

# *Whitepaper*

## Mit VPN Fernzugriff auf Dunkermotoren realisieren



Markus Weishaar | Produktmanager IIoT  
Dunkermotoren GmbH

## Das vorliegende Whitepaper beschreibt die Konfiguration einer VPN Verbindung für den Fernzugriff auf einen Dunkermotoren dPro Ethernet Motor über das Internet mit der Dunkermotoren Standardsoftware „Drive Assistant“ und der Open-Source-Software OpenVPN.

Hierzu wird ein Linux-basiertes Edge-Gateway als VPN Server konfiguriert. Das Edge-Gateway kommuniziert über 2 gebridgte Ports via Ethernet sowohl mit dem Motor als auch mit einem Router, der die Internetanbindung übernimmt. Auf der anderen Seite steht ein handelsüblicher Windows PC auf dem „Drive Assistant“ und openVPN installiert sind. Auf dem PC wird OpenVPN als Client eingerichtet, der über das Internet eine VPN-Verbindung zum VPN-Server auf dem Gateway aufbaut. Über diese Verbindung kann der Motor über „Drive Assistant“ ausgelesen und angesteuert werden oder auch ein Firmware-Update durchgeführt werden. Verfügt der Motor über eine bekannte statische IP-Adresse kann die VPN-Verbindung als Tunnel konfiguriert werden, da die Verknüpfung zweier Subnetze über Routing ausreichend ist. Verfügt der Motor über keine oder keine bekannte IP-Adresse muss die VPN-Verbindung als Brücke aufgebaut werden, die den Client in das gleiche Subnetz zieht, in dem sich auch der Server befindet. Das ist nötig da der „Drive Assistant“ Broadcasts zur Antriebssuche verwendet, und Broadcasts nur im selben Subnetz funktionieren.

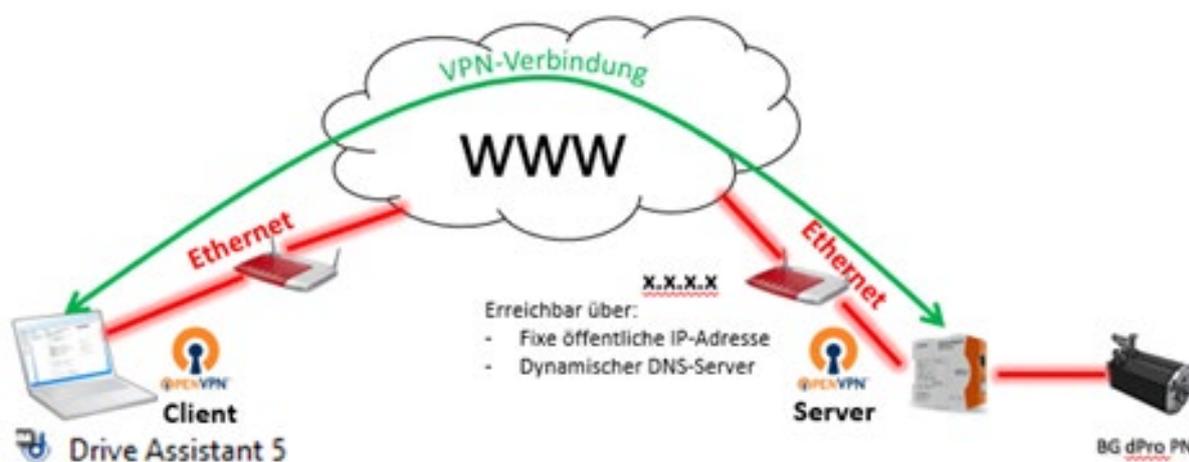


Abbildung 1: Übersicht VPN Netzwerke

## Inhalt:

<b>1</b>	<b>Voraussetzungen/ Vergleichskonfiguration.....</b>	<b>4</b>
<b>2</b>	<b>Konfiguration OpenVPN Server (Raspberry Pi/ Linux).....</b>	<b>4</b>
2.1	Installation OpenVPN.....	4
	Schritt 1 Raspberry updaten und OpenVPN installieren	
2.2	Ethernet-Einstellungen.....	4
2.2.1	VPN Tunnel (TUN).....	5
2.2.2	VPN Brücke (TAP).....	6
2.3	Zertifikate und Schlüssel erstellen.....	8
2.4	Konfiguration OpenVPN Server.....	9
2.4.1	VPN Tunnel (TUN).....	9
2.4.2	VPN Brücke (TAP).....	11
2.5	Konfiguration Linux-Firewall.....	12
2.5.1	VPN Tunnel (TUN).....	13
2.5.2	VPN Brücke (TAP).....	14
2.5.3	Init-File aktivieren.....	14
2.5.4	IP Forwarding statisch aktivieren.....	14
2.6	Konfiguration OpenVPN Client.....	15
2.6.1	VPN Tunnel (TUN).....	16
2.6.2	VPN Brücke (TAP).....	16
2.7	Erzeugung und Export Konfigurations-Files für Clients.....	16
<b>3</b>	<b>Konfiguration OpenVPN Client (Windows).....</b>	<b>17</b>
3.1	Installation OpenVPN.....	17
3.2	Konfiguration OpenVPN Client.....	17
3.3	Konfiguration TAP-Windows-Adapter V9.....	18
<b>4</b>	<b>Allgemeine Netzwerkeinstellungen &amp; Verbindungsaufbau.....</b>	<b>18</b>
4.1	Portweiterleitung auf Routern aktivieren.....	18
4.2	Einrichtung dynamischer DNS Server.....	18
4.3	VPN Verbindung aufbauen und testen.....	19
<b>5</b>	<b>Drive Assistant.....</b>	<b>19</b>

## 1 Voraussetzungen/ Vergleichskonfiguration:

- » Dunkermotoren „Drive Assistant 5“ Version 8.0.0
- » Dunkermotoren BG XX dPro PN (Ethernet)
- » openVPN Version 2.4.7
- » Hardware Gateway: Kunbus Revolution Pi Connect (Raspberry Pi Compute Module 3)
- » Betriebssystem VPN-Server: Raspbian (Linux)
- » Betriebssystem VPN-Client: Windows 10
- » Statische öffentliche IP-Adresse oder dynamischer DNS Server für serverseitigen Router
- » Rechte zur Konfiguration des serverseitigen Routers (Portweiterleitung)
- » Rechte zur Konfiguration der Firewall des openVPN Servers

## 2 Konfiguration Open VPN Server (Raspberry Pi/ Linux)

### 2.1 Installation Open VP

*Schritt 1 Raspberry updaten und Open VPN installieren*

Vor der Installation von OpenVPN wird empfohlen nach Updates für das Betriebssystem des des Raspberry Pi zu suchen und diese auszuführen:

Nun muss die Software OpenVPN und für die Verschlüsselung OpenSSL mit dem folgenden Befehl geladen und installiert werden:

```
sudo apt-get update  
sudo apt-get upgrade
```

```
sudo apt-get install openvpn openssl
```

### 2.2 Ethernet Einstellungen

Um ein Routing zwischen den beiden Ethernet-Ports des Raspberry Pis zu verzichten und trotzdem über die VPN-Verbindung an eth0 auf den Motor an eth1 zugreifen zu können werden in diesem Beispiel die beiden Ports gebridget und mit einer gemeinsamen Adresse versehen.

