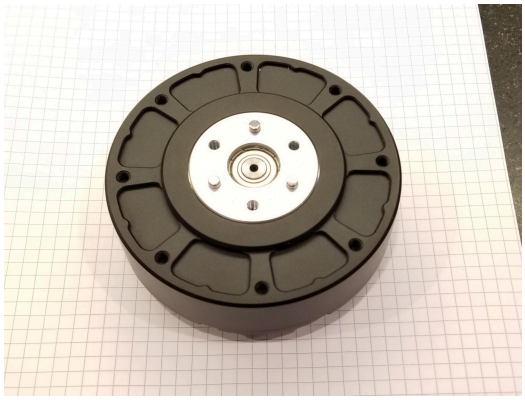
MIT Cheetah/ Research
HobbyKing Robot Motor
AKA SteadyWin V3

Hey SteadyWin: If you are reading this, **please contact me**. I have many questions and you have failed to respond to many email and support requests. Your product is far from what is advertised. That is OK, but we need some answers to improve the product.



AliExpress Page
https://www.aliexpress.com/32985671853.html

Steady Win Company Page
http://www.steadywin.cn/

Video of someone using a different model
https://www.youtube.com/watch?v=ecSQZINda6g

My Videos
https://www.youtube.com/watch?v=Fb6HQNZ4PzQ
https://twitter.com/buildlog/status/1219807520816017409
https://twitter.com/buildlog/status/1220372055776022528

Ben Katz Blog
https://build-its-inprogress.blogspot.com/search/label/HobbyKing%20Cheetah

Ben Katz Github

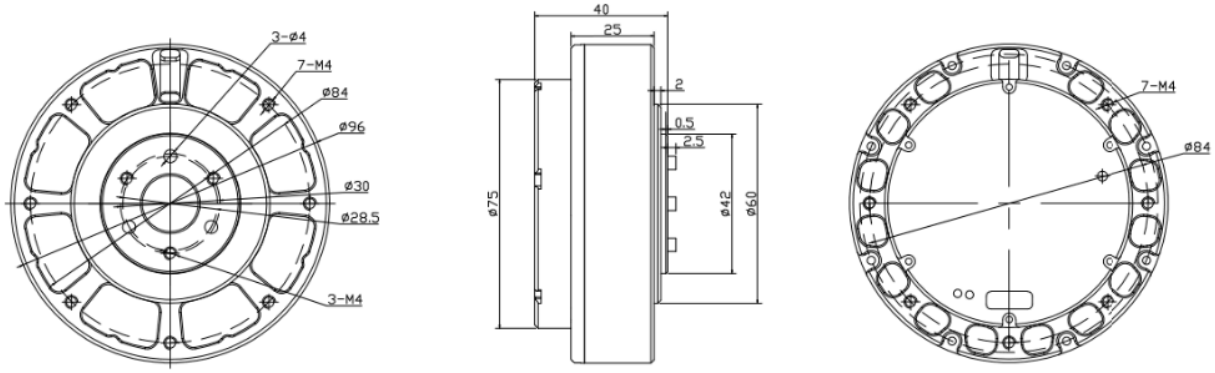
https://github.com/bgkatz/3phase_integrated

Motor Drive documentation
https://docs.google.com/document/d/1dzNVzblz6mqB3eZVEMyi2MtSngALHdgpTaDJIW_BpS4/edit

Controller Schematic PDF
https://drive.google.com/open?id=1LKZBExanS721uNWVH1Bye9HUD9dOXb_F

mBed (The Firmware)
https://os.mbed.com/users/benkatz/code/Hobbyking_Cheetah_Compact/

Python Library
https://github.com/bgkatz/USBtoCAN/tree/master/python%20library



Notes:

The motor will turn on with a red LED. If you enable the motor it will turn green, but the rotor will not lock. If you send it to a position, it will go there and lock.

Parameters

Kp is desired position stiffness. If you set all commands to zero except for Kp, the motor will behave like a spring with stiffness Kp about the 0 angle.

Kd is velocity gain. Kd acts like a damper. If you set all the commands to zero except Kd and try to spin the motor by hand, you will feel some drag, proportional to Kd.

The feed-forward torque is a bias torque. If you set all the commands to zero except for the feed forward torque, the motor will just apply the torque you set.

All the commands get summed up in the motor drive, so the final torque is:

Kp*position_error* + *Kd*velocity_error + feedforward_torque

Tip: If you want to go at a specific velocity, the Kd should be higher than the kp

Command Packet Structure (CAN Speed is 1Mbps)

The driver uses one packet to combine 5 commands. The commands are:

- 16 bit position command, scaled between P_MIN and P_MAX in CAN_COM.cpp
- 12 bit velocity command, scaled V_MIN and V_MAX in CAN_COM.cpp

- 12 bit Kp
- 12 bit Kd
- 12 bit Feed-Forward Current

```
// from ...
https://os.mbed.com/users/benkatz/code/Hobbyking_Cheetah_Compact//file/6cc428f3431d/CAN/CAN_com.h/

#define P_MIN -12.5f  // -4*pi
#define P_MAX 12.5f   // 4*pi
#define V_MIN -45.0f
#define V_MAX 45.0f
#define KP_MIN 0.0f
#define KP_MAX 500.0f
#define KD_MIN 0.0f
#define KD_MAX 5.0f
#define T_MIN -18.0f
#define T_MAX 18.0f
```

