Rudolf Diesel Fachschule 18.10.2019

Installation und Konfiguration der virtuellen Maschine

Verwendetes Betriebssystem	Microsoft Windows 10 64bit
RAM	4096MB
Grafikspeicher	128MB
Netzwerk	Intel PRO/10000 MT Desktop
Verwendetes Tool	Vagrant/ Oracle VM VirtualBox

Installation der Software „Vagrant" und „Oracle VM VirtualBox"

Vagrant ist eine freie Ruby-Anwendung zum Erstellen und Verwalten virtueller Maschinen. VirtualBox ist eine Virtualisierungssoftware von Oracle, die kompatibel mit der Software Vagrant ist. Nachdem die Software von den Herstellerwebseiten (www.vagrantup.com/downloads und https://www.virtualbox.org/wiki/Downloads) heruntergeladen und erfolgreich auf dem Host-System installiert wurden, wurde auf selbigem ein geeignetes Verzeichnis zur Erstellung der virtuellen Maschine angelegt. Die Wahl für das Betriebssystem der Maschine fiel auf Microsoft Windows 64bit.

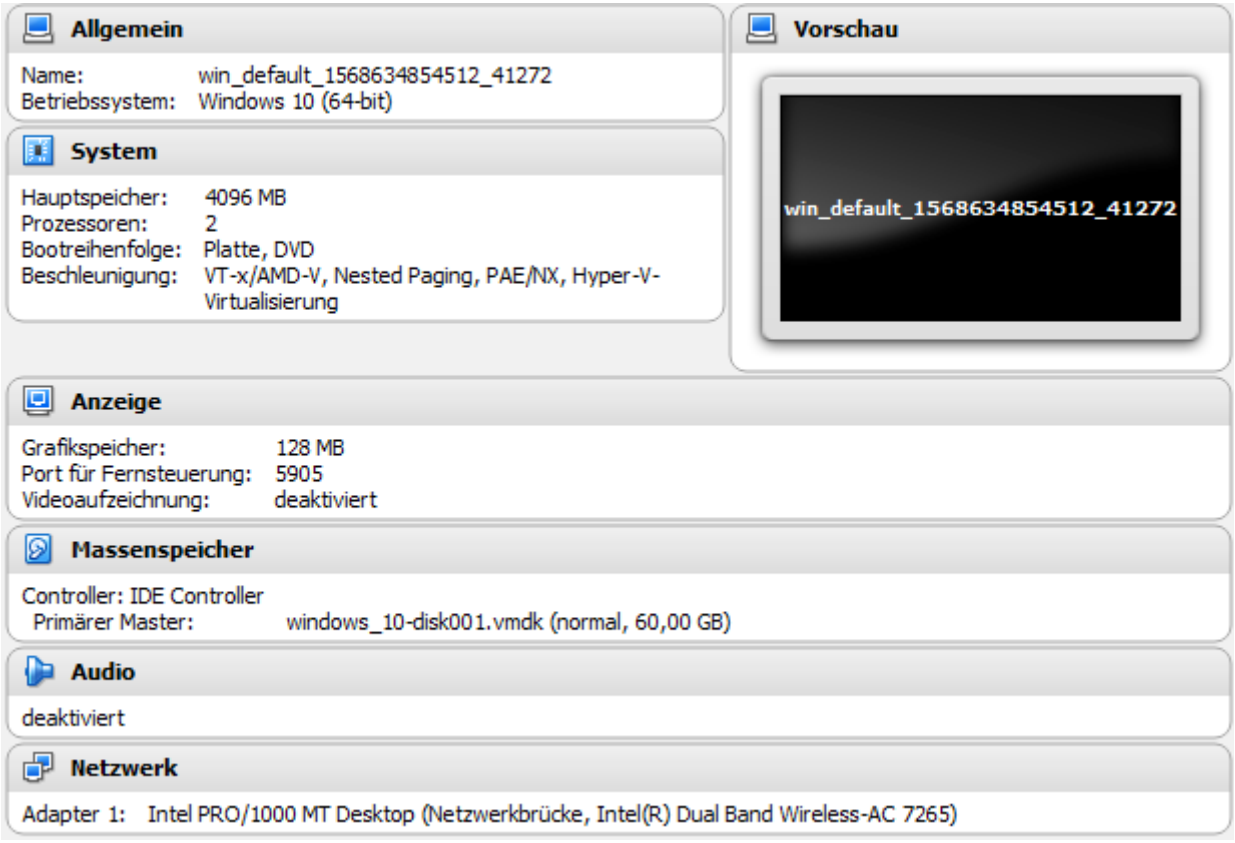
Installation des Betriebssystems

Die Iso-Datei wurde dann über die Eingabeaufforderung mit dem Kommando **vagrant init** in dem zuvor auf unserem Host eingerichtetes Verzeichnis aufgespielt.

