Whitepaper

Implement remote access to Dunkermotoren with VPN

Markus Weishaar | Product Manager IIoT
Dunkermotoren GmbH

This Whitepaper describes the configuration of a VPN connection for the remote access of a Dunkermotoren dPro Ethernet engine via the Internet with the Dunkermotoren standard software „Drive Assistant“ and the open source software OpenVPN.

A Linux-based Edge-Gateway is configured as a VPN server for this purpose. The Edge-Gateway communicates with the engine as well as with a router, which accepts the Internet connection, over 2 bridged ports via Ethernet. On the other side is a standard Windows PC on which „Drive Assistant“ and openVPN are installed. OpenVPN is configured as a client on the PC which sets up a VPN connection to the VPN server on the Gateway via the Internet. By means of this connection, the engine can be selected and driven via „Drive Assistant“ or a Firmware Update can be installed. If the engine has a known static IP address, the VPN connection can be configured as a tunnel since the linking of two subnets via routing is sufficient. If the engine has no IP address or no known IP address, the VPN connection must be set up as a bridge which draws the client into the same subnet in which the server is also located. This is necessary because the „Drive Assistant“ uses broadcasts for drive search and broadcasts only function in the same subnet.

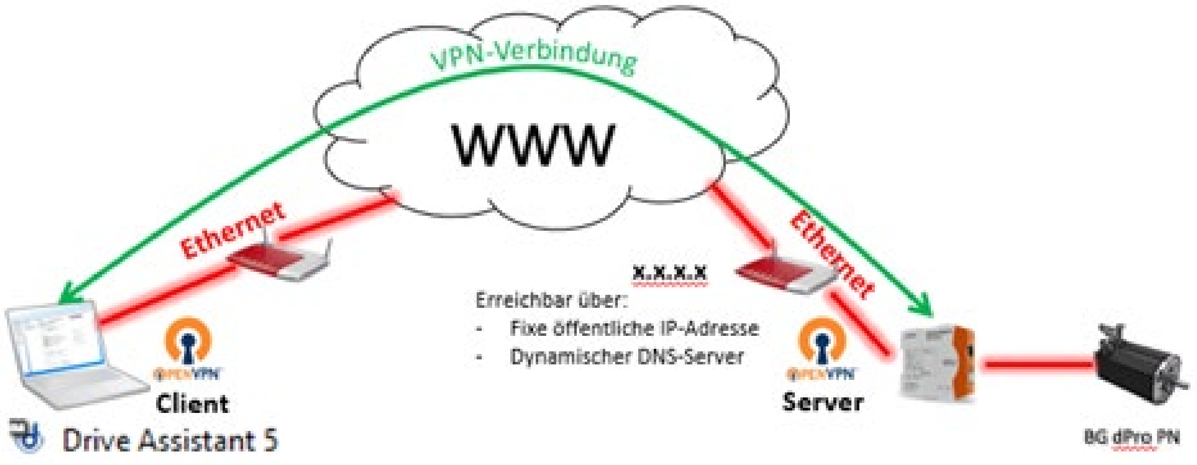


Figure 1: VPN networks

Author: Markus Weishaar
Date: 11.05.2019

Contents:

1 Requirements / Comparative Configuration....4

2 Configuration OpenVPN Server (Raspberry Pi/ Linux).... 4
2.1 Installation OpenVPN.... 4
Step 1 Update Raspberry and install OpenVPN
2.2 Ethernet-Settings.... 4
2.2.1 VPN Tunnel (TUN)....5
2.2.2 VPN Bridge (TAP).... 6
2.3 Create certificate and key....8
2.4 Configuration OpenVPN Server....9
2.4.1 VPN Tunnel (TUN)....9
2.4.2 VPN Bridge (TAP).... 11
2.5 Configuration Linux-Firewall....12
2.5.1 VPN Tunnel (TUN)....13
2.5.2 VPN Bridge (TAP).... 14
2.5.3 Activate Init-File.... 14
2.5.4 Statically Activate IP Forwarding.... 14
2.6 Configuration OpenVPN Client....15
2.6.1 VPN Tunnel (TUN)....15
2.6.2 VPN Bridge (TAP).... 16
2.7 Generation and Export Configurations Files for Clients....16

