# Anweisungen für das Jetson Orin Nano Developer Kit

## 1. Image auf die microSD-Karte schreiben

Um Ihr Jetson Orin Nano Developer Kit einzurichten, müssen Sie die microSD-Karte mit dem richtigen Image vorbereiten. Folgen Sie diesen Schritten:  
  
1. Das Jetson Orin Nano Developer Kit-Image herunterladen:  
 - Besuchen Sie das Jetson Download Center und laden Sie das neueste SD-Karten-Image für das Jetson Orin Nano Developer Kit herunter.  
 - Die heruntergeladene Datei liegt normalerweise im `.zip`-Format vor. Entpacken Sie die Datei, um die `.img`-Datei zu erhalten.  
  
2. microSD-Karte in den Host-Computer einlegen:  
 - Verwenden Sie einen microSD-Kartenleser, um die Karte in Ihren Computer einzulegen.  
 - Stellen Sie sicher, dass die microSD-Karte von Ihrem Betriebssystem erkannt wird.  
  
3. Image auf die microSD-Karte schreiben:  
 - Nutzen Sie ein Tool, um das Image auf die Karte zu flashen. NVIDIA empfiehlt die Verwendung von Balena Etcher, das für Windows, Mac und Linux verfügbar ist.  
  
 Schritte mit Balena Etcher:  
 - Installieren und öffnen Sie Balena Etcher.  
 - Wählen Sie die zuvor entpackte `.img`-Datei aus.  
 - Wählen Sie die microSD-Karte als Zielgerät aus.  
 - Klicken Sie auf die Schaltfläche „Flash“, um das Image auf die Karte zu schreiben.  
  
4. Image überprüfen:  
 - Balena Etcher überprüft automatisch die geschriebenen Daten, um sicherzustellen, dass der Vorgang erfolgreich war.  
 - Wenn die Überprüfung fehlschlägt, wiederholen Sie den Flash-Vorgang oder verwenden Sie eine andere microSD-Karte.  
  
5. microSD-Karte sicher entfernen:  
 - Sobald der Flash- und Überprüfungsvorgang abgeschlossen ist, werfen Sie die microSD-Karte sicher aus Ihrem Computer aus.  
 - Vermeiden Sie Datenbeschädigungen, indem Sie die Karte nicht entfernen, während sie verwendet wird.

## 2. Einrichtung und erster Start

Nachdem Sie das Image auf die microSD-Karte geschrieben haben, führen Sie die folgenden Schritte durch, um Ihr Jetson Orin Nano Developer Kit einzurichten und zum ersten Mal zu starten:  
  
1. Verbindungen herstellen:  
 - microSD-Karte einlegen: Setzen Sie die vorbereitete microSD-Karte in den entsprechenden Steckplatz des Jetson Orin Nano Developer Kits ein.  
 - Peripheriegeräte anschließen:  
 - Schließen Sie eine USB-Tastatur und -Maus an die USB-Anschlüsse an.  
 - Verbinden Sie einen Monitor über den DisplayPort-Anschluss.  
 - Netzwerkverbindung herstellen: Falls erforderlich, verbinden Sie ein Ethernet-Kabel mit dem Gigabit-Ethernet-Port für Netzwerkzugang.  
 - Stromversorgung anschließen: Verbinden Sie das mitgelieferte 19V-Netzteil mit dem DC-Eingang des Developer Kits.  
  
2. Erster Start:  
 - Einschalten: Das Developer Kit startet automatisch, sobald die Stromversorgung angeschlossen ist. Eine grüne LED neben dem USB-C-Anschluss leuchtet auf, um anzuzeigen, dass das Gerät eingeschaltet ist.  
 - Ersteinrichtung: Beim ersten Start werden Sie durch die folgenden Schritte geführt:  
 - Überprüfung und Akzeptanz der NVIDIA Jetson Software-Endbenutzer-Lizenzvereinbarung (EULA).  
 - Auswahl der Systemsprache, Tastaturlayout und Zeitzone.  
 - Verbindung mit einem drahtlosen Netzwerk (falls ein WLAN-Modul installiert ist).  
 - Einrichtung eines Benutzerkontos mit Benutzername und Passwort.  
  
