不断重复步骤 3，直至达到最佳。

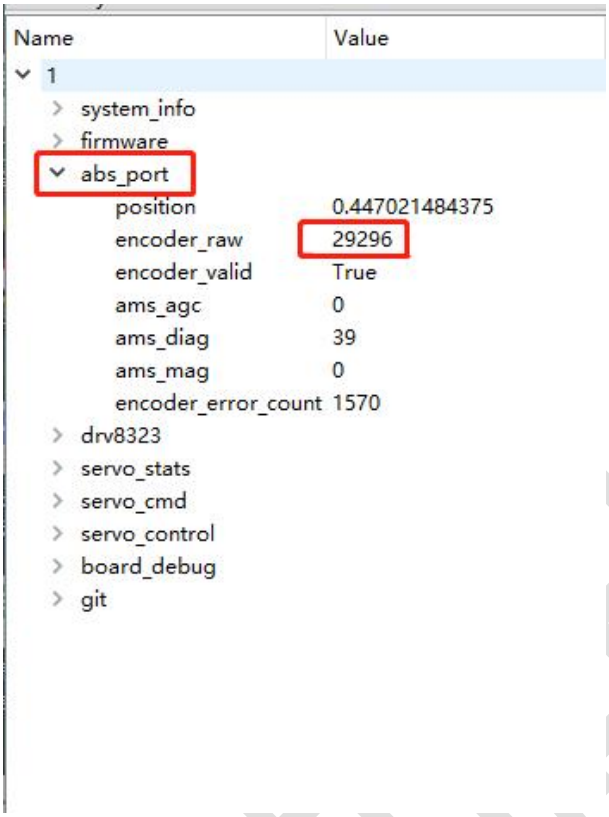
5. 确定最终参数后，需要输入保存指令“conf write”保存参数

2.3 双编码器设置

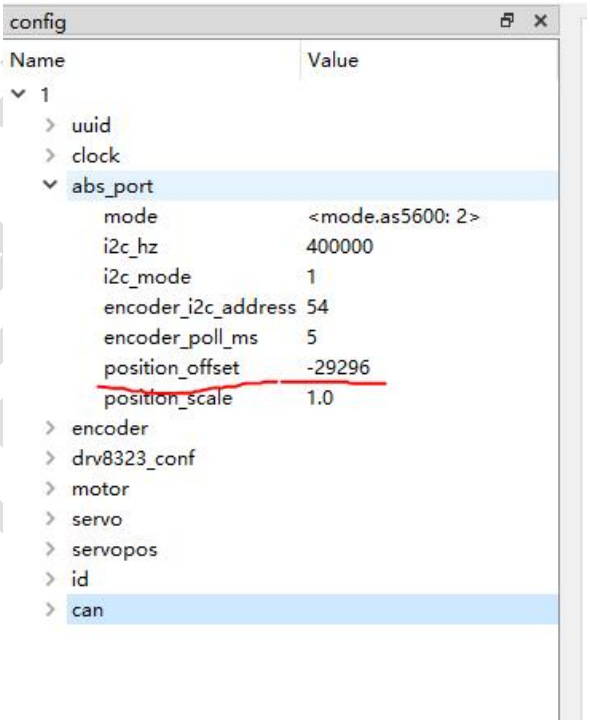
1.打开上位机 config 选项，在 abs_port 选项中打开 mode 选项,选择 mode.as5600:2;encoder_i2c_address 修改为 54。（AS5048B 的 i2c 地址为 64，AS5600 编码器的地址为 54，看编码器型号修改。）



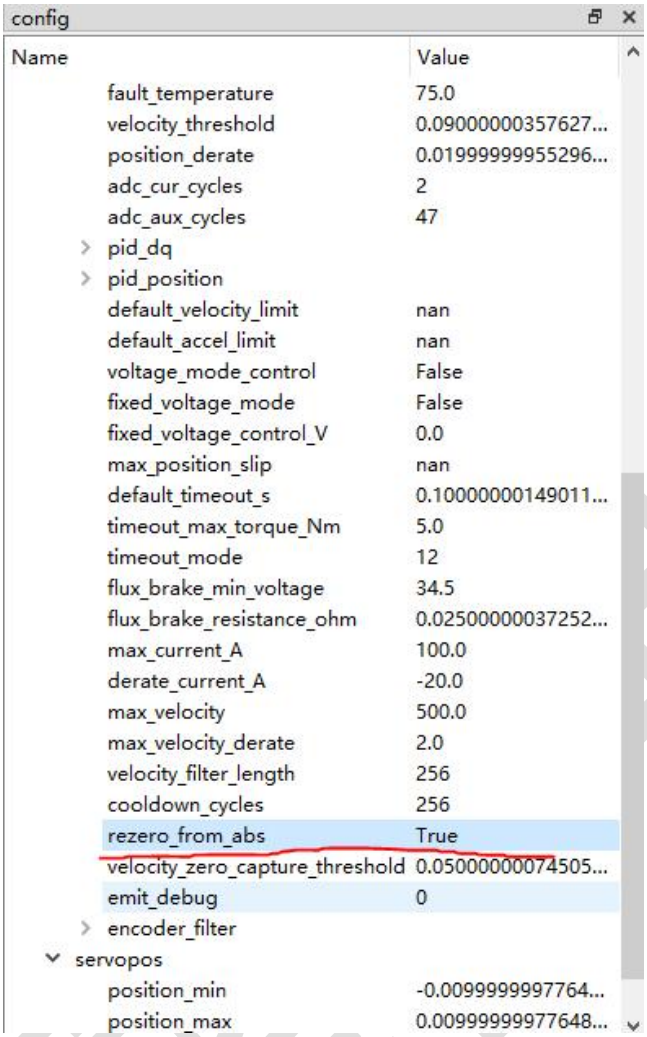
2.打开 telemetry 选项，找到第一个原始编码器数值，复制该数值



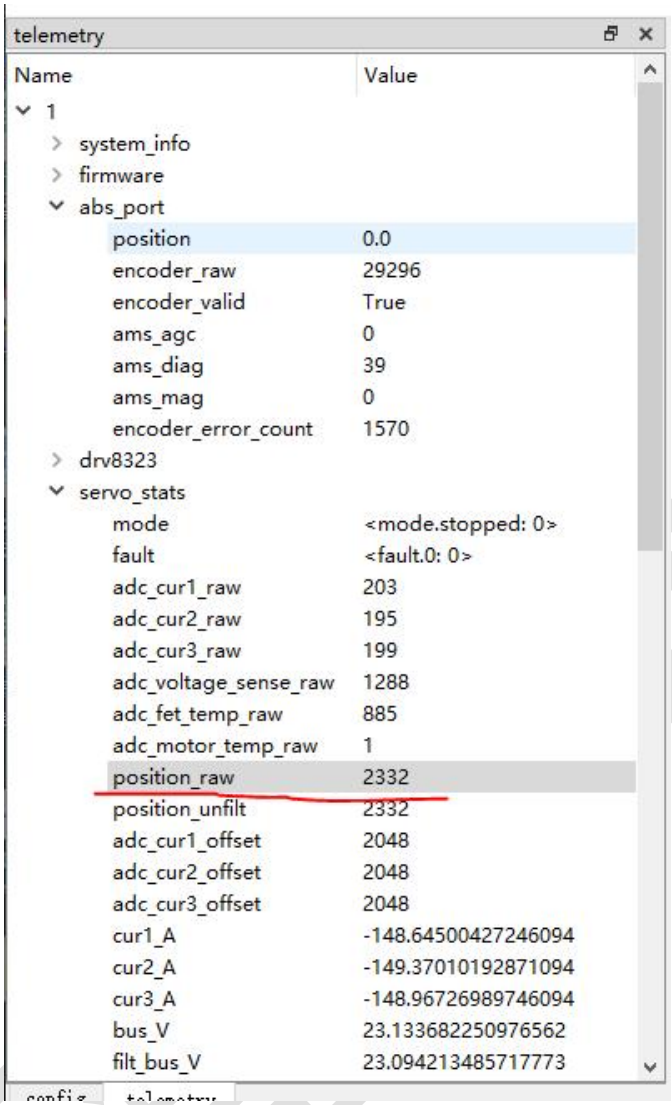
3.打开 config 设置，找到下面的选项，将第一个编码器的原始数据粘贴上去，需要在数值前加一个负号。