3 Configuration OpenVPN Client (Windows)....17
3.1 Installation OpenVPN.... 17
3.2 Configuration OpenVPN Client....17
3.3 Configuration TAP-Windows-Adapter V9....18

4 General Network Settings & Connection Establishment....18
4.1 Activate Port Forwarding to Routers....18
4.2 Establishment of Dynamic DNS Server.... 18
4.3 Building and Testing VPN Connection.... 19

5 Drive Assistant....19

1 Requirements / Comparative Configuration:

» 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)
» Operating system VPN-Server: Raspbian (Linux)
» Operating system VPN-Client: Windows 10
» Static public IP adress or dynamic DNS Server for server-side router
» Permission for configuration of the server-side router (port forwarding)
» Permission of configuration of the openVPN server‘s firewall

2 Configuration Open VPN Server (Raspberry Pi/ Linux)

2.1 Installation Open VP

Step 1 Update Raspberry and install Open VPN

Prior to installation of OpenVPN, it is recommended to search for updates for the Raspberry Pi operating system and to install them:
Now the OpenVPN software and the OpenSSL for the encryption must be loaded and installed with the following command:

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

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

2.2 Ethernet Settings

To forgo a routing between both Raspberry Pi Ethernet ports and still be able to access the VPN connection at eth0 to the engine at eth1, both ports are bridged and provided with a common address in this example.
Alternatively, it is also possible to work with only one port and provide it with a fixed IP address. The engine and the VPN connection can be connected to the port by means of a switch. This scenario is not detailed here.
To configure the Ethernet-Settings of the Raspberry Pi, the file „Interfaces“ must be opened as follows and adjusted according the following chapter:

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

The virtual Loopback-Adapter is always registered by default and should also always be retained in the configuration:

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

Now the existing network interfaces are created. Since our Gateway has two separated Ethernet ports, the two interfaces **eth0** and **eth1** are created. The attached command **„allow-hotplug**

ethX"„ causes the interface to be automatically activated and configured on a kernel event. This entry is important because the interface must otherwise be manually started via the command „sudo ifup eth0".

```
auto eth0
allow-hotplug eth0

auto eth1
allow-hotplug eth1
```

The configuration file must not be closed yet since the interfaces in the current state have no addresses and configuration and the Raspberry Pi would not be accessible anymore. The configuration is then carried out on a case-specific basis:

2.2.1 VPN Tunnel (TUN)

First, both Ethernet adapters are set to „manual" mode. This is important as they are configured via the bridge. For both adapters the following line is added:

```
iface ethX inet manual
```

Next, the Brücke **br0** is created as adapter and statically configured:

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

Afterwards, the network settings for the adapter are set up. An example configuration could appear as follows:

IP-Adresse:	192.168.0.200
Subnet mask:	255.255.255.0
Standard-Gateway:	192.168.0.1
Network:	192.168.0.0
Broadcast:	192.168.0.255

In the configuration file, the entries appear as follows:

```
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
```

Finally, the two interfaces are added to the bridge via the following line:

```
bridge_ports eth0 eth1
```

The complete network configuration entries to be made should then appear as follows:

```
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
```

The change can be saved with „Ctrl + O“ adn the editor can be closed with „Ctrl + X“.

2.2.2 VPN Bridge (TAP)

The fundamental setting of the ports and bridge are identical to the previous configuration for this variant. Only the bridge is supplemented in this respected so that the VPN adapter tap0 is likewise added to the bridge. **„Pre-up“** commands are given here before the bridge is built and **„post-up“** commands are executed immediately after the bridge is created. The same applies when ending the bridge for the commands **„pre-down“** and **„post-down“.**
First, the bridge is given a defined MAC address that the bridge uses to report to the network. This facilitates the diagnosis and enables the MAC address to be made known on the router if MAC filtering is active on it. If the command is omitted, the bridge receives the MAC address in the best case scenario but will not receive any MAC address in the worst case scenario.