Alternativ kann auch nur mit einem Port gearbeitet werden und dieser mit einer fixen IP-Adresse versehen werden. Der Motor kann und die VPN-Verbindung können dann mittels eines Switchs an diesen Port angeschlossen werden. Auf dieses Szenario wird aber nicht weiter eingegangen.

Um die Ethernet-Einstellungen des Raspberry Pis zu konfigurieren muss die Datei „interfaces“ wie folgt geöffnet und entsprechend der folgenden Kapitel angepasst werden:

```
sudo nano /etc/network/interfaces
```

Der virtuelle **Loopback-Adapter** ist per Default immer eingetragen und sollte auch immer in der Konfiguration belassen werden:

```
auto lo  
iface lo inet loopback
```

Nun werden die vorhandenen Netzwerkschnittstellen angelegt. Da unser Gateway über zwei getrennte Ethernet-Ports verfügt, werden die beiden Schnittstellen **eth0** und **eth1** angelegt.

Der angehängte Befehl „**allow-hotplug ethX**“ bewirkt, dass die Schnittstelle bei einem Kernel-Event automatisch aktiviert und konfiguriert. Dieser Eintrag ist wichtig, da die Schnittstelle ansonsten über das Kommando „sudo ifup eth0“ manuell gestartet werden muss.

```
auto eth0
allow-hotplug eth0

auto eth1
allow-hotplug eth1
```

Das Konfigurationsfile darf jetzt noch nicht geschlossen werden, da im jetzigen Zustand die Schnittstellen noch über keine Adressen und Konfiguration verfügen und der Raspberry Pi so nicht mehr erreichbar wäre. Die Konfiguration wird nachfolgen Fallspezifisch durchgeführt:

### 2.2.1 VPN Tunnel (TUN)

Zuerst werden die beiden Ethernet-Adapter in den Mode „manual“ versetzt, dies ist wichtig, da sie durch die Brücke konfiguriert werden. Dazu wird bei beiden Adaptern folgende Zeile ergänzt:

```
iface ethX inet manual
```

Danach wird die **Brücke br0** als Adapter angelegt und statisch konfiguriert:

```
auto br0
iface br0 inet static
```

Danach werden die Netzwerksettings für den Adapter eingestellt. Eine Beispielkonfiguration könnte wie folgt aussehen:

IP-Adresse: 192.168.0.200  
Subnetzmaske: 255.255.255.0  
Standard-Gateway: 192.168.0.1  
Netzwerk: 192.168.0.0  
Broadcast: 192.168.0.255

Im Konfigurationsfile sehen die Einträge dazu wie folgt aus:

```
address xxx.xxx.xxx.xxx
netmask xxx.xxx.xxx.xxx
gateway xxx.xxx.xxx.xxx
network xxx.xxx.xxx.xxx
broadcast xxx.xxx.xxx.255
```

Abschließend werden der Brücke über folgende Zeile noch die beiden Schnittstellen hinzugefügt:

```
bridge_ports eth0 eth1
```

Die kompletten vorzunehmenden Einträge der Netzwerkkonfiguration sollte dann so aussehen:

```
auto lo
iface lo inet loopback
auto eth0
allow-hotplug eth0
iface eth0 inet manual
auto eth1
allow-hotplug eth1
iface eth1 inet manual
auto br0
iface br0 inet static
address xxx.xxx.xxx.xxx
netmask xxx.xxx.xxx.xxx
gateway xxx.xxx.xxx.xxx
network xxx.xxx.xxx.xxx
broadcast xxx.xxx.xxx.255
bridge_ports eth0 eth1
```

Mit "Strg + O" kann die Änderung gespeichert und mit „Strg + X“ der Editor geschlossen werden.

### 2.2.2 VPN Brücke (TAP)

Das grundlegende Setting der Ports und der Brücke sind bei dieser Variante identisch wie bei der vorhergehenden Konfiguration. Lediglich die Brücke wird insoweit ergänzt, dass der VPN-Adapter tap0 ebenfalls zur Brücke hinzugefügt wird. „**Pre-up**“ Befehle werden hier vor Aufbau der Brücke ausgeführt und „**post-up**“ Befehle werden unmittelbar nach erstellen der Bridge ausgeführt. Das selbe gilt beim Beenden der Brücke für die Befehle „**pre-down**“ und „**post-down**“.

Als erstes wird der Brücke eine definierte MAC-Adresse vergeben, mit der sich die Brücke im Netzwerk meldet. Das erleichtert zum einen die Diagnose und ermöglichtes zum anderen die MAC-Adresse auf dem Router bekanntzumachen falls auf diesem die MAC-Filterung aktiv ist. Wird dieser Befehl weg gelassen bekommt die Brücke im besten Fall die MAC-Adresse von eth0 aber im schlechten Fall irgendeine MAC-Adresse.

```
post-up ip link set br0 address 28:2B:1b:e1:55:2F
```

Mit den nächsten Befehlen wird zuerst vor Aufbau der Brücke OpenVPN aufgefordert ein virtuelles Netzwerk Decive tap0 zu erstellen und dieses wird dann nach Aufbau der Brücke selbiger hinzugefügt.

```
pre-up openvpn --mktun --dev tap0
post-up brctl addif br0 tap0
```

