

Autonomes Fahren erleben:

Vom Simulator auf die Rennstrecke

Dein Einstieg in die Welt der künstlichen Intelligenz



Keine grauer Theorie sondern ein praktischer Schritt für Schritt Einstieg in:

- Künstlicher Intelligenz und Deep-Learning
- Erzeugen synthetischer Daten im Simulator
- Training eines Autopiloten / neuronalen Netzes
- Testen des Autopiloten im Simulator
- Schritt für Schritt Anleitung zum Bau eines echten Modellroboter-Autos

Autor: Ingmar Stapel (Version 0.10 | 2021)

E-Book Download Seite: <https://custom-build-robots.com/donkey-car-e-book-de>

Inhalt

1	Einführung	5
2	Die Grundlagen des Projektes.....	6
2.1	Wie lernen Neuronale Netze? 6	
3	Der Donkey Car Simulator	8
3.1	Die Windows Host PC Installation 10	
3.1.1	Notwendige Tools: 10	
3.1.2	Installation des Donkey Car Frameworks 11	
3.1.3	Donkey Car Framework mit CPU Unterstützung einrichten 12	
3.1.4	Donkey Car Framework mit GPU Unterstützung einrichten 13	
3.1.5	Einrichten des Simulators unter Windows 13	
3.2	Die Ubuntu Host PC Installation 15	
3.2.1	Kleinen Helfer und Tipps: 15	
3.2.2	Installation - Miniconda 17	
3.2.3	Donkey Car Framework mit CPU Unterstützung einrichten 19	
3.2.4	Donkey Car Framework mit GPU Unterstützung einrichten 20	
3.2.5	Einrichten des Simulators unter Ubuntu 26	
3.3	Konfiguration des Donkey Car Frameworks (myconfig.py) 28	
3.3.1	Gamepad Konfiguration unter Ubuntu 32	
3.4	Vom Trainingsdaten erstellen bis zum autonomen Fahren 35	
3.4.1	Trainingsdaten im Simulator aufzeichnen 36	
3.4.2	Das neuronale Netz trainieren Windows 37	
3.4.3	Das neuronale Netz trainieren Ubuntu 38	
3.4.4	Das fertige neuronale Netz testen 40	
3.5	Rennen fahren im Simulator 41	
3.5.1	Anpassung der myconfig.py 42	
3.5.2	Überholen trainieren - Trainingsdaten erfassen 43	
3.6	Was Sie bis hier erreicht haben 43	
4	Baue Dein eigenes Donkey Car - Einführung in die benötigten Komponenten 45	
4.1	Auswahl der Hardware 45	
4.1.1	Zentrale Recheneinheit und Zubehör 46	
4.1.2	Auswahl des Chassis 49	
4.1.3	Servo Kontroller 50	
4.1.4	Gamepad 50	
4.1.5	OLED Display 50	
4.1.6	Kabel und Adapter 50	
4.1.7	RC Akku - Spannungsanzeige 51	
4.1.8	Komponentenliste - online 51	

5	Aufbau des Roboter-Autos und einrichten des Jetson Nano.....	52
5.1	Benötigtes Werkzeug	52
5.2	Vorbereiten der Grundplatte	52
5.3	Komponenten befestigen	53
5.3.1	Überrollkäfig befestigen	53
5.3.2	Die Position für Elektronische Komponenten festlegen	54
5.3.3	Löcher bohren	55
5.3.4	Intel AC8265 Wireless Antennen befestigen	55
5.4	Verkabelung der Komponenten im Roboter-Auto	56
5.4.1	I ² C Bus eine kurze Einführung	56
5.4.2	Verkabelung I ² C Bus	56
5.4.3	Fahrtenregler mit aktiver BEC:	59
5.4.4	Fahrtenregler ohne BEC:	59
5.4.5	Kamera anschließen	59
6	Einrichten des Betriebssystem des Jetson Nano	61
6.1.1	SWAP Datei einrichten	62
6.2	SSD Festplatte einrichten – empfohlen (aber optional)	63
6.2.1	Erster Eintrag – LABEL primary	65
6.2.2	Zweiter Eintrag – LABEL sd-card	65
6.3	Was Sie bis hier erreicht habe.	66
7	Installation des Donkey Car Frameworks und Kalibrierung des Roboter-Auto	67
7.1	Die virtuelle Umgebung einrichten	68
7.2	Donkey Car Framework installieren	69
7.2.1	I2C Bus und Servo Kontroller PCA9685 Anpassung	70
7.3	Konfiguration des Roboter-Auto	71
7.3.1	Voraussetzungen für die Kalibrierung	71
7.3.2	Lenkung Kalibrieren:	71
7.3.3	Fahrtenregler Kalibrieren:	71
7.3.4	Vertauschte Richtung bei der Beschleunigung	72
7.3.5	Konfigurationsdatei myconfig.py	72
7.4	ACHTUNG: Fehler im Part camera.py	75
7.5	Kamera mit fehlerhafter Farbdarstellung	76
7.6	Fahren üben	77
7.7	OLED Display aktivieren	77
7.7.1	Das Skript start-oled.sh selber anlegen	78
7.8	Was Sie bis hier erreicht habe.	79
8	Vorbereitung für die Trainingsdatenaufzeichnung und Trainings des Autopiloten.....	80
8.1	Strecken für das Roboter-Auto bauen	80