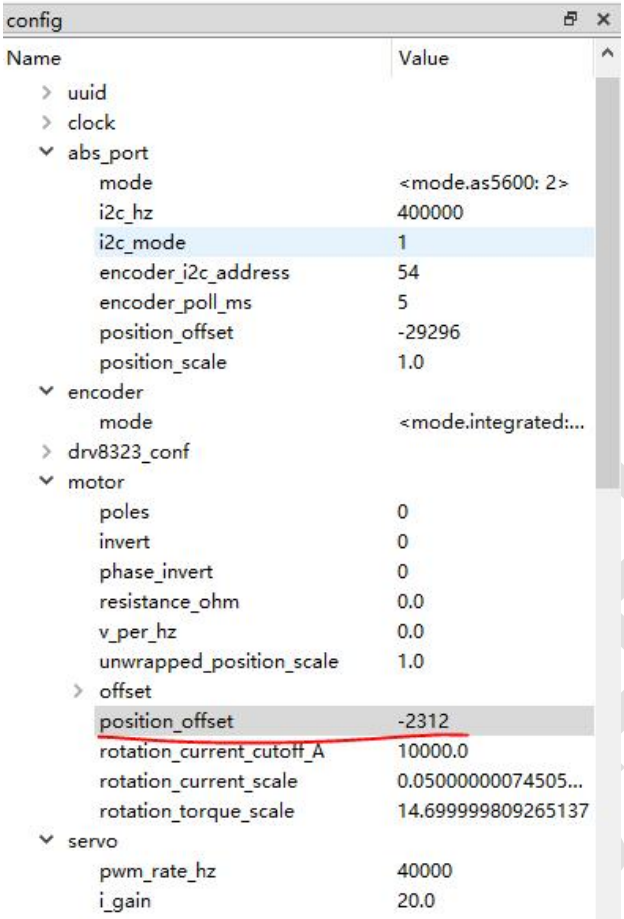
4.打开 config 设置选项，找到红线选项，修改为 True。



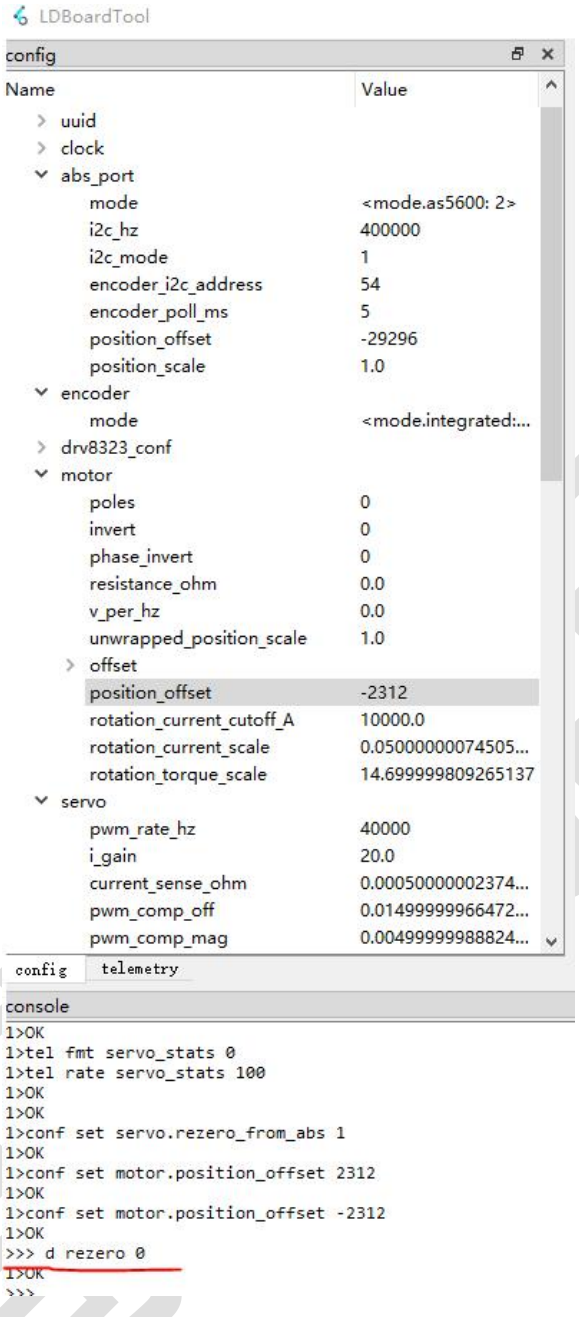
5.找到下图红线选项，复制编码器原生数值。



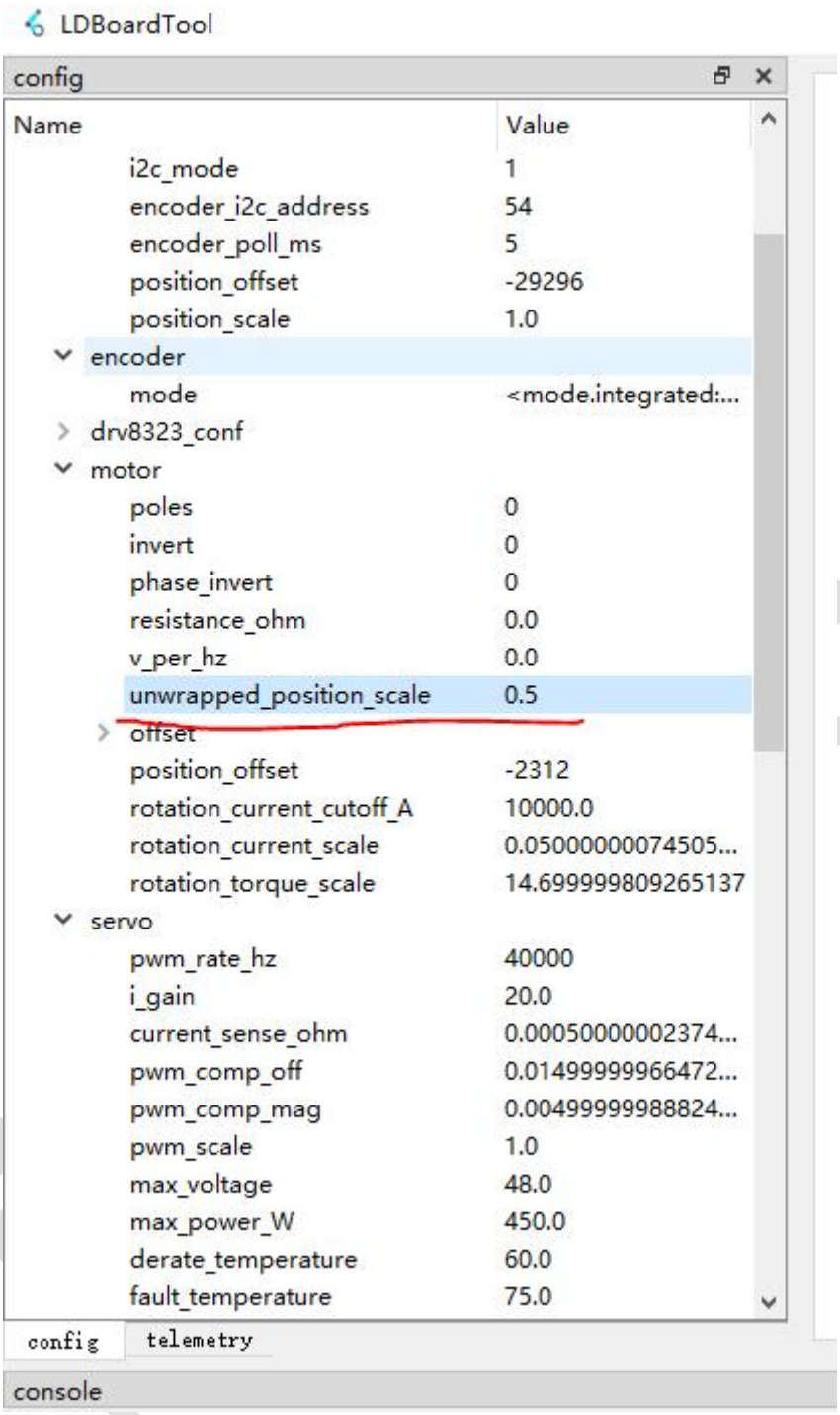
6.找到下图中红线的设置，粘贴步骤 5 中复制的数值，需要加负号。



7.输入编码器位置调 0 指令


LDBoardTool
config
Name
Value
uuid
clock
abs_port
mode
<mode.as5600: 2>
i2c_hz
400000
i2c_mode
1
encoder_i2c_address
54
encoder_poll_ms
5
position_offset
-29296
position_scale
1.0
encoder
mode
<mode.integrated:...
drv8323_conf
motor
poles
0
invert
0
phase_invert
0
resistance_ohm
0.0
v_per_hz
0.0
unwrapped_position_scale
1.0
offset
position_offset
-2312
rotation_current_cutoff_A
10000.0
rotation_current_scale
0.05000000074505...
rotation_torque_scale
14.699999809265137
servo
pwm_rate_hz
40000
i_gain
20.0
current_sense_ohm
0.00050000002374...
pwm_comp_off
0.01499999966472...
pwm_comp_mag
0.00499999988824...
config
telemetry
console
1>OK
1>tel fmt servo_stats 0
1>tel rate servo_stats 100
1>OK
1>OK
1>conf set servo.rezero_from_abs 1
1>OK
1>conf set motor.position_offset 2312
1>OK
1>conf set motor.position_offset -2312
1>OK
>>> d rezero 0
1>OK
>>>

