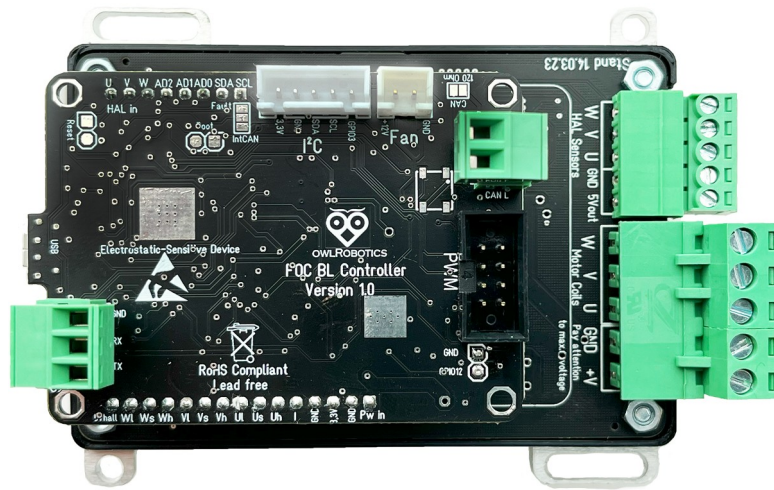




## OWLROBOTICS

Consulting, Development & Production of Robotic Solutions



OwlDrive 24-250 open frame

# owlDrive User Guide

Version: 25/10/2023

## Contents

Overview.....	2
Protocol overview.....	2
What is FOC control.....	3
Motion control overview.....	4
Hardware overview.....	5
Attention: owlDrive power should be 16-36V DC. The power supply must cover the max. motor current and the rated motor voltage. Use an external fuse to protect motor and motor driver. The fuse value must fit to the motor peak current.....	5
Quickstart: Getting your motor to run (motor torque control).....	6
Example application: motor velocity control.....	9
Open-loop velocity control.....	10
Example application: position control.....	11
Example application: motor angle synchronization.....	12
FOC commander protocol (UART/USB).....	14
CAN protocol.....	15
Motor parameters.....	19
BLDC motor formulas.....	20
Appendix A.....	22

## Overview

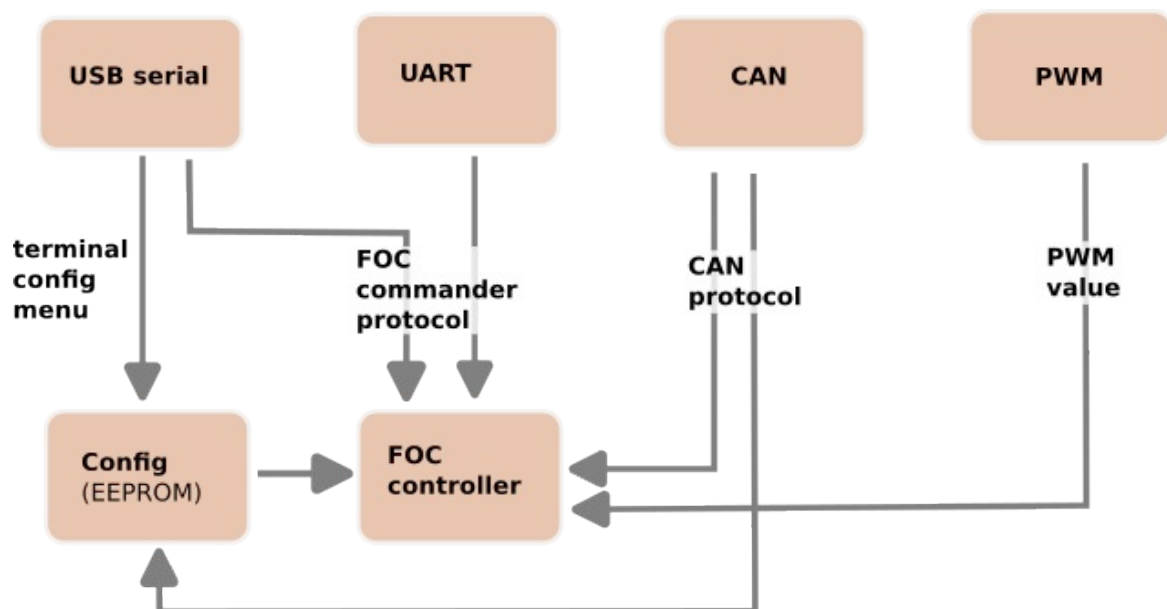
owlDrive is a Field Oriented Control (FOC) driver for nearly all kind of DC brushless motors. It offers various features to configure different motor types and the integrated controller can talk different protocols on different hardware. The easy configuration interface via USB serial terminal provides a simple but effective configuration menu to the user, without specific software apps or tools.

### Attribution

owlDrive is inspired by the open-source project 'SimpleFOC' ([www.simplefoc.com](http://www.simplefoc.com)). We use, under open-source licensing, parts of the SimpleFOC libraries and implemented several, additional features.

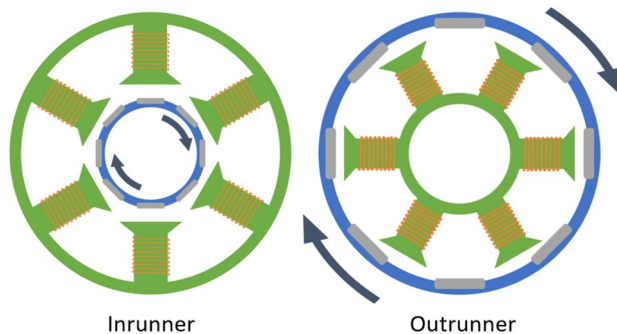
## Protocol overview

UART and USB serial uses FOC commander protocol which communicates directly with the FOC-controller. The configuration (motor config, motion config, CAN config etc.) can only be changed and saved via USB serial or CAN bus.

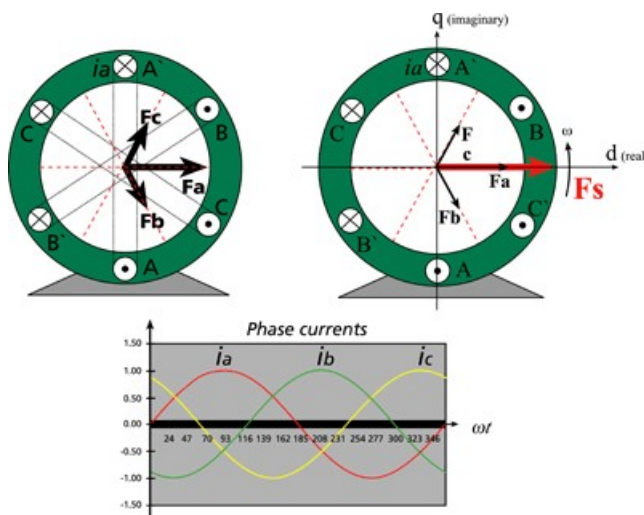


## What is FOC control

To understand how this works, we first look at the structure of the motor. A three-phase motor incorporates windings that are displaced by 120 degrees (or a fraction of that) along the stator.



