Modellbahn, Linux und Windows Software

Einleitung

Im H0-Modellbahnforum kamen Diskussionen bezüglich der nötigen Betriebssysteme für diverse Modellbahnsoftware, im speziellen Lokprogrammer auf. Einige spezielle Software für Modellbahn, ob Logprogrammer oder Gleisplanungssoftware, erfordert Windows als Betriebssystem. Für MacOS und Linux ist die Auswahl in diesen Bereichen nicht sehr üppig. Auch für die Steuerung der Modellbahn mit PC existiert wohl nur RocRail und iTrain, welches unter allen verbreiteten Betriebssystemen läuft. In einem Thread ging es um ein Programmiergerät für Fahrzeugdecoder, das zwar mit iOS und MacOS funktioniert, aber für das Update des Programmiergerätes selbst ist wieder ein Windows erforderlich.

Obiges gilt auch für diverse Amateurfunk-Software und diese kurze Beschreibung kann auch für den Betrieb von RufzXP, Morserunner, N1MM, HamOffice, UCXLog Logging Software verwendet werden.

Ich möchte hier aufzeigen, wie man mit einer virtuellen Maschine mit Windows 10 (es wäre auch Windows 7 oder gar XP möglich) als Betriebssystem für die sporadische Nutzung von Windows Software gut arbeiten kann. Denn weder ein Update eines LokProgrammers selbst oder eine Gleisplanungssoftware sind Anwendungen, die man in der Regel ständig benötigt. Nun kann man dafür natürlich auch einfach ein Notebook, z.B. von den vielen Angeboten der Leasing-Rückläufer, nehmen, aber man hat dann ein weiteres Gerät, was nur sporadisch benötigt wird, nur wenn es benötigt wird, sind meist erst viele Updates zu machen, wenn so ein Notebook nur alle paar Monate mal aus dem Regal geholt wird. Mein Vorschlag mit einer virtuellen Maschine mit Windows, sogar mit einer älteren oder gar alten Version werde ich im folgenden näher erläutern.

Virtuelle Maschine (VM)

Eine virtuelle Maschine ist im Prinzip eine Nachbildung eines ganzen Rechnersystems mit allen Schnittstellen per Software oder mit einer abstrahierenden Schicht. Man muss hier die Abgrenzung zu einer Emulation beachten, eine virtuelle Maschine kapselt die Hardware, die wirklich vorhanden ist, bei einem emulierenden System kann auch ein gänzlich anderer Prozessor emuliert werden.

Für die Einrichtung einer virtuellen Maschine gibt es zwei wesentliche Softwareprodukte, VirtualBox und VMware. Ich beschreibe hier den Einsatz von VirtualBox, welches es für Windows, Linux und MacOS gibt. Ich beschränke mich dabei auf den Einsatz von Linux als Hostsystem und Windows als Gastsystem. Das Hostsystem ist der PC mit Linux Betriebssystem, das Gastsystem die virtuelle Maschine mit dem Windows Betriebssystem. Das ist aber durchaus auch übertragbar, wenn ein MacOS als Hostsystem verwendet wird.

Der Vorteil einer virtuellen Maschine besteht darin, dass eine schnelle Wiederherstellung möglich ist, das sogenannte Snapshots angelegt werden können, wenn man nur schnell mal eine Software ausprobieren möchte, dann aber zum ursprünglichen Zustand zurück möchte. Während man bei einer realen Hardware dafür dann eine Neuinstallation ins Auge fassen muss, genügt es bei einer virtuellen

Maschine die vorher gesicherte virtuelle Festplatte wieder aus einem Archiv zu kopieren oder den Snapshot wieder zurückzusetzen. Das dauert alles nur wenige Sekunden.

Ich werde hierbei mein System beschreiben und eine neue virtuelle Maschine für die Nutzung von Modellbahnsoftware aufsetzen.

Linux als Hostsystem

Steigt man neu in Linux ein, stellt sich zuerst die Frage nach der passenden Linux Distribution. Ich arbeite seit Jahrzehnten mit Debian auf Serversystemen. Debian, weil es extrem stabil ist und Stabilität vor neuen Features und den neuesten Softwareversionen geht. Nun möchte man auf dem Desktop auch mal die ganz aktuellen Versionen von Software verwenden. Anzumerken ist hier, dass Debian im Repository nicht total veraltete Software hat, sondern nur eben nicht immer die aktuellsten Versionen mit den neuesten Funktionen. Updates und Patches für die Software im Repository gibt es trotzdem. So empfehle ich für den Desktoprechner MX-Linux. MX-Linux basiert auf Debian, verwendet das gleiche Paketmanagement wie Debian. Unter MX-Linux gibt es eine Reihe Konfigurationsprogramme mit grafischer Oberfläche, so dass eher selten die Kommandozeile benötigt wird. Das ist ein oft gehörtes Argument, dass unter Linux immer die Kommandozeile benötigt wird und dies für Umsteiger von Windows schwierig sein kann. MX-Linux ist da eher ein GUI lastiges Betriebssystem, mit einfacher Bedienung, einer schnellen grafischen Oberfläche und in der Regel findet sich ein Umsteiger schnell zurecht.

Während man unter anderen Betriebssystemen Software aus diversen Quellen erst über ein Setup installiert, nutzen viele Linux Distributionen ein Repository. Im Repository sind vorkonfigurierte Softwarepakete, z.B. für Office Anwendungen, für CAD Anwendungen usw. enthalten, die einfach installiert werden können und dann auch automatisch aktualisiert werden. Im Repository von Debian/MX-Linux befinden sich ca. 60.000 Softwarepakete

Die Hardware

Für ein MX-Linux sind die Hardwareanforderungen nicht besonders hoch. Ich habe 2021 allerdings einen neuen Rechner angeschafft, der gut dimensioniert ist und so noch einige Jahre ausreichend sein wird. Das ist aber nicht nötig, wenn man nur eine virtuelle Maschine gleichzeitig laufen lässt. Ich habe nur manchmal mehrere virtuelle Maschinen laufen. Wichtig ist genügend Festplattenplatz, möglichst eine SSD wegen der Geschwindigkeit und ausreichend RAM. Mit dem i7 Prozessor und den 32GByte RAM in meinem Rechner kann ich ohne weiteres 4-6 Windows Systeme parallel laufen lassen. Für eine virtuelle Maschine genügt aber auch ein i5 mit 8GByte RAM. Auch ein i3 würde gehen. Es kommt immer etwas darauf an, was man machen möchte und welches Betriebssystem als Gastsystem laufen soll. Da ich viel CAD (KiCad, Eagle) mache und dabei mehrere Monitore nutze, habe ich eine etwas leistungsfähigere Grafikkarte. Die kann zwar bis zu 4 Monitore bedienen, aber ist preislich im Rahmen und keine Gamer-Karte, aber für 3D Darstellungen bei CAD durchaus sehr performant. Ansonsten ist an meiner Hardware nichts besonderes.

MX-Linux

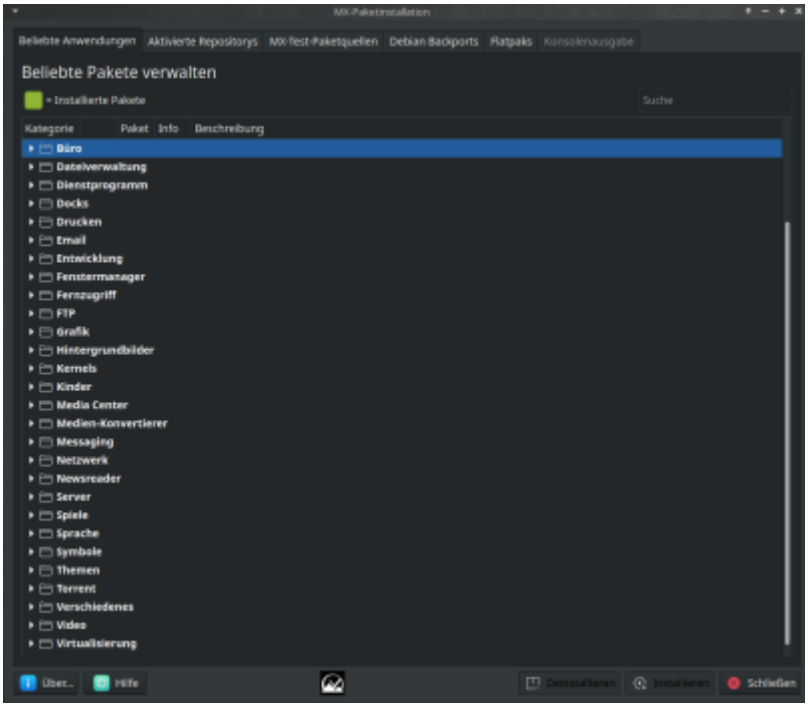
Ich werde nicht beschreiben, wie man MX-Linux installiert, weil ich das nicht einfach veranschaulichen kann, es läuft ja bei mir und ich wollte nicht neu installieren (vielleicht mal in einer virtuellen Maschine). Hier soll es mehr um die virtuelle Maschine mit Windows und Modellbahnsoftware gehen. Für die Installation ist es aber am einfachsten einen USB Stick vorzubereiten. Von dem Stick kann man MX-Linux auch direkt starten und sich das erst mal anschauen ohne das etwas am System verändert wird. Wer sein Windows System nicht gleich plätten möchte, kann sich auch einfach eine zweite Festplatte holen, die Windows Platte abklemmen und auf der zweiten Platte das Linux installieren. So geht nichts kaputt und man hat die Sicherheit, das das Windows System erhalten bleibt.

Software und Repository

Für den Heimanwender ist eigentlich alles dabei, was man so täglich benötigt. Meine verwendete Software habe ich hier aufgeführt (nicht vollständig, ändert sich auch mal).

- Office Packet (LibreOffice)
- Webbrowser (Firefox, Chromium)
- E-Book Reader (Calibre)
- PDF Betrachter (Okular)
- E-Mail (Thunderbird uva.)

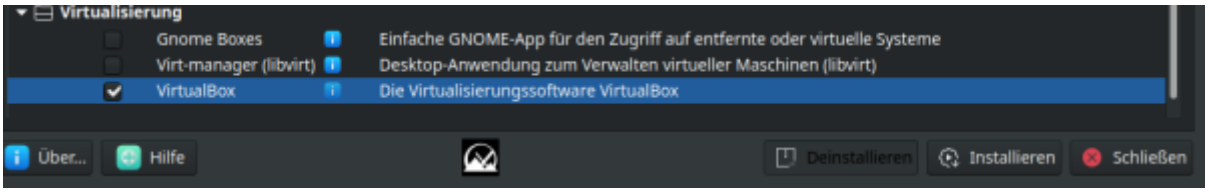
Die Software findet man in der MX-Paketinstallation (ganz simpel) oder in Synaptic (etwas anspruchsvoller). Die verfügbaren Softwarepakete sind in Kategorien sortiert.