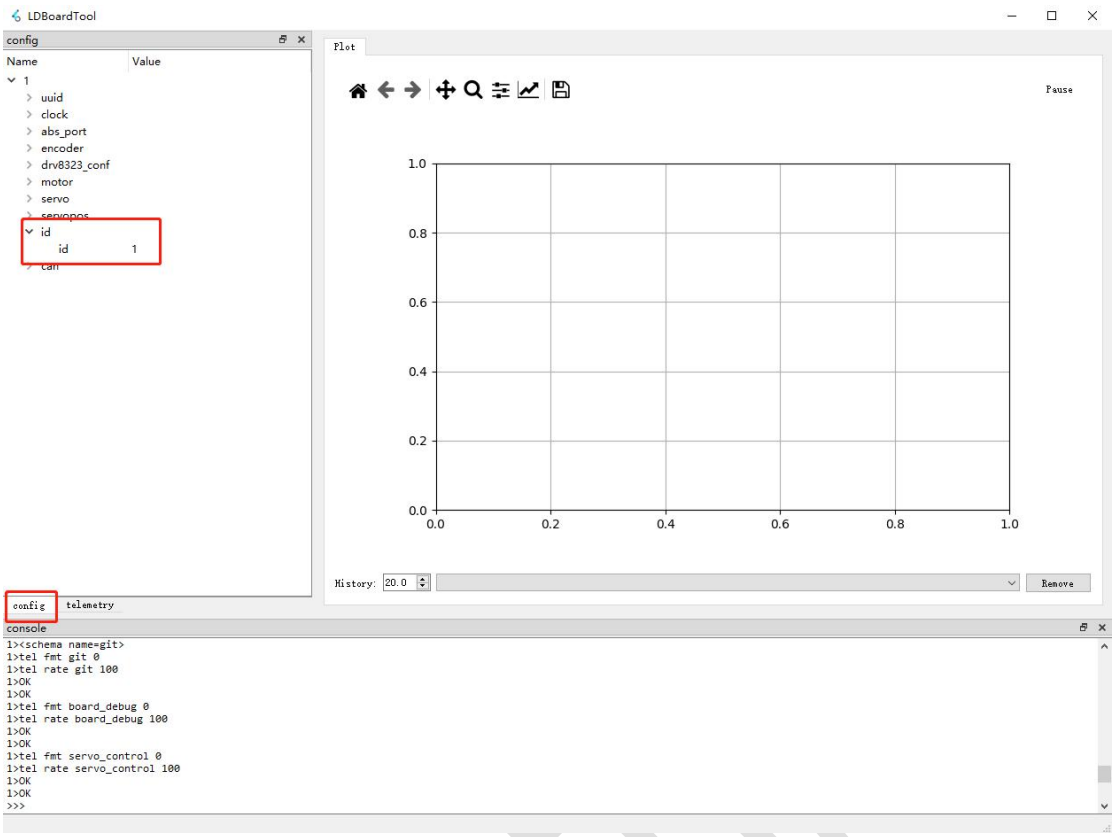
8.设置减速比

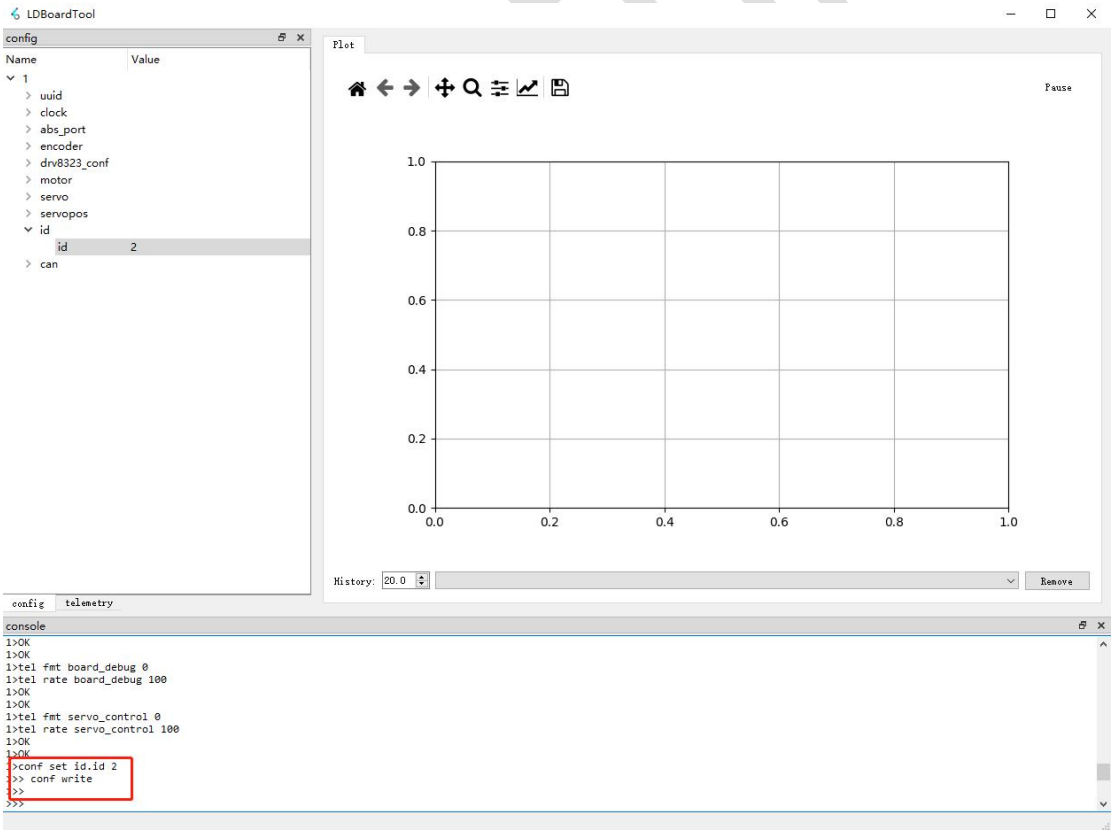


9.保存（conf write）,设置成功后能在 telemetry 选项的 abs_port 中看到编码器的数值变化。

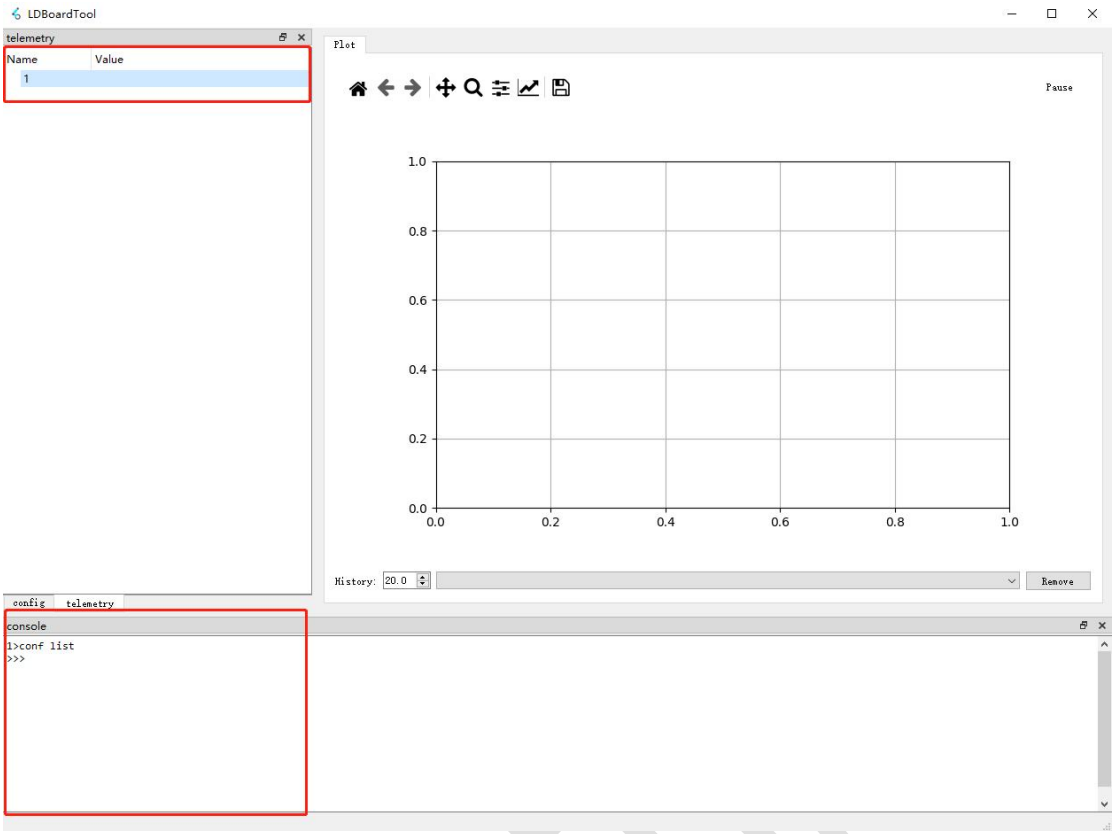
2.4 驱动板 ID 修改

1.打开上位机，打开 config 项，打开 id 选项，修改 id 号，



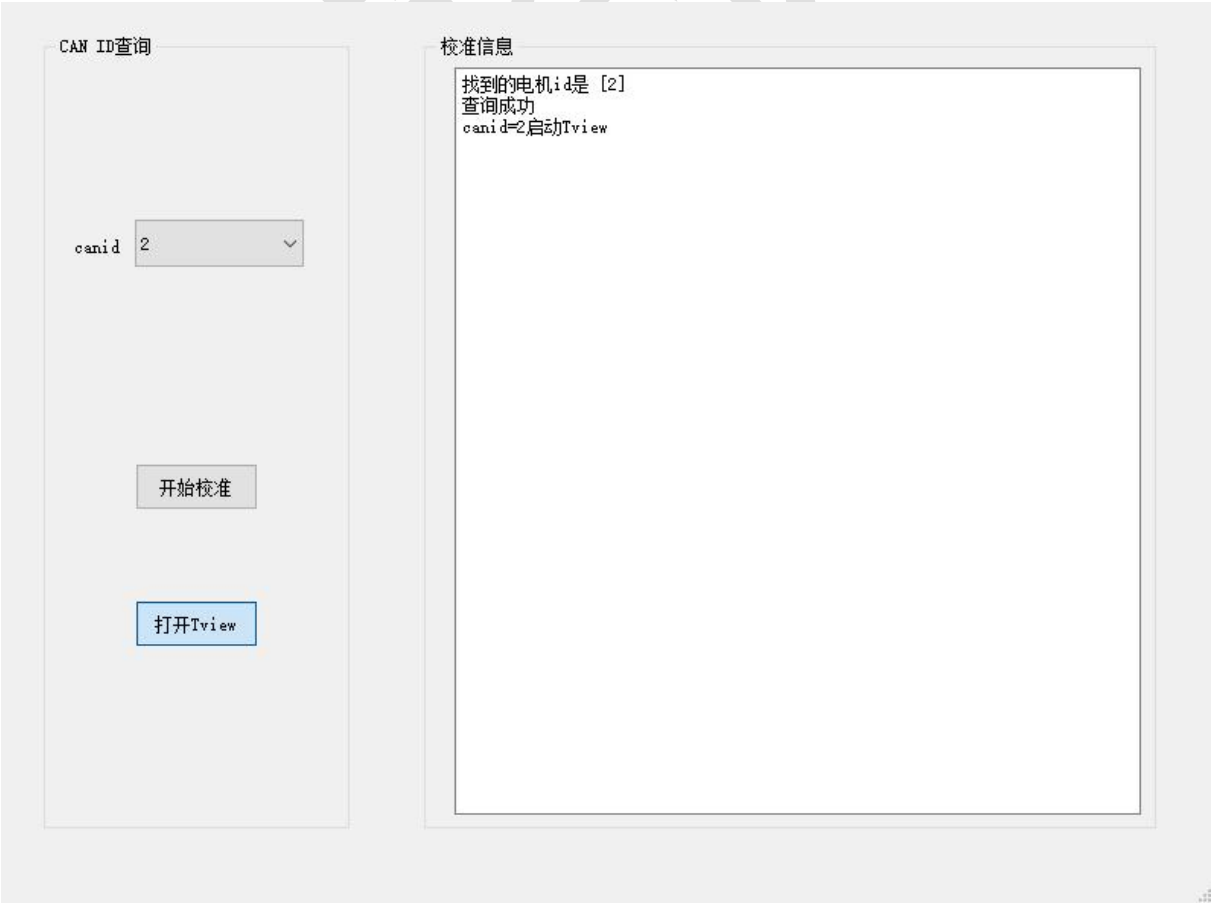


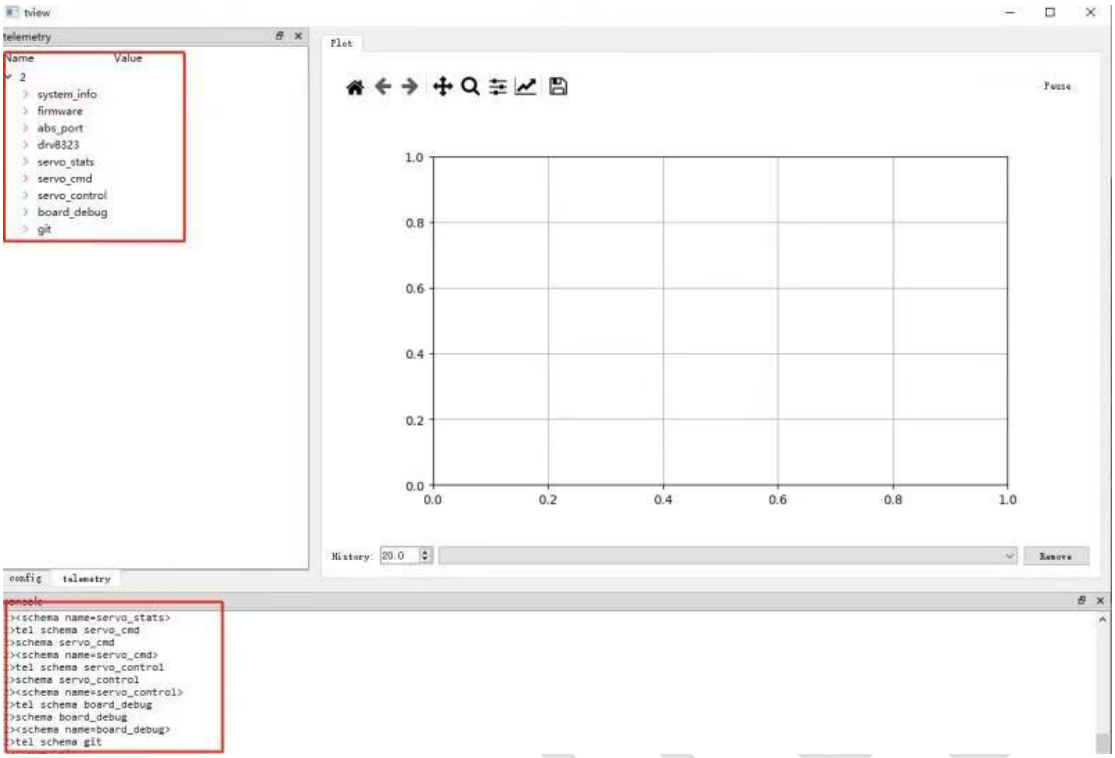
2. 修改后记得输入“conf write”命令保存参数



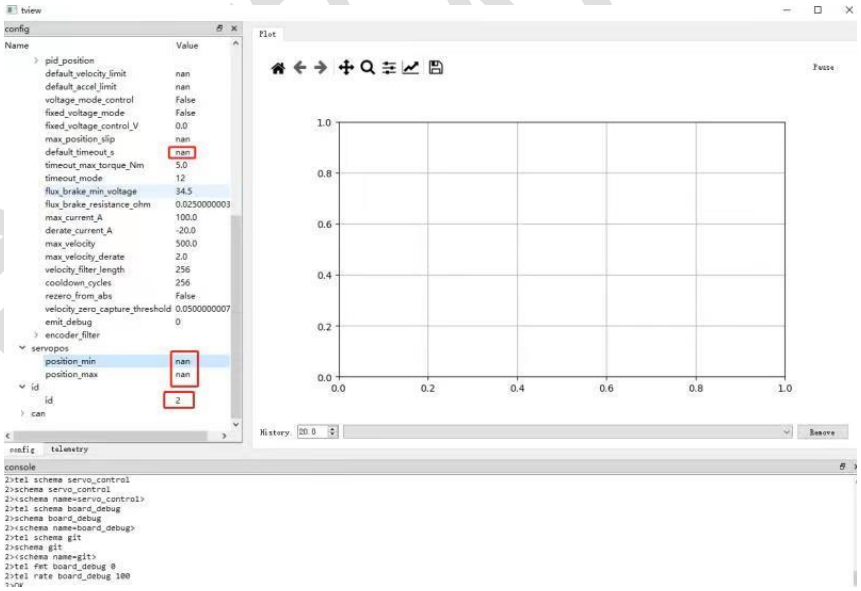
3. 保存后重新打开，可以发现上位机中没有驱动板信息，因为已经修改了驱动板 id 为 2，接下来我们再打开修改 id 后的驱动板

4、重新打开上位机，此时发现 ID 号已经改成 2 了，直接按‘打开 Tview’进入软件





如图所示，既为 id 修改成功，*id 修改后记得打开对应 ID 的软件重新再保存一次*



2.5 常用控制指令

2.5.1 FOC 位置控制指令：

d pos <pos> <vel> <max_torque> [options...]

每个可选元素由一个前缀字符和一个值组成。

p - kp 比例：设置电机转动的钢性参数

d - kd 比例：设置电机转动的阻尼参数

s - 停止位置：当给定非零速度时，当控制位置达到该值时运动停止。

f - 前馈扭矩，单位 Nm

t - timeout：如果在这么多秒内没有收到另一个命令，则进入超时模式。

v - 速度限制：给定值将在此命令期间覆盖全局速度限制。

a - 加速度限制：给定值将在此命令期间覆盖全局加速度限制。

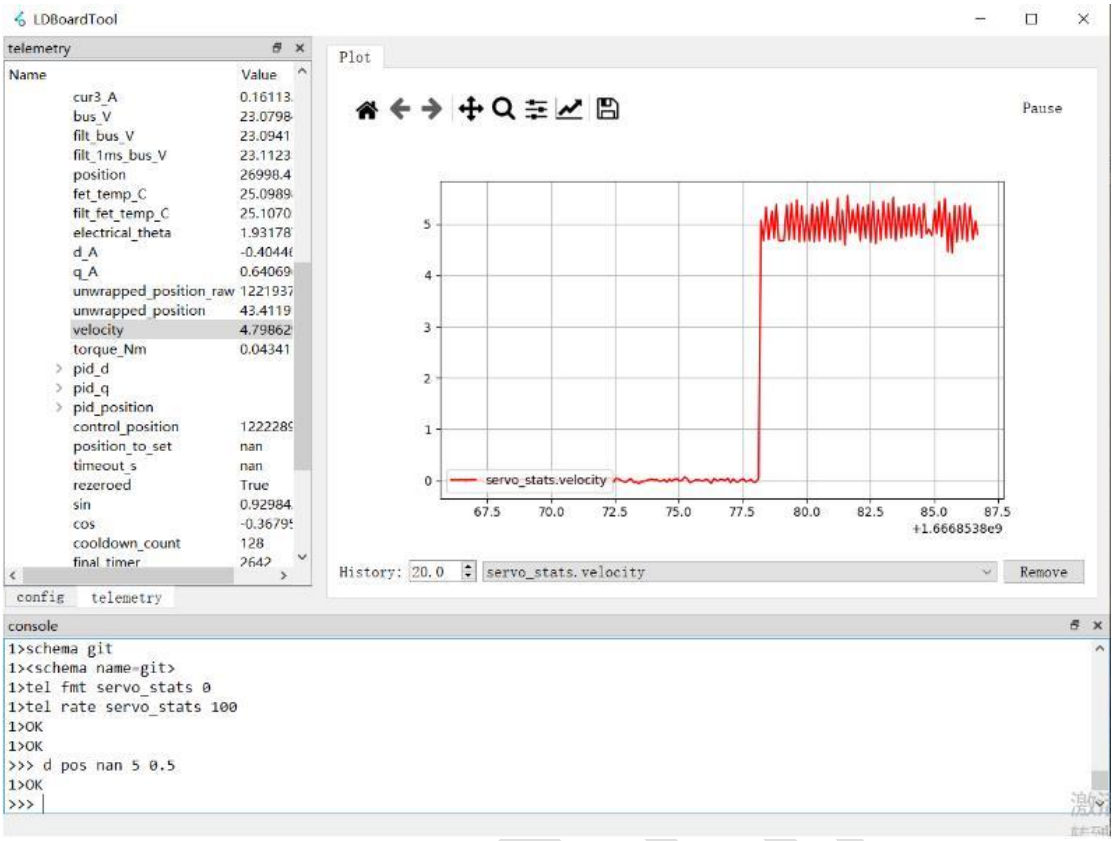
o - 固定电压覆盖：在生效时，将控制视为使用给定的电压固定电压模式启用

例:

d pos 10 1 0.1 解释：10 代表位置，1 代表速度，0.1 代表扭矩此命令是指：以 0.1n/m 的扭矩,以最大的加速度运动到 10 的位置。然后再以 0.1n/m 的扭矩,1 圈/s 的速度继续运转。

d pos nan 1 0.1 以 0.1n/m 的扭矩,1 圈/s 的速度运动

d pos nan 1 0.1 s10 以 0.1n/m 的扭矩,1 圈/s 的速度运动到 10 的位置，然后停止



2.5.2.停止指令

d stop

让控制器进入停止状态，从而控制电机停止

2.5.3 电压模式

d pwm

语法：d pwm <phase> <magnitude> [<phase_rate>]

Phase 是电向量方向（弧度），mangnitdue 是强度（电压伏特），可选 phase_rate 参数控制向量转速（弧度每秒）**注意：给电压伏特时数值从 0 开始给，幅度 0.1-0.2 左右增加，太大电压可导致电机发烫甚至冒烟损坏电机，谨慎使用！**

2.5.4 电流模式

d dp

语法：d dq <d_A> <q_A>

d_A 时磁铁方向电流，保持在 0A，q_A 时电机旋转力矩

建议 q_A 数值从 0 开始给，幅度 0.1 左右增加,谨慎使用！

2.5.5d zero

语法： d zero <pos> <vel> <max_torque> [options...]