8.2	Gute Trainingsdaten aufzeichnen	82
8.3	Gamepad Tastenbelegung	82
8.4	Alternative: Die Web-Steuerung	83
8.5	Trainingsdaten aufzeichnen	84
8.5.1	Daten Qualitätssicherung	84
8.6	Trainieren Sie ihren Autopiloten	85
8.7	Führen Sie ihren Autopiloten das erste Mal aus	86
8.8	Hilfe mein Roboter-Auto fährt nicht automatisch los	87
8.9	Was Sie bis hier erreicht haben	87
9	Tipps, Tricks und weiterführende Informationen zum Roboter-Auto	88
9.1	Zusätzlichen Arbeitsspeicher freigeben	88
9.2	Begrenzung der Leistungsaufnahme des Jetson Nano	89
9.3	Screen der Terminalmultiplexer	89
9.4	Trainingsbilder manipulieren	90
9.5	Hilfe mein Lüfter läuft nicht an	91
9.6	Roboter-Auto mit WIFI-Accesspoint	91
9.7	Was Sie bis hier erreicht haben	92

1 Einführung

Es gibt verschiedene Projekte, die sich dem Thema künstliche Intelligenz und autonom fahrende Modellautos annähern. Manche dieser Projekte gibt es als Simulator und andere für die reale Welt. Das Donkey Car Projekt gibt es sowohl als Simulator als auch als physisches Modellauto. Daher freue ich mich sehr Ihnen in diesem Buch das Donkey Car Projekt vorstellen zu dürfen.

Im November 2016 haben Adam Conway und Will Roscoe das Donkey Car Projekt ins Leben gerufen. Die Idee die beide antrieb war interessierten Bastlern eine Möglichkeit zu geben in die Welt der selbst fahrenden Autos im Modellformat einzusteigen. Die wichtigsten Gründe die für mich für den Einsatz des Donkey Car Frameworks sprechen sind einmal die große internationalen Community, die freie Verfügbarkeit der Software, der Donkey Car Simulator und das für den Bau des physischen Roboter-Autos Standard Modellbau Technik eingesetzt wird. Durch die Verwendung von klassischen Modellbauautos sinken die Kosten deutlich.

Ursprünglich wurde das Donkey Car Framework für den Raspberry Pi entwickelt, der der meist verkaufte Single Board Computer ist und mit seiner riesigen Community mit half den Grundstein für den Erfolg des Donkey Cars Projektes zu legen. Mit der Verfügbarkeit des NVIDIA Jetson Nano 2019 der über eine leistungsstarke GPU Architektur verfügt und dessen ähnlichen ARM Recheneinheit wie sie der Raspberry Pi verwendet, erkannte die Donkey Car Community sofort die Vorteile und erweiterte das Framework zusätzlich um eine Unterstützung für den Jetson Nano. Einer der größten Vorteile des Jetson Nano im Vergleich zum Raspberry Pi ist, dass das Training von neuronalen Netzen direkt auf dem Jetson Nano dank der GPU Architektur möglich ist. Mit den verbauten GPU-Einheiten kann der Jetson Nano parallele Berechnungen ausführen die es ihm ermöglichen performant neuronale Netze zu trainieren. Der Raspberry Pi 4 mit 4 GB RAM aber ohne GPU Unterstützung ist gegenüber dem Jetson Nano von seiner Rechenleistung chancenlos beim Ausführen und Training von neuronalen Netzen. Das in Python geschriebene Donkey Car Framework und dessen gute Inlinedokumentation erleichtern zusätzlich den Einstieg in das Vorhaben sich mit autonom fahrenden Roboter-Autos und in das Thema künstliche Intelligenz einzuarbeiten. An dieser Stelle sei auch der Simulator für das Donkey Car erwähnt. Dieser ermöglicht es Ihnen auf ganz leichte Art und wirklich preiswert den Einstieg in das Donkey Car Projekt zu finden. Das ist möglich, da Sie den Donkey Car Simulator auf bestehender Hardware wie einem Laptop oder PC ohne weiteres installieren können.