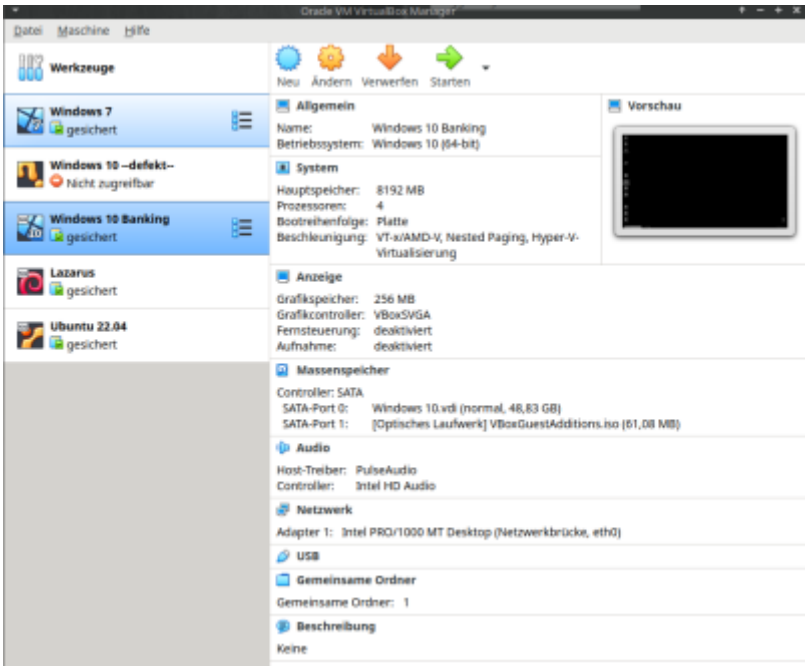
VirtualBox

In den Softwarepaketen findet man unter Virtualisierung auch VirtualBox. Für die Installation muss

dieses nur markiert werden und dann installiert werden.

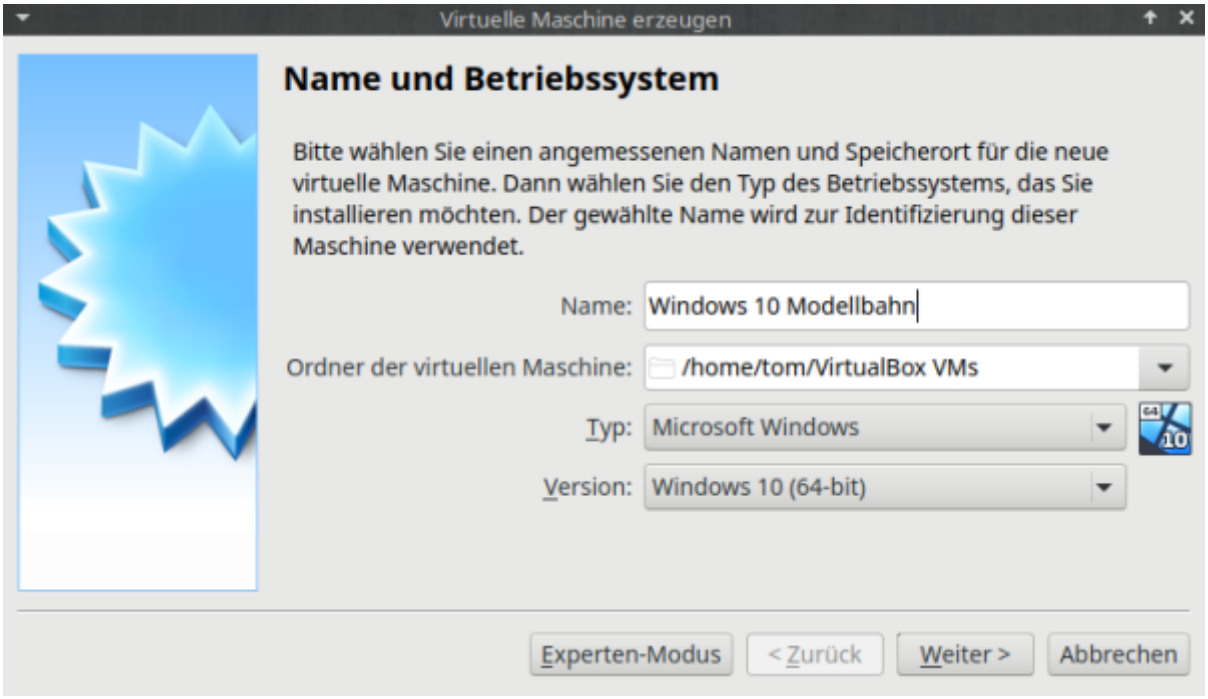


Nach der Installation und dem Aufruf aus dem Menüsystem stellt sich VirtualBox so dar (wenn man keine virtuellen Maschinen installiert hat, ist die Liste natürlich leer).



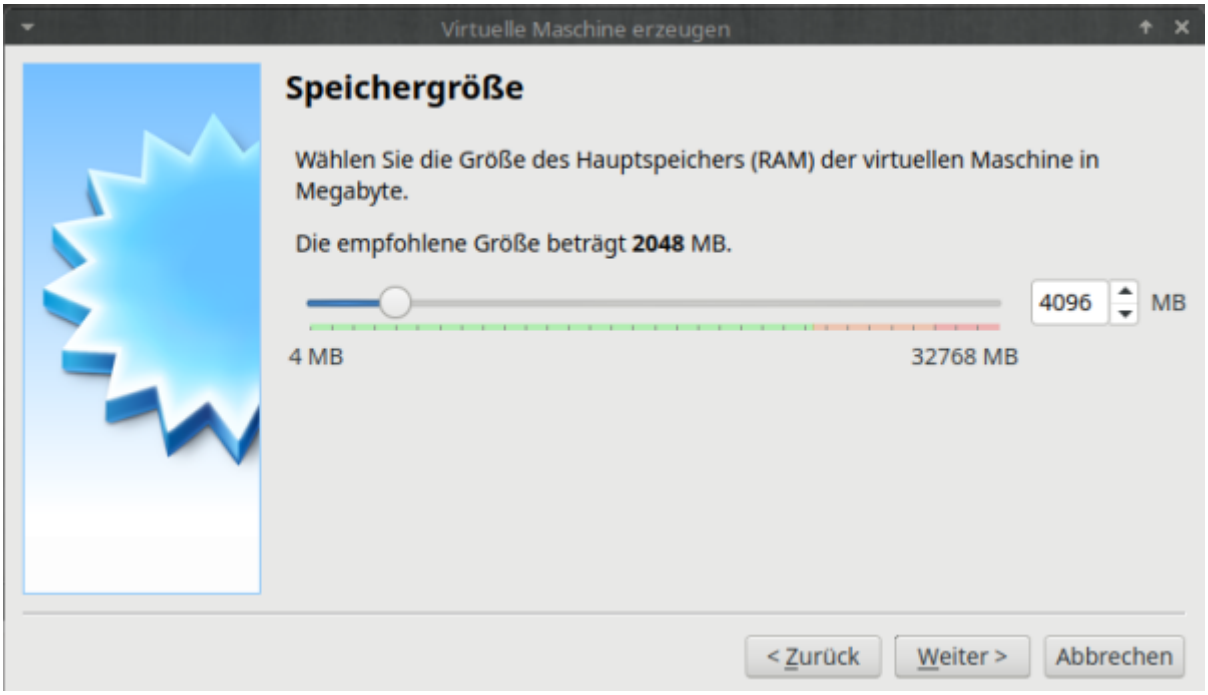
Virtuelle Maschine erstellen

Um nun eine neue virtuelle Maschine zu erstellen muss man nur auf den Button Neu klicken. Als Betriebssystem wählt man das für diese virtuelle Maschine vorgesehene System und gibt der virtuellen Maschine einen Namen. Für die virtuellen Maschinen legt man sich am besten einen Ordner an, in dem diese angelegt werden.

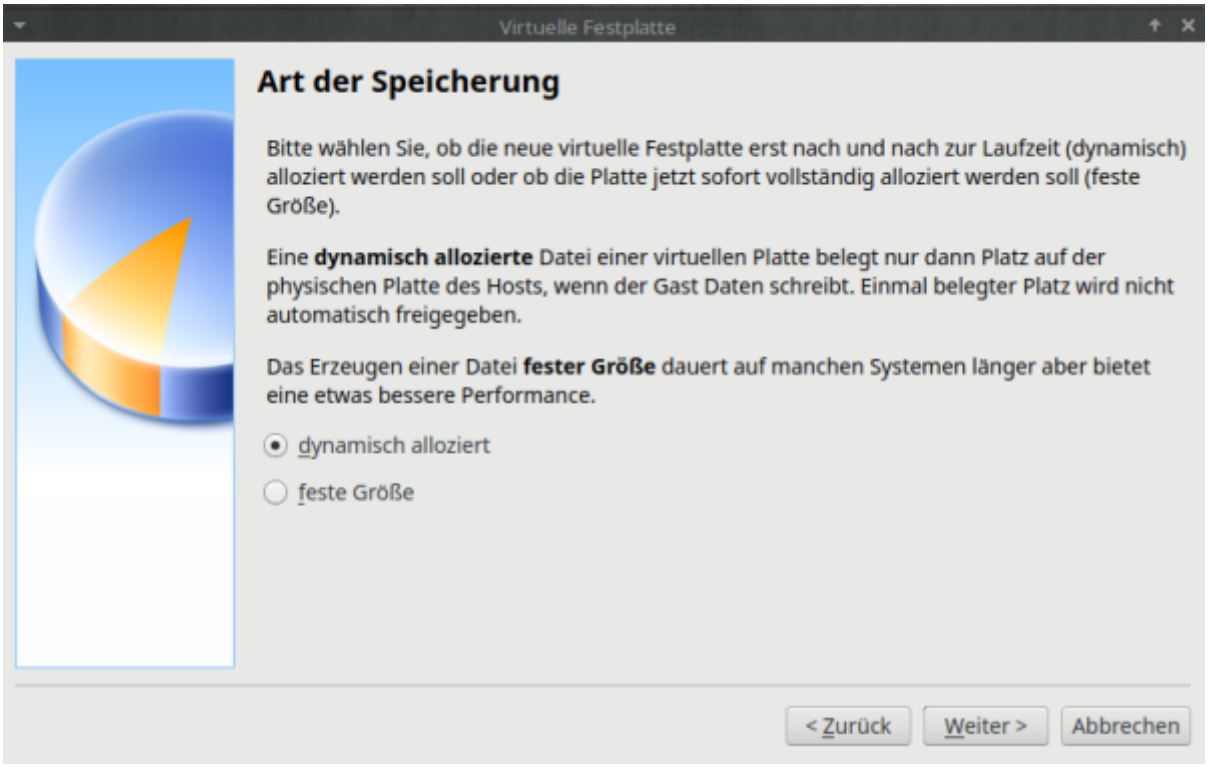


Mit einem Klick auf Weiter geht es dann, wer hätte das gedacht, weiter. Im folgenden definiert man die Hardware für die virtuelle Maschine, wobei einige Werte bereits vorgeschlagen werden. Allerdings halte ich 2GByte Speicher für eine virtuelle Maschine mit 64 Bit Windows 10 für zu wenig, wer aber nicht soviel Speicher im Rechner hat, lässt das erstmal so. Alle diese Werte können auch nachträglich geändert werden! Nur bei der Größe der Festplatte muss man wissen, wie man das macht, also besser nicht einfach loslegen und die Platte vergrössern. Möglich ist das aber auch und wenn man weiß wie, auch absolut problemlos. Aber weiter ...