Sample C code for packing the bits

```
// enter values to pack here

unsigned int pos = 0x1234; // 16 bit
unsigned int vel = 0x0567; // 12 bit
unsigned int kp = 0x089A;  // 12 bit
unsigned int kd = 0x0BCD; // 12 bit
unsigned int ff = 0x0EF1; // 12 bit

unsigned char can_msg[8];

can_msg[0] = pos >> 8;
can_msg[1] = pos & 0x00FF;
can_msg[2] = (vel >> 4) & 0xFF;
can_msg[3] = ((vel & 0x000F) << 4) + ((kp >> 8) & 0xFF);
can_msg[4] = kp & 0xFF;
can_msg[5] = kd >> 4;
can_msg[6] = ((kd & 0x000F)<<4) + (ff >> 8);
can_msg[7] = ff & 0xff;


printf("Test %02x %02x %02x %02x %02x %02x %02x %02x", can_msg[0],
can_msg[1], can_msg[2], can_msg[3], can_msg[4], can_msg[5],
can_msg[6], can_msg[7]);
```

Simple C code to unpack response

```c
int can_msg[6];
// example response ....
can_msg[0] = 1;
can_msg[1] = 0x12;
can_msg[2] = 0x34;
can_msg[3] = 0x56;
can_msg[4] = 0x78;
can_msg[5] = 0x9A;

unsigned int id = can_msg[0];
unsigned int pos = (can_msg[1] << 8) + can_msg[2];
unsigned int vel  = (can_msg[3] << 4) + ((can_msg[4] & 0xF0) >> 4);
unsigned int cur = ((can_msg[4] & 0x0F) << 8) + can_msg[5];

printf("\r\nid 0x%02X", id);
printf("\r\nPos 0x%03X", pos);
printf("\r\nVel 0x%03X", vel);
printf("\r\nCurrent 0x%03X", cur);
```

.

ESP32 CAN Library
https://github.com/sandeepmistry/arduino-CAN

My 3.3V CAN Adapter
https://www.amazon.com/gp/product/B00KM6XMXO

Special Commands:

Enter Motor Mode

[0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFC]

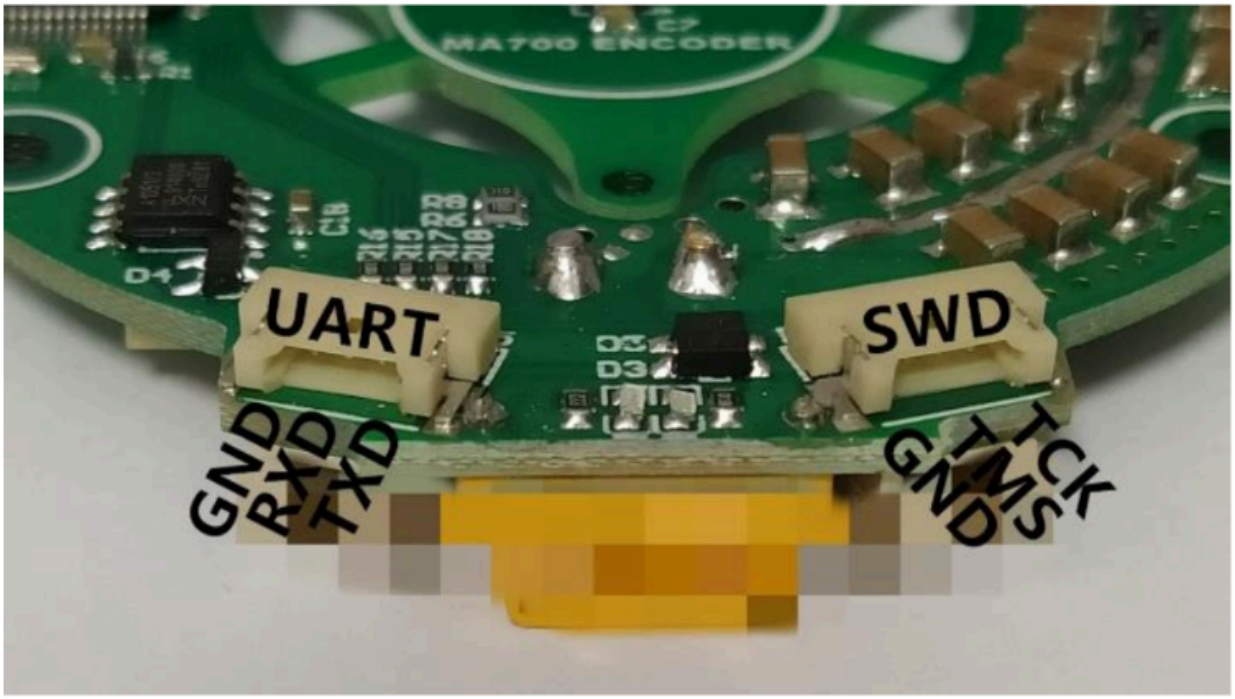
Exit Motor Mode

[0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFD]

Zero Position Sensor - sets the mechanical position to zero.

[0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFE]