3. Nach dem Login:  
 - Systemaktualisierung: Es wird empfohlen, das System auf den neuesten Stand zu bringen. Öffnen Sie ein Terminal und führen Sie die folgenden Befehle aus:  
 sudo apt update  
 sudo apt upgrade  
 - Installation zusätzlicher Software: Je nach Ihren Anforderungen können Sie zusätzliche Pakete oder Entwicklungswerkzeuge installieren.

## 3. Nächste Schritte

Nach der erfolgreichen Einrichtung Ihres Jetson Orin Nano Developer Kits können Sie folgende Schritte unternehmen, um Ihre Entwicklungsumgebung weiter zu optimieren und mit der Entwicklung von KI-Anwendungen zu beginnen:  
  
1. Installation zusätzlicher Software:  
 - Nutzen Sie den Jetson SDK Manager, um zusätzliche NVIDIA-Softwarepakete zu installieren, die für Ihre spezifischen Projekte erforderlich sind.  
 - Besuchen Sie die NVIDIA Jetson Download Center, um die neuesten Versionen von JetPack und anderen relevanten Tools herunterzuladen.  
  
2. Entwicklung von KI-Anwendungen:  
 - Nutzen Sie die Leistungsfähigkeit des Jetson Orin Nano, um KI-Modelle zu entwickeln und zu testen.  
 - Verwenden Sie Frameworks wie TensorFlow oder PyTorch, die auf dem Jetson Orin Nano unterstützt werden.  
  
3. Zugriff auf Ressourcen und Dokumentation:  
 - Besuchen Sie die NVIDIA Developer Website für umfassende Anleitungen, Tutorials und Beispiele.  
 - Nutzen Sie die NVIDIA Foren, um sich mit anderen Entwicklern auszutauschen und Unterstützung zu erhalten.  
  
Diese Schritte helfen Ihnen, das volle Potenzial Ihres Jetson Orin Nano Developer Kits auszuschöpfen und erfolgreiche KI-Projekte zu realisieren.

\*\*NVIDIA Jetson Orin Nano\*\*:  
  
---  
  
### \*\*1. Prozessor (CPU):\*\*  
- Architektur: ARM Cortex-A78AE  
- Kerne: 6 Kerne  
- Taktfrequenz: Bis zu 1,5 GHz  
- Merkmale: Hocheffizient und leistungsstark für KI- und Edge-Computing-Anwendungen.  
  
---  
  
### \*\*2. Grafikprozessor (GPU):\*\*  
- Architektur: NVIDIA Ampere  
- Tensor Cores: Unterstützt (64 Tensor Cores)  
- Rechenleistung: Bis zu 40 TOPS (Tera Operations per Second)  
- Zweck: Beschleunigung von KI- und Deep-Learning-Workloads, darunter Bildverarbeitung, Objektverfolgung und neuronale Netze.  
  
---  
  
### \*\*3. Speicher (RAM):\*\*  
- Kapazität: 4 GB oder 8 GB LPDDR5 (abhängig vom Modell)  
- Bandbreite: Bis zu 68 GB/s  
- Typ: LPDDR5 (Low Power DDR5) für hohe Geschwindigkeit und Energieeffizienz.  
  
---  
  
### \*\*4. Speicher (Storage):\*\*  
- Unterstützt: MicroSD-Karte, eMMC oder NVMe (über PCIe-Schnittstelle)  
- Externer Speicher: Erweiterbar über USB oder PCIe.  
  