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

The next commands first ask OpenVPN to create a virtual network Device tap0 before building the bridge and then add it after building the bridge.

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

Subsequently, a combined command is used to delete the IP addresses first assigned for the interfaces to the bridge and then to put the interfaces into **„promiscuous mode“** so that the bridge sees all data traffic arriving at these interfaces. Additionally, another command adds a fix route to the standard gateway for the bridge via which the Internet is accessed.

```
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
```

Finally, two command lines follow which remove the virtual network adapter from the bridge when the bridge is ended and ask OpenVPN to close the adapter.

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

The complete network configuration should then look as follows:

```
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
```

The change can be saved with „Ctrl + O“ adn the editor can be closed with „Ctrl + X“.

Alternatively, the construction and configuration of the bridge can also be realized via scripts, which are executed directly by OpenVPN and thus the network configuration itself can be kept narrow and independent. This variant is not considered in detail here.

2.3 Create certificate and key

The encryption used in this example is an example configuration for creating a functioning VPN connection quickly. Providing VPN clients with passwords is also avoided. For the concrete real use case, which goes beyond a connection test, it is recommended to select and configure a suitable encryption to achieve and guarantee the desired security levels.

First, the prefabricated „easy-rsa“ script is copied into the OpenVPN configuration directory. This creates different certificates and keys.

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

Next, the file „vars“ must be opened in the created directory and adjusted:

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

In the file, the line **„export EASY_RSA=“`pwd`““** must be replaced by the line **„export EASY_RSA=“/etc/openvpn/easy-rsa““**. You can also adjust the key length in the file in the line **„export KEY_SIZE=“** by changing the value. The key length determines the security level. For Raspberry Pi 3, a key length of **2048** presents no problem. For this reason, it is used in this example.

Now you have to change back to the configuration directory „easy-rsa“, assign root privileges there, execute the script „vars“ and make the resulting configuration file accessible via a symbolic link. These four steps are accomplished via the following four commands:

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

The certificate is created in the next step. The OpenVPN key files are reset and created anew:

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

A request to enter the two letter **„Country Name“** follows (DE for Germany, AT for Austria, and CH for Switzerland). All further queries can be skipped without entry by pressing Enter.

Finally, the **key file for the server** is created and here the „Country Name“ must also be entered and all further queries must be skipped. At the end of the dialog, the question on whether the certificate should be created should be confirmed twice with „Y“.

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

Next, the **key files for the clients** is created. It‘s important to note here that a key file must be created for each client who wishes to establish a connection with the VPN server. In our example we restrict ourselves to one client **„remote-pc-1“**. The procedure for certificate creation is analogous to the server (Country-Code, etc.)

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

If additional clients are required, the key files for these clients are created according to the same pattern:

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

For **clients equipped with a password, *„./build-key-pass client_name“*** must be used instead of the commands used above.

Key and certificate creation is now completed using the **Diffie-Hellman-key exchange** command. (This process takes approx. 20 min.)

```
./build-dh
```

Finally, the too-user is logged off after the end of key and certificate creation:

```
exit
```

2.4 Configuration OpenVPN Server

To configure the OpenVPN server, the file **„openvpn.conf“**must be opened as follows and adjusted according the following chapter:

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

2.4.1 VPN Tunnel (TUN)

First the routing over a tunnel is activated via **„dev tun“**, UDP is selected as transport protocol via **„proto udp“** and with **„port 1194“** the port is selected via which the tunnel is established. Alternatively, TCP can also be used during transport protocol. The port can be freely selected. The OpenVPN standard port 1194 is used in the example.

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

Next, an SSL/TLS root certificate (ca), a digital certificate (cert), and a digital key (key) are created via the directory „easy-rsa“. The correct bit-encryption is also entered. In this example, Diffie-Hellman with key length 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
```

Now the VPN server is given an IP address and a subnet mask. For this variant, a routing from this virtual VPN server network into the physical Raspberry Pi network occurs.