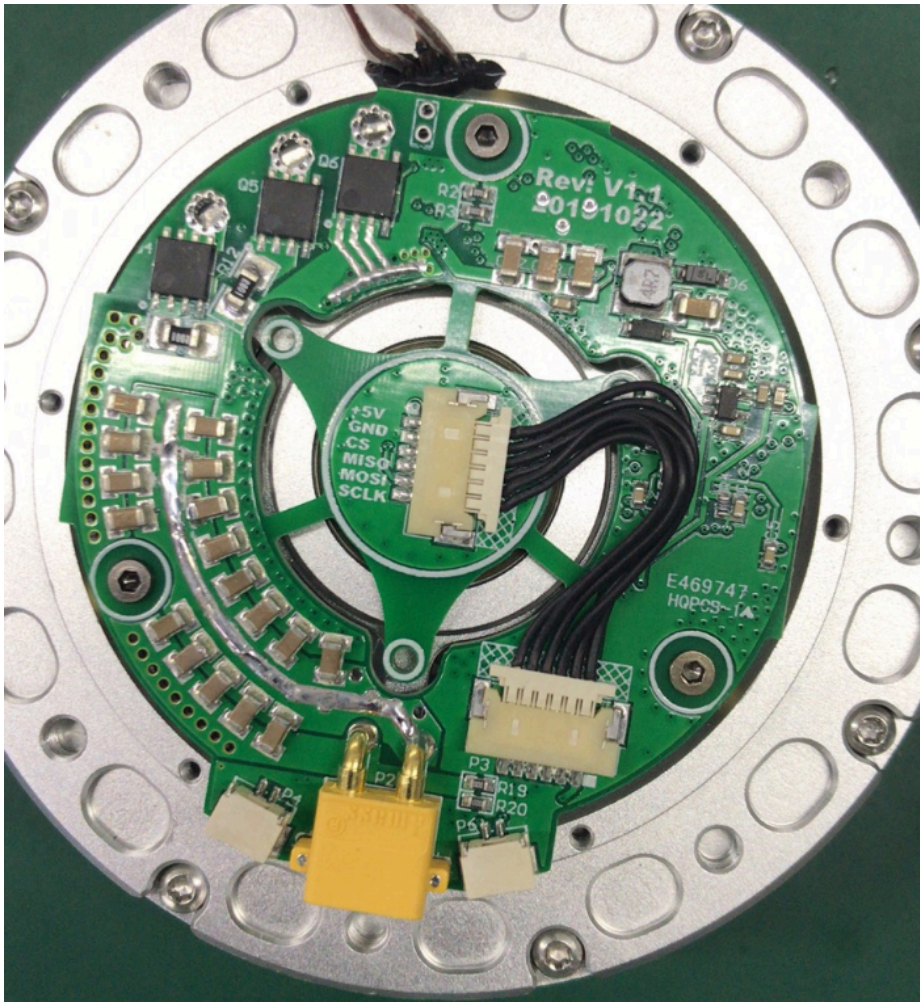

5V电源
24V电源
CAN
GND
5.0V
GND 24V
CANL
CANH

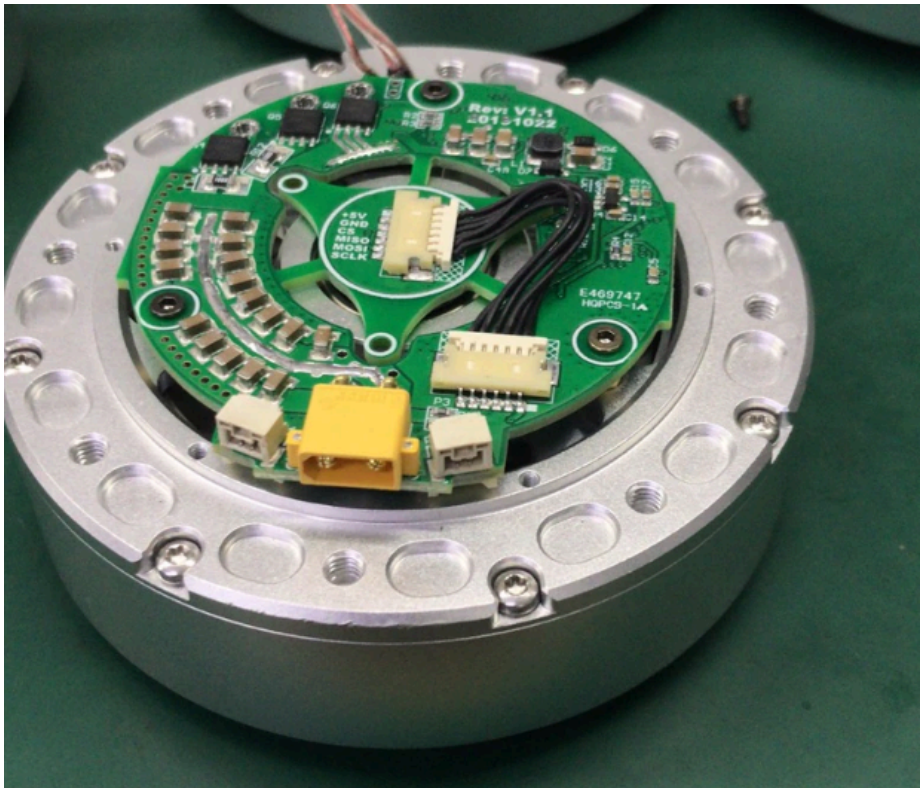


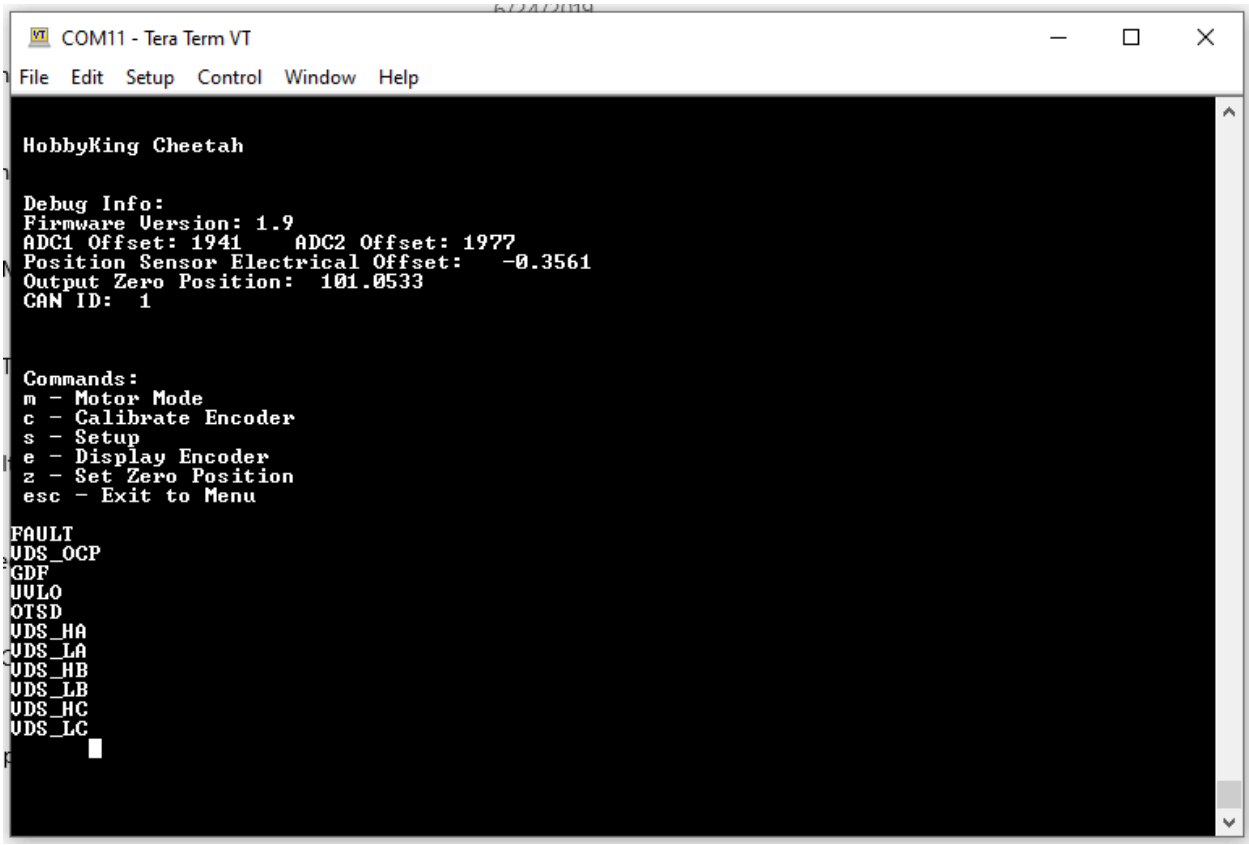