Den Speicher also auf 4GByte einstellen (4096) einstellen und dann weiter:

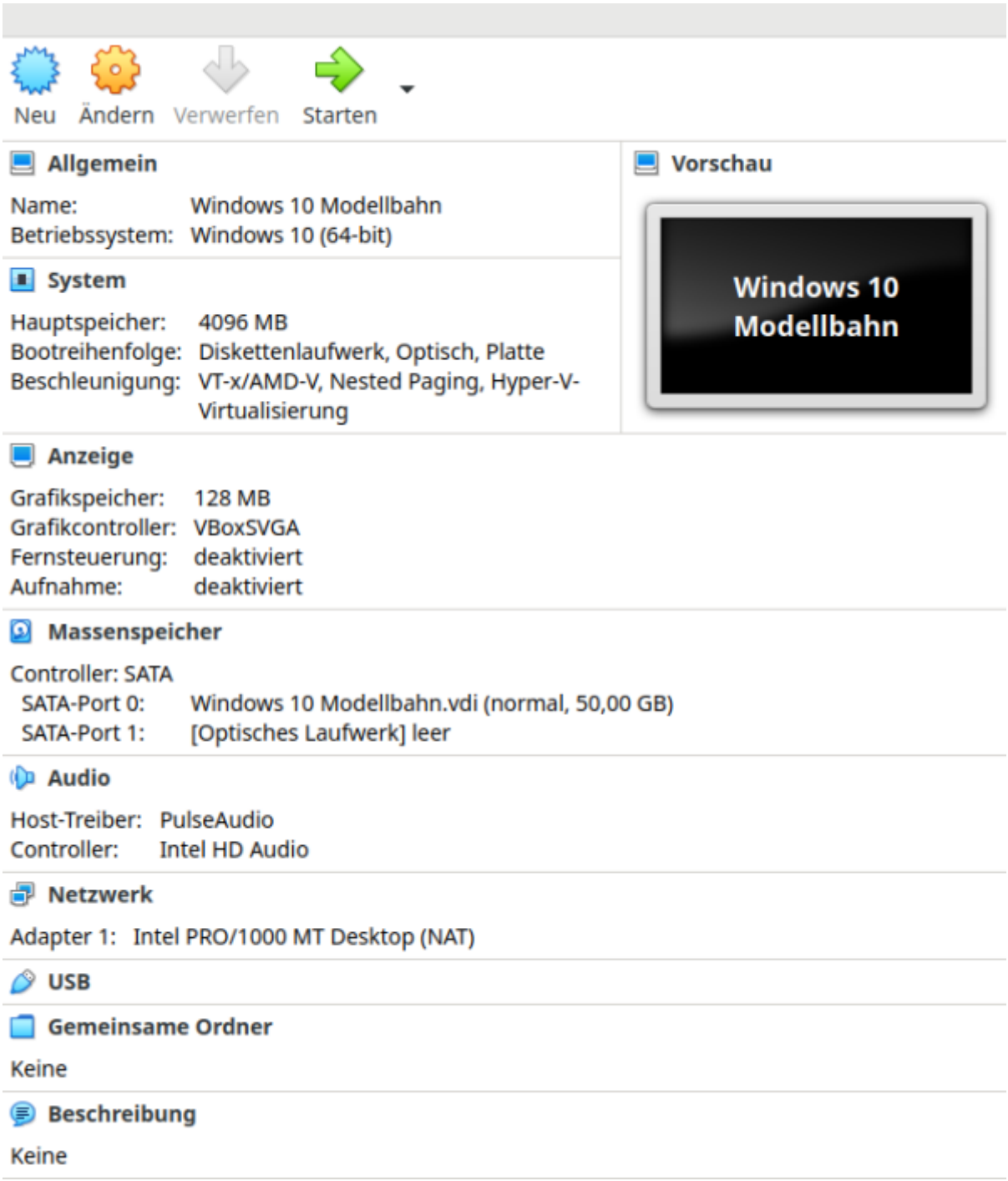


Da keine bereits bestehende virtuelle Festplatte eingesetzt werden soll, wird im nächsten Schritt "Festplatte erzeugen" ausgewählt. Im nächsten Fenster kann man den Dateityp für die Festplatte auswählen, einfach auf VDI stehen lassen. Nun kommt ein Punkt, den man nach Geschmack auswählt:

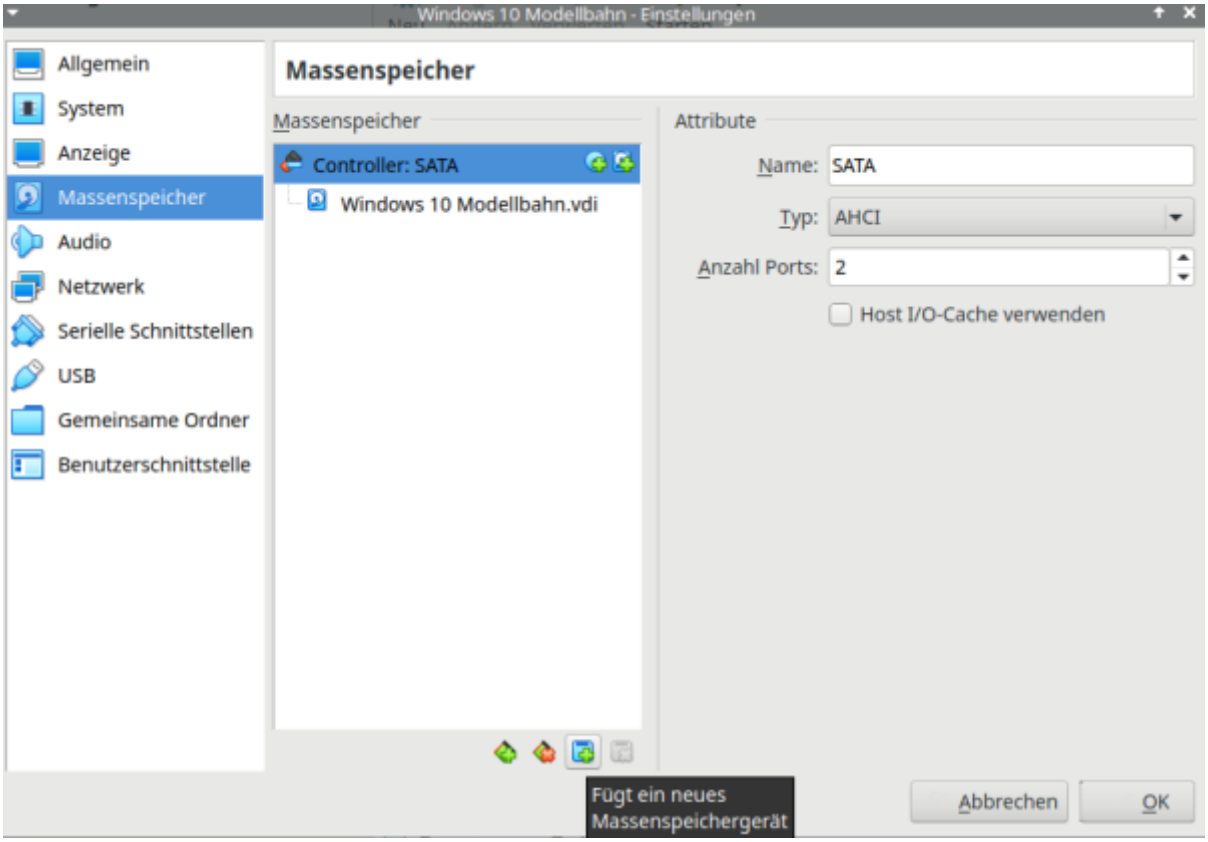


Die Unterschiede stehen im Text in dem Fenster, ich nutze, da ich wirklich viel Plattenplatz im Hostsystem habe, immer eine fest Größe, weil die Performance etwas besser ist. Nun wird noch der Pfad zur virtuellen Festplatte angegeben und die Größe der virtuellen Festplatte. Beides muss nicht geändert werden, der Pfad passt zum am Anfang erstellten Ordner für die virtuellen Maschinen und 50GByte sind für Windows 10 in der Regel ausreichend, da die Daten später auch nicht in der virtuellen Maschine gespeichert werden. Dazu komme ich dann aber später.

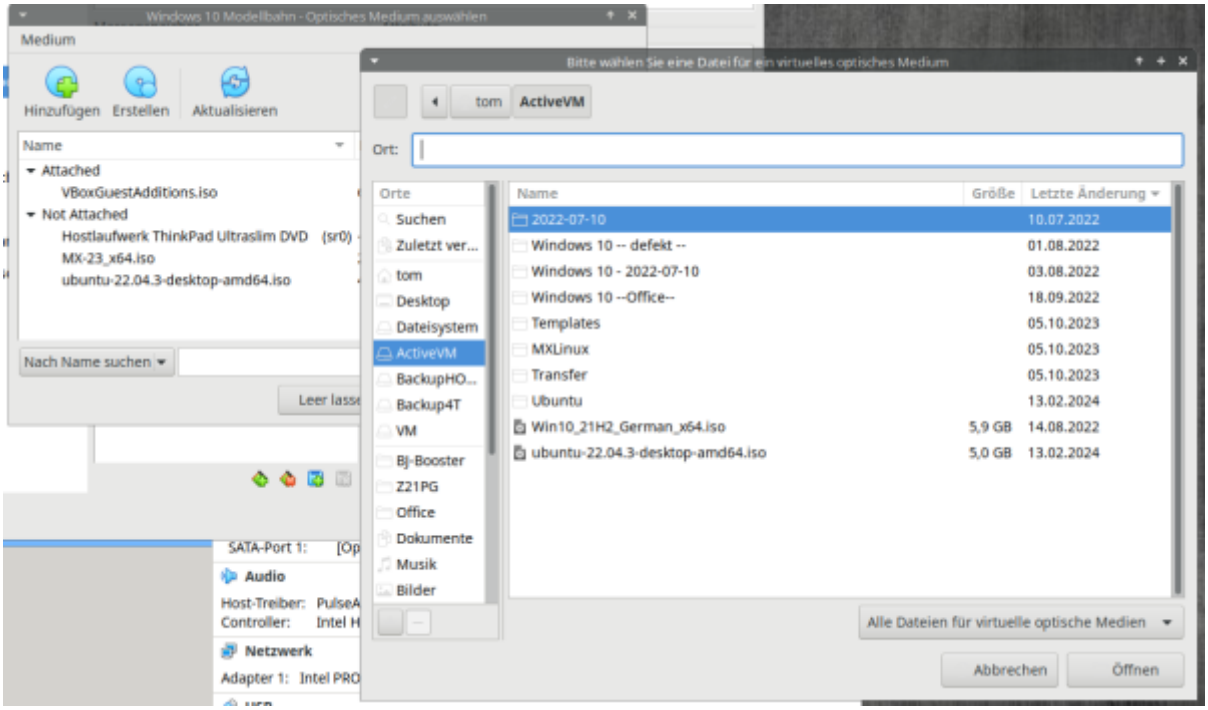
Klickt man nun auf "Erzeugen" wird die virtuelle Maschine erstellt. Man sieht dann in der Übersicht alle Parameter für diese virtualisierte Hardware.