```
server 10.8.0.0 255.255.255.0
```

via the command **„push „redirect-gateway def1 bypass-dhcp“”** , all IP server traffic is routed through the VPN tunnel depending on the application in regards to whether this setting makes sense or not. The following two commands name the DNS servers to be used for name resolution. In our example, this is a local DNS server of the router and the public DNS server from Google (8.8.8.8). However, these can be chosen at your discretion.

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

To save log information for connection in the file „/var/log/openvpn“, the following line is added:

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

The following is a standard set of commands. The command **„persist-key“** makes it so the key files are not read again and **„persist-tun“** ensures that the TUN and TAP network drivers are not restarted. The commands **„user nobody“** and **„group nobody“** set the rights of OpenVPN after a program start and thereby increase security. The line **„client-to-client“** enables communication between the clients and **„status /var/log/openvpn-status.log“** creates a status file which documents the current connection. The comprehensiveness of the logs is defined via **„verb x“**. Value „0“ means no outputs other than error messages. A value between 1 and 4 is suitable for normal use whereas a higher value is suitable for troubleshooting. To check the connection, **„keepalive 10 120“** is added. A ping is triggered every 10 seconds and when an answer is not received after 120 seconds, a connection interruption is diagnosed. To compress data in the VPN tunnel and to increase throughput, an LZO compression is activated via **„comp-lzo“**. The last command **„script-security x“** defines which applications and scripts may be carried out by OpenVPN. Value „0“ indicates a strict ban on conducting external applications. Value „1“ indicates exclusively „built-in“ applications such as ifconfig, ip, route, or netsh are to be carried out. These are necessary for the correct functionality of OpenVPN. Value „2“ indicates that additional user-defined scripts are allowed and value „4“ indicates that it is additionally allowed to deliver user passwords.

```
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
```

The complete configuration file for the server as VPN tunnel should then appear as follows:

```
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
```

The change can be saved with „Ctrl + O“ and the editor can be closed with „Ctrl + X“.

2.4.2 VPN Bridge (TAP)

Compared to the setting for a VPN tunnel, the bridged mode is activated first via **„dev tapX“**. TapX is the tap device assigned in the Ethernet configuration, in our case tap0.

```
dev tap0
```

Furthermore, a freely selectable VPN server is not assigned, but the server bridge that was configured in the network settings is specified (in the example, the default range 192.168.0.200). Together with an address range from which the VPN server can assign addresses to the clients, because with a bridge the client is „pulled“ into the subnet of the server. Here it must be ensured that the address range does not overlap with the address range that the router assigns on the service side via DHCP. Otherwise it can happen that there are duplicate IP addresses.

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

So that clients are always allocated the same addresses again, the command „ifconfig-pool-persist ipp.txt“ is added. This ensures that a client that dials in again gets their previous address from the address pool. The clients are thus indirectly assigned fixed IP addresses.

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

Otherwise, compared to the configuration of a VPN tunnel, only the „push“ commands are dropped. These are not needed, because we are on the same subnet as the server. All other standard commands are used identically.
The complete configuration file for the server as VPN bridge should then appear as follows:

```
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
```

The change can be saved with „Ctrl + O“ adn the editor can be closed with „Ctrl + X“.

2.5 Configuration Linux-Firewall

A forwarding to the local network Internet connection must be arranged for the firewall of the Raspberry Pi. The file „rpivpn“ must be created as follows and adjusted according the following chapter:

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

A header for a Linux-Init-Script is created by inserting the following comments:

```
#! /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 this variant, the IP-forwarding is initially activated via the following command:

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

Next, a forwarding for VPN packets is created with the packet filter „iptables“:

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

Finally, the clients are guaranteed access to the local network and to the Internet via the following commands:

```
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
```

The complete Init-file for the server as VPN bridge should then appear as follows:

```
#! /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
```

The change can be saved with „Ctrl + O“ adn the editor can be closed with „Ctrl + X“.