AliExpress source for the serial and prog cables.
https://www.aliexpress.com/item/32902429074.html

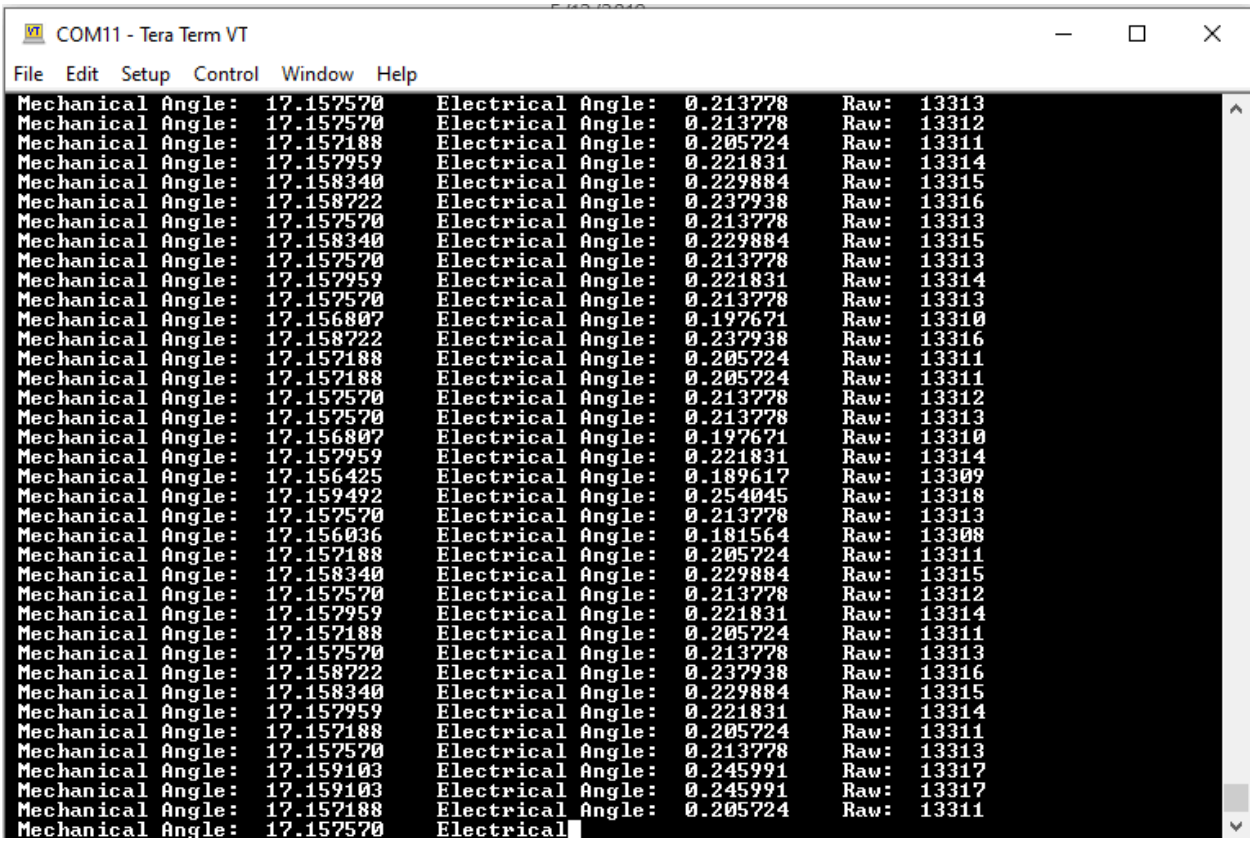
Serial port: 921600 baud, 8 bits, 1 stop bit, no parity bits