Michael Mölle

Konfiguration der virtuellen Maschine

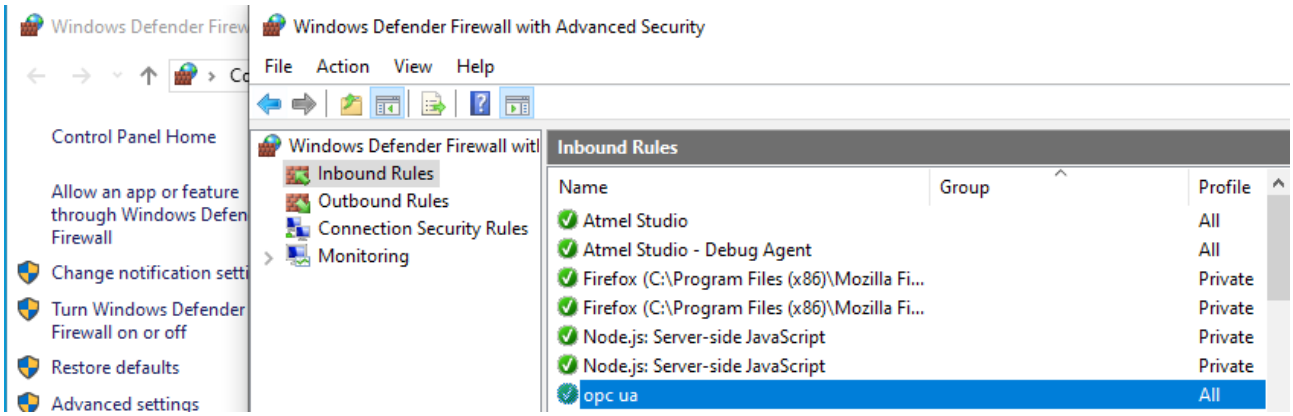
Nach der Installation der Iso-Datei wurde die virtuelle Maschine nach folgenden Einstellungen konfiguriert:



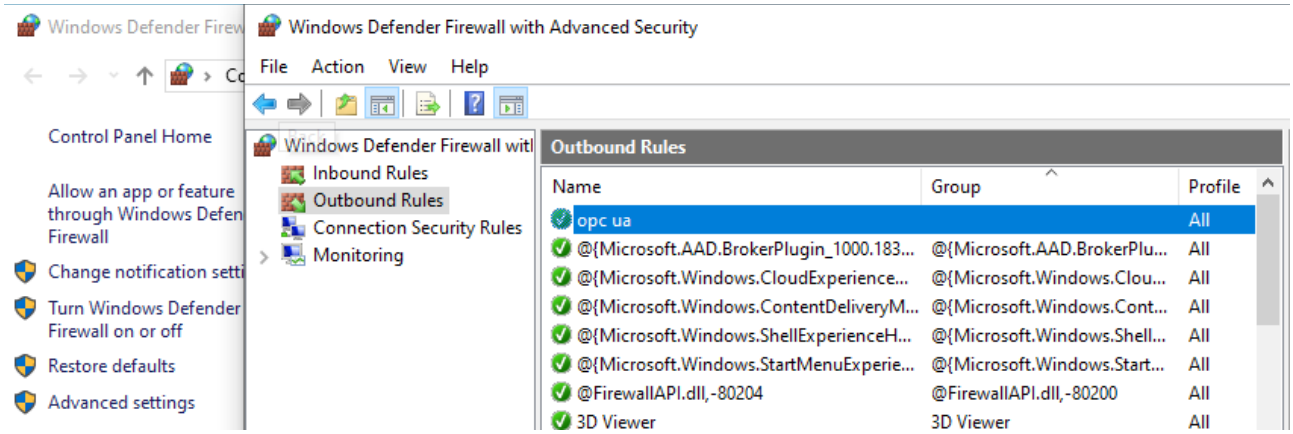
Konfiguration der virtuellen Maschine (Oracle VM VirtualBox)

Konfiguration der Netzwerkeinstellungen

Damit im späteren Projektverlauf der OPC-UA C++ Demo-Server, der auf der Windows 10 VM betrieben wird Konnektivität zur Software *Visual Components*, welche auf unserem Host-System installiert ist, und anderen Clients erhält, wurde in der **Systemsteuerung** im Reiter **Windows Defender Firewall** in den **erweiterten Einstellungen** ein- und ausgehende Regeln erstellt.
In den Regeleinstellungen wird letztendlich für den **Port 48010** die Verbindung in beide Richtungen zugelassen. Zu Testzwecken wurde die Windows-Firewall deaktiviert.