2.5.2 VPN Bridge (TAP)
In this case, the configuration is somewhat simpler; here, apart from IP forwarding via the following three lines, only the configured bridge is granted access to the local network and the Internet.

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

The complete Init-file for the server as VPN bridge should then appear as follows:

```
#! /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
```

The change can be saved with „Ctrl + O“ adn the editor can be closed with „Ctrl + X“.

Alternatively, the configuration of the firewall can also be realized via scripts, which are directly executed by OpenVPN and thus make an independent script unnecessary. This variant is not considered in detail here.

2.5.3 Activate Init-File
If the Init-file to the firewall-configuration is completed, the required rights must assigned to the file and the file must be installed as Init-script. This is done with the following two commands:

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

Finally, the script must be carried out and the OpenVPN server must be restarted:

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

2.5.4 Statically Activate IP Forwarding
As an alternative to the command „echo „1“ /proc/sys/net/ipv4/ip_forward‘ | sudo -s“, which temporarily activates the IP-forwarding upon each system start, the IP-forwarding can also be permanently activated statically. For this, the system file **„sysctl.conf“** must be opened:

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

The following line must then be activated by removing the commenting #.

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

The change can be saved with „Ctrl + O“ adn the editor can be closed with „Ctrl + X“.

2.6 Configuration OpenVPN Client

After the server has been configured, the configurations for the client must be created or correctly adapted. Although the configuration file can also be created directly on the client, creation on the server offers the advantage that both configurations are always maintained there for both the server and the client.
First, root-rights must be given again. Then the corresponding client file is opened. In our case, **„remote-pc-1“.**

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

The server address and the port through which the VPN server is accessible must be entered via the command „remote...“. This can be done either via a static public IP address or via a provider for a dynamic DNS which updates the address if this is newly given by the provider:

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

It is important that the Client Settings for **„dev“**, **„proto“, „verb“** and **„script-security“** correspond to those of the server. If **„comp-lzo“, „persist-key“** and **„persist-tun“** are activated on the server, these must also be used on the client. The command **„nobind“** is used to select that no port binding is forced locally and that the port can be arbitrary. The line **„remote-cert-tls server“** ensures that it is explicitly checked whether the opposite certificate has the type server. The line **„resolv-retry infinite**“ is added so that a DNS resolution is executed again after a server-side connection termination. In the client configuration, „det tun“ as opposed to „tap0“ is the only difference between tunnel and bridge.

The complete configurations files for the client are presented for both cases in the following chapters.

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
```

The change can be saved with „Ctrl + O“ adn the editor can be closed with „Ctrl + X“.

2.6.2 VPN Bridge (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
```

The change can be saved with „Ctrl + O“ adn the editor can be closed with „Ctrl + X“.

2.7 Generation and Export Configurations Files for Clients

Finally, the configuration file for the client is collected together with the relevant keys and certificates in a ZIP-file. So long as no ZIP-packet is installed on the Raspberry Pi, this can be done as follows.

```
apt-get install zip
```

Next, the ZIP file is created per client as follows. Here it is important that the correct client name is implemented.

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

Finally, the file rights must be adjusted and the root rights must be logged off.

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

The finished ZIP file can now by copied from the Raspberry Pi to the client via an FTP program such as Filezilla or via USB stick.

3 Configuration OpenVPN Client (Windows)

3.1 Installation OpenVPN

The OpenVPN can be obtained directly from the homepage www.openvpn.net . For the test set-up serving as an example, Open Source version 2.4.7 was used here. For use in a commercial application, the appropriate licenses and software packets can also be acquired via the OpenVPN homepage.
After downloading the correct software, this can be installed directly on the client PC and is accessible afterwards as „OpenVPN GUI“ via the start menu.