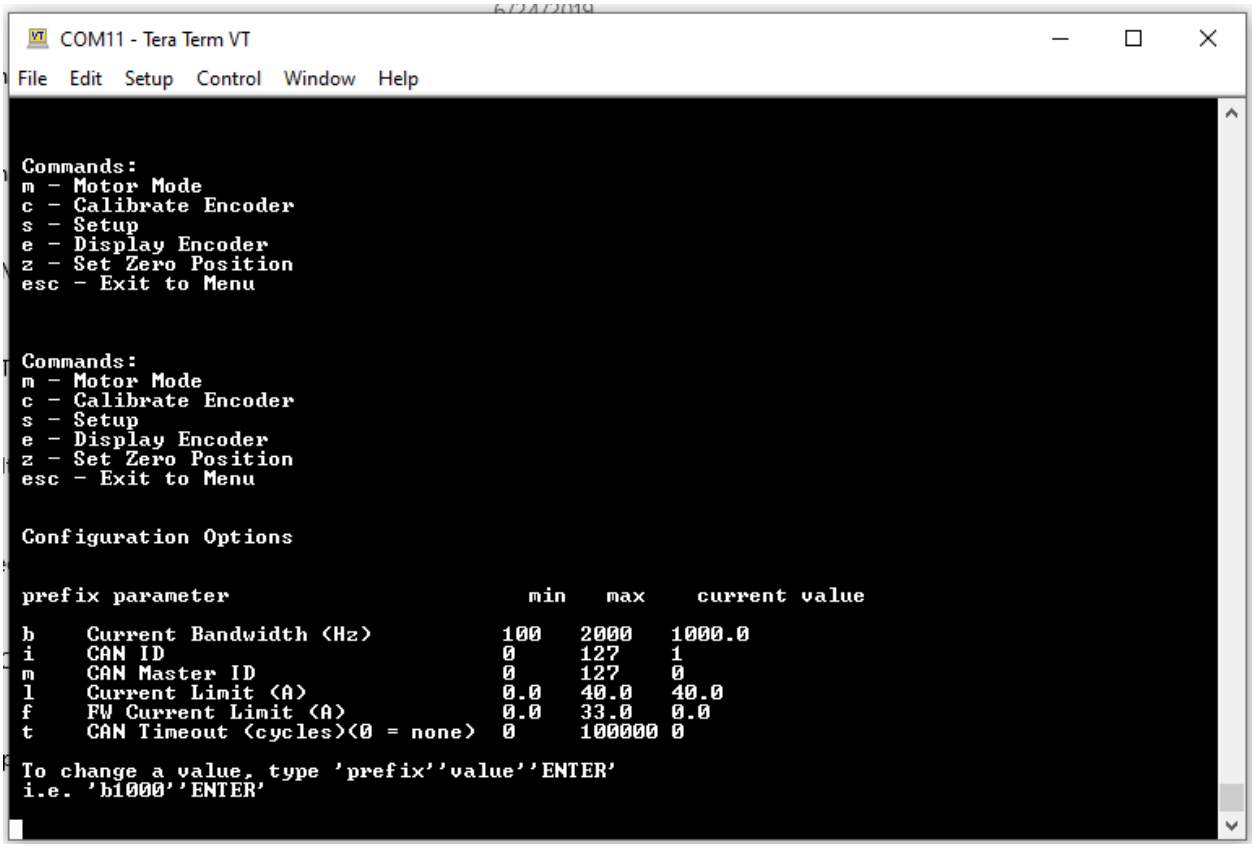
Picture of the controller. Note sure why the encoder is isolated and not connected on the PCB. It might be for thermal or mechanical isolation.



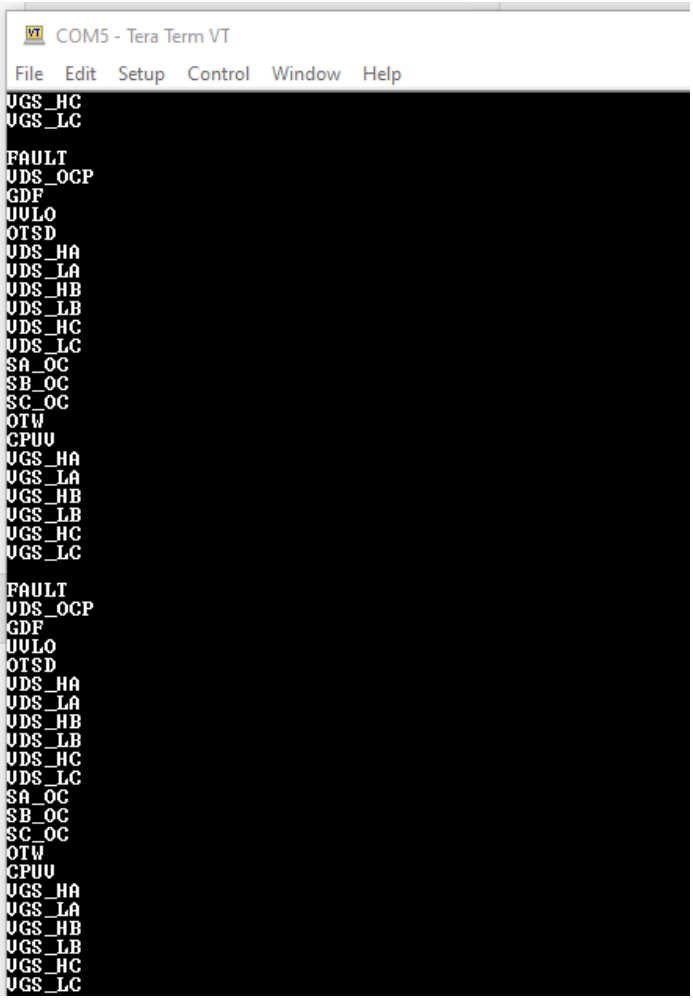

Rev: V1.1
+5V
GND
CS
MISO
MOSI
SCLK
E469747
HGPCB-1A


COM11 - Tera Term VT
File Edit Setup Control Window Help
HobbyKing Cheetah
Debug Info:
Firmware Version: 1.9
ADC1 Offset: 1941 ADC2 Offset: 1977
Position Sensor Electrical Offset: -0.3561
Output Zero Position: 101.0533
CAN ID: 1
Commands:
m - Motor Mode
c - Calibrate Encoder
s - Setup
e - Display Encoder
z - Set Zero Position
esc - Exit to Menu
FAULT
VDS_OCP
GDF
UVLO
OTSD
VDS_HA
VDS_LA
VDS_HB
VDS_LB
VDS_HC
VDS_LC