Im Anschluss werden für die zur Brücke befindlichen Schnittstellen über einen kombinierten Befehl erst vergebene IP-Adressen gelöscht und die Schnittstellen dann in den „promiscuous Mode“ versetzt, damit die Brücke den gesamten an diesen Schnittstellen ankommenden Datenverkehr sieht. Zusätzlich wird mit einem weiteren Befehl noch eine fixe Route zum Standardgateway für die Brücke hinzugefügt, über die das Internet erreicht wird.

```
post-up ifconfig tap0 0.0.0.0 promisc up
post-up ifconfig eth0 0.0.0.0 promisc up
post-up ifconfig eth1 0.0.0.0 promisc up
post-up route add default gw xxx.xxx.xxx.xxx br0
```

Abschließend folgen dann noch zwei Befehlszeilen, die beim Beenden der Brücke den virtuellen Netzwerkadapter wieder aus der Brücke entfernen und OpenVPN auffordern den Adapter aufzulösen.

```
pre-down brctl delif br0 tap0
post-down openvpn --rmtun --dev tap0
```

Die vollständige Netzwerkkonfiguration sollte dann wie folgt aussehen:

```
auto lo
iface lo inet loopback
auto eth0
allow-hotplug eth0
iface eth0 inet manual
auto eth1
allow-hotplug eth1
iface eth1 inet manual
auto br0
iface br0 inet static
address xxx.xxx.xxx.xxx
netmask xxx.xxx.xxx.xxx
gateway xxx.xxx.xxx.xxx
network xxx.xxx.xxx.xxx
broadcast xxx.xxx.xxx.xxx
bridge_ports eth0 eth1
post-up ip link set br0 address 28:2B:1b:e1:55:2F
pre-up openvpn --mktun --dev tap0
post-up brctl addif br0 tap0
post-up ifconfig tap0 0.0.0.0 promisc up
post-up ifconfig eth0 0.0.0.0 promisc up
post-up ifconfig eth1 0.0.0.0 promisc up
post-up route add default gw xxx.xxx.xxx.xxx br0
pre-down brctl delif br0 tap0
post-down openvpn --rmtun --dev tap0
```

Mit „Strg + O“ kann die Änderung gespeichert und mit „Strg + X“ der Editor geschlossen werden.

*Alternativ kann der Aufbau und die Konfiguration der Brücke auch über Skripte realisiert werden, die direkt von OpenVPN ausgeführt werden und somit die Netzwerkkonfiguration an sich schmal und unabhängig gehalten werden kann. Diese Variante wird hier aber nicht detailliert betrachtet.*

### 2.3 Zertifikate und Schlüssel erstellen

Bei der in diesem Beispiel verwendet Verschlüsselung handelt es sich um eine Beispielkonfiguration um schnell eine funktionierende VPN-Verbindung zu erstellen. Es wird auch darauf verzichtet VPN-Clients mit Passwörtern zu versehen. Für den konkreten realen Anwendungsfall, der über einen Verbindungstest hinausgeht wird empfohlen eine dafür geeignete Verschlüsselung auszuwählen und zu konfigurieren um die gewünschten Sicherheitslevels zu erreichen und zu gewährleisten.

Als erstes werden die vorgefertigten „easy-rsa“ Skripte in das Konfigurationsverzeichnis von OpenVPN kopiert. Dadurch werden verschiedene Zertifikate und Schlüssel angelegt.

```
sudo cp -r /usr/share/easy-rsa /etc/openvpn/easy-rsa
```

Als nächstes muss die Datei „vars“ im erstellten Verzeichnis geöffnet und angepasst werden:

```
sudo nano /etc/openvpn/easy-rsa/vars
```

In der Datei muss die Zeile „**export EASY\_RSA=“pwd”“**“ durch die Zeile „**export EASY\_RSA=“/etc/openvpn/easy-rsa”“**“ ersetzt werden. Zudem können sie in der Datei auch die in der Zeile „**export KEY\_SIZE=“**“ durch ändern des Wertes die Schlüssellänge anpassen. Die Schlüssellänge bestimmt das Sicherheitsniveau. Beim eingesetzten Raspberry Pi 3 ist eine Schlüssellänge von **2048** kein Problem, weswegen diese in diesem Beispiel verwendet wird.

Nun muss zurück ins Konfigurationsverzeichnis „easy-rsa“ gewechselt werden, dort Root-Rechte zugewiesen, das Skript „vars“ ausgeführt und die entstehende Konfigurationsdatei über einen symbolischen Link zugänglich gemacht werden. Diese vier Schritte werden durch die folgenden vier Befehle erledigt:

```
cd /etc/openvpn/easy-rsa
sudo su
source vars
ln -s openssl-1.0.0.cnf openssl.cnf
```

Im nächsten Schritt werden die Zertifikate erstellt. Dazu werden zunächst die **Schlüsseldateien von OpenVPN** zurückgesetzt und neu erstellt:

```
./clean-all
./build-ca OpenVPN
```

Es folgt eine Aufforderung den zwei Buchstaben langen „**Country Name**“ einzutragen (DE für Deutschland, AT für Österreich und CH für die Schweiz). Alle weiteren Abfragen können ohne Eingabe durch drücken von Enter übersprungen werden.

Anschließend wird die **Schlüsseldatei für den Server** erstellt, hier muss ebenfalls wieder der „Country Name“ eingetragen und alle weiteren Abfragen übersprungen werden. Am Ende des Dialogs muss die Frage ob das Zertifikat erstellt werden soll noch zwei mal mit „Y“ bestätigt werden.

```
./build-key-server server
```

Als nächstes werden die **Schlüsseldateien für die Clients** erstellt. Hier ist wichtig, dass für jeden Client, der eine Verbindung mit dem VPN-Server aufbauen möchte eine eigene Schlüsseldatei

erstellt werden muss. In unserem Fall beschränken wir uns auf einen Client „**remote-pc-1**“. Der Ablauf bei der Zertifikatserstellung ist analog zum Server (Country-Code, etc.)

```
./build-key remote-pc-1
```

Wenn weitere Clients benötigt werden, werden für diese die Schlüsseldateien nach demselben Muster erstellt:

```
./build-key client_name_xxx  
./build-key client_name_yyy  
./build-key client_name_zzz  
...
```

Sollen die **Clients mit einem Passwort** ausgestattet werden muss statt der oben verwendeten Befehle „./build-key-pass client\_name“ verwendet werden.