The blue rotor contains the permanent magnets (N/S pairs) in an inrunner and outrunner motor. The green stator holds the electromagnetic coils.

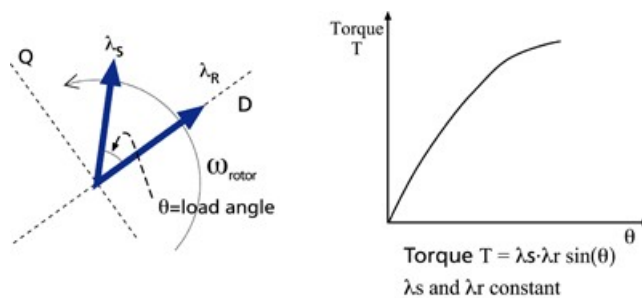


Feeding the windings with three voltages separated in phase by one-third of a cycle produces a rotating magnetic field.

The rotor in an induction machine consists of a closed circuit. When the stator magnetic field sweeps the rotor, an EMF is induced in the rotor circuit, and produces a current. The current produces its own magnetic field, and this induced magnetic field interacts with the stator magnetic field to producing mechanical

force upon the rotor.

These mechanical forces form a force couple, since the direction of force produced is opposite on opposite sides of the rotor, and result in a mechanical torque upon the rotor. The torque produced is proportional to the product of the magnitudes of the fluxes, and the sine of the angle between them.



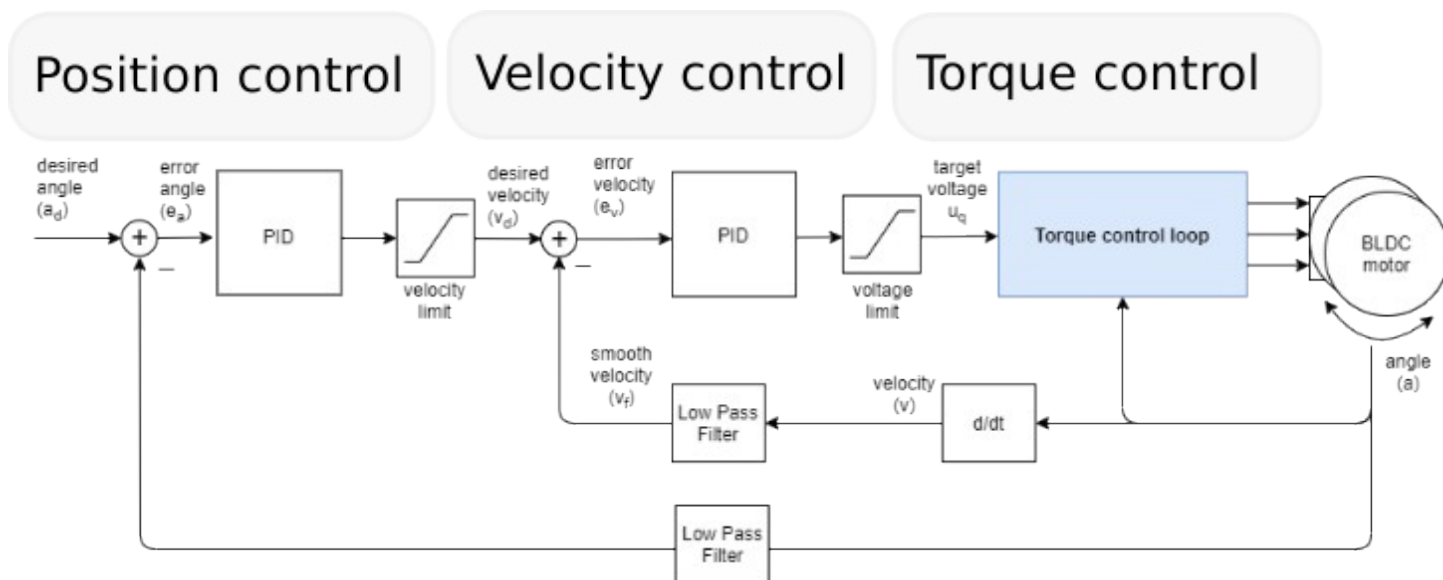
To implement the basic principle of FOC means to maintain a desired alignment between the stator flux and rotor flux. To do this, it is necessary to control the stator currents that produce the stator flux. An angle closer 90 degree produces more flux per unit current.

## Motion control overview

Three main motion control modes are supported

- torque (you define the **voltage** as target for the motor)
- velocity (you define the motor **shaft velocity** as target for the motor)
- position control (you define the **shaft angle** as target for the motor)

Because velocity control depends on proper working torque control, you have to start testing with torque control for a new motor. After that is working, you can test velocity control and after that is working you can try out position control.