进入零速度状态。 无论位置如何，都命令零速度。

2.5.6.d brake

进入“刹车”状态。 在这种模式下，所有电机相位都对地短路，导致被动“制动”动作。

2.5.7.d exact

语法：d exact <position>

将当前位置准确更新为给定值

2.5.8.conf enumerate

打印所有可配置参数的当前值

2.5.9.conf set

语法： conf set <item> <value>

获取单个可配置参数的值。

2.5.10.conf write

保存参数指令
将所有可配置参数的当前值从 RAM 写入内存

2.5.11.d within

位置限制

语法: d within <value> <value> <value>

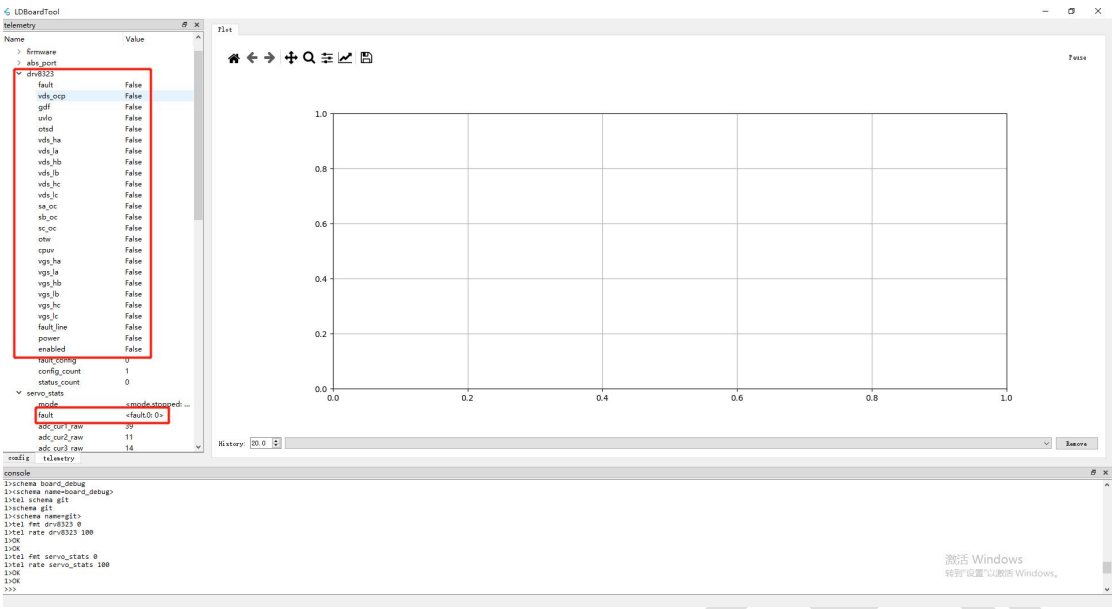
例：d within -0.4 0.5 0.3
d within nan 0.5 0.3
-0.4,0.5 为设定的位置上限和下限当超出设定的界限时，将会以 0.3 的扭力去限制扭动

2.5.12.d rezero 0

重新调整编码器 0 位，重设 0 点。

2.6 常见问题

当电机出现异常问题时，重新打开上位机，直接点击 ENTER，查看电机状态下的错误信息，与电机驱动上的错误信息，当出现问题时 drv8328 项会出现 true 项，如果出现 true 选项或者错误序号时，请参考下面错误警告提示



2.6.1 错误警告提示

如果主模式为 1（错误模式），这个寄存器将会指示错误信息。

32 校准故障-编码器在校准期间无法检测到磁体
*33*电机驱动器故障-最常见的原因是欠压，电机试图汲取比电源所能提供的电流更多的电流。其他电气故障也可能报告此错误，drv8323 诊断树有更多信息。
34 过电压-总线电压超过伺服 max 电压。这可能是由于配置错误，或者如果控制器使用无法吸收电力且未配置通量制动的电源再生电力。
35 编码器故障-编码器读数与磁铁不一致。
36 电机未配置-moteus_tool-校准程序尚未在此电机上运行。
37 脉宽调制周期超限-内部固件错误
38 温度过高-已超过最大配置温度
39 超出限制-试图在超出伺服位置配置的边界时开始位置控制。位置 min 和伺服位置 max。
40 电压过低-电压过低
41 -config changed-在需要停止的操作期间更改了配置值
42 0 无效-没有有效的换向编码器可用
43 位置无效-没有有效的输出编码器

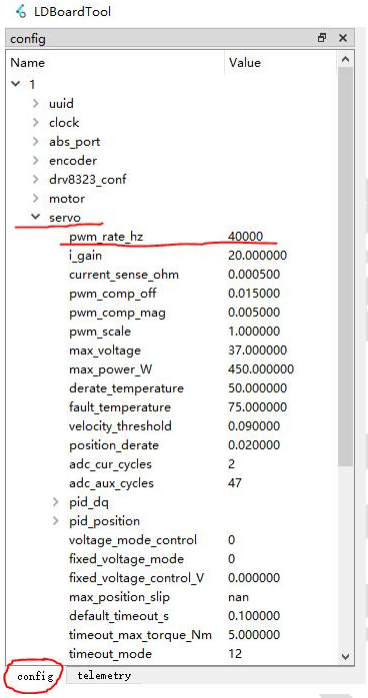
- VM 欠压闭锁 (UVLO)
- 电荷泵欠压 (CPUV)
- MOSFET 过流保护 (OCP)
- 栅极驱动器故障 (GDF)
- 热警告和热关断 (OTW/OTSD)
- 故障状态指示器 (nFAULT)

如果控制器出现 fault：39 的错误，这代表着是超出了配置的位置范围限制
可以在 config 里面修改以下两个参数：（修改为 nan，去掉最大范围）

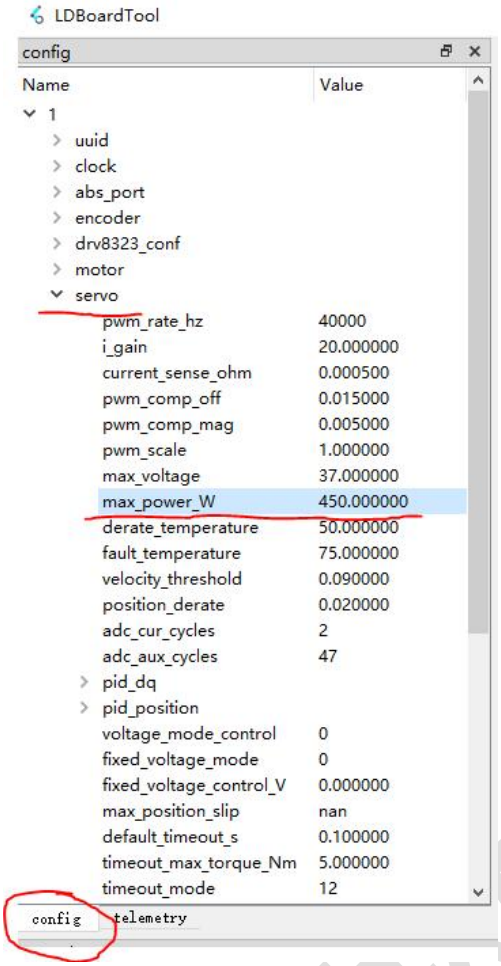
```
servopos.position_min        nan
servopos.position_max        nan
```

2.6.2 驱动板因温度限制扭矩

温度查看:在 telemetry 里面,点开 servo_stats,找到 filt_fet_temp_C,右键选择 plot right/left。
物理办法：加散热片/散热装置/冷空气
软件方法：
1、修改 PWM 的控制频率。配置值设置为 15000 到 60000 之间。默认为 40000
修改内容在 config 里面的 servo.pwm_rate_hz



如果对 servo.pwm_rate_hz 的参数进行修改，那么对应的也要修改其最大功率限制



2.6.3 为了可以更好地提高耐温，你可以设置以下参数

servo.derate_temperature　　当温度达到此值时，扭矩开始受到限制

servo.fault_temperature　　温度达到此值时，触发故障，停止所有扭矩

想要保证温度系数不受影响，首先要在良好的散热环境下运行（加散热片、散热铝壳、低温）等环境。
其次，可以降低其 PWM 的控制频率和最大限制功率（降频）。来控制温度的涨幅。
最后可以修改其受温度影响的扭矩参数来提高其性能（*必须在散热铝壳下进行）
servo.derate_temperature　　可以调至 80
servo.fault_temperature　　可以调至 90

第三章 Lively board 通讯协议说明

3.1.CANFD 指令格式

使用 CANFD 帧在 1Mbit 标准波特率和 5Mbit 数据波特率与 lively board 通信。
CAN ID:
ID 由 16 位数构成，高 8 位表示信号源，低 8 位表示数据的目的地。目的地址的最高位是 0，其余 7 位表示伺服电机的地址。源地址的低 7 位表示不同的地址，高 1 位若是被设置为 0，则表示此指令需要 lively board 回复。
如果帧 ID 大于 0x7FF，则必须是扩展帧。
ID：0x8001

* 发送地址是 0
* 目的地址是 1
* 最高位是 1，表示需要回复。

ID：0x100

* 发送地址是 1
* 目的地址是 0
* 最高位是 0，表示不需要回复。

config 的配置指令说明：
多路复用协议用于半双工或全双工串行连接。
它提供了一个基于逻辑包的接口，假设它是一个带有单个客户机和一个或多个服务器的客户机/服务器总线。它实现了许多服务。

常见的定义

endian-ness -- 所有基本类型都以最低位字节开头
Varuint -- 一个或多个 uint8_t 值的序列，以最不重要的一阶。对于每个值，这 7 条 lbs 中都包含了数据，如果设置了 MSB，则表示剩余的字节数更多。它最多代表一个 uint32_t，因此 5 字节是最大的有效长度。
Float -- 一阶最低有效字节的 IEEE 754 32 位浮点数
ID -- 每个节点由一个 7 位标识符标识，0x7f 是“广播”地址

Frame format
Header
- uint16_t => 0xab54
- uint8_t => source id
> if the high bit is set, that means a response is requested
- uint8_t => destination id

- varuint => size of payload
- bytes => Payload
- uint16_t => crc16 of entire frame including header assuming checksum field is 0x0000
Payload
subframe 1
subframe 2
Subframe
varuint => subframe type
[bytes] => possible subframe specific data

#Service: Register based RPC#
该服务型号的设备由多达 2**32 个“寄存器”组成。每个寄存器表示某种类型的值，该值可以映射为一种或多种不同的表示格式。客户端可以设置或查询任何寄存器的值。对于给定的寄存器，哪种表示格式有效取决于设备。可能的表示格式集合是:(int8_t, int16_t, int32_t, float)

Subframes
0x00, 0x04, 0x08, 0x0c - write (int8_t|int16_t|int32_t|float)
- varuint => number of registers (may be optionally encoded as a non-zero 2 LSBs)
- varuint => start register
- N x (int8_t|int16_t|int32_t|float) => values
0x10, 0x14, 0x18, 0x1c - read (int8_t|int16_t|int32_t|float)
- varuint => number of registers (may be optionally encoded as a non-zero 2 LSBs)
- varuint => start register #
0x20, 0x24, 0x28, 0x2c - reply (int8_t|int16_t|int32_t|float)
- varuint => number of registers (may be optionally encoded as a non-zero 2 LSBs)
- varuint => start register #
- N x (int8_t|int16_t|int32_t|float) => values
0x30 - write error
- varuint => register #
- varuint => error #
0x31 - read error
- varuint => register #
- varuint => error #
0x50 - nop
任何包含“读”命令的帧都有一个响应帧，其中每个请求的寄存器都被精确地命名一次。不要求响应使用完全相同的单一/多重公式，只要每个被提及一次。

Service: Tunneled Stream
通道流服务模型一个简单的字节流，其中客户端必须轮询服务器以获取数据。

Subframes
0x40 - client data on channel
- varuint => channel

- varuint => number of bytes sent from client
- N x uint8_t bytes
0x41 - server data on channel
- varuint => channel
- varuint => number of bytes sent from server
- N x uint8_t bytes
0x42 - client poll server
- varuint => channel
- varuint => maximum number of bytes to reply
当接收到 0x40 子帧时，无论当前是否有数据，从机都应该以 0x41 子帧响应。
作为对接收到 0x42 子帧的响应，无论是否有数据，从端都应该响应 0x41 子帧，其最大长度由客户端指定。
包含通道流子帧的帧可以恰好包含一个子帧总数。

```
#Register RPC #
    kWriteBase = 0x00,
    kWriteInt8 = 0x00,
    kWriteInt16 = 0x04,
    kWriteInt32 = 0x08,
    kWriteFloat = 0x0c,

    kReadBase = 0x10,
    kReadInt8 = 0x10,
    kReadInt16 = 0x14,
    kReadInt32 = 0x18,
    kReadFloat = 0x1c,

    kReplyBase = 0x20,
    kReplyInt8 = 0x20,
    kReplyInt16 = 0x24,
    kReplyInt32 = 0x28,
    kReplyFloat = 0x2c,

    kWriteError = 0x30,
    kReadError = 0x31,
# Tunneled Stream #
    kClientToServer = 0x40,
    kServerToClient = 0x41,
    kClientPollServer = 0x42,

    kNop = 0x50,
```

3.2 寄存器命令：

常见定义：
Endian:所有原始类型都以最低有效字节在前
Var:一个或多个 uint8 值的序列，以最低有效字节第一顺序排列。
对于每个值，7 个 LSB 包含数据，如果设置了 MSB，则意味着剩余的字节数更多。
它最多可以表示一个 uint32，因此 5 个字节是最大有效长度。
Float:一个 IEEE 754 32 位浮点数。