Die Erstellung der Schlüssel und Zertifikaten wird nun über den Befehl für den **Diffie-Hellman-Schlüsselaustausch** abgeschlossen. (Dieser Vorgang nimmt ca. 20min in Anspruch)

```
./build-dh
```

Abschließend wird nach Ende der Schlüssel- und Zertifikatserstellung noch der Too-Benutzer abgemeldet:

```
exit
```

## 2.4 Konfiguration OpenVPN Server

Um den OpenVPN Server zu konfigurieren muss die Datei „openvpn.conf“ wie folgt geöffnet und entsprechend der folgenden Kapitel angepasst werden:

```
sudo nano /etc/openvpn/openvpn.conf
```

### 2.4.1 VPN Tunnel (TUN)

Zunächst wird über „**dev tun**“ das Routing über einen Tunnel aktiviert, mittels „**proto udp**“ UDP als Transportprotokoll ausgewählt und mit „**port 1194**“ der Port ausgewählt, über den der Tunnel aufgebaut wird. Als Alternative kann beim Transportprotokoll auch TCP verwendet werden. Der Port kann frei gewählt werden, im Beispiel wird der OpenVPN Standardport 1194 verwendet.

```
dev tun  
proto udp  
port 1194
```

Als Nächstes wird über das Verzeichnis „easy-rsa“ ein SSL/TLS Root-Zertifikat (ca), ein digitales Zertifikat (cert) und ein digitaler Schlüssel (key) erstellt sowie die richtige Bit-Verschlüsselung eingetragen. In diesem Beispiel Diffie-Hellman mit Schlüssellänge 2048.

```
ca /etc/openvpn/easy-rsa/keys/ca.crt  
cert /etc/openvpn/easy-rsa/keys/server.crt  
key /etc/openvpn/easy-rsa/keys/server.key  
dh /etc/openvpn/easy-rsa/keys/dh2048.pem
```

Nun wird dem VPN-Server eine IP-Adresse und eine Subnetzmaske vergeben. Bei dieser Variante erfolgt später ein Routing von diesem virtuellen Netz des VPN-Servers in das physische Netz des Raspberry Pis.

```
server 10.8.0.0 255.255.255.0
```

Über das Kommando „push „**redirect-gateway def1 bypass-dhcp**““ wird der gesamte IP-Verkehr des Servers durch den VPN-Tunnel gelangt, hier hängt es von der Applikation ab, ob dieses Setting Sinn macht oder nicht. Mit den beiden folgenden Kommandos werden die zu verwendenden DNS Server zur Namensauflösung benannt. In unserem Beispiel ist dies ein lokaler DNS-Server des Routers und der öffentliche DNS-Server von Google (8.8.8.8). Diese können aber nach Belieben gewählt werden.

```
push „redirect-gateway def1 bypass-dhcp“  
push „dhcp-option DNS xxx.xxx.xxx.xxx“  
push „dhcp-option DNS 8.8.8.8“
```

Damit Log-Informationen zur Verbindung in der Datei „/var/log/openvpn“ gespeichert werden wird noch die folgende Zeile ergänzt:

```
log-append /var/log/openvpn
```

Nachfolgend folgt noch ein Standardsatz von Befehlen. Der Befehle „**persist-key**“ bewirkt, dass Schlüsseldateien nicht erneut gelesen werden und „**persist-tun**“ sorgt dafür, dass die TUN und TAP Netzwerktreiber nicht neu gestartet werden. Die beiden Befehle „**user nobody**“ und „**group nobody**“ setzen die Rechte von OpenVPN nach einem Programmstart herab und erhöhen so die Sicherheit. Die Zeile „**client-to-client**“ erlaubt die Kommunikation zwischen den Clients und „**status /var/log/openvpn-status.log**“ erstellt eine Statusdatei, die die aktuelle Verbindung dokumentiert. Die Ausführlichkeit der Logs wird über „**verb x**“ definiert. Wert „**0**“ bedeutet keine Ausgaben außer Fehlermeldungen. Ein Wert zwischen 1 und 4 eignet sich für den normalen Gebrauch, wohingegen sich darüber liegende Werte für die Fehlerbehebung eignen. Um die Verbindung zu prüfen wird „**keepalive 10 120**“ ergänzt. Dadurch wird alle 10 Sekunden ein Ping ausgelöst und bei einer fehlenden Antwort nach 120 Sekunden ein Verbindungsunterbruch diagnostiziert. Um die Daten im VPN-Tunnel zu komprimieren und den Durchsatz zu steigern wird über „**comp-lzo**“ noch eine LZO-Komprimierung aktiviert. Der letzte Befehl „**script-security x**“ definiert welche Anwendungen und Skripte durch OpenVPN ausgeführt werden dürfen. Wert „**0**“ bedeutet ein striktes Verbot, externe Anwendungen auszuführen. Wert „**1**“ bedeutet ausschließlich „**built-in**“ Anwendungen wie ifconfig, ip, route, oder netsh auszuführen, welche für die korrekte Funktionsweise von OpenVPN erforderlich sind. Wert „**2**“ bedeutet, dass zusätzlich benutzerdefinierte Skripte erlaubt sind und Wert „**4**“ bedeutet, dass es nochmals zusätzlich erlaubt ist User-Passwörter zu übergeben.

```
persist-key
persist-tun
user nobody
group nogroup
client-to-client
status /var/log/openvpn-status.log
verb 3
keepalive 10 120
comp-lzo
script-security 2
```

Das Vollständige Konfigurationsfile für den Server als VPN Tunnel sollte dann wie folgt aussehen:

```
dev tun
proto udp
port 1194
ca /etc/openvpn/easy-rsa/keys/ca.crt
cert /etc/openvpn/easy-rsa/keys/server.crt
key /etc/openvpn/easy-rsa/keys/server.key
dh /etc/openvpn/easy-rsa/keys/dh2048.pem
server 10.8.0.0 255.255.255.0
push „redirect-gateway def1 bypass-dhcp“
push „dhcp-option DNS xxx.xxx.xxx.xxx“
push „dhcp-option DNS 8.8.8.8“
log-append /var/log/openvpn
persist-key
persist-tun
user nobody
group nogroup
client-to-client
status /var/log/openvpn-status.log
verb 3
keepalive 10 120
comp-lzo
script-security 2
```

Mit “Strg + O” kann die Änderung gespeichert und mit „Strg + X“ der Editor geschlossen werden.