Mein Fazit in Summe ist, dass das Donkey Car Framework in seiner Kombination aus Simulator Lösung und physischen Modellauto sehr Praxis orientiert, leicht verständlich und daher ideal für den Einstieg in das Thema künstliche Intelligenz ist.

Die offizielle Donkey Car Projektseite erreichen Sie unter der folgenden URL.

URL: <https://www.donkeycar.com/>

Die neueste Version des Donkey Car E-Books von mir finden Sie immer auf meinem Blog unter der folgenden URL zum Download.

URL: <https://custom-build-robots.com/donkey-car-e-book-de>

Ich freue mich sehr über Ihr Feedback zu diesem E-Book. Sie erreiche mich wie folgt.

E-Mail: ebook@custom-build-robots.com

2 Die Grundlagen des Projektes

Sie wollten schon immer einmal verstehen wie autonom fahrende Autos funktionieren und sich selber ein Bild vom Stand der Technik machen? Dann sind Sie hier genau richtig! Ich werde Ihnen zeigen wie Sie im Simulator und in Echt einem neuronalen Netz beibringen ein Roboter-Auto zu steuern.

Mit diesem E-Book halten Sie eine Schritt für Schritt Anleitung in der Hand die das Augenmerk auf das selber machen legt. Beim Schreiben dieses E-Books kam es mir darauf an, die Grundlagen und Theorien die Sie kennen müssen wenn Sie sich mit künstlicher Intelligenz beschäftigen leicht verständlich zu vermitteln. So wird das E-Book nicht in die theoretischen Tiefen des Designs von neuronalen Netzen abtauchen. Sie werden stattdessen neuronal Netze schlicht und ergreifend anwenden so einfach wie das Abheben von Geld am Geldautomaten. Hier müssen Sie ja die Konzepte und Strukturen auch nicht verstehen die dazu beitragen den Geldautomaten nutzen und bedienen zu können. Vielmehr müssen wir als Gesellschaft erkenne das es an der Zeit ist sich mit künstlicher Intelligenz auseinanderzusetzen und diese einfach anzuwenden. Nur so können die jetzt folgenden Generationen damit aufwachsen und erhalten überhaupt die Möglichkeit die Technologie auch hinterfragen zu können.

2.1 Wie lernen Neuronale Netze?

Es gibt verschiedene Ansätze wie neuronale Netze angelernt werden können. Ein kleiner Überblick soll Ihnen dabei helfen die Möglichkeiten und wesentlichen Unterschiede kennenzulernen ohne dass Sie jetzt viel Zeit mit der Theorie verbringen müssen. Drei Konzepte wie neuronale Netze lernen möchte ich Ihnen vorstellen.

Behavioral Cloning: Beim Behavioral Cloning werden alle Aktionen die ein Mensch ausführt um ein bestimmtes Ziel zu erreichen aufgezeichnet. In einem nachfolgenden Schritt wird ein Neuronales Netz mit den aufgezeichneten Daten trainiert. So lernt die Maschine durch Beobachtung des menschlichen Verhaltens die gleiche Problemstellung zu lösen bzw. das mögliche Verhalten des Menschen so gut wie möglich vorherzusagen. Ein neuronales Netz ist ein Netz aus Knoten und Beziehungen. Die Knoten besitzen Gewichtungen und durch das Abschauen was der Mensch vorgemacht hat werden diese Gewichtungen gesetzt und das neuronale Netz fängt an zu lernen.