子帧
每一个 CANFD 帧都包含了一个或多个子帧。一个简短的关于可使用的子帧类型
CANFD 帧的任何需要尾部填充的字节都应该设置为 0x50。
0x00, 0x04, 0x08, 0x0c 表示写（int8|int16|int32|float)
Var:寄存器数量（可以选择编码为子帧类型的非零 2 LSBS）
Var:开始寄存器的 number
0x10, 0x14, 0x18, 0x1c 表示读（int8|int16|int32|float）
Var:寄存器数量（可以选择编码为非零 2 LSB）
Var:开始寄存器的 number
0x20, 0x24, 0x28, 0x2c 表示回复（int8|int16|int32|float）
Var:寄存器数量（可以选择编码为非零 2 LSB）
Var:开始寄存器的 number
0x30, 0x31 分别表示读/写错误
Var ： 寄存器 number
Var ： 错误 number
0x50 无操作

寄存器用法
每一个寄存器都作为潜在的多数据类型被访问，这一段描述了常用的对照表和每个寄存器的含义。

速度（单位，N·M）
* int8：最低有效位表示 0.1Hz，表示 36°每秒。
* int16：最低有效位表示 0.00025Hz，表示 0.09°每秒
* int32: 最低有效位表示 0.000001Hz。

3.3 寄存器的使用

每个寄存器都可以作为潜在的多种数据类型进行访问。以下就是常见的寄存器映射以及寄存器的意思

电流（以安培为单位）

· int8 => 1 LSB => 1A
· int16 => 1 LSB => 0.1A
· int32 => 1 LSB => 0.001A

扭矩（以 **N*m** 为单位）

· int8 => 1 LSB => 0.5 N*m
· int16 => 1 LSB => 0.01 N*m
· int32 => 1 LSB => 0.001 N*m

电压（以伏特为单位）

· int8 => 1 LSB => 0.5V
· int16 => 1 LSB => 0.1V
· int32 => 1 LSB => 0.001 V

温度（以摄氏度为单位）

· int8 => 1 LSB => 1 C
· int16 => 1 LSB => 0.1C
· int32 => 1 LSB => 0.001 C

时间（以秒为单位）

· int8 => 1 LSB => 0.01s
· int16 => 1 LSB => 0.001s
· int32 => 1 LSB => 0.000001s

位置（以转数测量）

· int8 => 1 LSB => 0.01 旋转 => 3.6 度（范围为 -1.27 到 1.27）
· int16 => 1 LSB => 0.0001 旋转 => 0.036 度（范围为 -3.2767 到 3.2767）
· int32 => 1 LSB => 0.00001 旋转 => 0.0036 度

速度（以转/秒测量）

· int8 => 1 LSB => 0.1Hz / 36 dps
· int16 => 1 LSB => 0.00025 Hz > 0.09 dps
· int32 => 1 LSB => 0.00001 Hz => 0.0036 dps

加速度（以转数/s^2 测量）

· int8 => 1 LSB => 0.05 l/s^2
· int16 => 1 LSB => 0.001 l/s^2
· int32 => 1 LSB => 0.00001 l/s^2

3.4. 寄存器

0x000 - 模式设置
可读写
当前可用的操作模式（并非所有值都可以写入）：
* 0：停止，r/w，清除错误
* 1：错误
* 2,3,4：准备运行
* 5：PWM 模式
* 6：电压模式
* 7：foc 电压模式
* 8：dq 电压模式
* 9：电流模式
* 10：位置模式
* 11：超时模式
* 12：零速模式
* 13：stay within 模式
* 14：测量电感模式
* 15：刹车模式

0x0001 位置
只读
当前伺服电机的位置，单位是输出轴的圈数。

0x002 - 速度
只读
当前伺服电机的速度，单位是输出轴的旋转频率。

0x003 - 转矩
只读
当前伺服电机的转矩。

0x004 - Q 相电流
只读
Q 相电流，单位是 A。

0x005 - D 相电流
只读
D 相电流，单位是 A。

0x006 - 绝对位置
只读
如果在 ABS 端口接了绝对值编码器，此处会报告绝对位置。

0x00d - 电压
只读
当前输入电压。

0x00e - 温度
只读
当前温度，单位是摄氏度。

0x00f - 错误代码
只读

0x010/0x011/0x012 - PWM 相位 A/B/C
读写
在 PWM 模式下，他控制 A,B 和 C 相的原始 PWM 值。

0x014/0x015/0x016 - 电压相位 A/B/C
读写
在电压模式下，它控制施加到 A、B 和 C 相的电压。
0x018 - 电压 FOC Theta
模式：读/写

当处于电压聚焦模式时，它控制所需的电相位。
0x019 - 电压 FOC 电压
模式：读/写

在电压聚焦模式下，它控制所需的施加相电压。

0x01a - D 电压
模式：读/写

在电压 Dq 模式下，它控制所需的施加 D 电压。
0x01b - Q 电压
模式：读/写

在 kVoltageDq 模式下，它控制所需的施加 Q 电压。如果未指定，则使用 0.0。

0x01c - 指令 Q 相电流
模式：读/写

在电流模式下，它控制所需的 Q 相电流。如果未指定，则使用 0.0。

0x01d - 指令 D 相电流
模式：读/写

在电流模式下，它控制所需的 D 相电流。

0x020 - 位置指令
模式：读/写

在位置模式下，它控制所需的位置。

0x021 - 速度命令
模式：读/写

在位置模式下，以给定的速度前进所需的位置。

0x022 - 前馈扭矩
模式：读/写

在位置模式下，在应用所有常规控制回路后添加给定的前馈转矩。

0x023 - Kp 比例
模式：读/写

在位置模式下，将比例控制项缩小给定因子。

0x024 - Kd 刻度
模式：读/写

在位置模式下，将微分控制项缩小给定因子。

0x025 - 最大扭矩
在位置模式下，要施加的最大扭矩。
0x026 - 指令停止位置
在位置模式下，并且命令非零速度时，到达给定位置时停止运动。NaN / 最大负数表示没有应用限制。如果未指定，则使用 NaN。

请注意，如果控制器曾经被命令离开停止位置，例如使用与开始和停止位置不一致的速

度命令，那么它将像命令 0 速度并且当前命令位置等于停止位置。

0x027 - 看门狗超时
模式：读/写

这决定了该命令有效的时间长度。
0x028 - 速度限制
模式：读/写

这可用于覆盖内部生成轨迹的全局速度限制。
0x029 - 加速度限制
模式：读/写

这可用于覆盖内部生成轨迹的全局加速度限制。
0x030 - 比例扭矩
模式：只读

PID 控制器中比例项的转矩。

0x031 - 积分扭矩
模式：只读

PID 控制器中积分项的转矩。

0x032 - 微分扭矩
模式：只读

PID 控制器中微分项的转矩。

0x033 - 前馈扭矩
模式：只读

PID 控制器中的前馈。

0x034 - 总控制扭矩
模式：只读

位置模式控制器的总指令扭矩。

0x040 - 保持在下限内
模式：读/写

当处于留在模式中时，它控制最小允许位置。
0x041 - 保持在上限内

模式：读/写

当处于留在模式中时，它控制最大允许位置。

0x042 - 前馈扭矩
0x022 寄存器的映射。

0x043 - Kp 比例
0x023 寄存器的映射。

0x044 - Kd 刻度
0x024 寄存器的映射。

0x045 - 最大扭矩
0x025 寄存器的映射。

0x046 - 看门狗超时
0x027 寄存器的映射。

这将返回一个 32 位型号。
0x110 - 通讯 ID
名称：Multiplex ID 模式：可配置
这控制用于通过多路 CAN 总线访问设备的主 ID。它只能在 1 到 127 之间。

0x130 - 重新归零
模式：只写

发送后，这会导致伺服器选择内部电机旋转的整数倍，以使最终位置尽可能接近给定位置。

数据发送示例

单个 CANFD 帧可用于命令伺服，并启动对某些寄存器的查询。示例框架可能如下所示，以十六进制编码并带有注释。

01- 写入单个 int8 寄存器（寄存器数量在 2 个 LSB 中编码）
00- 起始寄存器号“模式”
0a- “位置”模式
07- 写入 3 个 int16 寄存器（寄存器数量在 2 个 LSB 中编码）
20- 寄存器 0x 020
6000- 位置 = 0x0060 = 96 = 3.456 度
2001- 速度 = 0x0120 = 288 = 25.92 dps

50ff- 前馈扭矩 = 0xff50 = -176 = 1.76 N*m
14- 读取 int16 寄存器
04- 读取 4 个寄存器
00- 从 0x000 开始（所以 0x000 模式，0x001 位置，0x002 速度，0x003 扭矩）
13- 读取 3x int8 寄存器
0d- 从 0x00d 开始（所以 0x00d 电压，0x00e 温度，0x00f 故障代码）
因此，整个 CAN-FD 消息将是（十六进制）： （单片机发送）

01000a07206000200150ff140400130d

要使用 fdcanusb 转换器将其发送到配置为默认地址 1 的设备，您可以编写。

can send 8001 01000a07206000200150ff140400130d

这 80 在 ID 中用于两个目的。设置的高位强制设备响应（否则即使发送查询命令，它也不会响应）。其余位是要响应的“ID”。作为对该命令的响应，来自伺服的可能响应如下所示：

rcv 100 2404000a005000000170ff230d181400

解码，这意味着：

100 从设备“1”到设备“0”

24 回复 int16 值

044 个寄存器

00 从寄存器 0 开始

0a00 在模式 10 - 位置

5000 位置是 0x0050 = 80 = 2.88 度

0001 速度为 0x0100 = 256 = 23.04 dps

70ff 扭矩为 0xff70 = -144 = -1.44 Nm

23 回复 3 个 int8 值

0d 从寄存器 0x 00d 开始

18 电压为 12V

14 温度为 20℃

00 没有错

配置的值

CAN.ID
CAN 总线上显示的伺服 ID。修改后，您需要立即调整与哪个舵机 ID 通信，以便继续通信或保存参数。

can.prefix
一个 13 位整数，用作所有 CAN 通信 ID 的高 13 位。由于 id.id 这会立即生效，因此在更改后，必须使用正确的前缀重新启动通信才能执行保存配置等操作。

servopos.position_min
允许的最小控制位置值，以转数为单位。如果为 NaN，则不应用任何限制。

servopos.position_max
允许的最大控制位置值，以转数为单位。如果为 NaN，则不应用任何限制。

servo.pid_position
配置位置模式 PID 控制器。
kp/ki/kd- PID 增益，单位为：
kp - Nm 每转
ki - 每转 Nm/s
kd - Nm 每转/s
iratelimit- 积分项可以结束的最大速率，以 N*m/s 为单位。<0 表示“无限制”
ilimit- 总最大 I 项，以 Nm 为单位
max_desired_rate- 如果非零，则命令位置被限制以该速率（以 Hz 为单位）变化。
请注意，这些值是物理单位。因此，kp 值为 1 意味着对于输出的 1 转误差，将应用 1 Nm 的校正扭矩。同样，kd 值为 1 时，每秒 1 转的误差将导致 1 Nm 的校正扭矩。
再次注意，这些值是在输出端测量的，因此是在对位置、速度和扭矩进行任何缩放 之后 motor.unwrapped_position_scale 由 暗示的。

servo.pid_dq
这些与位置模式 PID 控制器具有相同的语义，并影响电流控制回路。

servo.default_velocity_limit/servo.default_accel_limit
对 lively board 内生成的轨迹进行限制。如果其中一个是 nan，则该限制未设置。这些限制也可以在每个命令的基础上单独覆盖。限制的语义如下：

均未设置（均 nan）在这种情况下，位置和速度命令立即生效。控制位置初始化为指令位置，控制速度设定为指令速度。控制位置将无限期地以给定的速度前进，或者直到到达命令停止位置。

仅设置速度限制：在这种情况下，在达到所需位置之前，控制速度将设置为正速度限制或负速度限制。一旦达到所需位置，速度将无限期地以指令速度继续。

两者都设置：在这种情况下，指令位置和指令速度都不会立即应用。相反，控制速度通过每个时间步长配置的加速度提前，同时限制为最大配置速度，以实现所需的位置和速度。一旦达到所需的位置和速度，则该最终速度将无限期地继续。请注意，如果当前和最终速度不允许单程进近，这可能需要“回溯”。

servo.voltage_mode_control
当设置为非零时，电流控制回路不闭合，所有以安培为单位的电流命令都被视为与校准相电阻相关的以伏特为单位的电压模式命令。对于高绕组电阻电机，默认电流检测电阻太小，无法进行准确的电流检测，从而导致显着的齿槽转矩和电流检测噪声。如果不能选择更换电流检测电阻，则可以使用该标志来实现平滑控制。缺点是实际扭矩将不再准确地跟随施加的扭矩以速度或面对外部干扰。

servo.fixed_voltage_mode
如果非零，则不进行基于反馈的位置或电流控制。相反，根据当前指令位置和配置的电机极数将固定电压施加到相端子。在这种模式下，编码器和电流检测电阻根本不用于控制。