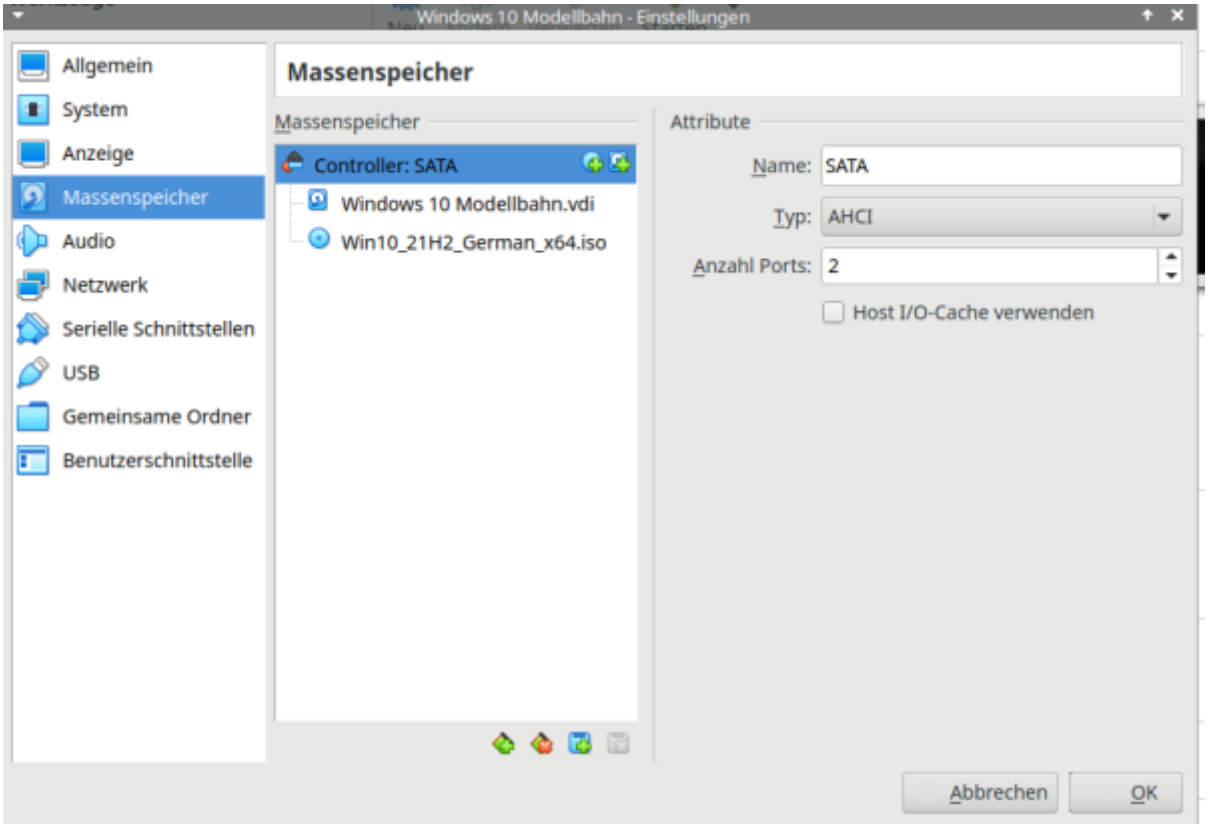
Jetzt kommt das Finetuning in dem man auf den "Ändern" Button klickt. Denn wir müssen der virtuellen Maschine noch ein Installationsmedium geben, also von wo das Betriebssystem installiert werden soll. Dazu lädt man sich von Microsoft ein ISO Image von Windows 10. Ich habe das bei mir sowieso im Archiv und muss das so nur auswählen. Das geht unter dem Punkt Massenspeicher. Hier sollte man das evtl. vorhandene optische Laufwerk entfernen und dann über das + unten in der Leiste ein neues optisches Laufwerk hinzufügen.



Dort wählt man dann das optische Laufwerk aus und klickt im folgenden Fenster auf "Hinzufügen" und wählt das heruntergeladene ISO Image aus und klickt auf "Öffnen".



Nun markiert man das ISO Image, welches unter Not Attached steht und klickt auf "Auswählen". Diese erscheint dann in der Liste der Massenspeicher.



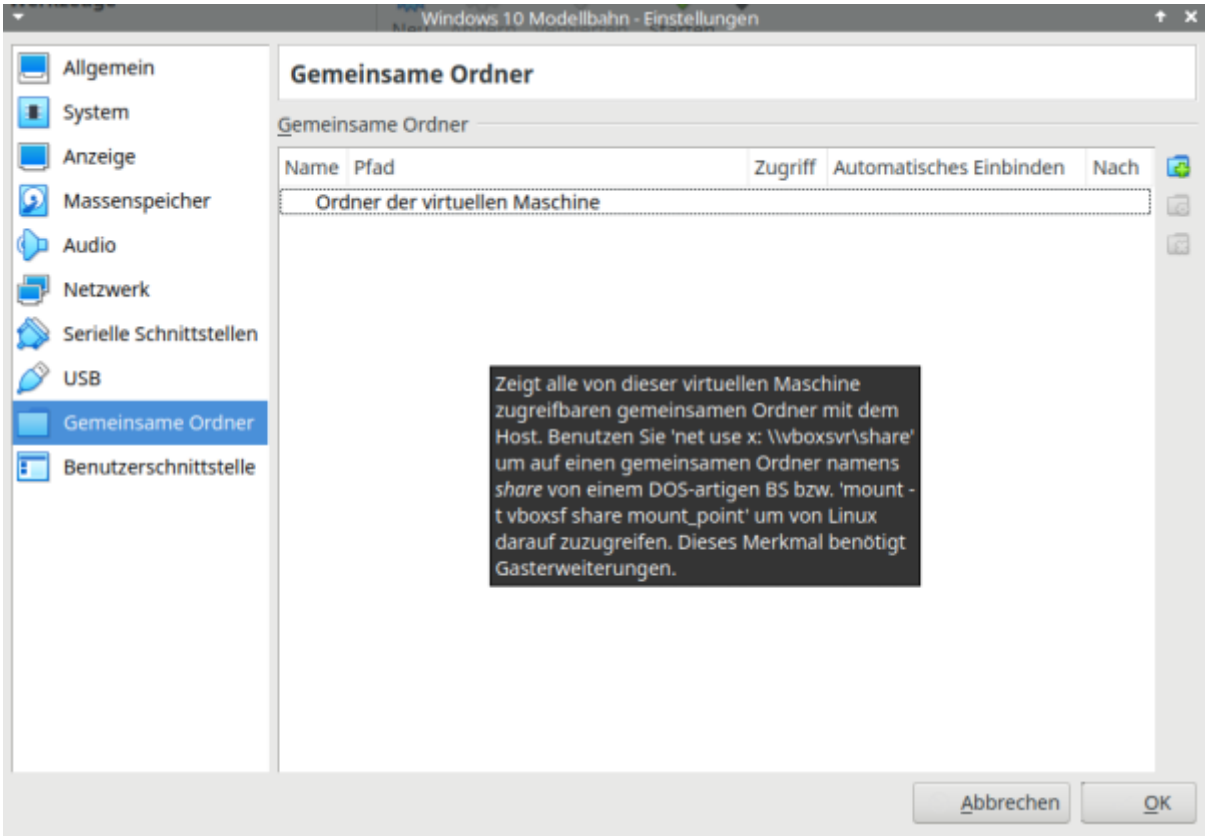
Hier markiert man das ISO Image und setzt rechts einen Haken bei Live-CD/DVD. Damit sind die Einstellungen für die Massenspeicher fertig. Nun geht man erneut auf "Ändern" und dann System. Hier kann eingestellt werden wie viele Prozessoren (Kerne) die virtuelle Maschine bekommen soll. Ich habe der mal 4 Kerne gegeben, mein i7 im Rechner hat 16. Das muss man halt etwas nach Gefühl anpassen, 2 Kerne sind aber schon sinnvoll.

Alle anderen Einstellungen bzgl. der Hardware können vorerst so bleiben, man kann die jederzeit ändern und benötigt dafür keinen Schraubendreher 😉

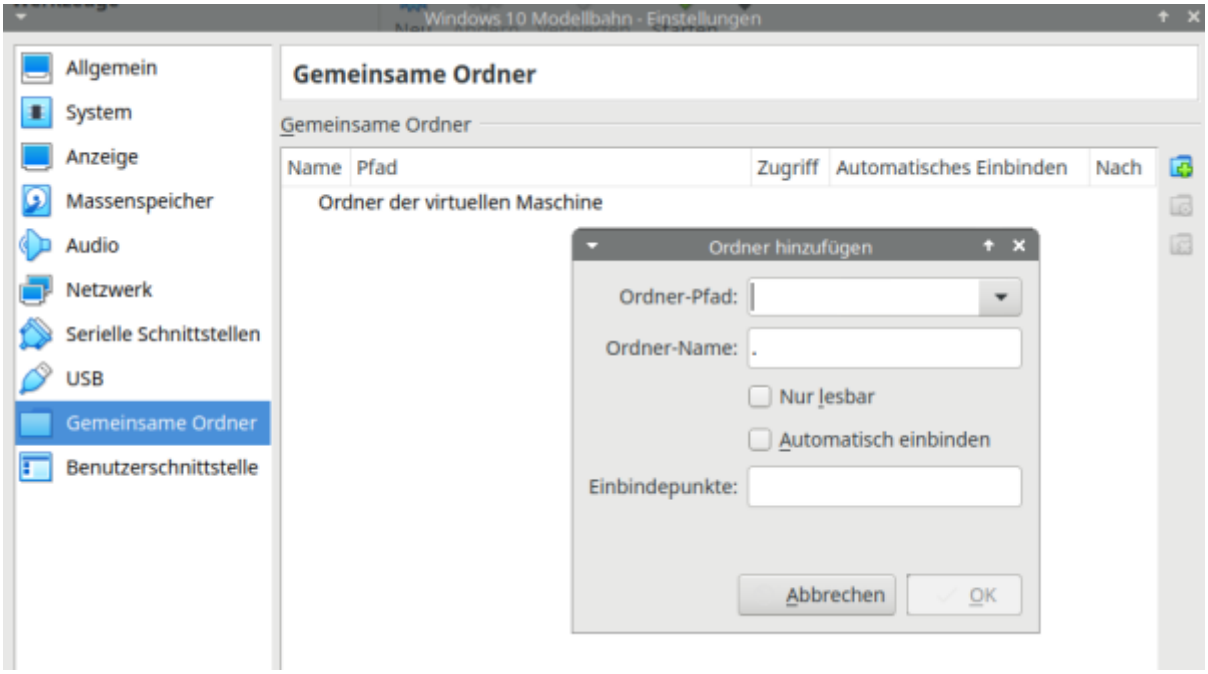
Gemeinsamer Ordner

Eine letzte Einstellung vor dem ersten Start der virtuellen Maschine nehmen wir jetzt aber noch vor. Um einen Datenaustausch zwischen dem Hostsystem und dem Gastsystem zu ermöglichen, kann ein gemeinsamer Ordner festgelegt werden.