3.2 Configuration OpenVPN Client

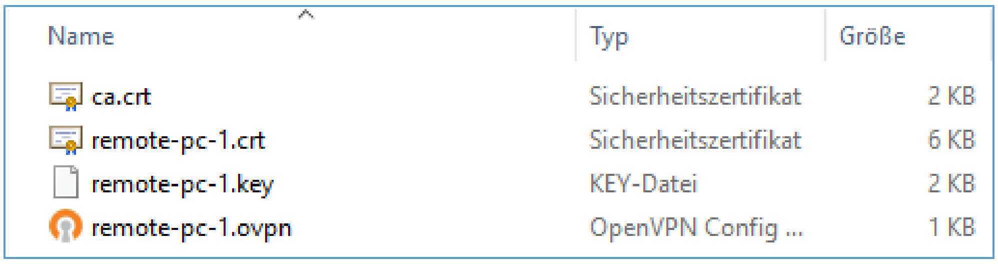
After starting „OpenVPN GUI“, the following symbol appears in the task bar which indicates that „OpenVPN“ has started.

Figure 2: Task bar symbol OpenVPN

First, the contents of the Zip-files which were copied from the server must be unpacked and stored in the configuration directory of OpenVPN. The directory of OpenVPN that was created in the user folder must be used here, not the general directory in the program folder. The directory tree should look something like this:

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

The unpacked folder, the following four files for key, certificate, and configuration should be available:



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

Figure 3: Files OpenVPN Client

The desired configuration can now be selected from all registered configurations via right-clicking on the OpenVPN symbol in the task bar. In the appearing submenu, the connection to the server can then be started, log information can be read, the password may be changed if necessary, or even the configuration file itself can be adjusted. If configuration changes are made to the server, either the new file from the server can be copied to the client or the existing file on the client can be adapted directly in parallel.

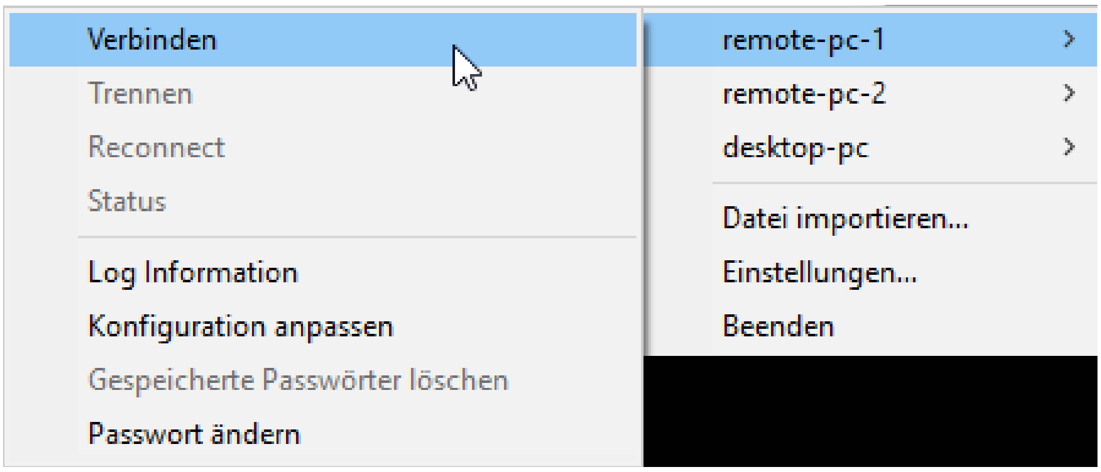


Figure 4: Optionen OpenVPN Client

3.3 Configuration TAP-Windows-Adapter V9

The TAP-Windows-Adapter V9 is a virtual network adapter which is already installed on many Windows computers and if not, it is installed with the installation of OpenVPN. OpenVPN builds the connection to the selected server via this adapter. The adapter can be configured the same as any other real network adapter in principle. In the case of a VPN connection, however, the VPN server assigns the configuration with regard to IP address independently of its own settings.
For the connection to a BG XX dPro PN and the use of the „Drive Assistant“, however, it is important that the adapter is assigned a fixed IP address in the normal setting and is not set to DHCP, otherwise it will not be recognized by the „Drive Assistant“. It does not matter which address is assigned, because it is overwritten as described.