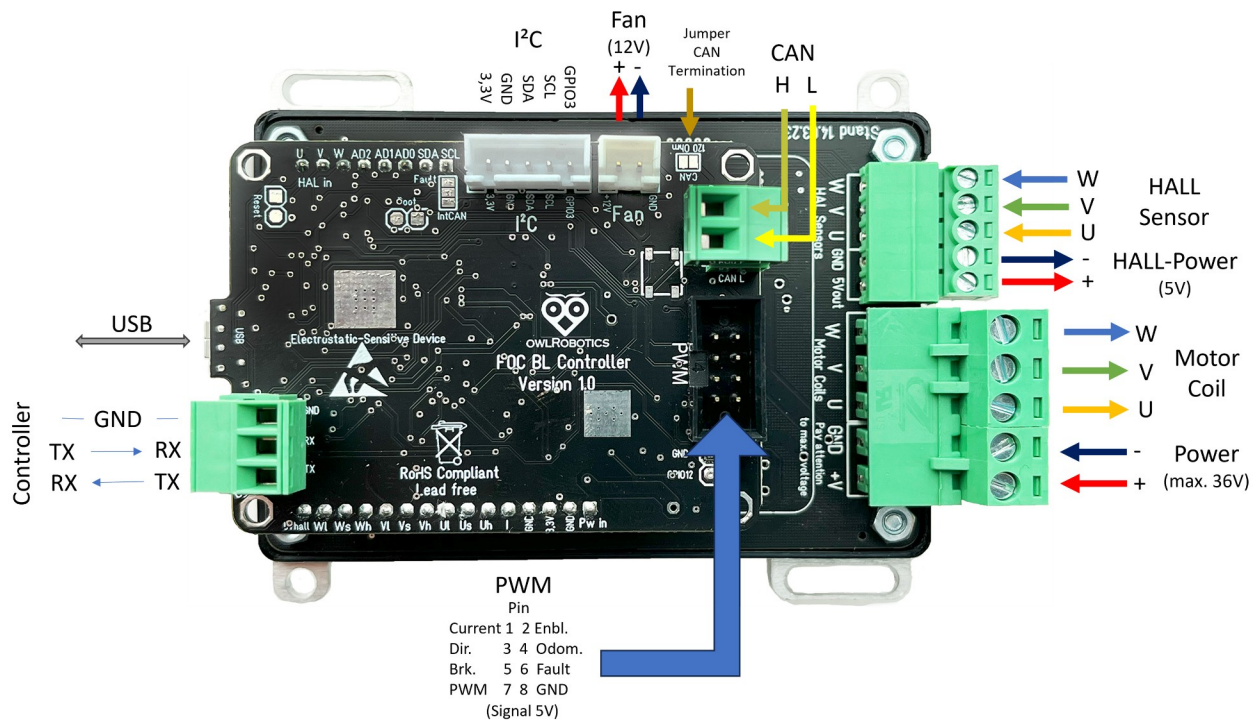


NOTE: The above schema shows a '**closed-loop**' Torque control which means that the position of the motor shaft is detected by a sensor (Hall etc.) and feed back to the torque controller. To test your motor without any sensors, you can use '**open-loop**' where no feedback sensors are required.



## Hardware overview

The owlDrive offers several connectors to connect the motor, sensor and the controller interface:



### Attention:

owlDrive power should be 16-36V DC. The power supply must cover the max. motor current and the rated motor voltage.

Use an **external fuse** to protect motor and motor driver. The fuse value must fit to the motor peak current.

### Attention:

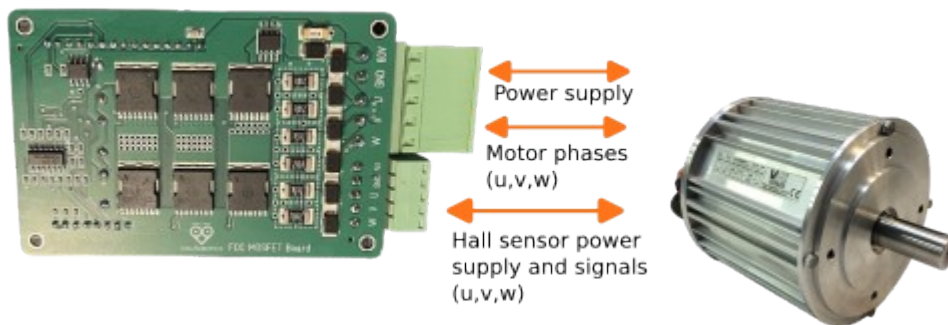
Do not use the owlDrive in applications where malfunctions of the owlDrive (e.g. due to problems in voltage, signal quality or other causes) can cause damage to property or personal injury. Or secure the use of the owlDrive with additional backup / security measures. Since malfunctions cannot be ruled out, owlRobotics GmbH cannot assume any liability for the application!

## Quickstart: Getting your motor to run (motor torque control)

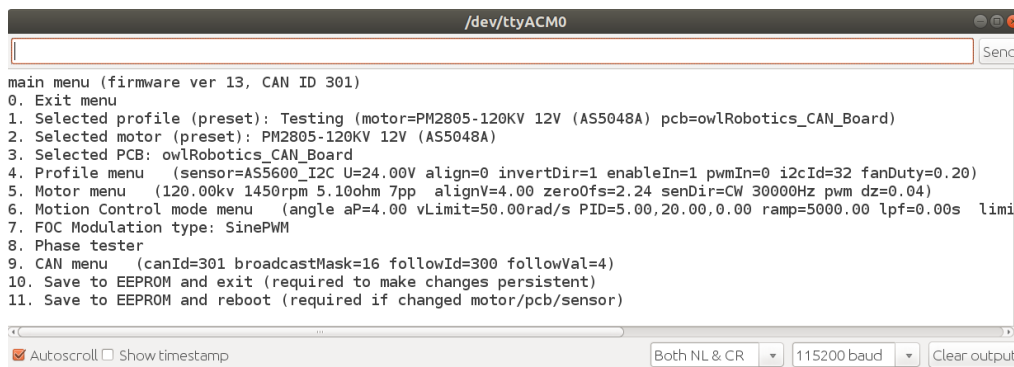
Before you start, refer to Appendix A and get familiar with the handling and warnings.

In this section you will install the serial terminal, connect your motor and configure it.

1. Connect your motor as shown below:



2. Install Arduino legacy IDE: <https://downloads.arduino.cc/arduino-1.8.19-windows.zip> and start it. From the menu, choose 'Tools → Serial Monitor' to start the serial terminal.



The following menu should appear:

To choose a menu point (or to send values), you will enter the menu number (or the value) and press ENTER.

**NOTE:** You can activate this menu at any time by sending '0' (and ENTER) to the driver. You can exit any menu by sending again '0' (and ENTER).

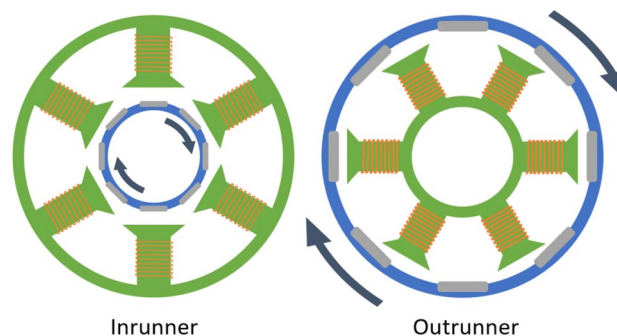
3. Choose 'Selected profile' and choose your profile (e.g. 'Testing') (**NOTE:** you can reset the complete configuration to a preset configuration with this at any time).
4. Choose 'Selected motor' and choose your motor. If your motor is not listed, you can choose 'Generic'.
5. Choose 'Selected PCB' and choose your PCB.