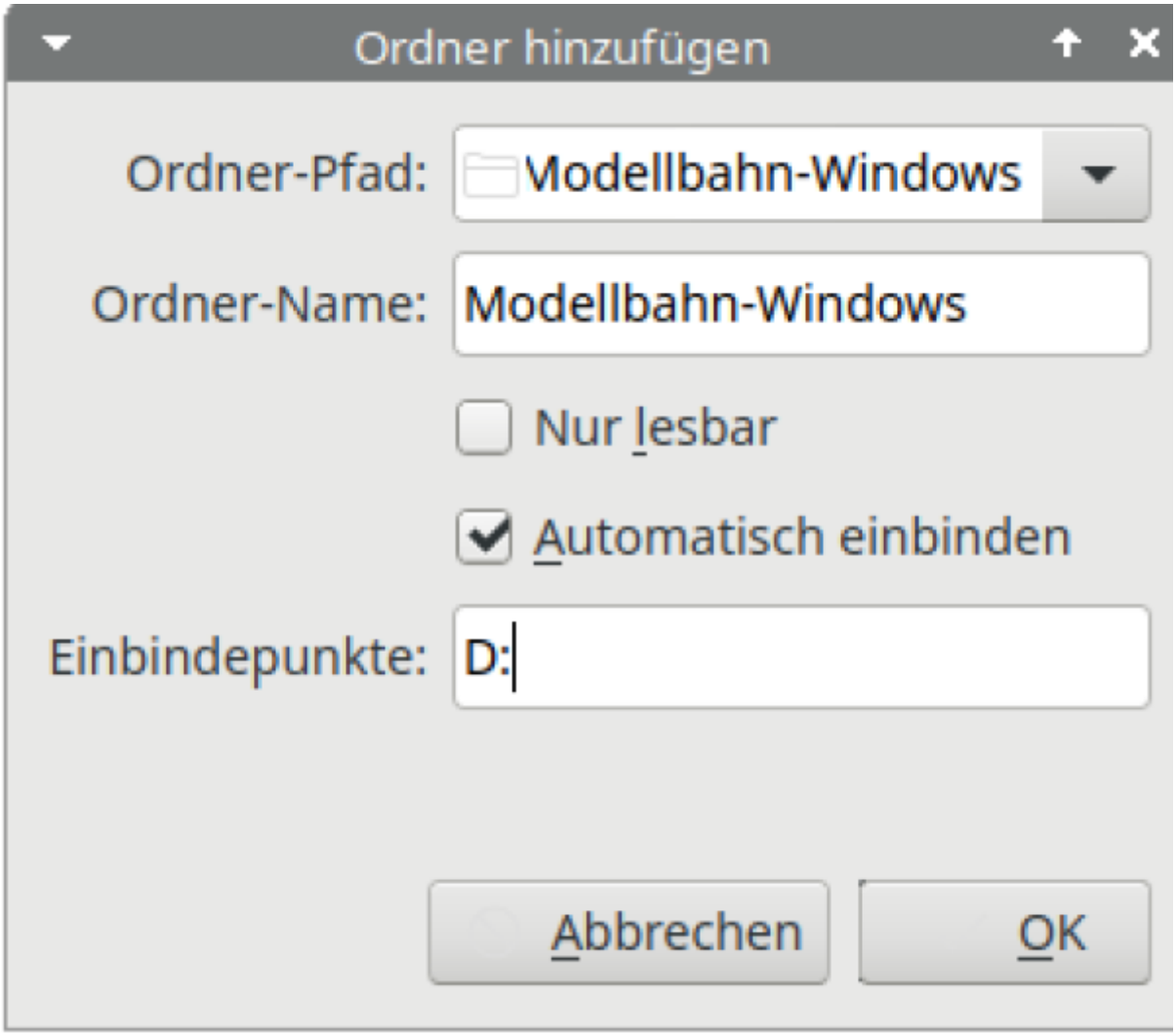
Wenn man ein Betriebssystem wie Windows XP oder Windows 7 als virtuelle Maschine verwendet, können über diesen gemeinsamen Ordner Daten ausgetauscht werden und auch Installationspakete mit dem Hostsystem (MX-Linux) heruntergeladen werden und in diesen gemeinsamen Ordner abgelegt werden. Diese stehen dann im Gastsystem (Windows) zur Verfügung und man muss mit dem Gastsystem gar nicht ins Internet, was bei so alten Systemen durchaus sinnvoll ist, diese nur ausschließlich lokal zu verwenden



Mit einem Klick auf das kleine + rechts, kann ein neuer gemeinsamer Ordner angelegt werden.

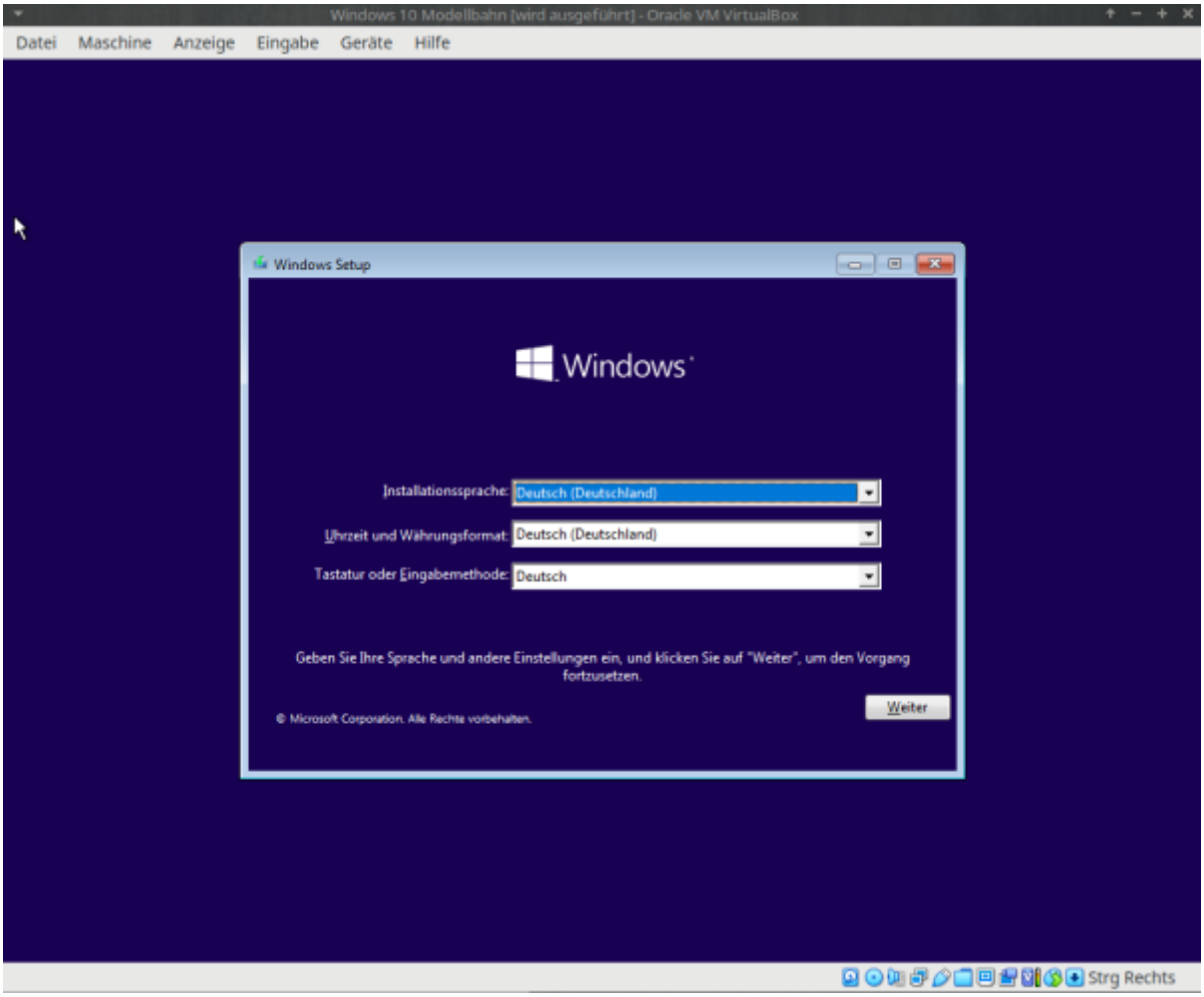


Hier gibt es noch ein Sicherheitsfeature, die Checkbox "Nur lesbar". Wenn diese ausgewählt wird, kann das Gastsystem nur lesend auf den Ordner zugreifen. Unter Einbindepunkte kann ein Laufwerksbuchstabe definiert werden, unter dem dieser gemeinsamer Ordner im Windows in der virtuellen Maschine erscheint.



Der erste Start

So, nun ist soweit erst mal alles eingestellt, dass die virtuelle Maschine das erste Mal gestartet werden kann, oben im Menü mit dem dicken grünen Pfeil.

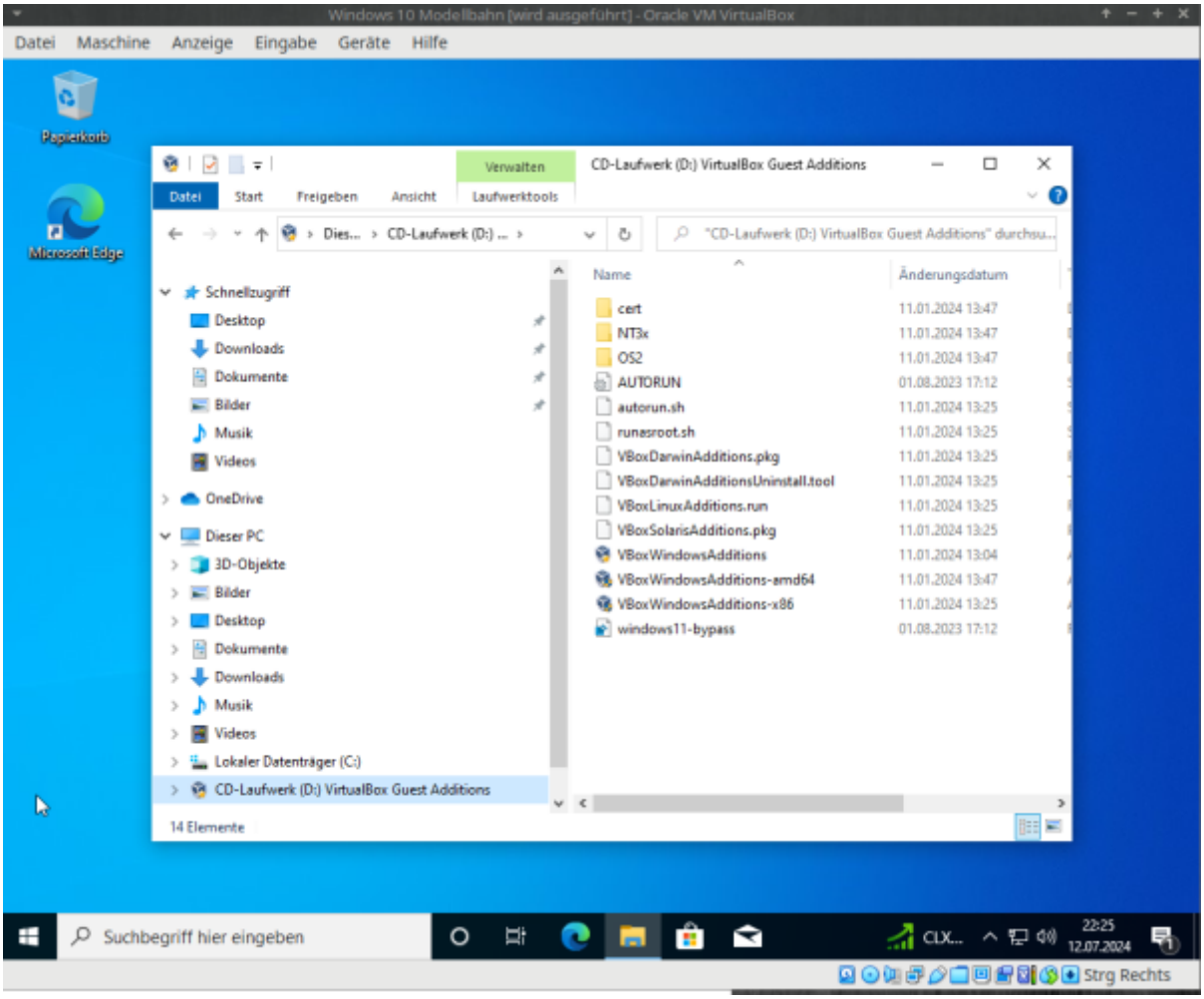


Wenn man in dem Fenster der virtuellen Maschine ist, mit dem Mauszeiger, kommt man da nur wieder heraus, indem man die Host-Taste betätigt, das ist in den Standardeinstellungen die rechte STRG (CTRL) Taste