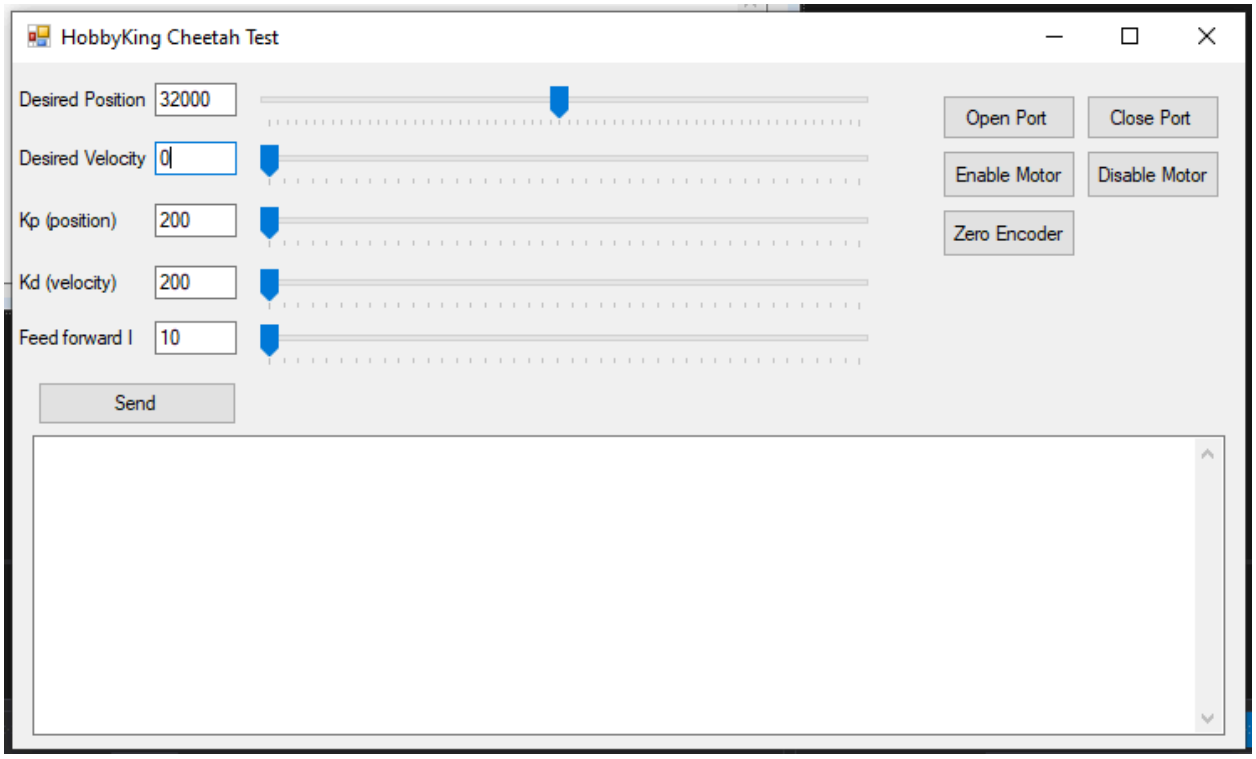

COM11 - Tera Term VT
File Edit Setup Control Window Help
Mechanical Angle: 17.157570 Electrical Angle: 0.213778 Raw: 13313
Mechanical Angle: 17.157570 Electrical Angle: 0.213778 Raw: 13312
Mechanical Angle: 17.157188 Electrical Angle: 0.205724 Raw: 13311
Mechanical Angle: 17.157959 Electrical Angle: 0.221831 Raw: 13314
Mechanical Angle: 17.158340 Electrical Angle: 0.229884 Raw: 13315
Mechanical Angle: 17.158722 Electrical Angle: 0.237938 Raw: 13316
Mechanical Angle: 17.157570 Electrical Angle: 0.213778 Raw: 13313
Mechanical Angle: 17.158340 Electrical Angle: 0.229884 Raw: 13315
Mechanical Angle: 17.157570 Electrical Angle: 0.213778 Raw: 13313
Mechanical Angle: 17.157959 Electrical Angle: 0.221831 Raw: 13314
Mechanical Angle: 17.157570 Electrical Angle: 0.213778 Raw: 13313
Mechanical Angle: 17.156807 Electrical Angle: 0.197671 Raw: 13310
Mechanical Angle: 17.158722 Electrical Angle: 0.237938 Raw: 13316
Mechanical Angle: 17.157188 Electrical Angle: 0.205724 Raw: 13311
Mechanical Angle: 17.157188 Electrical Angle: 0.205724 Raw: 13311
Mechanical Angle: 17.157570 Electrical Angle: 0.213778 Raw: 13312
Mechanical Angle: 17.157570 Electrical Angle: 0.213778 Raw: 13313
Mechanical Angle: 17.156807 Electrical Angle: 0.197671 Raw: 13310
Mechanical Angle: 17.157959 Electrical Angle: 0.221831 Raw: 13314
Mechanical Angle: 17.156425 Electrical Angle: 0.189617 Raw: 13309
Mechanical Angle: 17.159492 Electrical Angle: 0.254045 Raw: 13318
Mechanical Angle: 17.157570 Electrical Angle: 0.213778 Raw: 13313
Mechanical Angle: 17.156036 Electrical Angle: 0.181564 Raw: 13308
Mechanical Angle: 17.157188 Electrical Angle: 0.205724 Raw: 13311
Mechanical Angle: 17.158340 Electrical Angle: 0.229884 Raw: 13315
Mechanical Angle: 17.157570 Electrical Angle: 0.213778 Raw: 13312
Mechanical Angle: 17.157959 Electrical Angle: 0.221831 Raw: 13314
Mechanical Angle: 17.157188 Electrical Angle: 0.205724 Raw: 13311
Mechanical Angle: 17.157570 Electrical Angle: 0.213778 Raw: 13313
Mechanical Angle: 17.158722 Electrical Angle: 0.237938 Raw: 13316
Mechanical Angle: 17.158340 Electrical Angle: 0.229884 Raw: 13315
Mechanical Angle: 17.157959 Electrical Angle: 0.221831 Raw: 13314
Mechanical Angle: 17.157188 Electrical Angle: 0.205724 Raw: 13311
Mechanical Angle: 17.157570 Electrical Angle: 0.213778 Raw: 13313
Mechanical Angle: 17.159103 Electrical Angle: 0.245991 Raw: 13317
Mechanical Angle: 17.159103 Electrical Angle: 0.245991 Raw: 13317
Mechanical Angle: 17.157188 Electrical Angle: 0.205724 Raw: 13311
Mechanical Angle: 17.157570 Electrical