- Choose 'Save to EEPROM and reboot' to restart the driver with the new settings.
- If your motor was not listed above, you will have to configure the motor. Choose 'Motor menu'. The following menu will appear:

```
motor menu:  
0. Exit menu  
1. kv-rating (rpm/V) 120.00  
2. max rpm 1450  
3. rotor pole pairs (N/S) 7  
4. sensor align voltage 4.00  
5. sensor direction (0=CCW, 1=CW) 1  
6. zero electric ofs (rad) 2.24  
7. phase resistance (ohm) 5.10  
8. PWM frequency (Hz) 30000  
9. deadzone / deadtime 0.04  
10. Perform motor sensor align (auto-align)
```

## 8. Configure:

- The kv-rating (rpm per volt) (if you don't know the value, keep the default)
- nominal rpm (if you don't know the value, keep the default)
- number of rotor pole pairs (N/S): (if you don't know the value, you can find it out via the menu point 'perform motor sensor align')



The blue rotor contains the permanent magnets (N/S pairs) in an inrunner and outrunner motor. The

green stator holds the electromagnetic coils.

- sensor direction: how the hall sensor direction is (counter-clockwise or clockwise) in relation to the motor rotor (if you don't know the value, you can find it out via the menu point 'perform motor sensor align')
  - zero electric offset: which mechanical offset (degree in radiant) the hall sensor has in relation to the motor rotor.
- To find out any missing parameters of your motor (or to verify they are correct), choose 'Perform motor sensor align'. The motor will now rotate slightly for a few seconds. The output will be as shown:



# owlDrive - FOC Brushless Motor Controller

```
/dev/ttyACM0
10. Perform motor sensor align (auto-align)
user input: 10
MOT: Align sensor.
MOT mid_angle (deg): 73.48
MOT end_angle (deg): 24.79
MOT moved (deg): 48.69
MOT: sensor_direction==CW
MOT: PP check: OK!
MOT: Zero elec. angle: 2.42
MOT: Align current sense.
MOT: Success: 1
MOT: Ready.
motor menu:
Autoscroll Show timestamp Both NL & CR 115200 baud Clear output
```

If a wrong number of rotor pairs (pole pairs) was detected, an estimation will be shown (and you have to round up that estimation to an integer number). Any found zero electric offset and sensor direction will be automatically changed in the configuration.

10. Now it's time to test the new settings. In the main menu, choose 'Save to EEPROM and exit' to permanently save the configuration.

11. Send 'M5' to the driver (which means the target voltage should be 5 volts).

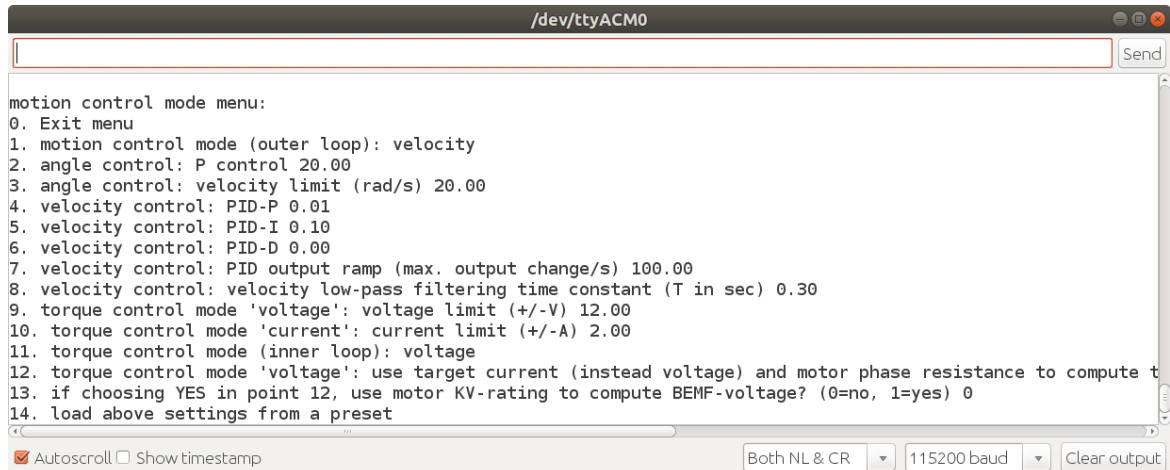
```
/dev/ttyACM0
M5
Target: 5.000
canId 301 sec 26 lps 59986 focps 6059 pwnCnt 0 tg(torque->voltage) 5.00 v 25.0(239rpm) BEMF 0.00 Uq 5.0
canId 301 sec 27 lps 60011 focps 6065 pwnCnt 0 tg(torque->voltage) 5.00 v 37.2(355rpm) BEMF 0.00 Uq 5.0
canId 301 sec 28 lps 59999 focps 6067 pwnCnt 0 tg(torque->voltage) 5.00 v 43.4(415rpm) BEMF 0.00 Uq 5.0
canId 301 sec 29 lps 60025 focps 6066 pwnCnt 0 tg(torque->voltage) 5.00 v 46.6(445rpm) BEMF 0.00 Uq 5.0
canId 301 sec 30 lps 60035 focps 6068 pwnCnt 0 tg(torque->voltage) 5.00 v 49.7(474rpm) BEMF 0.00 Uq 5.0
canId 301 sec 31 lps 60014 focps 6067 pwnCnt 0 tg(torque->voltage) 5.00 v 51.2(489rpm) BEMF 0.00 Uq 5.0
canId 301 sec 32 lps 60009 focps 6069 pwnCnt 0 tg(torque->voltage) 5.00 v 51.1(488rpm) BEMF 0.00 Uq 5.0
canId 301 sec 33 lps 59999 focps 6066 pwnCnt 0 tg(torque->voltage) 5.00 v 52.2(499rpm) BEMF 0.00 Uq 5.0
canId 301 sec 34 lps 60003 focps 6067 pwnCnt 0 tg(torque->voltage) 5.00 v 53.0(506rpm) BEMF 0.00 Uq 5.0
canId 301 sec 35 lps 60012 focps 6069 pwnCnt 0 tg(torque->voltage) 5.00 v 51.5(492rpm) BEMF 0.00 Uq 5.0
canId 301 sec 36 lps 60001 focps 6069 pwnCnt 0 tg(torque->voltage) 5.00 v 53.0(507rpm) BEMF 0.00 Uq 5.0
canId 301 sec 37 lps 59992 focps 6067 pwnCnt 0 tg(torque->voltage) 5.00 v 53.1(507rpm) BEMF 0.00 Uq 5.0
Autoscroll Show timestamp Both NL & CR 115200 baud Clear output
```

The motor should start turning. You can monitor certain motor values in realtime (target, velocity in rpm/radians per second, current, angle in radiant etc.).

## Example application: motor velocity control

In this application the velocity of the motor shaft is controlled. It is assumed that your motor is running properly via torque control mode (as used the section above).