Festlegung der eingehenden Regel



Festlegung der ausgehenden Regel

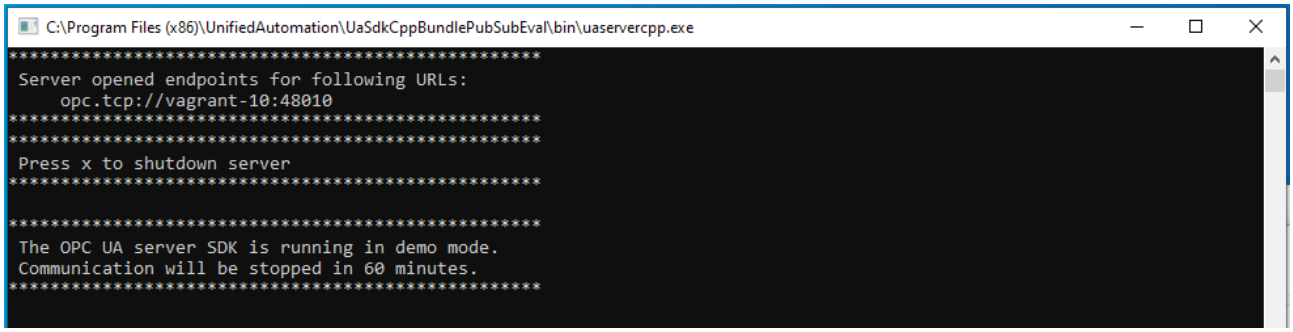
Installation der OPC-UA-Tools

Nach der erfolgreichen Erstellung unserer Regeln in den Firewall-Einstellungen von Windows geht es nun darum die verschiedenen Tools, die wir für die Konfiguration und Verwaltung des OPC UA Servers und des OPC UA Clients benötigen zu downloaden und zu installieren.

Diese sind kostenlos auf der Website der „**Unified Automation**" (https://www.unified-automation.com/) zu erhalten. Allerdings ist es notwendig, zuvor einen Benutzeraccount anzulegen und sich zu registrieren. Nach vollständiger Registrierung und beantworten der Bestätigungsmail kann nun die Software „***UAExpert***" (https://www.unified-automation.com/de/downloads/opc-ua-clients.html) und das „**C++ based OPC UA Client/ Server SDK + Pub/ Sub Bundle v1.7.1**" heruntergeladen werden. Letzteres enthält den C++ Demo-Server und den C++ Demo-Client zu Testzwecken.
„**UAExpert**" ist ein universeller OPC-UA Testclient und unterstützt OPC-UA Features wie **DataAccess**, **Alarms & Conditions**, **HistoricalAccess** und den Aufruf von OPC-UA **Methoden.**

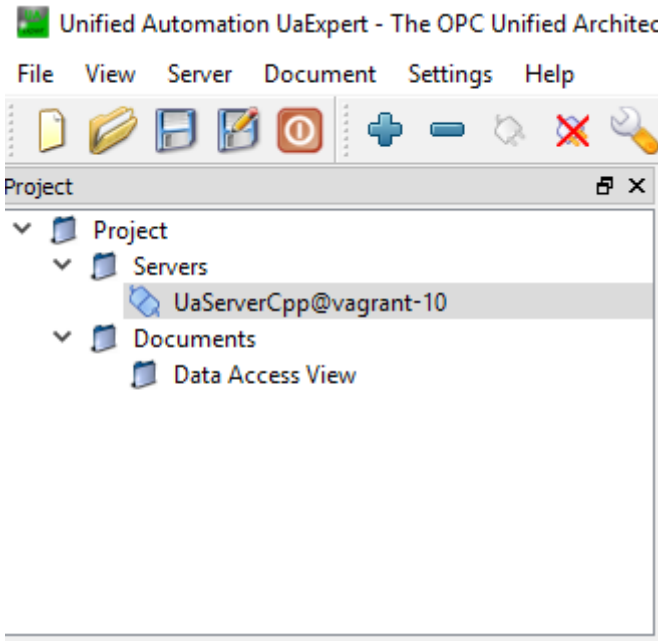
Funktionalität der Netzwerkkonnektivität testen

Nachdem der *UAExpert* und der Demo-Client, so wie der Demo-Server auf der virtuellen Maschine installiert wurden und die ein- und ausgehenden Regeln der Firewalleinstellungen erstellt wurden, kann nun die Konnektivität von Demo-Server (virtuelle Maschine) zum Client (Host-System) getestet werden. Dies haben wir mit zwei Clients getestet.
Für den ersten Client-Server-Verbindungstest wurde der OPC UA Server auf der virtuellen Maschine gestartet.