If only 5V is applied via the small 2 pin connector I get a continuous stream of faults on serial port….probably normal behavior.


COM5 - Tera Term VT
File Edit Setup Control Window Help
VGS_HC
VGS_LC
FAULT
VDS_OCP
GDF
UVLO
OTSD
VDS_HA
VDS_LA
VDS_HB
VDS_LB
VDS_HC
VDS_LC
SA_OC
SB_OC
SC_OC
OTW
CPUV
VGS_HA
VGS_LA
VGS_HB
VGS_LB
VGS_HC
VGS_LC
FAULT
VDS_OCP
GDF
UVLO
OTSD
VDS_HA
VDS_LA
VDS_HB
VDS_LB
VDS_HC
VDS_LC
SA_OC
SB_OC
SC_OC
OTW
CPUV
VGS_HA
VGS_LA
VGS_HB
VGS_LB
VGS_HC
VGS_LC

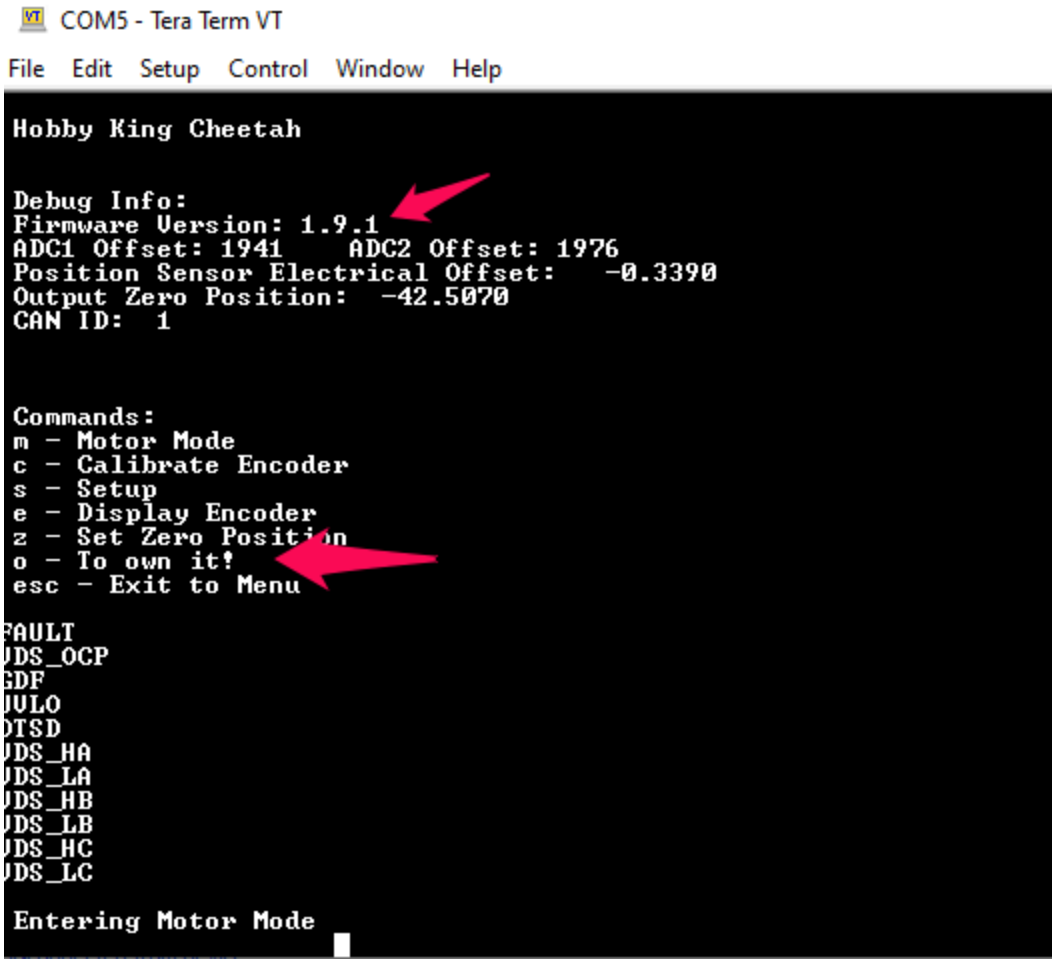
C# Test Program



Programing Firmware

I am able to make basic changes to the firmware and upload them. [Here are some instructions]

Example...