#### 2.4.2 VPN Brücke (TAP)

Gegenüber der Einstellung bei einem VPN Tunnel wird als erstes der Bridged-Mode über „**dev tapX**“ aktiviert. TapX ist dabei, dass in der Ethernet-Konfiguration vergebene tap device, in unserem Fall tap0.

```
dev tap0
```

Des Weiteren wird nicht ein frei wählbarer VPN-Server vergeben, sondern die Server-Bridge angegeben, die in den Netzwerkeinstellungen konfiguriert wurde (Im Beispiel exemplarisch im Standardbereich 192.168.0.200). Zusammen mit einem Adressbereich, aus dem der VPN-Server Adressen an die Clients vergeben kann, da bei einer Bridge der Client in das Subnetz des Servers

„gezogen“ wird. Hier ist darauf zu achten, dass sich der Adressbereich nicht mit dem Adressbereich überschneidet, den der Router Serviceseitig per DHCP vergibt. Ansonsten kann es passieren, dass es doppelte IP-Adressen gibt.

```
server-bridge 192.168.0.200 255.255.255.0 192.168.0.201 192.168.0.220
```

Damit die Clients immer wieder dieselben Adressen zugewiesen bekommen wird noch der Befehl „ifconfig-pool-persist ipp.txt“ ergänzt. Dieser sorgt dafür, dass ein Client der sich erneut einwählt wieder seine vorherige Adresse aus dem Adresspool bekommt. Den Clients werden somit indirekt fixe IP-Adressen zugeordnet.

```
ifconfig-pool-persist ipp.txt
```

Ansonsten fallen gegenüber der Konfiguration eines VPN-Tunnels lediglich noch die „push“ Befehle weg. Diese werden nicht benötigt, da wie uns bei einer Bridge im selben Subnetz befinden wie der Server. Alle anderen Standardbefehle werden identisch verwendet.

Das Vollständige Konfigurationsfile für den Server als VPN Brücke sollte dann wie folgt aussehen:

```
dev tap0
proto udp
port 1194
ca /etc/openvpn/easy-rsa/keys/ca.crt
cert /etc/openvpn/easy-rsa/keys/server.crt
key /etc/openvpn/easy-rsa/keys/server.key
dh /etc/openvpn/easy-rsa/keys/dh2048.pem
ifconfig-pool-persist ipp.txt
server-bridge 192.168.0.200 255.255.255.0 192.168.0.201 192.168.0.220
log-append /var/log/openvpn
persist-key
persist-tun
user nobody
group nogroup
client-to-client
status /var/log/openvpn-status.log
verb 3
keepalive 10 120
comp-lzo
script-security 2
```

Mit „Strg + O“ kann die Änderung gespeichert und mit „Strg + X“ der Editor geschlossen werden.

## 2.5 Konfiguration Linux-Firewall

Für die Firewall des Raspberry Pis muss eine Weiterleitung zum Internetanschluss des lokalen Netzwerks eingerichtet werden. Dazu muss die Datei „rpivpn“ wie folgt angelegt und entsprechend der folgenden Kapitel angepasst werden:

```
sudo nano /etc/init.d/rpivpn
```

Durch einfügen der nachfolgenden Kommentare wird ein Header für ein Linux-Init-Skript erstellt:

```
#!/bin/sh
### BEGIN INIT INFO
# Provides: rpivpn
# Required-Start: $remote_fs $syslog
# Required-Stop: $remote_fs $syslog
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: VPN initialization script
### END INIT INFO
```

### 2.5.1 VPN Tunnel (TUN)

In dieser Variante wird als erstes das IP-Forwarding über den folgenden Befehl aktiviert:

```
echo ,echo „1“ > /proc/sys/net/ipv4/ip_forward' | sudo -s
```

Als nächstes wird mit dem Paketfilter „iptables“ eine Weiterleitung für VPN-Pakete angelegt:

```
iptables -A INPUT -i tun+ -j ACCEPT
iptables -A FORWARD -i tun+ -j ACCEPT
```

Abschließend wird über die nachfolgenden Befehle dann noch den Clients selbst der Zugang zum lokalen Netzwerk und zum Internet gewährt:

```
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -t nat -F POSTROUTING
iptables -t nat -A POSTROUTING -o 10.8.0.0 -o br0 -j MASQUERADE
```

Das Vollständige Init-File für den Server als VPN Brücke sollte dann wie folgt aussehen:

```
#!/bin/sh
### BEGIN INIT INFO
# Provides: rpivpn
# Required-Start: $remote_fs $syslog
# Required-Stop: $remote_fs $syslog
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: VPN initialization script
### END INIT INFO
echo ,echo „1“ > /proc/sys/net/ipv4/ip_forward' | sudo -s
iptables -A INPUT -i tun+ -j ACCEPT
iptables -A FORWARD -i tun+ -j ACCEPT
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -t nat -F POSTROUTING
iptables -t nat -A POSTROUTING -o 10.8.0.0 -o br0 -j MASQUERADE
```

Mit “Strg + O” kann die Änderung gespeichert und mit „Strg + X“ der Editor geschlossen werden.

## 2.5.2 VPN Brücke (TAP)

In diesem Fall ist die Konfiguration etwas einfacher, hier wird neben dem IP-Forwarding über die nachfolgenden drei Zeilen lediglich der konfigurierten Brücke Zugang zum lokalen Netzwerk und zum Internet gewährt.

```
iptables -A INPUT -i tap0 -j ACCEPT
iptables -A INPUT -i br0 -j ACCEPT
iptables -A FORWARD -i br0 -j ACCEPT
```

Das Vollständige Init-File für den Server als VPN Brücke sollte dann wie folgt aussehen:

```
#!/bin/sh
### BEGIN INIT INFO
# Provides: rpivpn
# Required-Start: $remote_fs $syslog
# Required-Stop: $remote_fs $syslog
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: VPN initialization script
### END INIT INFO
echo ,echo „1“ > /proc/sys/net/ipv4/ip_forward' | sudo -s
iptables -A INPUT -i tap0 -j ACCEPT
iptables -A INPUT -i br0 -j ACCEPT
iptables -A FORWARD -i br0 -j ACCEPT
```

Mit „Strg + O“ kann die Änderung gespeichert und mit „Strg + X“ der Editor geschlossen werden.