Betrieb des OPC UA C++ Demo-Servers

Für diesen Konnektivätstest ist es von Nöten, dass das Tool *UAExpert* ebenfalls auf dem Host-System installiert ist. Dieser wird geöffnet und unter dem Reiter **Add Server** die Endpint-Adresse des Servers eingegeben. Nach erfolgreichem Eintragen der Adresse, kann nun unter **Connect Server** eine Verbindung aufgebaut werden.
Aus der Ausgabe können mögliche Fehler entnommen werden. Taucht dort die Zeile *„Connection status of server 'UaServerCpp@vagrant-10' changed to 'Connected'"* war das Verbinden mit dem Server erfolgreich.

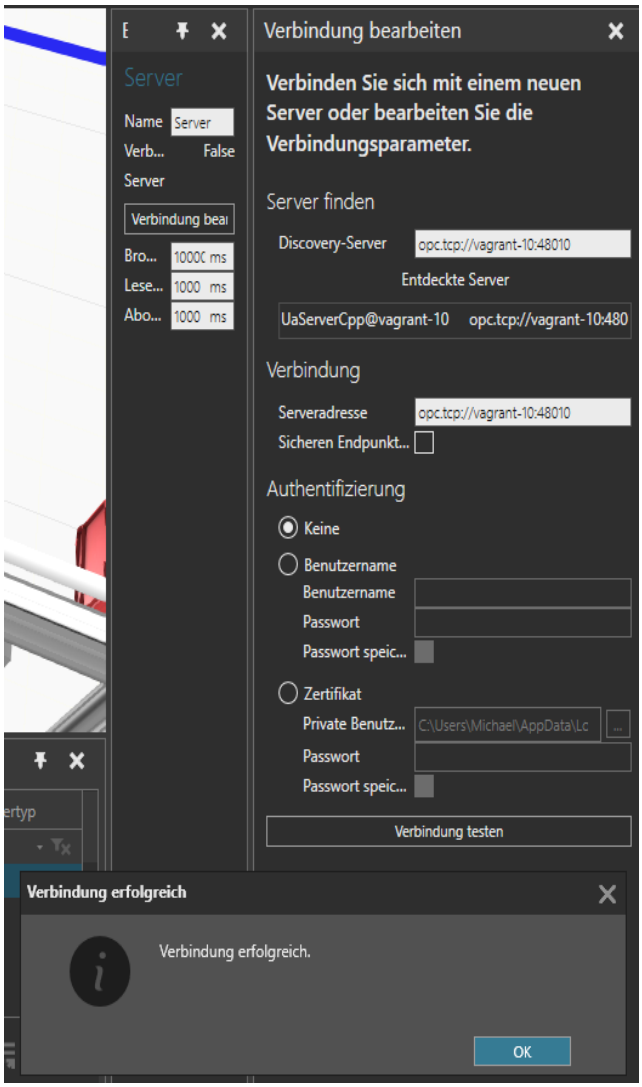


Konnektivität zum OPC UA C++ Demo-Server mit UAExpert hergestellt

Timestamp	Source	Server	Message
18.10.2019 10:35...	Server Node	UaServerCpp@...	Security policy: 'http://opcfoundation.org/UA/SecurityPolicy#None'
18.10.2019 10:35...	Server Node	UaServerCpp@...	ApplicationUri: 'urn:vagrant-10:UnifiedAutomation:UaServerCpp'
18.10.2019 10:35...	Server Node	UaServerCpp@...	Used UserTokenType: Anonymous
18.10.2019 10:35...	AddressSpaceM...	UaServerCpp@...	Registered for ModelChangeEvents
18.10.2019 10:35...	Server Node	UaServerCpp@...	Connection status of server 'UaServerCpp@vagrant-10' changed to 'Connected'.
18.10.2019 10:35...	Server Node	UaServerCpp@...	Revised values: SessionTimeout=1200000, SecureChannelLifetime=3600000
18.10.2019 10:35...	AddressSpaceM...	UaServerCpp@...	Browse on node 'i=84' succeeded.
18.10.2019 10:35...	TypeCache	UaServerCpp@...	Reading type info of NodeId NS0|Numeric|35 succeeded
18.10.2019 10:35...	TypeCache	UaServerCpp@...	Reading type info of NodeId NS0|Numeric|33 succeeded
18.10.2019 10:35...	TypeCache	UaServerCpp@...	Reading type info of NodeId NS0|Numeric|31 succeeded

Ausgabefenster in UAExpert

Der zweite und wohl genauso wichtige Test bestand darin, zu sehen ob wir eine Konnektivät vom Server zur Software *Visual Components*, die auf unserem Host-System betrieben wird aufbauen können.
Zu diesem Zweck wurde in *Visual Components* unter **Optionen** den Reiter **Konnektivität** aktiviert und nach einem Neustart der Software eine OPC UA Verbindung zu unserem Demo-Server eingerichtet. Der anschließende Funktionstest war erfolgreich.



Konnektivät zum OPC UA C++ Demo-Server mit Visual Component hergestellt