How this will be used with Grbl_ESP32.

The Cheetah motor works a bit like a hobby servo. It has a limited range in rotation and that range is mapped across a 16 bit address. That mapping range is adjustable in firmware, but I will use the existing range for now.

That 16 bit range will be mapped as steps in Grbl in machine space. You can use the steps/mm setting to set a real world unit like degrees. So G0X360 might move one revolution. Mapping in machine space will still allow you to zero the axis, but it will respect the range of the machine (ie motor)

At startup, or whenever Grbl is in stepper_idle mode the torque will be turned off. It will constantly read the Cheetah motor's position and update Grbl's axis location. This means you can manually move the motor and Grbl will track it.

When a Grbl move is made, stepper_idle ends, the torque is turned on and the Cheetah motor begins tracking Grbl's motion. At first, a high update rate (100Hz) of CAN messages will be used. This means the motor does not need to do a rapid uncontrolled move to Grbl's current position. It also means that the speed, position and acceleration of Grbl is tracked by the motor. Later, step/direction signals could be hacking to the firmware.

I hacked Grbl_ESP32 enough to demonstrate the motor. This is not final code, but functional. You can use it for reference. See this file on a branch of the main code.

Step and Direction Control

Step and direction is now working.

Links to Progress Videos (Tweets)

Control
Feedback

Discussion

If you have read this far you deserve a link to the Discord Server. Use the bldc_servo_motors channel. **Lots of good stuff appearing on the Slack Channel !!!**

Donation

If you consider this doc helpful, please consider a donation to support my open source projects via PayPal

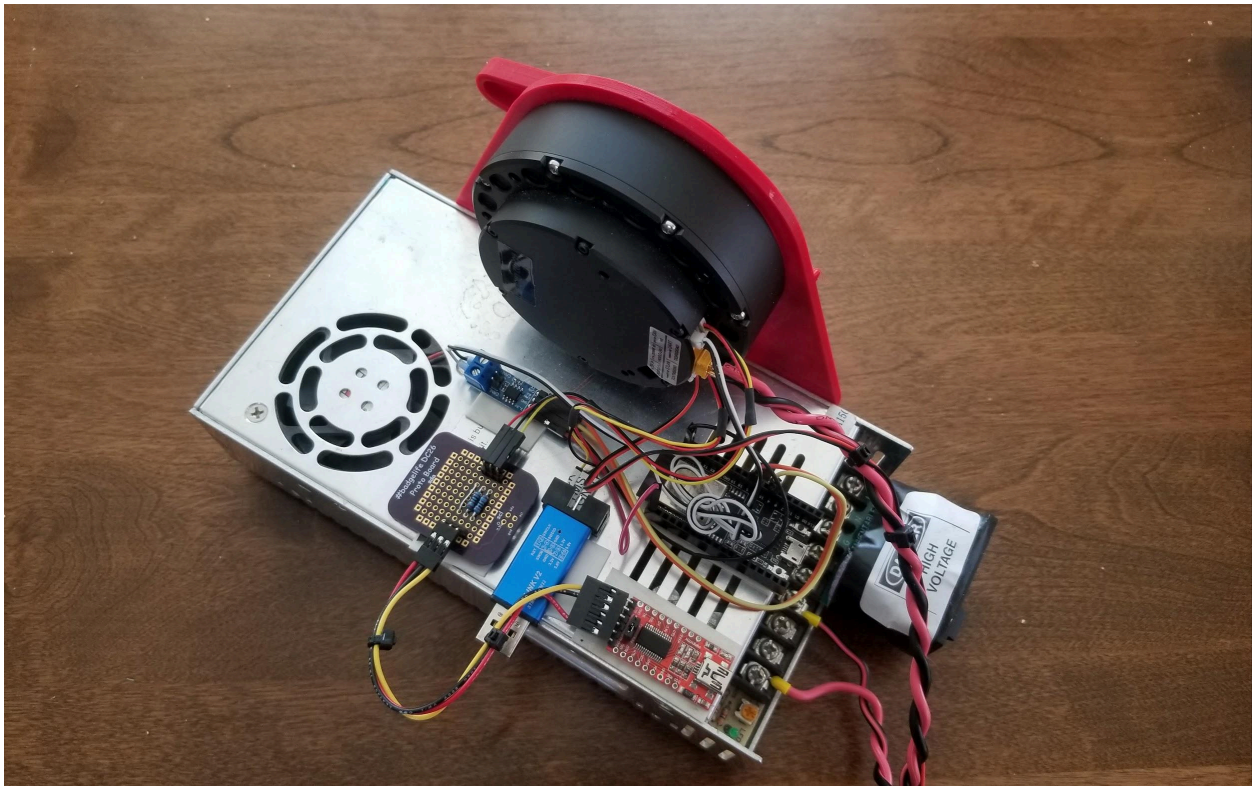
Suggestions

I keep getting blank notices of suggestions. If you have a suggestion, do it on Slack.

Extra Photos

Here is a photo of my test rig. The motor needs a lot of weight to keep it from jumping around. Even with the weight of this 400 watt power supply it can do some serious jumps.

- ESP32 near the power plug is acting as a USB to CAN adapter. It goes through the skinny blue CAN PCB near the motor.
- The red PCB is a 3.3V FTDI USB UART that goes through a breadboard with some resistors. They limit the current, if I screw anything up playing with those pins.
- The blue dongle in the middle is the programmer.
- Not shown is another ESP32 running Grbl_ESP32 that generates step and direction signals. It plugs into the breadboard instead of the FTDI when in step/direction mode.


HIGH
VOLTAGE