1. Enter the main menu, and choose 'Motion control mode menu'.



```
/dev/ttyACM0
motion control mode menu:
0. Exit menu
1. motion control mode (outer loop): velocity
2. angle control: P control 20.00
3. angle control: velocity limit (rad/s) 20.00
4. velocity control: PID-P 0.01
5. velocity control: PID-I 0.10
6. velocity control: PID-D 0.00
7. velocity control: PID output ramp (max. output change/s) 100.00
8. velocity control: velocity low-pass filtering time constant (T in sec) 0.30
9. torque control mode 'voltage': voltage limit (+/-V) 12.00
10. torque control mode 'current': current limit (+/-A) 2.00
11. torque control mode (inner loop): voltage
12. torque control mode 'voltage': use target current (instead voltage) and motor phase resistance to compute t
13. if choosing YES in point 12, use motor KV-rating to compute BEMF-voltage? (0=no, 1=yes) 0
14. load above settings from a preset

Autoscroll Show timestamp Both NL & CR 115200 baud Clear output
```

2. Choose motor control mode (outer loop): velocity.
3. Configure the velocity control via the 'PID' parameters.
  - PID-P: The higher the P-value (gain), the faster the controller tries to reach the velocity target (set-point).
  - PID-I: The higher the I-value (integration), the more the controller accumulates for control errors (set-point to current-point velocity differences).
  - PID-D: keep this to zero (0).
  - PID output ramp: allows you to restrict acceleration

**NOTE:** You can get example preset values via the menu point 'load above settings from a preset'.

4. If using a hall sensor, make sure to use a low-pass filter for the sensor. Choose 'velocity low-pass filtering time constant' and the value '0.3'. For magnetic encoders, choose low-pass filtering value '0'.
5. In the main menu, choose 'Save to EEPROM and exit'.
6. Send 'M10' to the driver (which means the velocity target should be 10 radians per second).

The motor should start turning. You can monitor certain motor values in realtime (target, velocity in rpm/radians per second, current, angle in radiant etc.).

```

/dev/ttyACM0
M10
canId 301 sec 425 lps 59919 focps 6036 pwnCnt 0 tg(velocity->voltage) 10.00 v 10.1(96rpm) BEMF 0.00 Uq
canId 301 sec 426 lps 59910 focps 6031 pwnCnt 0 tg(velocity->voltage) 10.00 v 10.0(96rpm) BEMF 0.00 Uq
canId 301 sec 427 lps 59901 focps 6029 pwnCnt 0 tg(velocity->voltage) 10.00 v 10.1(96rpm) BEMF 0.00 Uq
canId 301 sec 428 lps 59876 focps 6029 pwnCnt 0 tg(velocity->voltage) 10.00 v 10.0(95rpm) BEMF 0.00 Uq
canId 301 sec 429 lps 59916 focps 6024 pwnCnt 0 tg(velocity->voltage) 10.00 v 10.0(95rpm) BEMF 0.00 Uq
canId 301 sec 430 lps 59883 focps 6034 pwnCnt 0 tg(velocity->voltage) 10.00 v 10.1(97rpm) BEMF 0.00 Uq
canId 301 sec 431 lps 59901 focps 6031 pwnCnt 0 tg(velocity->voltage) 10.00 v 10.1(96rpm) BEMF 0.00 Uq
canId 301 sec 432 lps 59912 focps 6027 pwnCnt 0 tg(velocity->voltage) 10.00 v 10.0(96rpm) BEMF 0.00 Uq
canId 301 sec 433 lps 59902 focps 6032 pwnCnt 0 tg(velocity->voltage) 10.00 v 9.9(95rpm) BEMF 0.00 Uq
canId 301 sec 434 lps 59875 focps 6023 pwnCnt 0 tg(velocity->voltage) 10.00 v 9.8(94rpm) BEMF 0.00 Uq
canId 301 sec 435 lps 59881 focps 6032 pwnCnt 0 tg(velocity->voltage) 10.00 v 10.0(95rpm) BEMF 0.00 Uq
canId 301 sec 436 lps 59872 focps 6026 pwnCnt 0 tg(velocity->voltage) 10.00 v 10.0(95rpm) BEMF 0.00 Uq
canId 301 sec 437 lps 59898 focps 6033 pwnCnt 0 tg(velocity->voltage) 10.00 v 10.1(97rpm) BEMF 0.00 Uq
Autoscroll Show timestamp Both NL & CR 115200 baud Clear output

```

## Open-loop velocity control

If you cannot get your motor to run with both torque and velocity control, you might have issues with your feedback sensor (Hall etc.). In this case it is recommended to try out the open-loop velocity control.

1. Enter the main menu, and choose 'Motion control mode menu'.

```

/dev/ttyACM1
motion control mode menu:
0. Exit menu
1. motion control mode (outer loop): velocity_openloop
2. angle control: P control 20.00
3. angle control: velocity limit (rad/s) 20.00
4. velocity control: PID-P 0.01
5. velocity control: PID-I 0.10
6. velocity control: PID-D 0.00
7. velocity control: PID output ramp (max. output change/s) 100.00
8. velocity control: velocity low-pass filtering time constant (T in sec) 0.30
9. torque control mode 'voltage': voltage limit (+/-V) 3.00
10. torque control mode 'current': current limit (+/-A) 50.00
11. torque control mode (inner loop): voltage
12. torque control mode 'voltage': use target current (instead voltage) and motor phase resistance to compute target voltage? (0=no, 1=yes) 0
13. if choosing YES in point 12, use motor KV-rating to compute BEMF-voltage? (0=no, 1=yes) 0
14. load above settings from a preset
Autoscroll Show timestamp Both NL & CR 115200 baud Clear output

```

2. Choose motor control mode (outer loop): velocity\_openloop.
3. Choose 'torque control mode voltage limit' and the value '3.0' to limit the voltage.
4. In the main menu, choose 'Save to EEPROM and exit'.
5. Send 'M10' to the driver (which means the velocity target should be 10 radians per second).

## Example application: position control

In this application the position of the motor shaft is controlled. It is assumed that your motor is running properly via torque and velocity control mode (as used the sections above).