---  
  
### \*\*5. Schnittstellen und I/O:\*\*  
- \*\*PCIe:\*\*   
 - PCIe Gen 3 x4  
 - Gen 4 Unterstützung in bestimmten Konfigurationen  
- \*\*USB:\*\*   
 - 1× USB 3.2 (SuperSpeed)  
 - 1× USB 2.0  
- \*\*Ethernet:\*\* Gigabit-Ethernet  
- \*\*Serielle Schnittstellen:\*\* UART, SPI, I2C  
- \*\*Kamera-Schnittstellen:\*\* 2× MIPI CSI-2 Kameraport (Unterstützung für bis zu 4 Kameras je nach Konfiguration)  
- \*\*HDMI/Display:\*\* HDMI 2.1, DP 1.2 für Monitorausgabe bis zu 4K.  
  
---  
  
### \*\*6. Leistung und Energieverbrauch:\*\*  
- Leistung: 40 TOPS (Tensor Operations per Second) für KI-Anwendungen.  
- Energieverbrauch:   
 - Einstellbar zwischen 7 W und 15 W.  
 - Optimiert für Edge- und Low-Power-Anwendungen.  
  
---  
  
### \*\*7. Softwareunterstützung:\*\*  
- Betriebssystem:   
 - \*\*Linux for Tegra (L4T)\*\* – basiert auf Ubuntu mit NVIDIA-Anpassungen.  
 - Unterstützung durch \*\*NVIDIA JetPack SDK\*\*, das CUDA, TensorRT und DeepStream enthält.  
- Entwicklungsumgebung:  
 - Unterstützung für CUDA 11, C++ und Python.  
 - Plug-and-Play mit populären Frameworks wie TensorFlow, PyTorch und ONNX.  
  
---  
  
### \*\*8. Anwendungen:\*\*  
Der Jetson Orin Nano ist optimiert für Anwendungen wie:  
- Robotik und autonome Systeme  
- Computer Vision (z. B. Objekterkennung und Bildklassifikation)  
- Internet der Dinge (IoT)  
- Überwachungssysteme (z. B. Edge-KI-Kameras)  
- Medizinische Geräte  
- Drohnen und autonome Fahrzeuge  
  
---  
  
### \*\*Vergleich der Modelle:\*\*  
  
| \*\*Modell\*\* | \*\*RAM\*\* | \*\*Leistung (TOPS)\*\* | \*\*Energieverbrauch\*\* |   
|----------------------|---------|---------------------|-----------------------|   
| Jetson Orin Nano 4GB | 4 GB | 20 TOPS | 7–15 W |   
| Jetson Orin Nano 8GB | 8 GB | 40 TOPS | 7–15 W |   
  
---  
  
### \*\*Größe und Formfaktor:\*\*  
- \*\*Abmessungen:\*\* Kompakt, ca. 70 mm × 45 mm  
- \*\*Formfaktor:\*\* SODIMM-Modul, kompatibel mit bestehenden Jetson-Plattformen.

Betroffene Schwierigkeiten:

Folgende Softwares wurde auf zwei Laptops ausgeführt. Der einer ist Hochschul-Laptop, den ich zur C++ Tutorium nutze. Der andere ist mein persönlicher Laptop.

Hochschul-Laptop:

Betribssystem: Windows 10

Speicher: 237GB

RAM: 8GB

Weiteres: Core i3, 2 Kerne, 4 log. Prozessoren

Persönlicher Laptop:

Betribssystem: Windows 10

Speicher: 222GB SSD + 1T DDR

RAM: 16GB

Weiteres: Core i5, 4 Kerne, 8 log. Prozessoren

- Es gibt zwei images von der nvidia. Ein image funktionert gar nicht, wenn ich die SD Karte nach dem Flash-Prozess durch FUSU in die Hardware einsteke und einschalte. Es heißt: jetson …. . Das andere Image aus diesem Link …. und heißt …. . Anscheinden funktioniert aber wie Sie unten lesen werden, System bootet nicht.

- SD Card Formatter funktioniert nur auf den persönlichen PC.