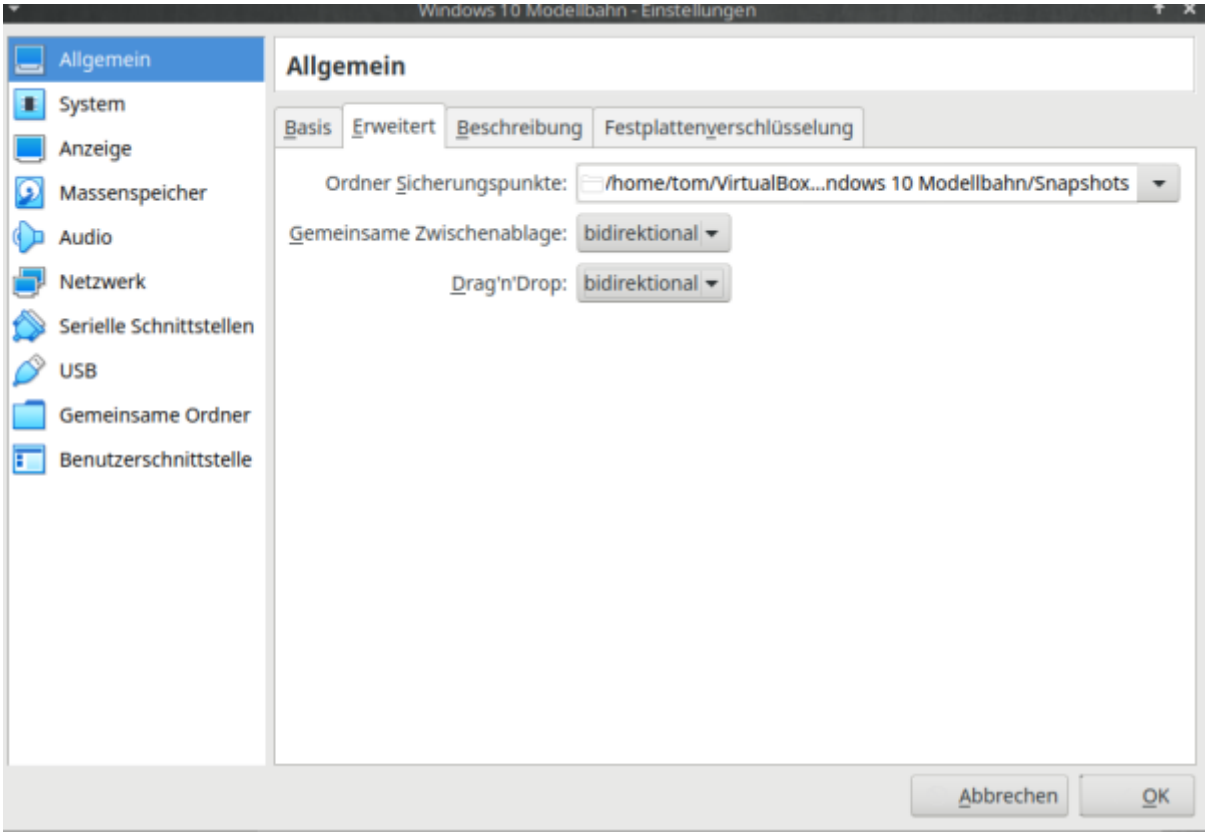
Eigentlich kann ich hier jetzt aufhören, denn nun ist es einfach nur eine Windows Installation auf einem PC  Die weitere Installation beschreibe ich nun nicht weiter, aber es ist natürlich ein Product-Key für Windows erforderlich. Die virtuelle Maschine ist einfach nur ein PC, auf dem man Windows installiert, somit ist auch zwingend dafür eine Lizenz erforderlich. Die Installation kann aber erstmal ohne Product-Key erfolgen, Windows wird später oft genug dazu auffordern, einen Product-Key einzugeben, dann kann dies nachgeholt werden. Ansonsten läuft die Installation nun genauso ab, als hätte man einen PC aus Blech vor sich.

Details

Damit die virtuelle Maschine im Host-System richtig eingebettet erscheint, sollten die VirtualBox Guest Additions installiert werden. Diese sind auch nötig, wenn man die Zwischenablage zwischen Host und Gast bidirektional verwenden möchte. Für die Installation der Guest-Additions gibt es einen Menüpunkt unter Geräte: "Gasterweiterungen einlegen". Dann erscheint im Explorer von Windows (in der virtuellen Maschine) ein Laufwerk. Dort findet man die Datei `VBoxWindowsAdditions-amd64` die man dann installiert.



Nach der Installation und dem Neustart kann das Fenster mit der virtuellen Maschine fließend angepasst werden, in den Seamless-Mode geschaltet werden (die laufenden Windows-Anwendungen integrieren sich dann in den Linux-Desktop), Vollbild, Fenstermodus usw. kann umgeschaltet werden. Evtl. muss die Auflösung von Windows über rechte Maustaste, Anzeigeeinstellungen (in der VM) noch höher gestellt werden, denn wenn diese nur auf 1024x768 steht, wird das natürlich nicht größer, auch wenn man das Fenster der virtuellen Maschine auf Fullscreen schaltet. Nun wird die virtuelle Maschine heruntergefahren, um noch ein paar Details einzustellen. Dazu in der VirtualBox wieder auf "Ändern" klicken und dann unter Allgemein die gemeinsame Zwischenablage und Drag'n'Drop auf bidirektional stellen.



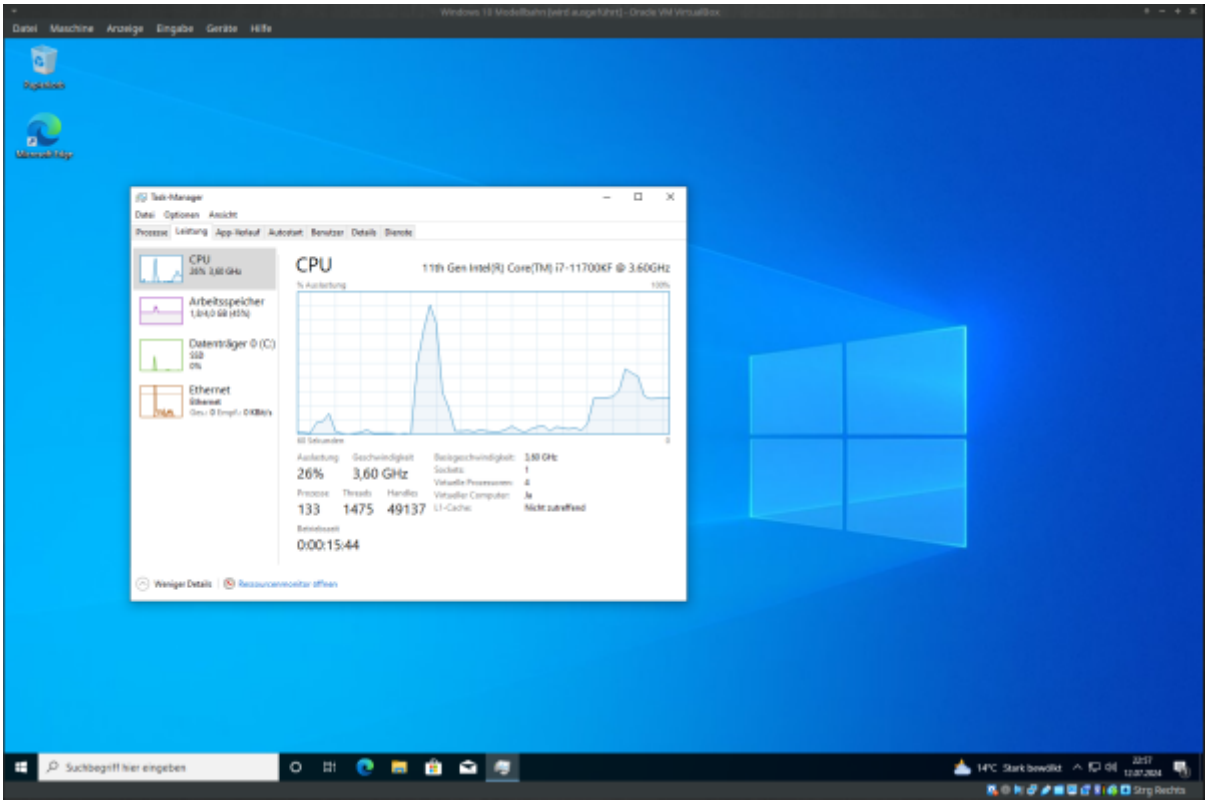
Auch das CD/DVD Laufwerk kann nun ganz entfernt werden, unter Massenspeicher. Das wurde nur für die Installation vom ISO Image benötigt. Die Netzwerkeinstellungen könnten noch angepasst werden, aber NAT, die Standardeinstellungen, sind auch ok.

Hat der Host-Rechner serielle Schnittstellen, können diese unter serielle Schnittstellen aktiviert werden und stehen dann auch im Gastsystem zur Verfügung. Werden für serielle Verbindungen nur USB-Seriell Konverter verwendet, ist das nicht nötig, USB ist in den Standardeinstellungen bereits vorhanden.

Unter den Anzeigeeinstellungen kann noch die 3D Beschleunigung aktiviert werden, wenn die Grafikkarte dies unterstützt. Das merkt man sehr schnell und schaltet das dann einfach wieder ab. Eigentlich ist die virtuelle Maschine mit Windows 10 nun soweit fertig, dass Software installiert werden kann.