4 General Network Settings & Connection Establishment

Before the connection can be established, a few general settings must be made on the server-side IT infrastructure and the mapping in the public IP address space must be ensured.

4.1 Activate Port Forwarding to Routers

On the router or all higher-level routers via which the OpenVPN server communicates with the Internet, the port forwarding of the VPN port (1194 in the example) must be activated so that VPN requests arriving at the router are forwarded to the server. Forwarding can be activated device specifically for the individual gateway.
The specific configuration depends here on the router used which is why the process is not described here in detail on principle.

4.2 Establishment of Dynamic DNS-Server

So that the OpenVPN server can always be addressed, it must always be accessible at the identical address even in the public IP address range. One possibility here would be to use a static public IP address or the use of a dynamic DNS provider, which ensures that even if the Internet provider assigns new addresses to the router and thus also to the end devices after 24 hours or after a disconnection, the VPN server still remains identically accessible.
For this purpose, an account must first be opened with an appropriate provider, e.g. Secure Point(www.spdyn.de) and the route to the server-side router must be made known.
Afterwards, the corresponding dynamic DNS provider must also be made known on the router, so that it can be transmitted if the addresses have changed and it can follow the route. The specific configuration here depends on the router used and the selected dynamic DNS provider, which is why the procedure is described here only in principle and not in detail.