这是一种类似于廉价无刷云台控制器的控制模式，并且依赖于在电机绕组中连续燃烧固定量的功率。

此模式激活时，驱动器禁用时报告的位置和速度将为 0，启用时与控制位置完全相同。

在此模式下，各种降额限制不起作用：

扭矩降额温度
超出位置界限时的扭矩降额
最大电流限制
指令最大扭矩
过热仍会触发故障。

servo.fixed_voltage_control_V
在固定电压控制模式下，施加到输出的电压。

servo.max_position_slip
当有限时，这会对控制位置和当前测量位置之间的差值施加限制，以转数为单位。当控制器在速度模式下使用时，它可以用来防止“追赶”。

servo.max_voltage
如果输入电压达到此值，则会触发故障并停止所有扭矩。

servo.max_power_W
控制器会将输出功率限制为该值。该值是相对于 40kHz 的 PWM 速率定义的，并相对于 PWM 速率进行线性缩放。

servo.pwm_rate_hz
要使用的 PWM 速率，默认为 40000。允许的值介于 15000 和 60000 之间。较低的值会提高效率，但会限制峰值功率并降低最大速度和控制带宽。

servo.derate_temperature
当温度达到此值时，扭矩开始受到限制。

servo.fault_temperature
如果温度达到此值，则会触发故障并停止所有扭矩。

servo.flux_brake_min_voltage
当输入电压高于此值时，控制器使电机充当具有电阻的“虚拟电阻器”
servo.flux_brake_resistance_ohm。所有额外的能量都被倾倒到电机的 D 相中。如果输入直流链路无法接受足够的能量，这可用于处理过多的再生能量。

servo.max_current_A
相电流永远不会超过这个值。可以减少它以限制控制器使用的总功率。超出出厂配置值可能会导致硬件损坏。

servo.max_velocity
如果速度超过此阈值，输出功率将受到限制。

servo.max_velocity_derate
一旦速度达到 max_velocity 加上这个值，允许的输出功率就会降低到 0。

servo.rotation_*
这些值为电机配置了更高阶的扭矩模型。
servo.rotation_current_cutoff_A 如果小于该值，则 motor.v_per_hz 使用隐含的线性关系

servo.rotation_current_cutoff_A 如果相电流小于该值，则 motor.v_per_hz 使用隐含的线性关系。
一旦高于该截止值，则使用以下公式从相电流确定转矩：

torque = cutoff * tc + torque_scale * log2(1 + (I - cutoff) * current_scale)
tc 是从 导出的扭矩常数 motor.v_per_hz。

此模型不会自动校准，需要手动确定和配置。

servo.default_timeout_s
通过 CAN 发送位置模式命令时，有一个可选的看门狗超时。如果没有以特定速率接收到命令，则控制器将锁定到“位置超时”状态，需要停止命令才能恢复操作。此配置值控制控制器在收到命令后进入此状态之前等待的时间长度。如果设置为，nan 则控制器将永远不会进入此超时状态。

可以使用 0x027 寄存器或诊断接口命令的 t 可选标志在每个命令的基础上覆盖它。d pos

servo.timeout_max_torque_Nm
当处于“位置超时”模式时，控制器用于抑制输出。此参数控制可用于此类阻尼的最大扭矩。

servo.timeout_mode
选择在位置超时模式下将发生什么行为。允许的值是顶级模式的子集。

0 - “停止” - 驱动程序脱离
12 - “零速度”
15 - “刹车”
servo.rezero_from_abs
如果设置为 1，则在启动后不久，位置值将初始化为与 ABS 端口测量的位置最接近的值。

abs_port.mode
配置 ABS 端口的操作模式：

0 - 禁用
1 - AS5048B（I2C 地址默认为 64）
2 - AS5600（I2C 地址默认为 54）
abs_port.i2c_mode
配置将使用的 I2C 模式：

abs_port.encoder_i2c_address
与辅助编码器通信的 I2C 地址。

abs_port.encoder_poll_ms
轮询辅助编码器的频率（以毫秒为单位）。必须不少于 5。

举个例子:
"conf write" :
can send 01 40010b636f6e662077726974650a
“conf set id.id 2\n” :

can send 01 400111636f6e66207365742069642e696420320a
“conf set abs_port.mode 3\n” :
can send 03 400119636f6e662073657420616273f706f72742e6d6f646520330a

第四章 固件更新配置

3.1 Pid 调参方法以及对不同电机之间的抖动问题

该章针对 2023/3/27 后使用的最新固件。为了提高电机驱动的兼容性，为了去适配更多的电机，解决不同电机之间的齿槽转矩与电机抖动的问题，新版本的固件更新了电机齿槽转矩参数，以及如何调节电机转动钢度和转动时阻尼大小的参数，假使使用者觉得电机高转速时抖动，可通过指令调节参数，确认并保存。但需要提醒使用者，电机高转速抖动通过调整参数解决后，低转速时会发生抖动，这是一个取舍问题，取决于使用者的使用方式和使用情况。以下两种方法均可对不同点击之间的抖动情况有效。

4.1.1 齿槽转矩参数调节：

点击 config，选中 motor 选项，找到 cogging_dq_scale 选项，该选项参数为校准后自动识别的齿槽转矩参数，不同电机之间发生的抖动问题可以通过修改该参数经行微调。

4.1.2 电机钢度及阻尼：

参考 FOC 位置控制指令：**d pos <pos> <vel> <max_torque> [options...]**

例：d pos nan 5 0.5 p0.5 d0.5

上面例子中的 p 和 d 参数为电机的转动钢度（P）以及旋转阻尼(d)的调整，当电机发生轻微抖动时，可通过调节 P 和 d 的参数经行微调。为了更加明显地感受到该指令 P 和 d 之间的区别，可以使用低速转动时或者让点击不动时调节 P 和 d 的参数，能明显地感受得到钢度以及阻尼对电机转动的影响。例：d pos nan 0 0.5 p1 d0.5,通过该指令使电机停在原位置，通过修改 Pd 之间的参数后，使用手转动电机，可以明显感受到区别。通过上述方法在电机转动时，电机出现轻微抖动可以使用该方法调节电机。

Pd 参数调到最佳状态后，使用 P 参数乘以当前 PID 参数的 P，结果作为新的 PID 参数中的 P 参数，d 参数与上述操作相同。

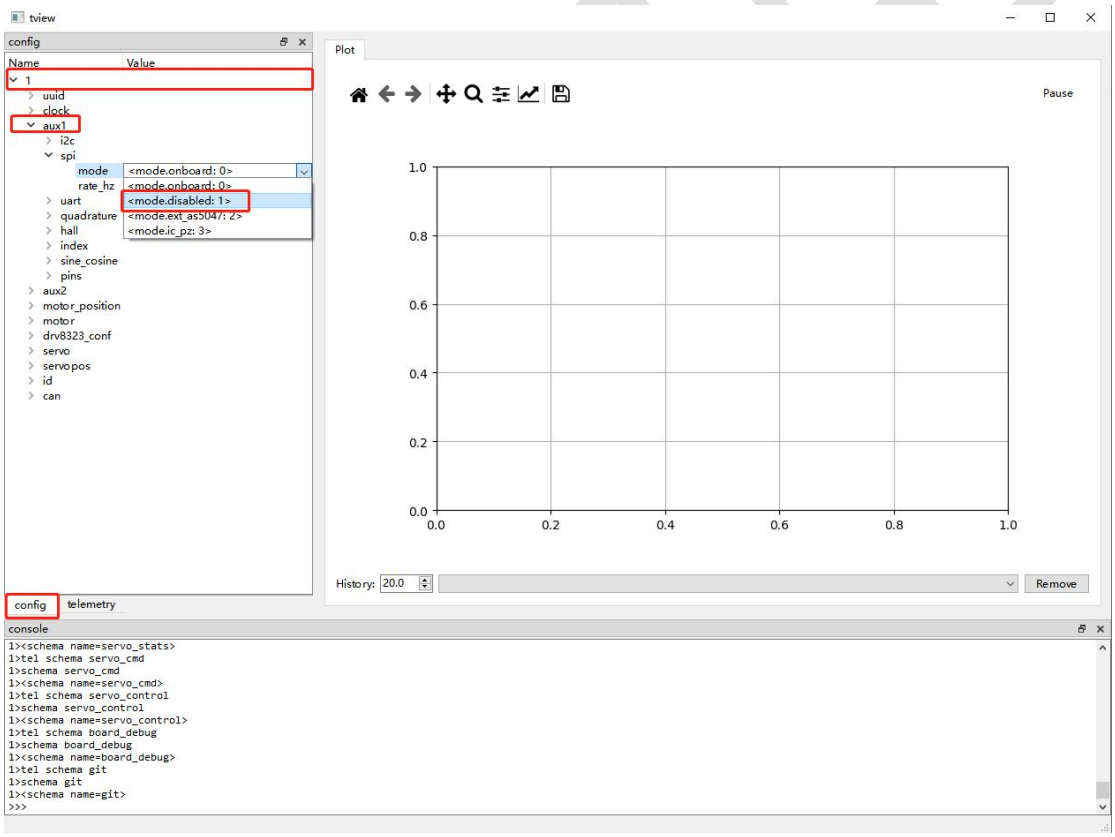
4.2 不同编码器源选择

4.2.1 使用内部编码器 AS5047P

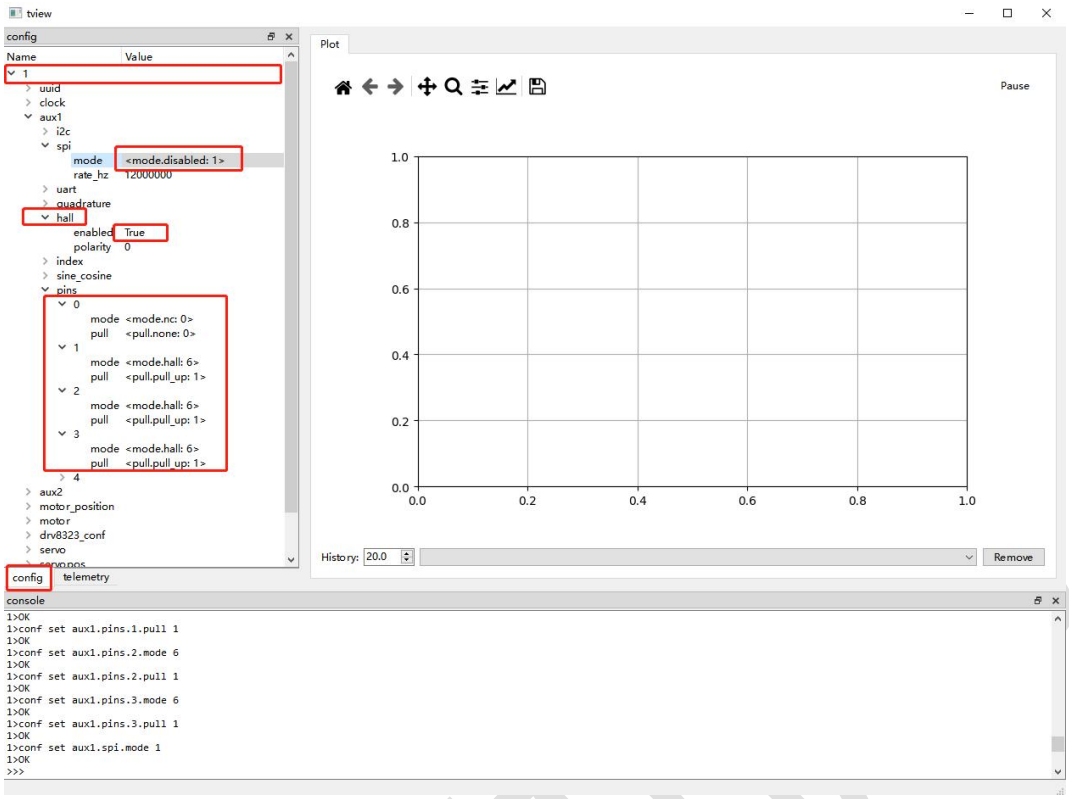
使用通过内部SPI总线连接的板载AS5047P绝对编码器不需要进行任何的操作，拿到新的驱动器后只需要通过校准即可使用。

4.2.2 外置霍尔编码器：支持扭矩控制、速度控制一般、位置控制一般

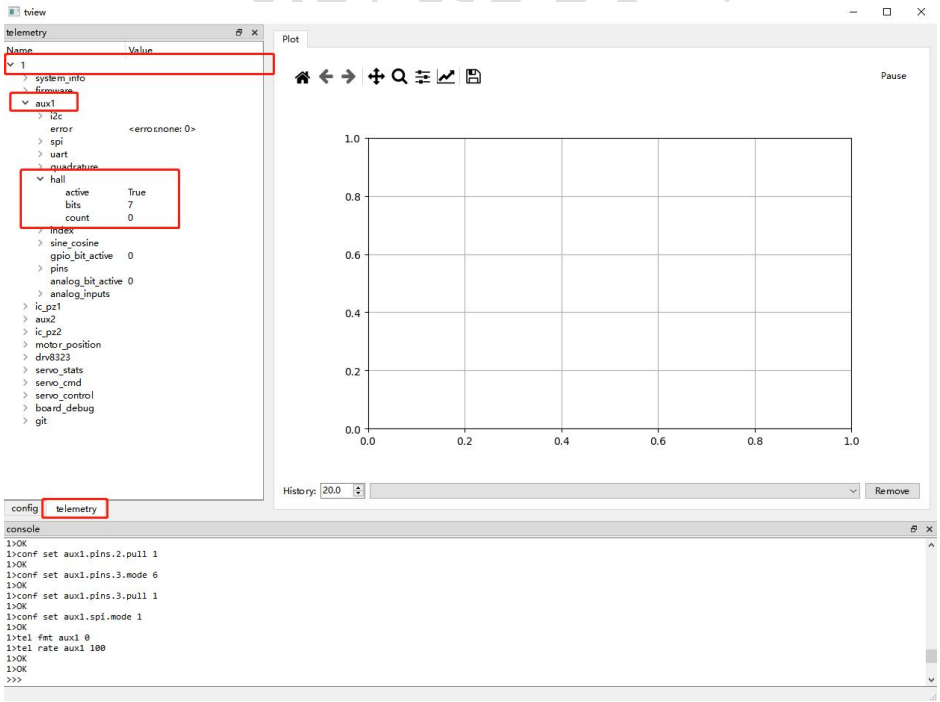
许多常见类型的电机，比如滑板车电机，部分轮毂电机会集成霍尔传感器，可用于换向以及基本的速度控制，使用驱动板控制带霍尔传感器的电机需要外置编码器操作，同时驱动板可供电压3.3V,若不兼容霍尔编码器电压需要使用额外的电瓶转换电路。确保线路没有接错之后，下面是设置方法和步骤。