1. Enter the main menu, and choose 'Motion control mode menu'.
2. Choose motor control mode (outer loop): angle.
3. Configure the velocity control via the 'PID' parameters (see section above for more details).
4. Configure the angle control via the P control value. The higher the angle P value, the faster the controller tries to reach the target angle.
5. Configure the velocity limit (radian per second). Allows you to limit the velocity for the controller to reach the target angle.
6. In the main menu, choose 'Save to EEPROM and exit'.
7. Send 'M31.4' to the driver (which means the target angle should be 5 turns ( $2 * \pi * 5 = 31.4$  radians)).

```
/dev/ttyACM0
M31.4
Target: 31.400
canId 301 sec 7 lps 59725 focps 6022 pwnCnt 0 tg(angle->voltage) 31.40 v 23.4(224rpm) BEMF 0.00 Uq -12.00 Ud 0.00 Iq -0.91 Id -0.11 ang 28.52 EN 0
canId 301 sec 8 lps 59953 focps 6054 pwnCnt 0 tg(angle->voltage) 31.40 v 0.0(0rpm) BEMF 0.00 Uq 2.37 Ud 0.00 Iq 0.03 Id 0.02 ang 31.30 EN 0 DIR
canId 301 sec 9 lps 59929 focps 6023 pwnCnt 0 tg(angle->voltage) 31.40 v 0.0(0rpm) BEMF 0.00 Uq 0.07 Ud 0.00 Iq -0.02 Id 0.01 ang 31.41 EN 0 DIF
canId 301 sec 10 lps 59970 focps 6029 pwnCnt 0 tg(angle->voltage) 31.40 v 0.0(0rpm) BEMF 0.00 Uq -0.40 Ud 0.00 Iq -0.02 Id 0.01 ang 31.41 EN 0
canId 301 sec 11 lps 59923 focps 6028 pwnCnt 0 tg(angle->voltage) 31.40 v 0.0(0rpm) BEMF 0.00 Uq -0.71 Ud 0.00 Iq -0.02 Id 0.00 ang 31.40 EN 0
canId 301 sec 12 lps 59889 focps 6021 pwnCnt 0 tg(angle->voltage) 31.40 v 0.2(2rpm) BEMF 0.00 Uq -1.85 Ud 0.00 Iq -0.03 Id 0.03 ang 31.40 EN 0
canId 301 sec 13 lps 59921 focps 6023 pwnCnt 0 tg(angle->voltage) 31.40 v 0.2(2rpm) BEMF 0.00 Uq -1.81 Ud 0.00 Iq -0.02 Id 0.03 ang 31.40 EN 0
canId 301 sec 14 lps 59922 focps 6026 pwnCnt 0 tg(angle->voltage) 31.40 v 0.0(0rpm) BEMF 0.00 Uq -0.89 Ud 0.00 Iq -0.03 Id 0.00 ang 31.40 EN 0
canId 301 sec 15 lps 59932 focps 6027 pwnCnt 0 tg(angle->voltage) 31.40 v -0.2(-2rpm) BEMF 0.00 Uq 0.12 Ud 0.00 Iq -0.01 Id 0.01 ang 31.40 EN 0
canId 301 sec 16 lps 59922 focps 6027 pwnCnt 0 tg(angle->voltage) 31.40 v 0.0(0rpm) BEMF 0.00 Uq -0.74 Ud 0.00 Iq -0.03 Id 0.00 ang 31.40 EN 0
canId 301 sec 17 lps 59951 focps 6026 pwnCnt 0 tg(angle->voltage) 31.40 v 0.2(2rpm) BEMF 0.00 Uq -1.69 Ud 0.00 Iq -0.03 Id 0.02 ang 31.40 EN 0
canId 301 sec 18 lps 59925 focps 6024 pwnCnt 0 tg(angle->voltage) 31.40 v -0.2(-2rpm) BEMF 0.00 Uq -0.69 Ud 0.00 Iq -0.01 Id 0.01 ang 31.40 EN 0
canId 301 sec 19 lps 59926 focps 6022 pwnCnt 0 tg(angle->voltage) 31.40 v -0.2(-2rpm) BEMF 0.00 Uq 0.23 Ud 0.00 Iq -0.01 Id 0.01 ang 31.40 EN 0
canId 301 sec 20 lps 59953 focps 6025 pwnCnt 0 tg(angle->voltage) 31.40 v 0.0(0rpm) BEMF 0.00 Uq -0.74 Ud 0.00 Iq -0.02 Id 0.00 ang 31.40 EN 0
canId 301 sec 21 lps 59940 focps 6026 pwnCnt 0 tg(angle->voltage) 31.40 v 0.0(0rpm) BEMF 0.00 Uq -0.74 Ud 0.00 Iq -0.03 Id 0.02 ang 31.40 EN 0
```

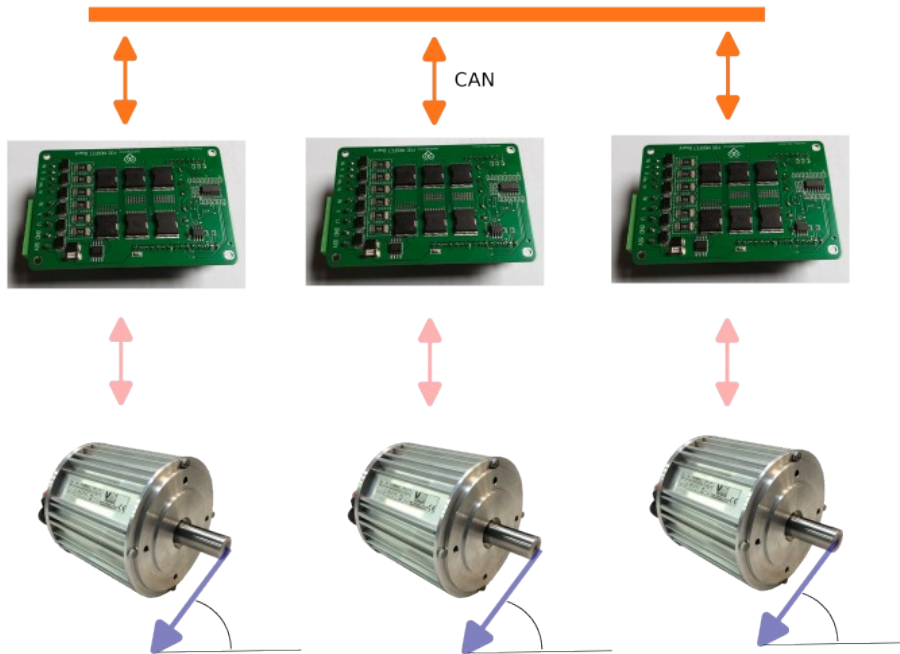
The motor should start turning. You can monitor certain motor values in realtime (target, velocity in rpm/radians per second, current, angle in radian etc.).



## Example application: motor angle synchronization

In this application two additional motors are synchronized to the first motor. The first motor (master) will broadcast its shaft angle and the remaining motors (slave) will follow this shaft angle (they will use it as their target angle).

example application (motor angle synchronization):



It is assumed that your motor is running properly via torque, velocity and angle control mode (as used the sections above) and the motion control mode is configured for angle (see section above).