*Alternativ kann die Konfiguration der Firewall auch über Skripte realisiert werden, die direkt von OpenVPN ausgeführt werden und somit ein eigenständiges Skript erübrigen. Diese Variante wird hier aber nicht detailliert betrachtet.*

## 2.5.3 Init-File aktivieren

Wenn das Init-File zur Firewall-Konfiguration fertig erstellt wurde müssen dem File noch die benötigten Rechte zugewiesen werden und das File als Init-Skript installiert werden. Dies führt man mit den beiden folgenden Befehlen aus:

```
sudo chmod +x /etc/init.d/rpivpn
sudo update-rc.d rpivpn defaults
```

Abschließend muss das Skript noch ausgeführt und der OpenVPN Server neu gestartet werden:

```
sudo /etc/init.d/rpivpn
sudo /etc/init.d/openvpn restart
```

## 2.5.4 IP Forwarding statistisch aktivieren

Alternativ zum Befehl „echo ,echo „1“ > /proc/sys/net/ipv4/ip\_forward' | sudo -s“ der das IP-Forwarding bei jedem Systemstart temporär aktiviert, kann das IP-Forwarding auch permanent statistisch aktiviert werden. Hierzu muss dir Systemdatei „**sysctl.conf**“ geöffnet werden:

```
sudo nano /etc/sysctl.conf
```

Die nachfolgende Zeile muss dann durch das Entfernen des auskommentierenden # aktiviert werden.

```
net.ipv4.ip_forward=1
```

Mit "Strg + O" kann die Änderung gespeichert und mit „Strg + X“ der Editor geschlossen werden.

## 2.6 Konfiguration OpenVPN Client

Nachdem der Server soweit konfiguriert ist, müssen noch die Konfigurationen für den Client erstellt beziehungsweise richtig angepasst werden. Das Konfigurationsfile kann zwar auch direkt auf dem Client erstellt werden, die Erstellung auf dem Server bietet aber den Vorteil, dass dort immer beide Konfigurationen sowohl für den Server als auch den Client gepflegt sind.

Zuerst müssen wieder Root-Rechte vergeben werden. Dann wird die entsprechende Client Datei geöffnet. In unserem Fall „**remote-pc-1**“.

```
sudo su
cd /etc/openvpn/easy-rsa/keys
nano remote-pc-1.ovpn
```

Über den Befehl „remote...“ muss die Adresse des Servers und der Port über den der VPN Server erreichbar ist eingetragen werden. Entweder kann dies über eine statische öffentliche IP-Adresse erfolgen oder über einen Anbieter für eine dynamische DNS der die Adresse nachführt, wenn diese vom Provider neu vergeben wird:

```
remote xyz.dynDNSServer.com 1194 // oder StatischeIP 1194
```

Es ist wichtig, dass die Client Settings bei „dev“, „proto“, „verb“ und „script-security“ denen des Servers entsprechen. Wenn auf dem Server „comp-lzo“, „persist-key“ und „persist-tun“ aktiviert sind müssen diese auf dem Client ebenfalls verwendet werden. Über den Befehl „nobind“ wird ausgewählt, dass lokal keine Portbindung erzwungen wird und der Port beliebig sein kann. Die Zeile „remote-cert-tls server“ sorgt dafür dass explizit geprüft wird, ob das gegenüberliegende Zertifikat den Type Server besitzt. Damit auch nach einem serverseitigen Verbindungsabbruch wieder eine DNS-Auflösung ausgeführt wird, wird noch die Zeile „resolv-retry infinite“ ergänzt. In der Client-Konfiguration ist somit „dev tun“ gegenüber „dev tap0“ der einzige Unterschied zwischen Tunnel und Brücke.

Die vollständigen Konfigurations-Files für den Client sind für beide Fälle in den nachfolgenden Kapiteln dargestellt.

### 2.6.1 VPN Tunnel (TUN)

```
Client
dev tun
proto udp
remote xyz.dynDNSServer.com 1194 // oder StatischeIP 1194
```

```
resolv-retry infinite
nobind
persist-key
persist-tun
ca ca.crt
cert remote-pc-1.crt
remote-cert-tls server
key remote-pc-1.key
comp-lzo
verb 3
script-security 2
```

Mit "Strg + O" kann die Änderung gespeichert und mit „Strg + X“ der Editor geschlossen werden.

## 2.6.2 VPN Brücke (TAP)

```
Client
dev tap0
proto udp
remote xyz.dynDNSServer.com 1194 // oder StatischeIP 1194
resolv-retry infinite
nobind
persist-key
persist-tun
ca ca.crt
cert desktop-pc.crt
remote-cert-tls server
key desktop-pc.key
comp-lzo
verb 3
script-security 2
```

Mit "Strg + O" kann die Änderung gespeichert und mit „Strg + X“ der Editor geschlossen werden.

## 2.7 Erzeugung und Export Konfigurations Files für Clients

Abschließend wird noch die Konfigurationsdatei für den Client zusammen mit den zugehörigen Schlüsseln und Zertifikate in eine ZIP-Datei zusammengefasst. Sofern noch kein ZIP-Paket auf dem Raspberry Pi installiert ist, kann das wie folgt erledigt werden.

```
apt-get install zip
```

Danach wird pro Client wie folgt die ZIP Datei erstellt. Hier ist wichtig das überall der richtige Client-Name eingesetzt wird.

```
zip /home/pi/remote-pc-1.zip ca.crt remote-pc-1.crt remote-pc-1.key remote-pc-1.ovpn
```

Abschließend müssen noch die Rechte der Datei angepasst und die Root-Rechte abgemeldet werden.

```
chown pi:pi /home/pi/remote-pc-1.zip
exit
```

Die fertige ZIP Datei kann nun über ein FTP-Programm wie z.B. Filezilla oder via USB-Stick vom Raspberry Pi auf den Client kopiert werden.

### 3 Konfiguration OpenVPN Client (Windows)

#### 3.1 Installation OpenVPN

Die Software von OpenVPN kann direkt über die Homepage [www.openvpn.net](http://www.openvpn.net) bezogen werden. Für den als Beispiel dienenden Versuchsaufbau wurde hier auf die Open Source Version 2.4.7 zurückgegriffen. Für den Einsatz in einer kommerziellen Anwendung können die entsprechenden Lizenzen und Softwarepakete aber ebenfalls über die OpenVPN Homepage erworben werden. Nach dem Download der richtigen Software kann diese direkt auf dem Client-PC installiert werden und ist danach als „OpenVPN GUI“ über das Startmenü erreichbar.

#### 3.2 Konfiguration OpenVPN Client

Nach dem Start von „OpenVPN GUI“ erscheint nachfolgendes Symbol in der Taskleiste, das signalisiert, dass „OpenVPN“ gestartet ist.