要配置他首先使用“aux1”配置，禁用SPI功能，然后打开我们的外置端口设置编码器IO口。



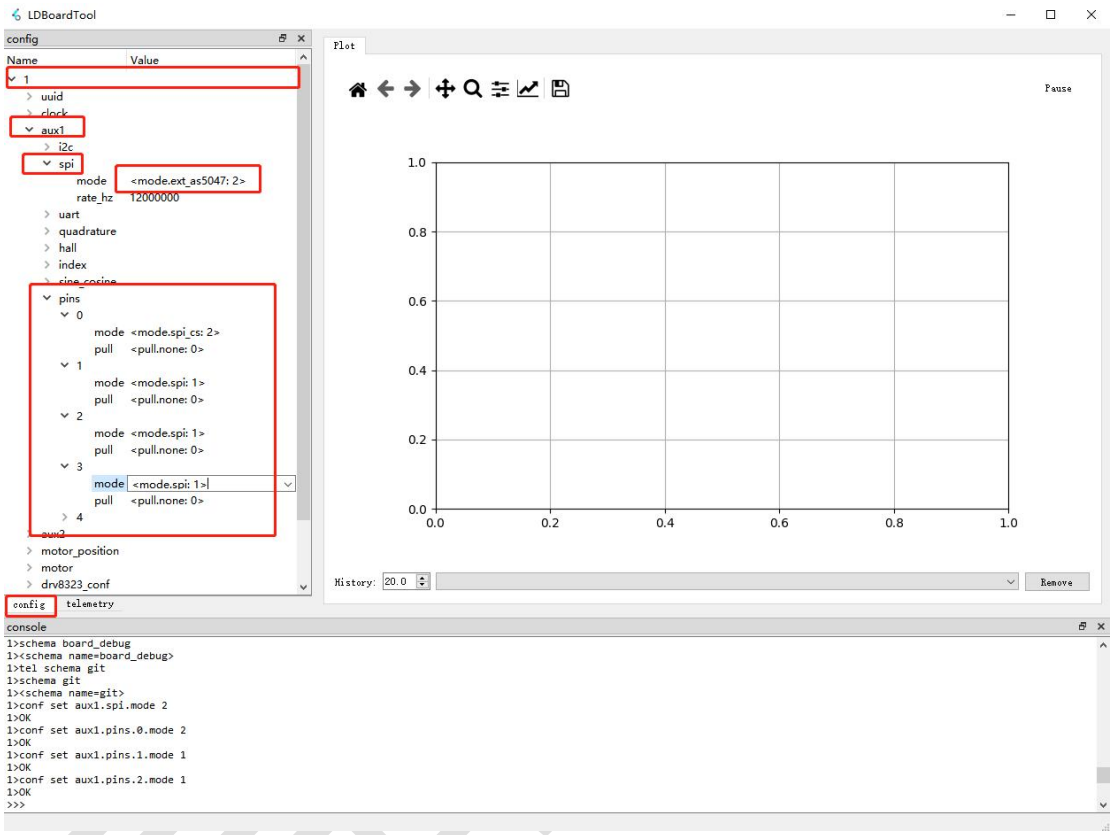
如上图所示，配置完成后打开终端



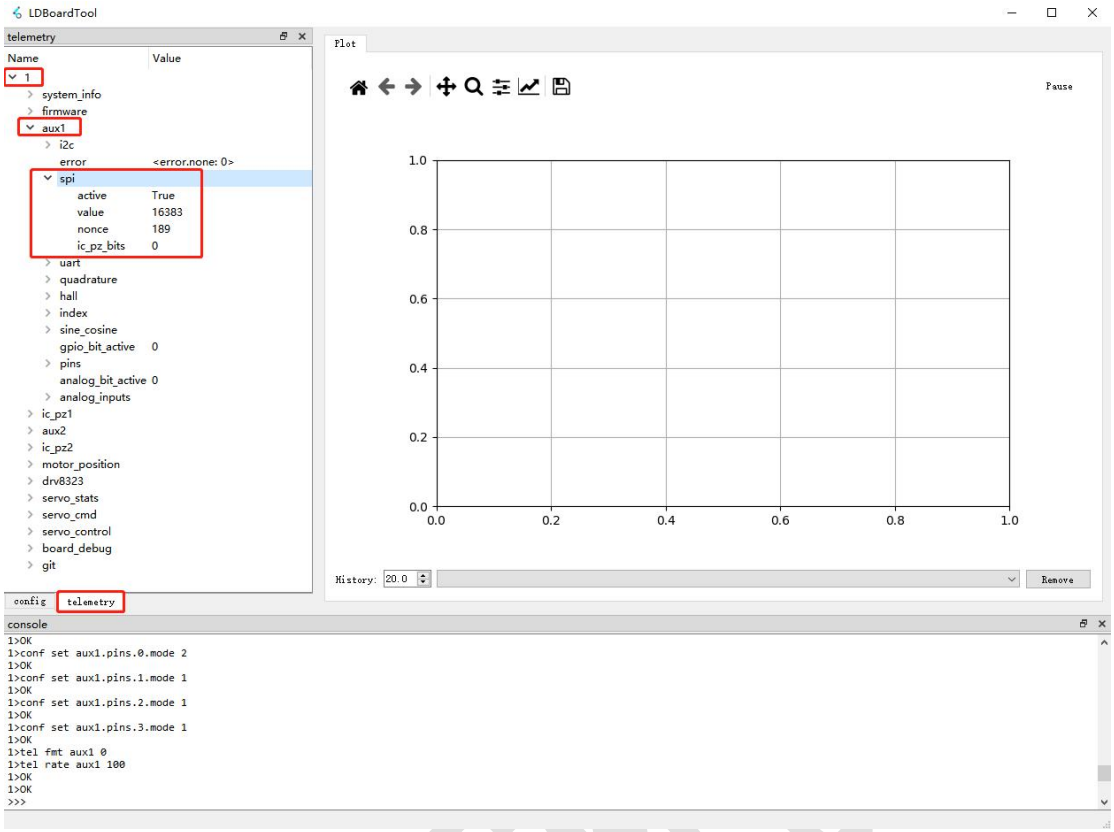
此时转动电机，即可在 hall 中看到数值的变动，说明霍尔编码器配置成功。

4.2.3 外置 AS5047P 编码器：

如果驱动器不能位于电机磁铁后面，那么可以使用外置的 AS5047P 连接到驱动器上。使用外置的编码器，需要将外部编码器的 CS 引脚接到驱动板的 ENC 引脚上。