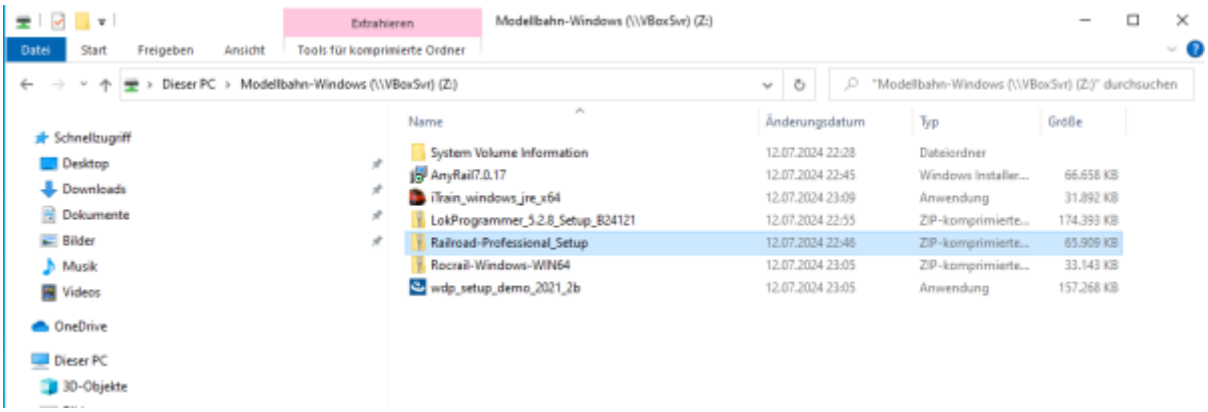
Wie jetzt hier mit Windows 10 als Beispiel gezeigt, geht das natürlich auch mit allen anderen Windows-Versionen, auch mit Windows 3.11, Windows 95 usw.

Wer möchte, sichert sich nun den Ordner mit der virtuellen Maschine. Im Falle eines Falles kopiert man einfach diesen Ordner wieder zurück und hat sofort wieder eine lauffähige virtuelle Maschine mit Windows. Ausserdem sollten noch die persönlichen Windowseinstellungen gemacht werden, Farben, Schriften usw. Aber das ist nichts anderes, als wie auf einer Blechkiste, kein Unterschied.



Modellbahnsoftware

Für die Installation der gewünschten Software müssen die Installationspakete in der virtuellen Maschine zur Verfügung stehen. Das geht am einfachsten, in dem man diese in dem gemeinsamen Ordner ablegt. Bei mir erscheint der gemeinsame Ordner nun als Laufwerk Z: in der virtuellen Maschine.



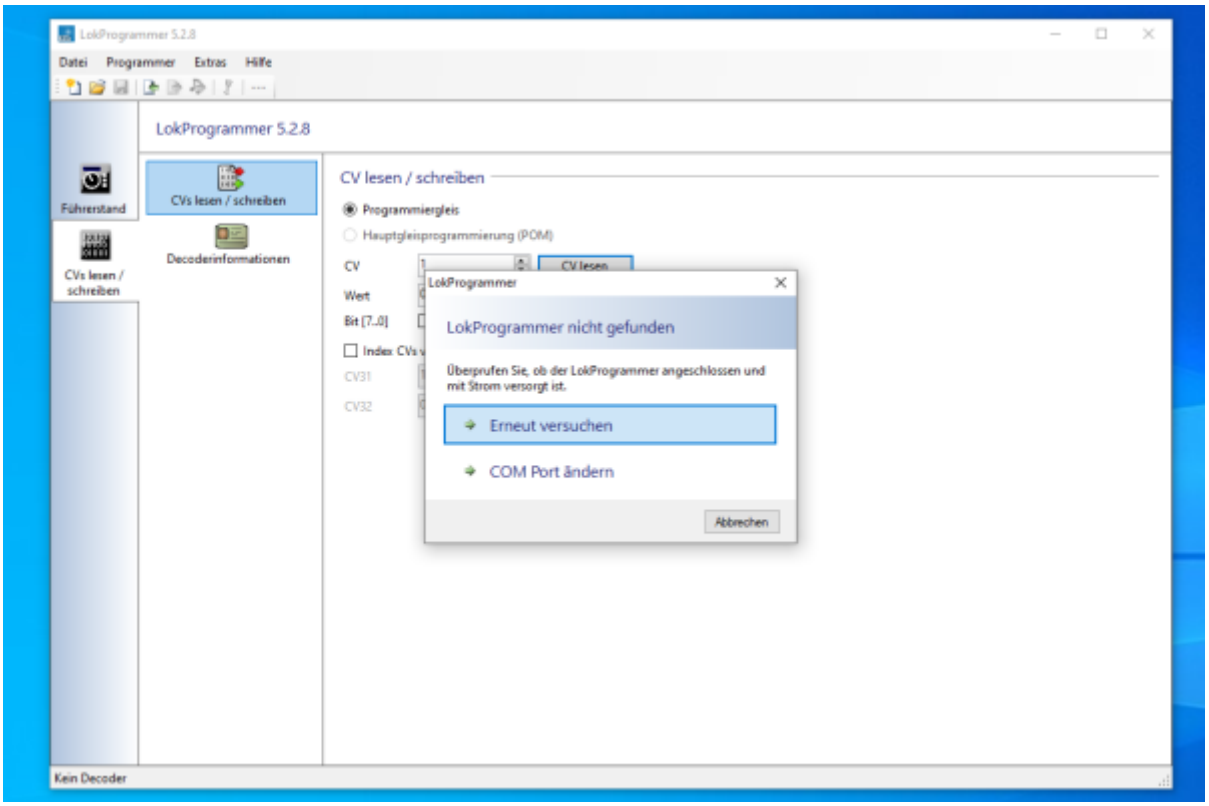
Von hier aus kann nun jede Software auch installiert werden. Manche Installationen funktionieren nicht von entfernten Laufwerken, dann muss das jeweilige Setup-Programm auf C: kopiert werden. Dies betrifft direkt ausführbare Setups, das sind Sicherheitseinstellungen von Windows und der gemeinsame Ordner ist für Windows kein lokales Laufwerk.

ESU LokProgrammer

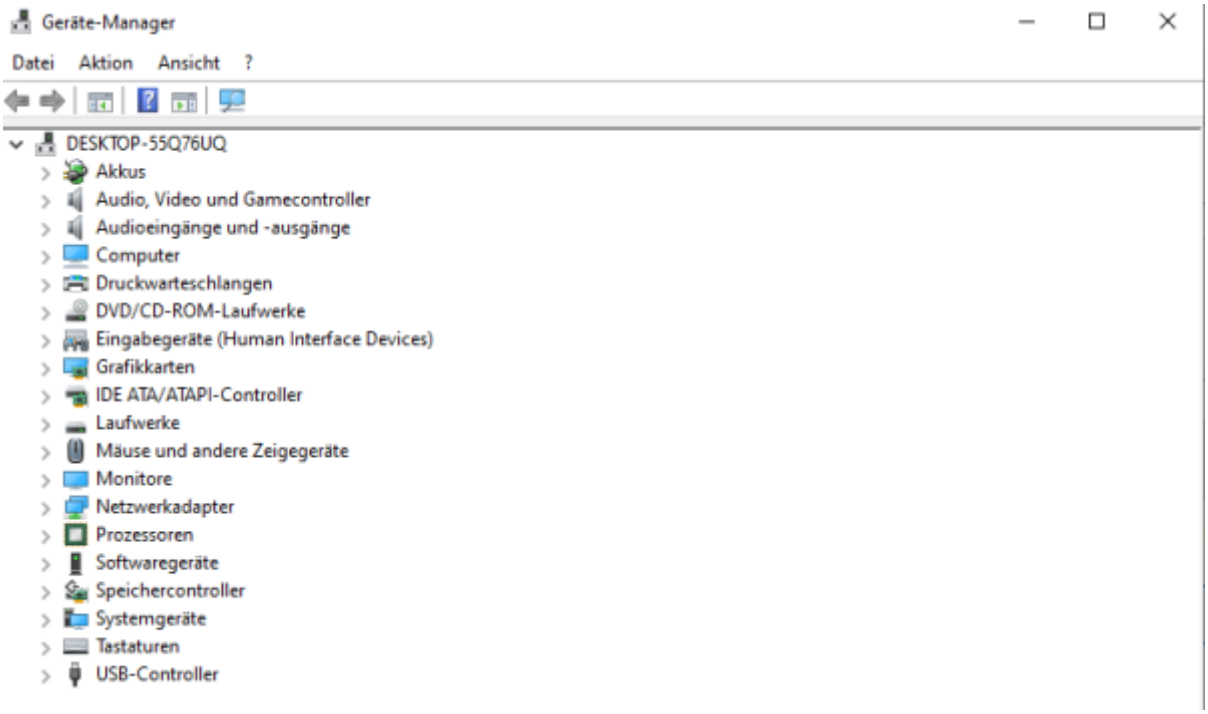
Als Beispiel für die Nutzung von seriellen Schnittstellen dient hier der ESU LokProgrammer, der mit

einem USB-Seriell Adapter geliefert wird. Der im Adapter enthaltene FTDI Chip wird von Linux sofort erkannt. Wie man das unter Linux nachschaut, erläutere ich noch in einem gesonderten Beitrag 🔧 Fix Me!.

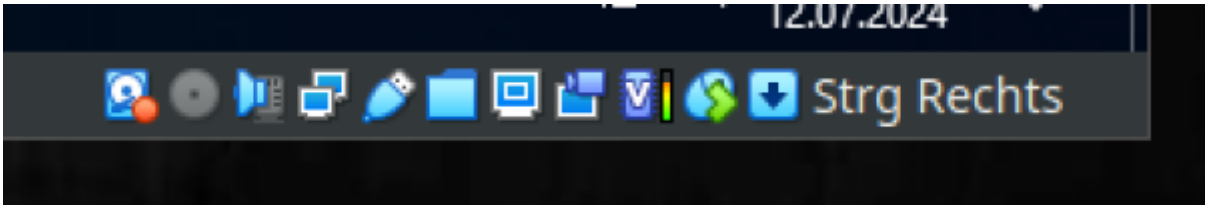
Zuerst muss die Software für den ESU LokProgrammer unter Windows in der virtuellen Maschine installiert werden. Dazu lädt man mit Linux die Software von der ESU Webseite herunter, speichert diese in dem gemeinsamen Ordner und kopiert die unter Windows von dort nach Laufwerk C:. Da es sich bei dem Setup-Programm um eine ausführbare Datei handelt, lässt sich diese nicht direkt in dem gemeinsamen Ordner ausführen. Nach der Installation hat man das Icon vom LokProgrammer auf dem Desktop und kann die Software ganz normal starten. Versucht man nun aber einen Dekoder auszulesen, erhält man die Fehlermeldung, dass der LokProgrammer nicht gefunden wurde.



Das liegt nun daran, dass Linux zwar den LokProgrammer kennt (das ist nicht ganz korrekt, Linux kennt den USB-Seriell Adapter vom LokProgrammer), aber die virtuelle Maschine hat gar keinen COM-Port, bzw. kennt den USB Seriell Adapter nicht, weil dieser nicht mit der virtuellen Maschine verbunden ist. Dies lässt sich auch im Gerätemanager sehen, da ist kein COM-Port.