Abbildung 2: Taskleistensymbol OpenVPN

Als erstes muss nun der Inhalt des Zip-Files, das vom Server kopiert wurde entpackt und im Konfigurationsverzeichnis von OpenVPN abgelegt werden. Es muss hier das Verzeichnis von OpenVPN verwendet werden das im User-Ordner angelegt wurde, nicht das allgemeine im Programmordner. Der Verzeichnisbaum müsste in etwa wie folgt aussehen:

```
C:\Users\XYZ\OpenVPN\config\remote-pc-1
```

In dem entpackten Ordner sollten dann die folgenden vier Files für Schlüssel, Zertifikate und Konfiguration vorhanden sein:

Name	Typ	Größe
 ca.crt	Sicherheitszertifikat	2 KB
 remote-pc-1.crt	Sicherheitszertifikat	6 KB
 remote-pc-1.key	KEY-Datei	2 KB
 remote-pc-1.ovpn	OpenVPN Config ...	1 KB

Abbildung 3: Files OpenVPN Client

Über rechtsklick auf das OpenVPN-Symbol in der Taskleiste kann nun aus allen bekannten Konfigurationen die gewünschte ausgewählt werden. Im erscheinenden Untermenü kann dann der Verbindungsaufbau zum Server gestartet, Log Informationen eingesehen, falls vergeben das Passwort geändert oder auch das Konfigurationsfile selbst angepasst werden. Wenn am Server Konfigurationsänderungen vorgenommen werden kann somit entweder das neue File vom Server auf den

Client kopiert werden oder direkt parallel das vorhandene File auf dem Client angepasst werden.

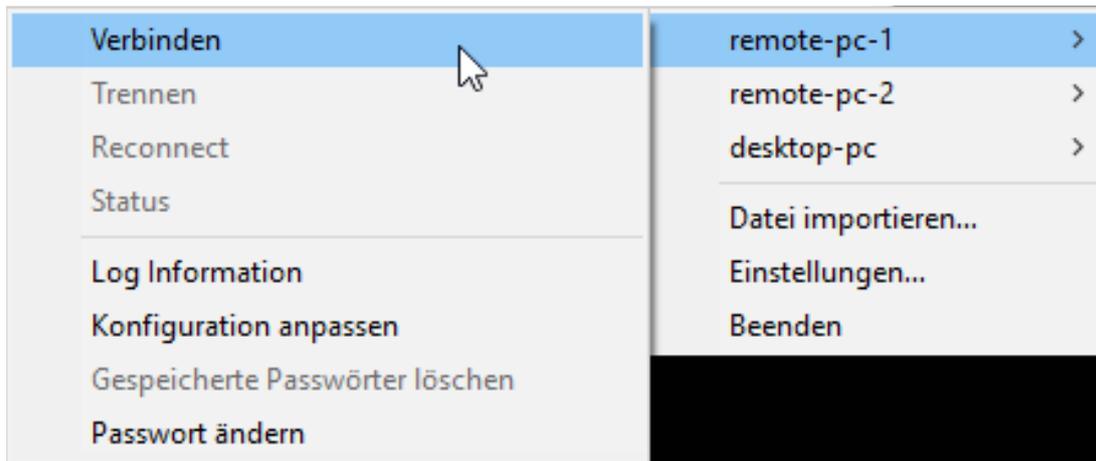


Abbildung 4: Optionen OpenVPN Client

### 3.3 Konfiguration TAP-Windows-Adapter V9

Der TAP-Windows-Adapter V9 ist ein virtueller Netzwerkadapter, der bei vielen Windows Rechnern bereits vorinstalliert ist und falls nicht mit der Installation von OpenVPN mitinstalliert wird. Über diesen Adapter baut OpenVPN die Verbindung zum ausgewählten Server auf. Der Adapter kann prinzipiell wie jeder andere reale Netzwerkadapter konfiguriert werden, im Falle der VPN-Verbindung bekommt er aber unabhängig von seinen eigenen Einstellungen die Konfiguration in Bezug auf IP-Adresse etc. vom VPN-Server zugewiesen.

Für die Verbindung zu einem BG XX dPro PN und dem Einsatz des „Drive Assistant“ ist allerdings wichtig, dass der Adapter im normalen Setting eine fixe IP-Adresse vergeben bekommt und nicht auf DHCP steht, da er ansonsten vom „Drive Assistant“ nicht erkannt wird. Welche Adresse vergeben wird ist dabei egal, da sie wie beschrieben überschrieben wird.

## 4 Allgemeine Netzwerkeinstellungen & Verbindungsaufbau

Bevor die Verbindung aufgebaut werden kann müssen noch ein paar allgemeine Einstellungen an der serverseitigen IT-Infrastruktur vorgenommen werden und der das Mapping im öffentlichen IP-Adressraum sichergestellt werden.

### 4.1 Portweiterleitung auf Routern aktivieren

Auf dem Router bzw. allen übergeordneten Routern über den/die der OpenVPN-Server mit dem Internet kommuniziert muss die Portweiterleitung des VPN-Ports (im Beispiel 1194) aktiviert werden, damit am Router ankommende VPN-Anfragen zum Server weitergeleitet werden. Die Weiterleitung kann Gerätespezifisch für das einzelnen Gateway aktiviert werden.

Die spezifische Konfiguration ist hier vom eingesetzten Router abhängig, weswegen hier nur prinzipiell und nicht im Detail das Vorgehen beschrieben wird.

### 4.2 Einrichtung dynamischer DNS-Server

Damit der OpenVPN Server immer angesprochen werden kann muss er auch im öffentlichen IP-Adressbereich immer unter der identischen Adresse erreichbar sein. Eine Möglichkeit wäre hier eine statische öffentliche IP-Adresse zu verwenden oder die Verwendung eines dynamischen DNS Anbieters, der dafür sorgt, dass auch wenn der Internet-Provider nach 24h oder nach einem Verbindungsabbruch dem Router und dadurch auch den Endgeräten neue Adressen vergibt, der VPN-Server trotzdem identisch erreichbar bleibt.

Hierzu muss bei einem entsprechenden Anbieter z.B. Secure Point ([www.spdyn.de](http://www.spdyn.de)) als erstes ein Konto eröffnet werden und die Route zum serverseitigen Router bekannt gemacht werden.

Danach muss auch auf dem Router der entsprechende dynamische DNS Anbieter bekannt gemacht werden, damit diesem übermittelt werden kann, wenn sich die Adressen geändert haben und dieser die Route nachführen kann.

Die spezifische Konfiguration ist hier vom eingesetzten Router und vom ausgewählten dynamischen DNS Anbieter abhängig, weswegen hier nur prinzipiell und nicht im Detail das Vorgehen beschrieben wird.