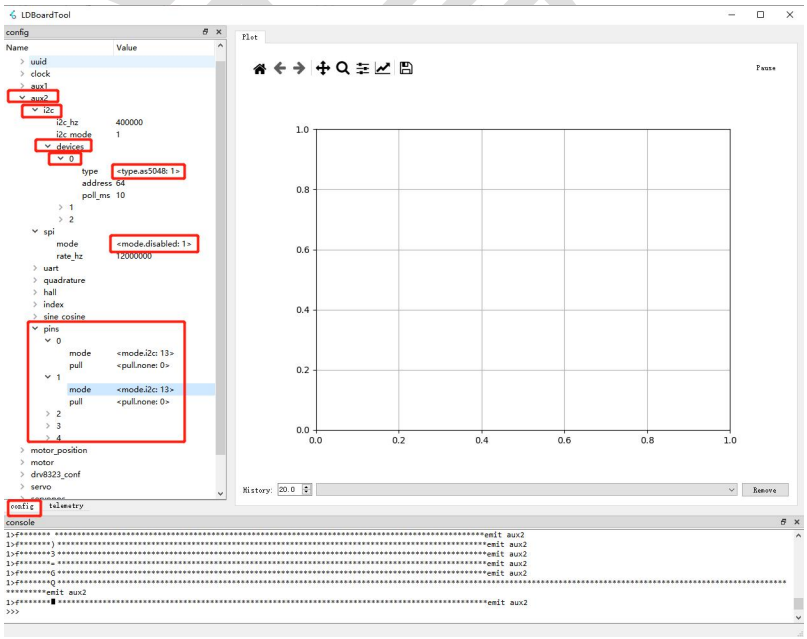


设置完成后，打开终端 telemetry，确认连接无误后即可看到外部编码器的数值。

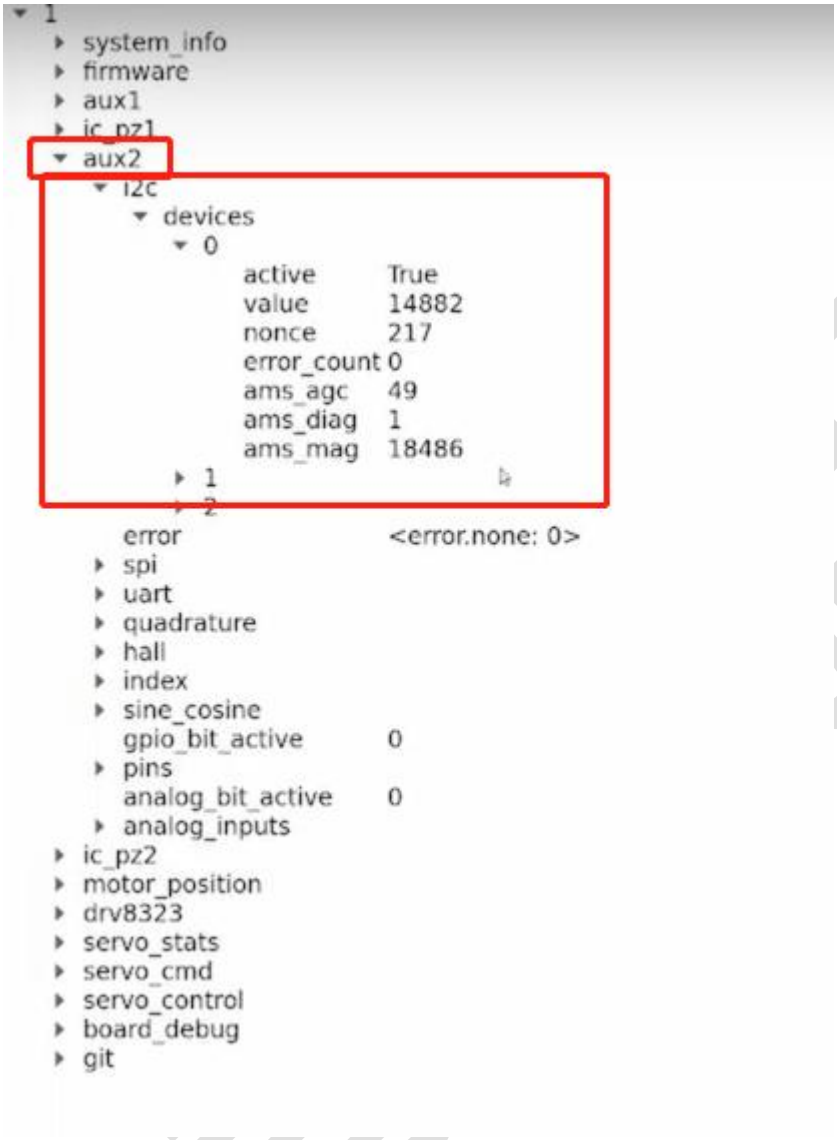


4.4.4 IIC 双编码器设置

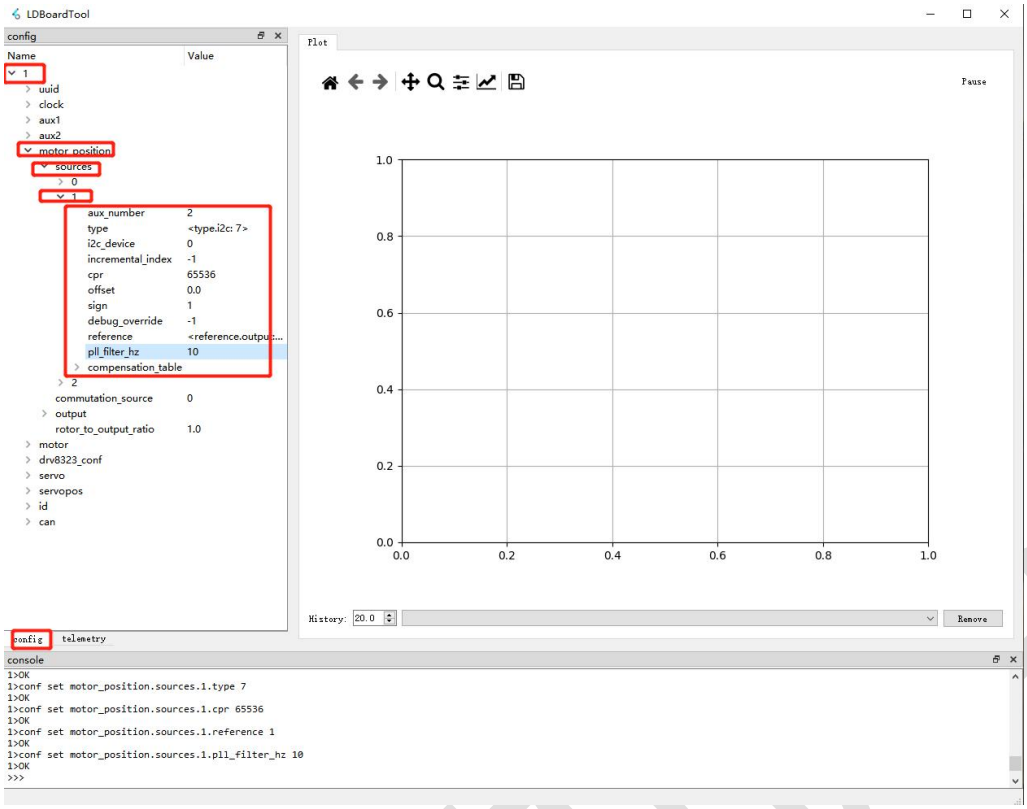
双编码器设置，主编码器测量转子位置，支持 IIC 协议与第二个编码器相连接。



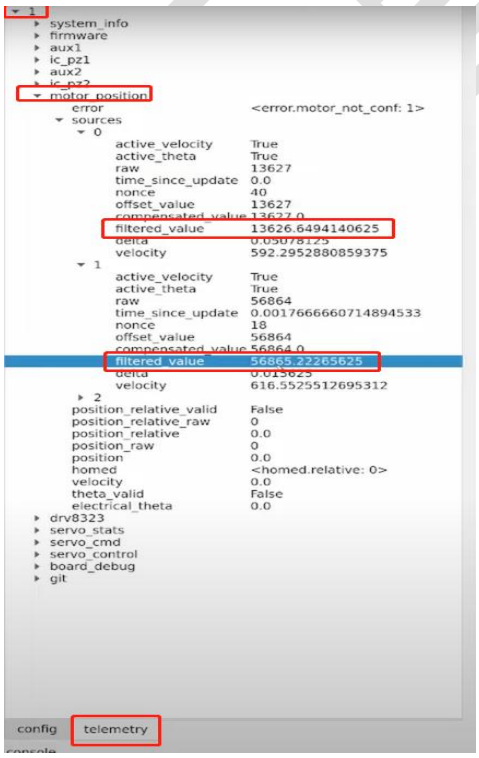
设置好后，打开终端 telemetry,此时打开相应的选项时能看到编码器数值变化。

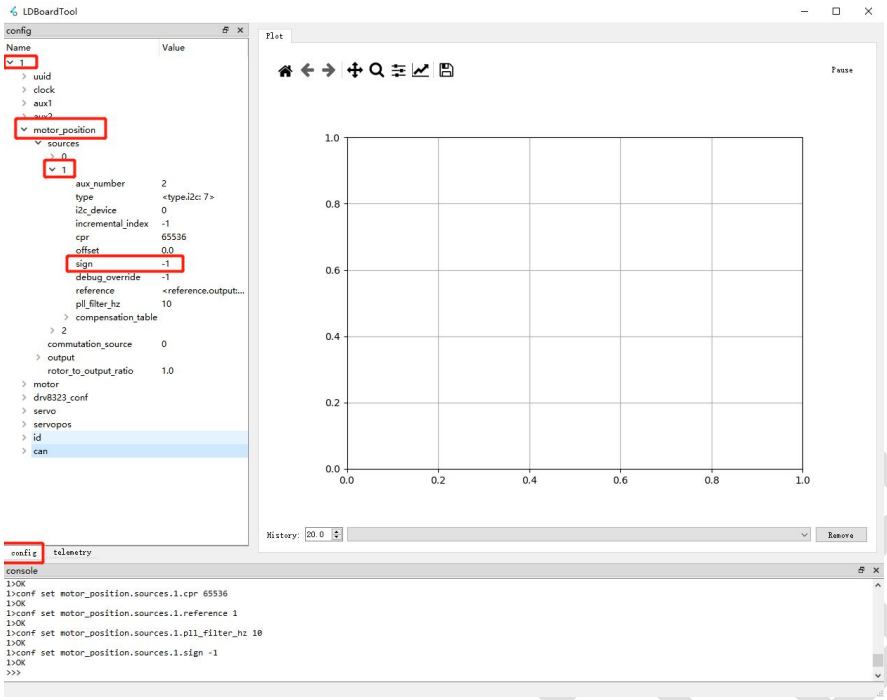


上一步成功后，我们开始设置第二个源。

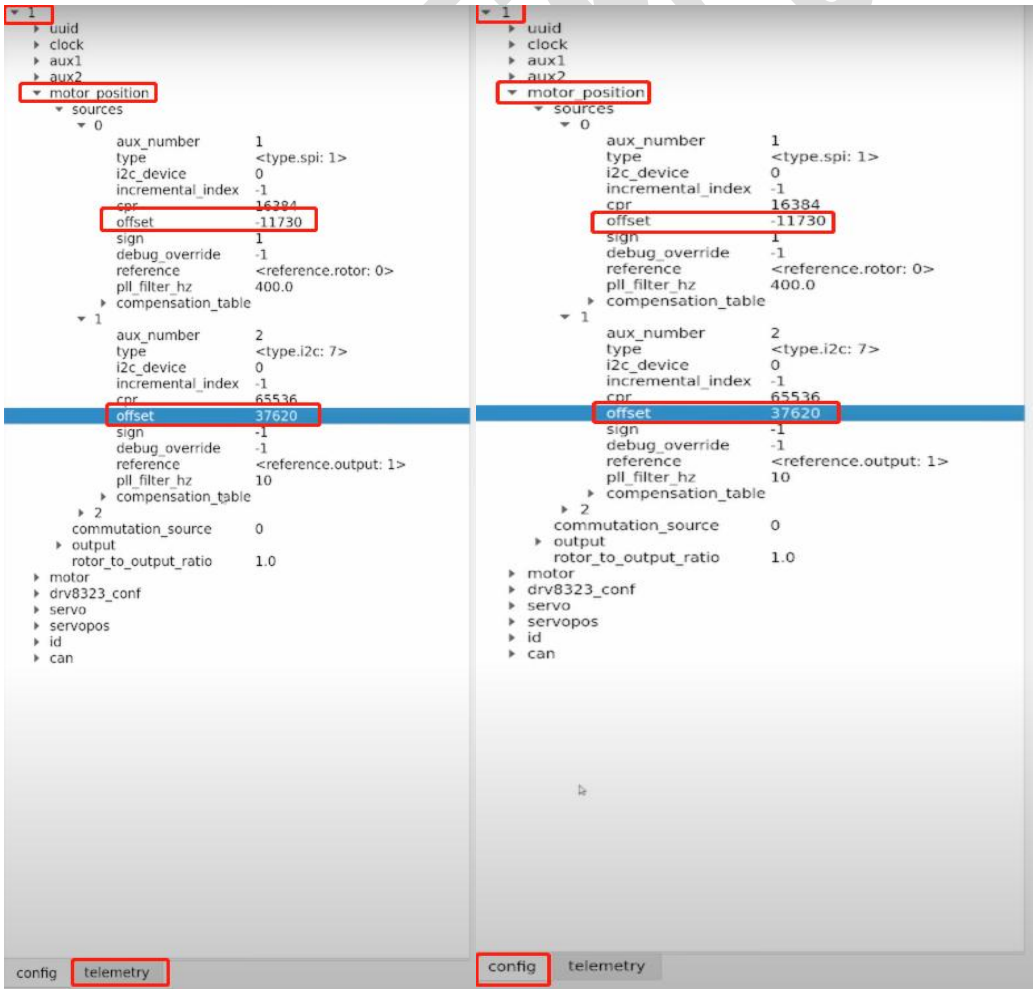


此时设置好之后，我们应确保两个值的符号的偏移量保持一致。看下图中的两个编码器的数值，使用双编码器是当电机正向转动时，另外一个编码器的轮子应该是反向旋转，此时就应该设置编码器的符号，使电机两个参数相反，我们只需要将其中以个编码器源的符号更改，使其相反就可以，如下图操作。

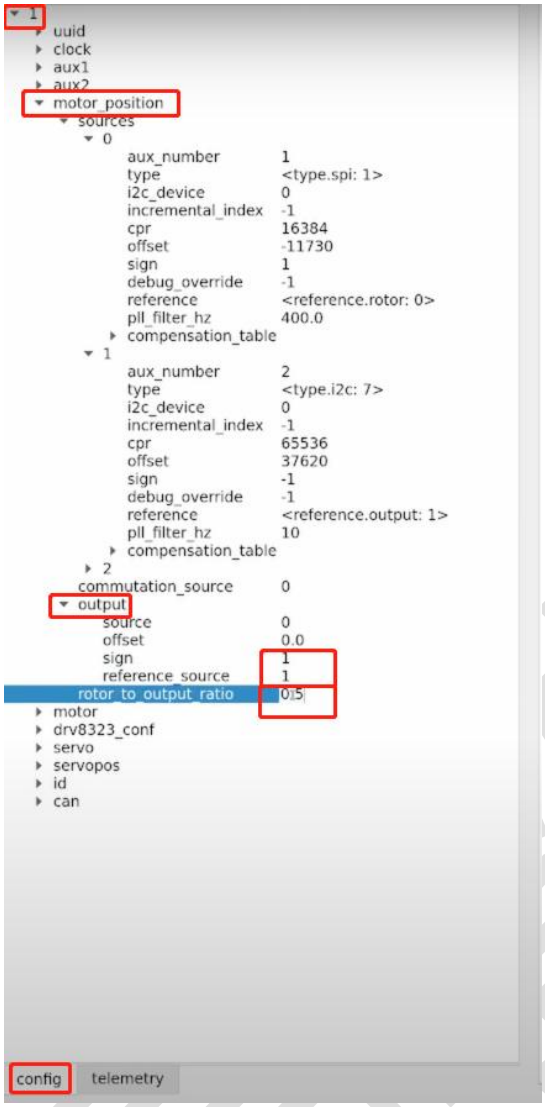




双编码器最后一步设置为将当前检测到的编码器的数值，输入进去编码器的偏移量，作为一个数据校准。如下图操作



最后，配置减速比。上述示例中的减速比为 2：1，因此在最后的配置项中设置为 0.5。设置完成后，在上位机软件中即可捕获到双编码器的数值。



第五章 树莓派操作

*此操作只能在树莓派的系统上运行！

准备硬件：驱动板，树莓派转接板，Can 线，电源

安装相关的包：`pip3 install moteus`

安装 python 库：`sudo pip3 install moteus-pi3hat`

安装 pyside2 库：

sudo apt install python3-pyside2* python3-serial python3-can python3-matplotlib python3-qtconsole

sudo pip3 install asyncqt importlib_metadata pyelftools

sudo pip3 install --no-deps moteus moteus_gui

连接端口：

比如：ID1 出现在端口 1/总线 1 上

```
sudo python3 -m moteus_gui.tview --pi3hat-cfg '1=1' -t 1
```

比如：ID1 和 ID6 出现在端口 1/总线 1 上，ID3 和 ID7 出现在端口 2/总线 2 上：

```
sudo python3 -m moteus_gui.tview --pi3hat-cfg '1=1,6;2=3,7' -t 1,3,6,7
```

选择其他端口（哪个 ID 出现在哪个端口）：

```
--pi3hat-cfg 'BUS1=ID1,ID2,ID3;BUS2=ID4,ID5,ID6
```

最后举例简单例程：

```
import asyncio
import math
import moteus
import moteus_pi3hat
import time

async def main():
#假设有 4 个舵机，每个连接到单独的 pi3hat 总线。
#servo_bus_map 参数描述在哪条总线上找到哪些 ID。
    transport = moteus_pi3hat.Pi3HatRouter(
        servo_bus_map = {
            1:[11],
            2:[12],
            3:[13],
            4:[14],
        },
    )

    servos = {
        servo_id : moteus.Controller(id=servo_id, transport=transport)
        for servo_id in [11, 12, 13, 14]
    }
#向所有舵机发送“停止”
    await transport.cycle([x.make_stop() for x in servos.values()])

    while True:
        now = time.time()
```

```
        commands = [
            servos[11].make_position(
                position=math.nan,
                velocity=0.1*math.sin(now),
                query=True),
            servos[12].make_position(
                position=math.nan,
                velocity=0.1*math.sin(now + 1),
                query=True),
            servos[13].make_position(
                position=math.nan,
                velocity=0.1*math.sin(now + 2),
                query=True),
            servos[14].make_position(
                position=math.nan,
                velocity=0.1*math.sin(now + 3),
                query=True),
        ]
#打印 ID、位置和速度
        print(", ".join(
            f"({result.arbitration_id} " +
            f"{result.values[moteus.Register.POSITION]} " +
            f"{result.values[moteus.Register.VELOCITY]})"
            for result in results))

        await asyncio.sleep(0.02)


if __name__ == '__main__':
    asyncio.run(main())
```

第六章 Linux 操作

准备相关硬件：驱动板、USB 转 FDCAN、电源、数据传输线

1、安装相关依赖环境：pip3 install moteus_gui

2、把 USB 转 FDCAN 协议挂载上去

```
# A udev rule to create a /dev/fdcanusb symlink.
#
# Install into /etc/udev/rules.d
```

```
#
# Then run:
#     udevadm control --reload-rules
#     udevadm trigger --subsystem-match=tty

SUBSYSTEM=="tty",ATTRS{manufacturer}=="mjbots",ATTRS{product}=="fdcanusb",MODE="0666",
SYMLINK+="fdcanusb"
```

3、运行：python3 -m moteus_gui.tview --devices=1

如果使用虚拟机来操作的话可能会存在 PySide2 和 WSL 不兼容的问题。具体环境配置问题需自行解决。