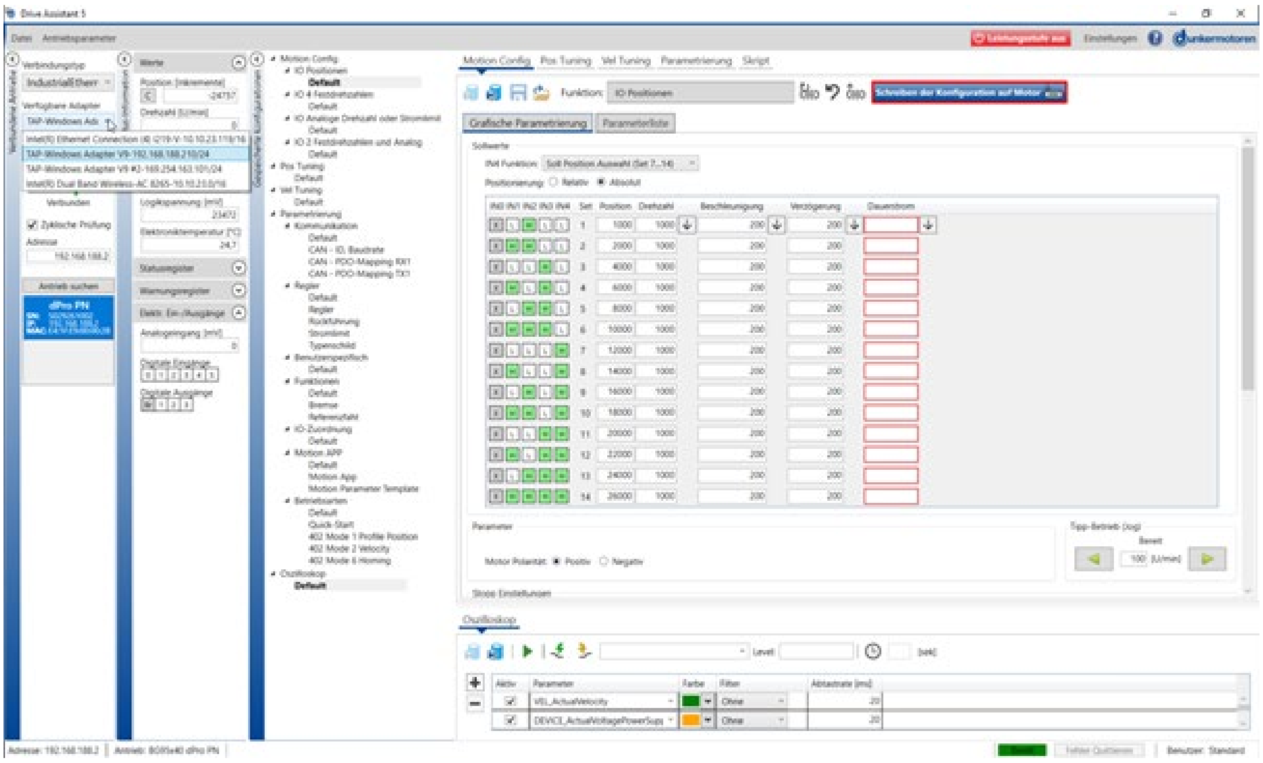
4.3 Building and Testing VPN Connection

If all settings have been executed as described, the connection to the VPN server can be established. On the client, right-click on the OpenVPN symbol and select the correct configuration of the menu item „Connect“.
The OpenVPN symbol in the task bar now turns yellow and a log window appears which displays the current status of the connection establishment.
If no error occurs, the log window closes again automatically as soon as the connection has been successfully established and the OpenVPN symbol in the task bar turns green. The connection to the OpenVPN server has now been established.
As a first check, it makes sense to check what has been assigned to the virtual network adapter for an IP address. For a VPN tunnel, the address must be in the range of the VPN server (10.8.8.X). For a VPN bridge, it must be an address from the free address pool of the VPN bridge and correspond to the network there.
Finally, the connection can still be tested using ping. Here it is recommended to ping the VPN server first. If this is accessible, the connection to the Gateway is already established. If the ping does not go through, it is recommended to firstly check the router and firewall settings and secondly to ping a registered device in the VPN server‘s network. If this ping goes through, the VPN connection is fully functional. If the second ping does not go through, the recommendation is to firstly check the routing and the firewall setting on the VPN server.

5 Drive Assistant

No special settings need to be carried out in „Drive Assistant 5“. If everything has been configured as a VPN bridge according to the instructions and the VPN connection is established, the **„TAP-Windows Adapter V9“** can be selected under Available Adapters for Connection Type **„Industrial Ethernet“** and after starting the Drive Search, drives located in the network are found.

Since the „Drive Assistant 5“ recognizes unknown motors via broadcast commands, it is important that the connection is implemented as a VPN bridge. If the IP address of the drive is permanently assigned and known, a VPN tunnel can be used. However, in this case the drive search does not work and the IP address of the motor must be set permanently in the corresponding field.



Sollwerte

IN4 Funktion: Soll Position Auswahl (Set 7...14)

Positionierung: ○ Relativ ● Absolut

IN0	IN1	IN2	IN3	IN4	Set	Position	Drehzahl	Beschleunigung	Verzögerung	Dauerstrom
X	L	H	L	L	1	1000	1000	200	200	
X	H	H	L	L	2	2000	1000	200	200	
X	L	L	H	L	3	4000	1000	200	200	
X	H	L	H	L	4	6000	1000	200	200	
X	L	H	H	L	5	8000	1000	200	200	
X	H	H	H	L	6	10000	1000	200	200	
X	L	L	L	H	7	12000	1000	200	200	
X	H	L	L	H	8	14000	1000	200	200	
X	L	H	L	H	9	16000	1000	200	200	
X	H	H	L	H	10	18000	1000	200	200	
X	L	L	H	H	11	20000	1000	200	200	
X	H	L	H	H	12	22000	1000	200	200	
X	L	H	H	H	13	24000	1000	200	200	
X	H	H	H	H	14	26000	1000	200	200	

Parameter

Motor Polarität: ● Positiv ○ Negativ

Stopp Einstellungen

Aktiv	Parameter	Farbe	Filter	Abtastrate [ms]
✓	VEL_ActualVelocity	🟩	Ohne	20
✓	DEVICE_ActualVoltagePowerSupp	🟧	Ohne	20

Figure 5: Drive Assistant 5: Network Adapter Selection

Your Contact For Public Relations:
Janina Dietsche | janina.dietsche@ametek.com
Tel: +49 (0)7703/930-546

dunkermotoren AMETEK® ADVANCED MOTION SOLUTIONS