1. Enter the main menu, and choose 'CAN menu'

```
/dev/ttyACM0
user input: 9
CAN menu - (configure CAN and send commands to other motor driver modules):
0. Exit menu
1. local CAN bus ID: 301
2. value broadcast mask (0=off, 1=target, 2=voltage, 4=current, 8=velocity, 16=angle, 32=motion_ctl_mode): 16
3. follow remote CAN bus ID (remote CAN bus ID that broadcasts a value, 0=off): 300
4. follow remote value (local target is defined by this remote motor driver broadcast value: 0=target, 1=voltage, 2=current, 3=velocity, 4=angle,
5. remote CAN ID (for next remote commands, use remote CAN bus ID): 300
6. send target to remote
7. request value from remote
8. send local motion control params to remote (NOTE: remote driver must use same firmware ver)
9. send complete local config to remote (NOTE: remote driver must use same firmware ver)
```

2. Choose 'local CAN ID' and choose a unique ID for each driver:  
1<sup>st</sup> motor (master): 300  
2<sup>nd</sup> motor (slave): 301  
3<sup>rd</sup> motor (slave): 302
3. For the master motor, choose 'value broadcast mask (angle)', so the driver broadcasts the shaft angle.

4. For the slave motors, choose 'follow remote CAN bus ID (300)', so the slave motors will follow the master motor.
5. For the slave motors, choose 'follow remote value (angle)', so the slave motors will use the master shaft angle as their angle target.
6. In the main menu, choose 'Save to EEPROM and exit'.
7. Send 'M31.4' to the master motor driver (which means the target angle should be 31.4 radians). The master motor should start turning. The slave motors should start turning as well and follow the shaft angle of the master motor.

## FOC commander protocol (UART/USB)

### general commands

MC0	activate torque control
MC1	activate velocity control
MC2	activate angle control
ME0	enable motor status
ME1	disable motor status
MVP, MAP, etc.	get a specific setting (velocity PID P, angle PID P in this example)

### torque control examples

M5	set target 5 volt
----	-------------------

### velocity control examples

M20	set velocity target 20 rad/s
M20 12	set velocity target 20 rad/s with torque limit 12 volt
MVP1	set velocity PID P (proportional gain) to 1
MVI1	set velocity PID I (integral gain) to 1
MVD1	set velocity PID D (derivative gain) to 1
MVR100	set velocity PID output ramp (maximum speed of change of the output value) to 100
MVL100	set velocity PID limit (maximum output value) to 100
MVF0.1	set velocity low-pass filtering time constant to 0.1

### angle control examples

M3.14 20	set angle target 3.14 rad with velocity limit 20 rad/s
M3.14 12 20	set angle target 3.14 rad with torque limit 12 volt and velocity limit 20 rad/s
MAP1	set angle PID P (proportional gain) to 1
MAL100	set angle PID limit (maximum output value) to 100
MAR10000	set angle PID output ramp to 10000

**monitoring** examples ([https://docs.simplefoc.com/commander\\_motor](https://docs.simplefoc.com/commander_motor))

MMG / MMG0	get variable - target
MMG1	get variable – voltage q
MMG2	get variable - voltage d
MMG3	get variable – current q
MMG4	get variable – current d
MMG5	get variable – velocity
MMG6	get variable - angle
MMG7	get all variable

**monitoring output formater** examples

@	get monitor output mode
@1	set on user request no add. output
@2	set user friendly mode
@3	get machine_readable mode

The full FOC commander protocol is described here:

<https://docs.simplefoc.com/communication>

A graphical user interface (SimpleFOCStudio) is available for this protocol (to use it, deactivate the serial monitor output in the profile menu): <https://docs.simplefoc.com/studio>

A web interface (simplefoc-webcontroller) is available for this protocol (to use it, deactivate the serial monitor output in the profile menu):

<https://github.com/geekuillaume/simplefoc-webcontroller>

## CAN protocol

In this section the CAN protocol is described.

The Controller Area Network protocol (CAN or CAN Bus) is a two-wire (twisted-pair), bidirectional serial bus communication method that allows electronic subsystems to be linked together and interact in a network.

Please inform yourself about the basics of CAN bus wiring and use appropriate CAN twisted pair wire.



The termination of the CAN bus is possible with an external 120 Ohm resistor at the beginning and the end of the bus or use the possibility of the solder bridge on the first and last owlDrive to activate the terminating resistors on board.

CAN frame protocol format

CAN address	CAN byte 0-1	CAN byte 2	CAN byte 3	CAN byte 4-7
configured message ID	source/destination	command	value	data

message ID: all motor drivers (nodes) that communicate with each other should use the same message ID.

Source/destination:

bits 0..5 – source node ID (valid IDs are 1-62)

bits 6..11 – destination node ID (valid IDs are 1-62 , 63 means all nodes)

Command types:

0 – driver value information

1 – request driver to send value information

2 – set driver value

3 – save driver value

Value types:

0 – target (data type: float)

1 – voltage (data type: float)

2 – current (data type: float)

3 – velocity (data type: float)

4 – angle (data type: float)

5 – motion control mode (data type: 1 byte)

6 – config memory (data type: ofs\_val)

7 – motor enable (data type: 1 byte)

8 – angle P controller (data type: float)

9 – max. velocity control of the position control (data type: float)

10 – velocity P (data type: float)

11 – velocity I (data type: float)

12 – velocity D (data type: float)

13 – velocity output ramp (max. output change/s) (data type: float)

14 – velocity low-pass filtering time constant (sec) (data type: float)

15 – error status (data type: byte)

# owlDrive - FOC Brushless Motor Controller

Data types:

<b>1-4 bytes</b>	byte 0	byte 1	byte 2	byte 3
<b>1 integer</b>	integer value (little endian)			
<b>float</b>	float value (little endian)			

<b>ofs_val</b>	short ofs (little endian)	byte value	-/-
----------------	---------------------------	------------	-----

Error status constants:

- 0 – no error
- 1 – no CAN communication
- 2 – no settings
- 3 – undervoltage triggered
- 4 – overvoltage triggered
- 5 – overcurrent triggered
- 6 – over-temperature triggered

Examples:

A) request driver (with nodeID=1) to send target value

Destination	command	value	data
1	1	0	not used

B) driver (with nodeID=1) sends target value

Source	command	value	data (4 byte float)
1	0	0	3.14

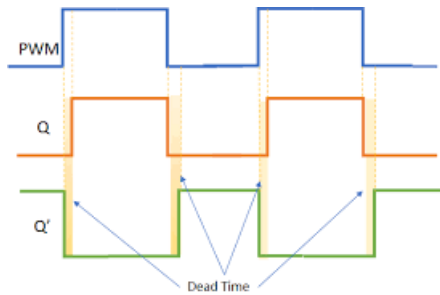
C) set driver (with nodeID=1) target value

Destinaton	command	value	data (4 byte float)
1	2	0	3.14

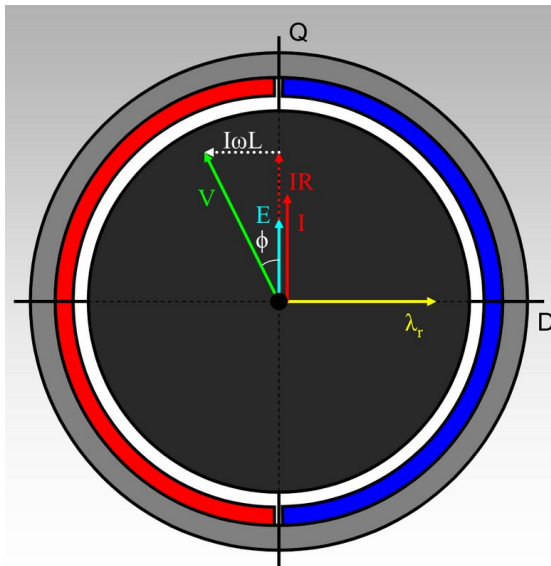
## Motor parameters

This section describes the motor parameters.