### 4.3 VPN Verbindung aufbauen und testen

Sind alle Settings wie beschrieben durchgeführt kann nun die Verbindung zum VPN Server aufgebaut werden. Auf dem Client wird nun über Rechtsklick auf das OpenVPN Symbol und Auswahl der richtigen Konfiguration der Menüpunkt „Verbinden“ gewählt.

Das OpenVPN Symbol in der Taskleiste wird nun gelb und ein Log Fenster erscheint, das den aktuellen Status des Verbindungsaufbaus anzeigt.

Treten keine Fehler auf schließt sich das Log-Fenster wieder automatisch sobald die Verbindung erfolgreich aufgebaut ist und das OpenVPN Symbol in der Taskleiste wird grün. Die Verbindung zum OpenVPN Server ist nun hergestellt.

Als erste Prüfung macht es Sinn zu kontrollieren was dem virtuellen Netzwerkadapter für eine IP-Adresse zugeordnet wurde. Bei einem VPN-Tunnel muss die Adresse im Bereich des VPN Servers (10.8.8.X) liegen. Bei einer VPN Brücke muss es eine Adresse aus dem freien Adress-Pool der VPN-Brücke sein und dem dortigen Netz entsprechen.

Abschließend kann die Verbindung noch mittels Ping zu testen. Hier empfiehlt es sich zuerst den VPN Server anzupingen. Ist dieser erreichbar steht auf jeden Fall schon die Verbindung zum Gateway. Kommt der Ping nicht durch empfiehlt es sich als erstes die Router- und Firewall-Einstellungen zu prüfen Als zweites ist sinnvoll noch ein bekanntes Gerät im Netzwerk des VPN Servers anzupingen. Kommt auch dieser Ping durch, ist die VPN-Verbindung voll funktionsfähig. Kommt der zweite Ping nicht durch ist die Empfehlung als erstes das Routing und die Firewall-Settings auf dem VPN-Server zu prüfen.

### 5 Drive Assistant

Im „Drive Assistant 5“ müssen keine speziellen Settings vorgenommen werden. Wenn alles entsprechend der Anleitung als VPN Brücke konfiguriert wurde und die VPN Verbindung steht kann bei Verbindungstyp „**Industrial Ethernet**“ unter Verfügbare Adapter der „**TAP-Windows Adapter V9**“ ausgewählt werden und nach starten der Antriebssuche werden im Netz befindliche Antriebe gefunden.

*Da der „Drive Assistant 5“ unbekannte Motoren via Broadcast Befehlen erkennt ist es wichtig, dass die Verbindung als VPN-Brücke realisiert wird. Ist die IP-Adresse des Antriebs fix vergeben und bekannt, kann mit einem VPN-Tunnel gearbeitet werden. Allerdings funktioniert in diesem Fall die Antriebssuche nicht und die IP-Adresse des Motors muss fix im entsprechenden Feld eingestellt werden.*

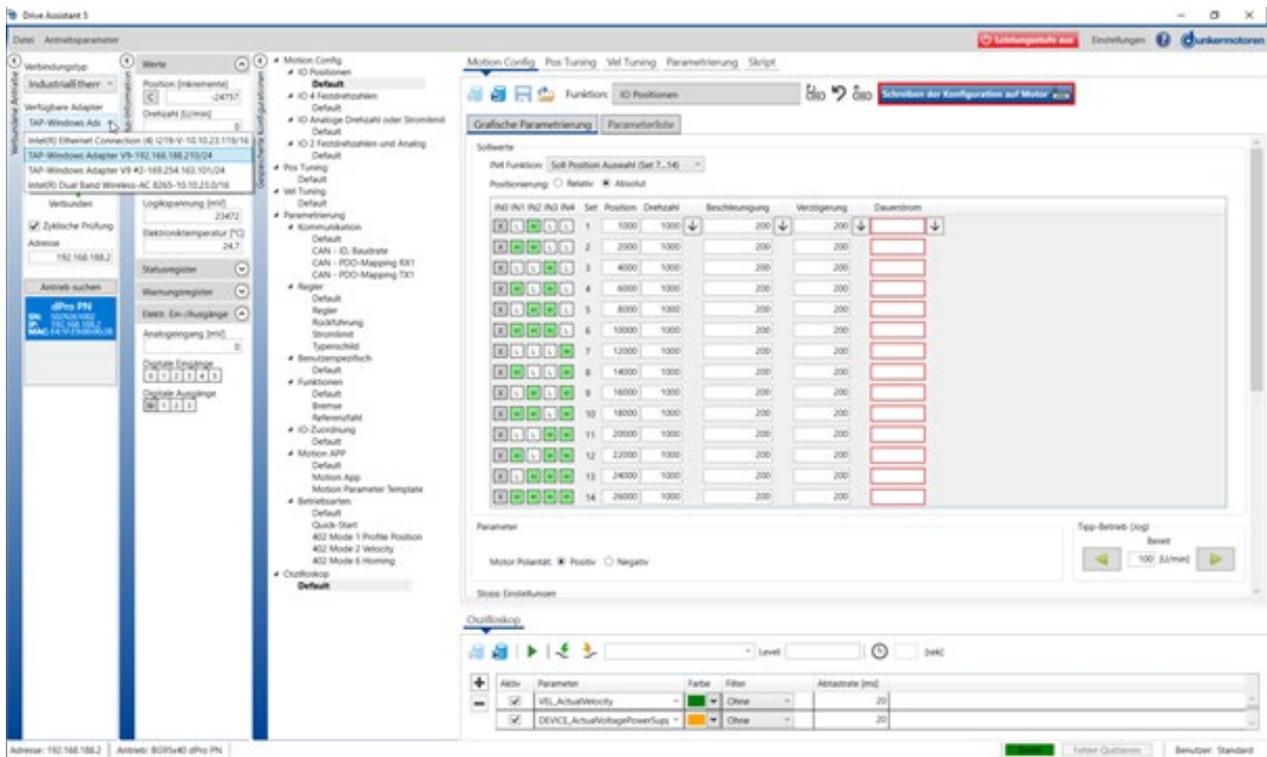


Abbildung 5: Drive Assistant 5 Auswahl Netzwerkadapter

Ihr Kontakt für Public Relations:  
 Janina Dietsche | janina.dietsche@ametek.com  
 Tel: +49 (0)7703/930-546