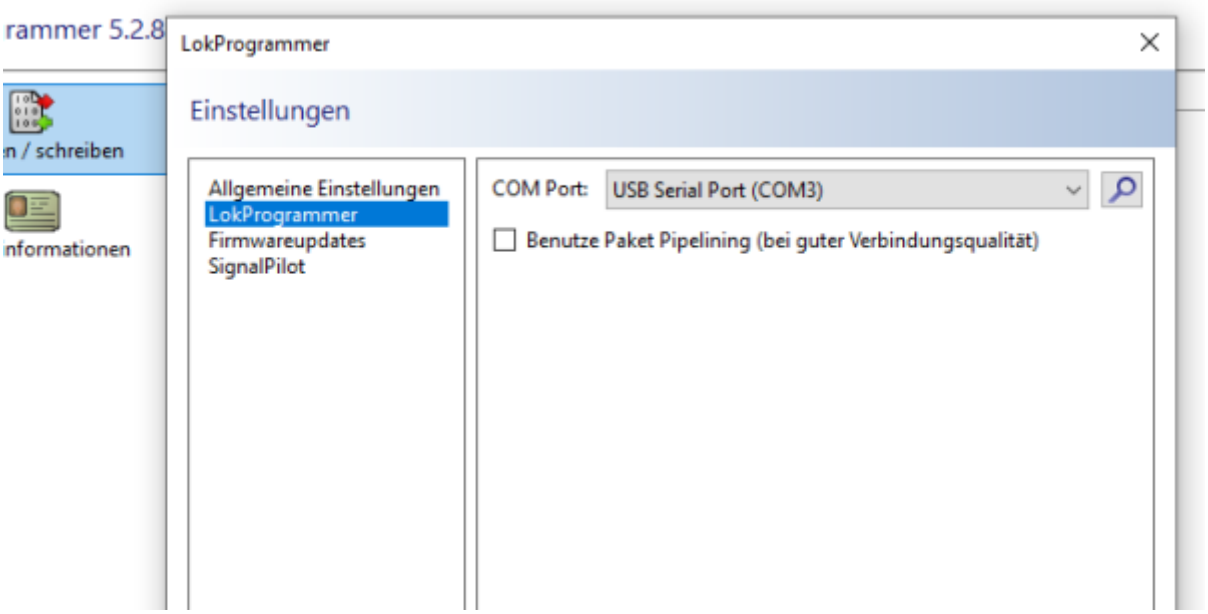
Also muss nun der USB-Seriell Adapter mit der virtuellen Maschine verbunden werden. Da kann man über die kleinen Icons machen, die in der Fensterdarstellung der virtuellen Maschine zu sehen sind.



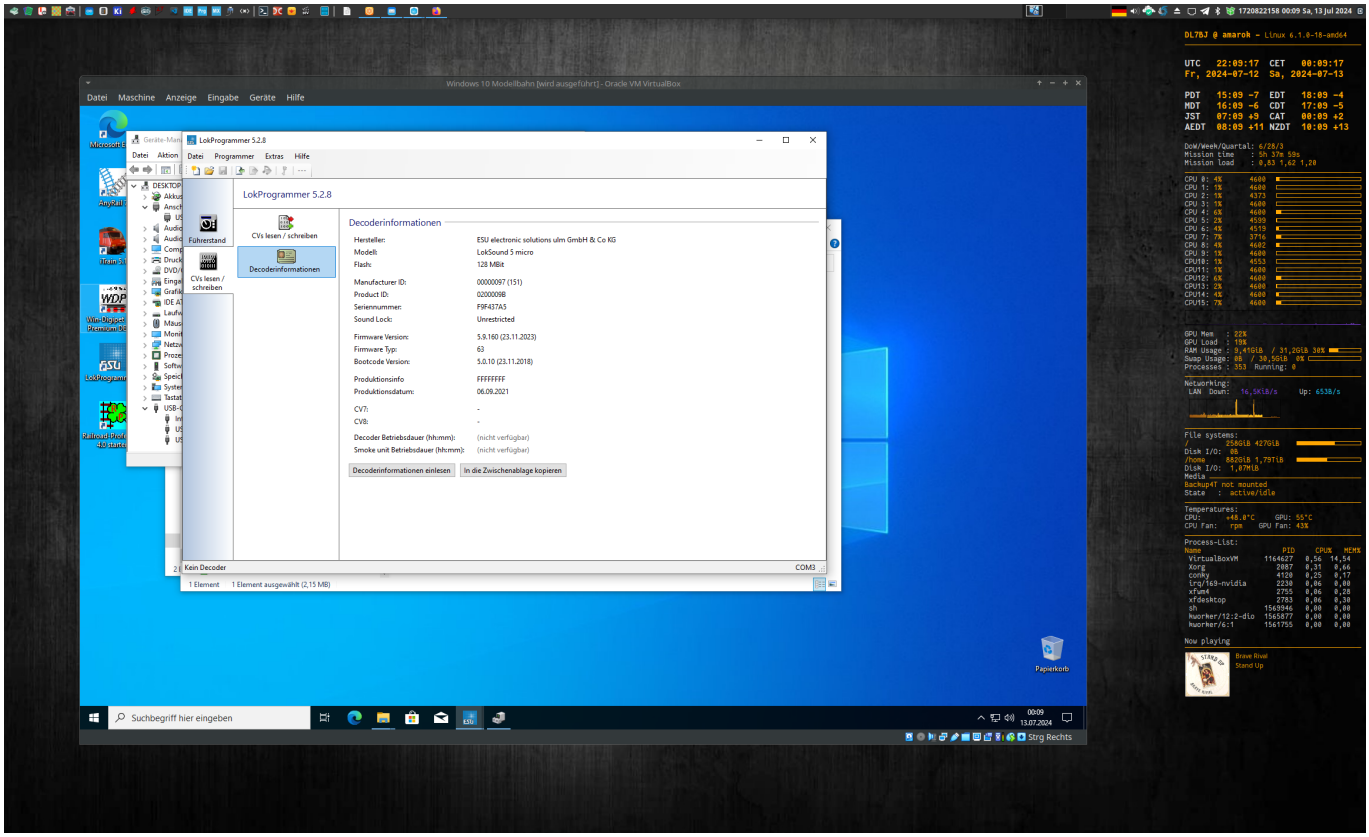
Hat man die virtuelle Maschine im Vollbild-Modus laufen, gibt es die Icons nicht, dann muss man über die Menüleiste am oberen Bildschirmrand gehen und unter Geräte schauen, unter dem Menüpunkt USB. Also entweder mit der rechten Maustaste auf das Icon-Symbol für USB (im obigen Bild das 5. von links) oder über das Geräte Menü. Verbindet man nun das dort erscheinende FTDI FT232R Gerät mit der virtuellen Maschine erscheint auch kurz darauf im Gerätemanager ein COM-Port oder aber nur der Hinweis, das Windows keinen Treiber für das Gerät kennt. Dann muss man den Treiber für Windows 10 bei FTDI herunterladen und installieren. Das geht genauso, wie man auch sonst unter Windows macht. Ist der Treiber installiert, erscheint im Gerätemanager auch der COM-Port.

Fehlende Treiber hat man unter Windows häufiger und die Installation dieser Treiber unterscheidet sich in keiner Weise, egal ob virtuelle Maschine oder Blechkiste.

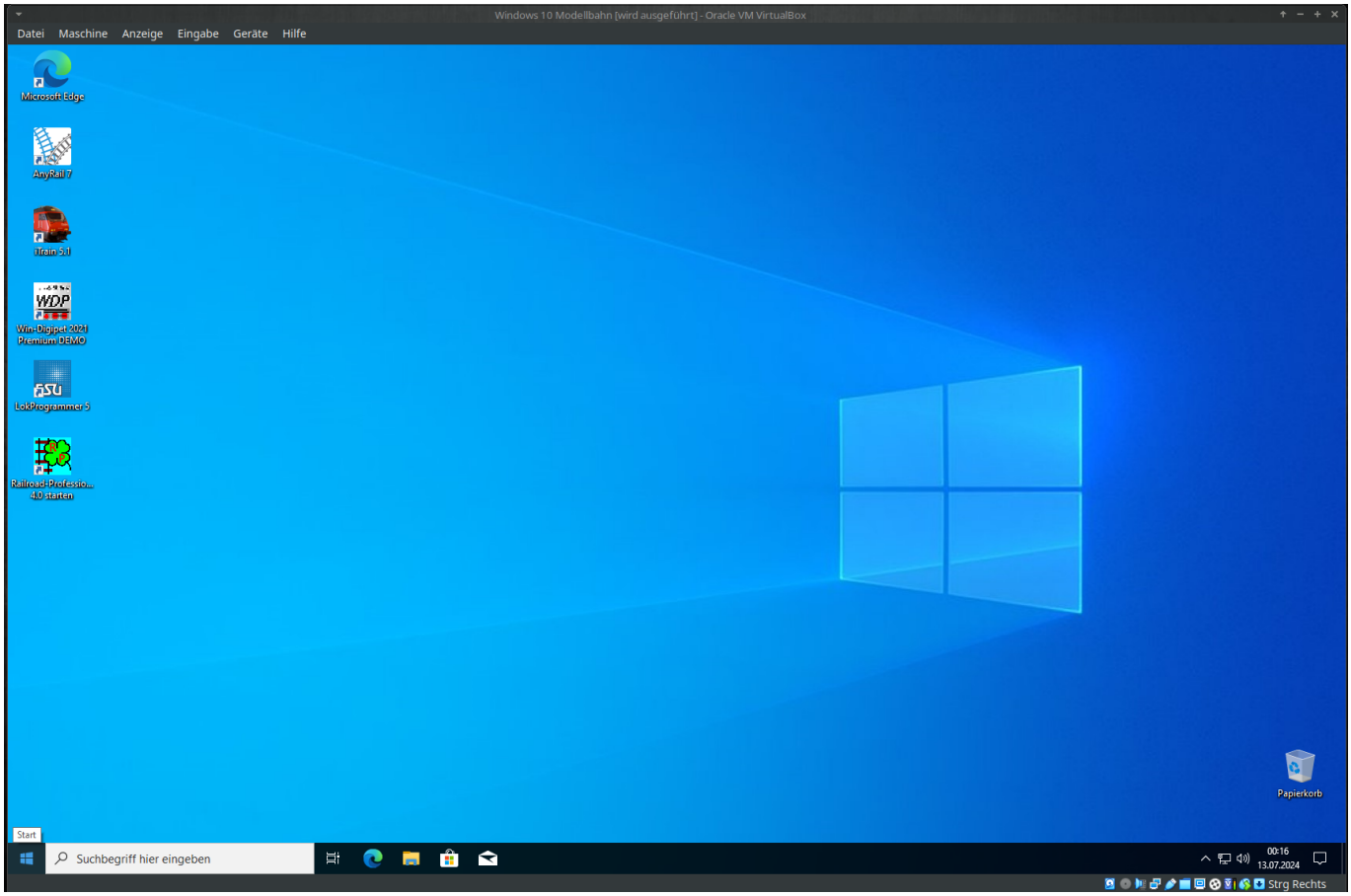
Nun kann im ESU LokProgrammer der korrekte COM-Port, hier ist es COM3, eingestellt werden. Der wird dann auch korrekt als USB Serial Port im LokProgrammer angezeigt.



An den LEDs des LokProgrammers kann man nun auch sofort die Funktion sehen und z.B. mit Decoderinformationen lesen die Funktion testen. Hier mal ein extra großer Screenshot der gesamten virtuellen Maschine auf dem MX-Linux Desktop (anklicken zum Vergrössern).



Fortsetzung folgt ...



Übrigens, wenn man das Fenster der virtuellen Maschine einfach oben rechts schließt, gibt es die Frage, ob die Maschine ausgeschaltet werden soll oder der akt. Zustand gespeichert werden soll. Wählt man den Zustand speichern, was nur ein paar Sekunden dauert und ruft die virtuelle Maschine wieder auf, kann man direkt an der Stelle weitermachen, wo man aufgehört hat.