**Deadtime:** The deadtime is the time where neither the High-side nor the Low-side of the a MOSFET coil is switched, so the coil is in high-impedance during that time. A higher deadtime will reduce motor heat however the supply voltage noise for the owlDrive may increase due to motor's back-electrical-force (EMF) and can lead to a reset of the owlDrive.



**Phase inductance:** At higher speeds, there is a lag between motor voltage and current due to the motor inductance. Setting the correct phase inductance will phase-shift the voltage to compensate this lag. As a consequence the motor achieves higher velocities.





## BLDC motor formulas

1. Motor Kv rating, rotation speed (RPM) and back EMF (volt):

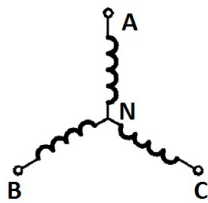
Kv: RPM/V      unloaded motor speed per Volt (peak)

$$\omega_{rpm} = K_v \cdot \text{Back EMF}$$

where  $\omega_{rpm}$  = rotation speed

2. Motor resistance and motor inductance

Usually BLDC motors are connected in WYE (Star) configuration. The other type of configuration is Delta connected which is very rare.



In WYE (Star) configuration, the phase windings are connected as shown in the above diagram. The terminals A, B, and C are generally accessible. Most motor manufacturers do not provide Neutral Point access. So, machine parameters (resistance, inductance here) can only

be measured across A-B, B-C, and C-A, which are line to line and these parameters are called line values (line resistance, line inductance). And what we engineers use for motor control is phase values (phase resistance, phase inductance). Here, phase values are nothing but half the line values.

Resistance:

You can use an Ohm Meter or a multimeter. Measure resistance across A-B, B-C, and C-A. Being a balanced system, they should result the same.

$$R_{phase} = \frac{R_{line - to - line}}{2}$$

What if you don't have an Ohm Meter handy?

Pass some known current through the phase winding (say A-B), measure the

voltage drop across the two terminals. Now, Voltage Drop over Current gives you the resistance of the winding (A-B).

Note: Make sure you start with lower values of currents as you wouldn't want to burn the motor winding by passing larger currents.

Inductance:

Connect a LCR meter on each phase. Start with A-B. Slowly turn the rotor for one

revolution, in small increments (say 30 degrees) and record the Inductance value.

Repeat the same for B-C and C-A. Tabulate the readings and observe carefully the values of each phase. If the values are found to be similar, you can average them out and use the averaged value as Line-to-Line inductance, half of which is Phase Inductance.

Now, what if you don't have a LCR meter.

Have a low voltage ac source handy. Supply the ac voltage to one pair of the three wires (say A-B). Measure the current consumed and work out the following equations.

$$\text{impedance} = \frac{\text{Voltage}}{\text{Current}}$$

$$\sqrt{\text{reactance}^2 + \text{resistance}^2} = \text{impedance}$$

$$\sqrt{\text{impedance}^2 - \text{resistance}^2} = \text{reactance}$$

$$\text{reactance} = 2 * \pi * \text{frequency} * \text{inductance}$$

Since line parameters are being calculated, the resistance that need to be fed in the above equations is line-to-line and the frequency component is the frequency of the ac voltage used.

### 3. Motor torque, motor torque coefficient

Kv: RPM/V

Kt: Motor torque coefficient =  $60 / (2\pi * Kv)$

current: amps

Torque: Nm

Torque [Nm] =  $60 / (2\pi * Kv) * \text{current}$

Further reading

1. Brushless motors animations

<https://fab.cba.mit.edu/classes/865.18/motion/brushless/index.html>

2. How to calculate motor poles & motor KV

<https://www.tytorobotics.com/blogs/articles/how-to-calculate-motor-poles-and-brushless-motor-kv>

- 3.

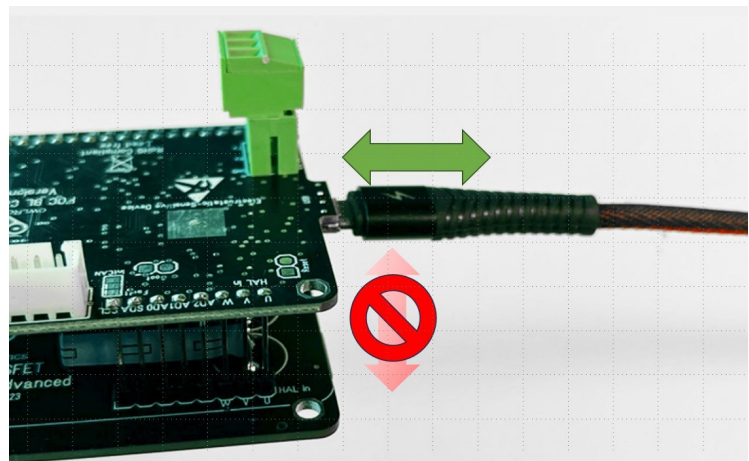
## Appendix A

### Handling the owlDrive.

#### USB plug in and out

Plug in the USB plug into the owlDrive USB connector as shown in the picture (green arrow).

If the USB plug is connected, do **not** bend the plug in any other direction. This can damage the USB connector on the owlDrive board.



#### Attention:

owlDrive 250-24 power should be 16-36V DC. The power supply must cover the max. motor current and the rated motor voltage.

Use an **external fuse** to protect motor and motor driver. The fuse value must fit to the motor peak current.

#### Attention:

**Do not use the owlDrive in applications where malfunctions of the owlDrive (e.g. due to problems in voltage, signal quality or other causes) can cause damage to property or personal injury. Or secure the use of the owlDrive with additional backup / security measures. Since malfunctions cannot be ruled out, owlRobotics GmbH cannot assume any liability for the application!**