- Etscher bleibt anhängend auf den Hochschul-Laptop. Zeiget, dass es gerade mit Flash angefangen wird, fängt aber nie.

- Etscher führt auf meinem persönlichen PC zu Blue Screen. Das habe ich vielmals bekommen. Genau an dem Punkt wo der Flash-Prozes in die SD Karte fertig ist, scheitert das System meines Laptops und zeigt den blauen Bildschirm.

- Fusu: als Lösung dafür habe ich Fusu installiert. Fusu führt manchmal zu Blue screen auch. Doch das macht er nur wenn er die SD Karte selbst formatieren muss. Formatierte SD Karte verursacht dabei keine Problem und der Flash-Prozess geht bis zu FERTIG. Das führt aber nicht das Ende.

- Win32 Disk Manager: Alternative zu Fusu habe ich Win32 Disk Manager benutzt. Mein Rechner erkennt dabei die SD Karte nicht. Deswegen konnte ich diese Möglichkeit auf meinen Rechner nicht nutzen. Auf den Hochschul-Laptop wird die SD Karte zwar erkennt, kann aber der Flash-Prozess nicht gestartet werden.

- Flash-Prozess: Er hat bei Fusu mehrmals zum Ende funktioniert. Einmal auch sogar bei Etscher. Das Image konnte aber nicht in die Hardware geflscht werden. System startet und zeigt das Logo von Nvidia und fängt zu booten von der SD Karte an. In ein paar Sekunden geht mein Bildschirm aus und zeigt dass keine HDMI Signale von meiner Hardware gibt. Die Ventilator geht dabei kurz aus und läuft wieder ohne dass sich etwas auf meinen Bildschirm ändert.

-System bootet nicht: Habe mir das Forum von nvidia geguckt. Da haben sich auch einigen mehrmals beschworen. Am meisten haben sie aber Linux genutzt, wobei ich auf Windows arbeite. Diese vorhandenen Feedbacks waren mir nicht hilfreich.

- SDK Manager: aus den Foren hatte ich die Idee auf Linux zu arbeiten. Anweisungen für die Installation werden folgen. Nötige bzw. verwendete Softwares: VirtualBox, Ubuntu, 35GB Mindestspeicher.

- Speicher ist wenig: das löst man, indem man aus VirtualBox, also außerhalb Ubuntu, den Speicher vergrößt. Stellen Sie sicher, dass Sie genug Speicher auf Ihren Windows haben. Nach der Vergrößerung die Virtuelle Maschine schließen und neu starten.

- SDK Mnager beschwert sich: Obwohl der Speicher schon über 70GB ist, beschwert sicher die SDK Manager Anwendung. Problem konnte so sein: Speicher ist in mehreren Partitionen geteilt. SDK Manager nutzt dabei die kleine Partition oder es gibt nur eine Partition, die klein ist. Der Rest des Speichers ist dabei nicht zugeteilt und bleibt nicht benutzt. Lösung: Nuten Sie Gparted um die Größe der Partition zu vergrößern.

- Recovery Mode. SDK Manager erkennt keine Hardware. Der Grund dafür, dass der USB-Anschluss mit Type C für die Stromversorgung und für die Datenübertragung genutzt werden kann. Wenn Die dies einfach schließen, wird die Hardware nicht erkennt. Die Hardware muss sich in das Recovery Mode befinden. Recovery Mode wird erst geschaltet, wenn die zwei PINS Ground und REC verbunden sind. Nutzen Sie Female zu Female Jumber/ Kabel, um die Zwei Pins zu schalten. Danach können Sie die Stromversorgung anschließen. So wird die Hardware von SDK Manager erkennt und man kann weiter mit der Installation fortsetzen.

- Gerät wieder nicht erkannt: Nach der Installation aller Paketen und wenn der Flash-Prozess angefangen hat, meldet sich SDK Manager, dass die Hardware nicht gefunden werden kann.

- Freezing der Virtuelle Maschine VirtualBox.