Un-Super Wise Learning: Das Un-Super Wised Learning ist möglich geworden da sehr viele Daten zu z. B. einem fachlichen Thema bereits gesammelt wurden und maschinell lesbar verfügbar sind. Zu dieser Entwicklung hinzugekommen ist die kostengünstige Rechenleitung in Form von GPUs die es erst ermöglicht hat massenhaft unstrukturierte Daten nach Mustern zu durchsuchen. Diese Kombination aus vielen Daten und günstiger Rechenpower ermöglichte es, dass Neuronale Netze nach Mustern in riesen Datenbeständen suchen die vom allgemeinen Rauschen abweichen. Durch die Anpassung der Gewichtungen der Knoten (Neuronen) des Neuronalen Netzes lernt dieses anhand der Daten dann wieder Vorhersagen zu treffen.

Reinforcement learning: Beim Reinforcement learning (RL) wird in einem Simulator das Neuronale Netz trainiert. Zu Beginn ist diesem die Aufgabenstellung nicht bekannt und durch ein System von Belohnung und Bestrafung lernt es im Simulator die richtigen Entscheidungen zu treffen also die Gewichtungen der Knoten im Netz zu setzen. Nach vielen Wiederholungen lernt das neuronale Netz im Simulator die Problemstellung zu lösen. Entspricht die Simulierte Welt stark der Realität wie z. B. ein digitales Abbild einer Werkshalle kann das so trainierte Neuronale Netz außerhalb des Simulators in der realen Welt zum Einsatz kommen. Ein Beispiel ist hier z. B. der DeepRacer von Amazon bei dem im Simulator durch RL das neuronale Netz trainiert wird und den DeepRacer anschließend um den Racetrack steuert.

Der menschliche BIAs: Wenn Sie dieses Buch durchgearbeitet haben werden Sie feststellen, wie Sie als Mensch bewusst oder unbewusst mit Ihrem Verhalten auf die Vorhersagen des Neuronalen Netzes einflussnehmen werden. Genau dieses unbewusste weitergeben von persönlichen Erfahrungen und Verhaltensmustern nennt man den menschlichen BIAs. In Projekten die sich mit dem Training von

neuronalen Netzen beschäftigen kann der BIAs zu Problemen führen. Fahren Sie z. B. sehr langsam mit dem Donkey Car während der Aufzeichnung der Trainingsdaten wir das neuronale Netz später das Roboter-Auto ebenfalls sehr langsam um die Rennstrecke steuern. Das war jetzt ein ganz einfaches Beispiel aber sehr gut in diesem Projekt auszuprobieren. Oder fahren Sie die Rennstrecke nur im Uhrzeigersinn ab und lassen dann das Donkey Car gegen den Uhrzeigersinn fahren. Sehr wahrscheinlich wird es das nicht schaffen da es nicht ausreichend gelernt hat auch nach links zu lenken.

Jetzt reicht es auch schon wieder mit der Theorie. Legen Sie im folgenden Kapitel gleich los mit der Installation des Donkey Car Simulators. Probieren Sie mit dem Simulator aus einem neuronalen Netz das Fahren auf einer Rennstrecke beizubringen.

3 Der Donkey Car Simulator

Bevor Sie jetzt Geld für Hardware wie einem RC Modellauto und dazugehöriger Elektronik ausgeben, starten Sie mit dem Simulator. Hier können Sie schon vieles der Konzepte hinter dem Begriff Deep Learning erleben die notwendig sind ein neuronales Netz zu trainieren. Aber seien Sie sich sicher, in der realen Welt verhält sich ein neuronales Netz wieder ganz anders.

Für das Donkey Car Framework gibt es neben dem Roboter-Auto aus Hardware auch einen extra Simulator. Diesen müssen Sie auf einer x86 Umgebung dem sogenannten Host PC installieren wenn Sie diesen ausprobieren möchten. Der Hintergrund hierfür ist, dass dieser Simulator die Unity Engine verwendet und diese steht nur für die x86 CPU Umgebung zur Verfügung. Daher kann der Simulator aktuell unter Windows, Linux und MAC OS installiert werden. Der Simulator stellt Ihnen exakt ein Abbild des Donkey Car Frameworks zur Verfügung.

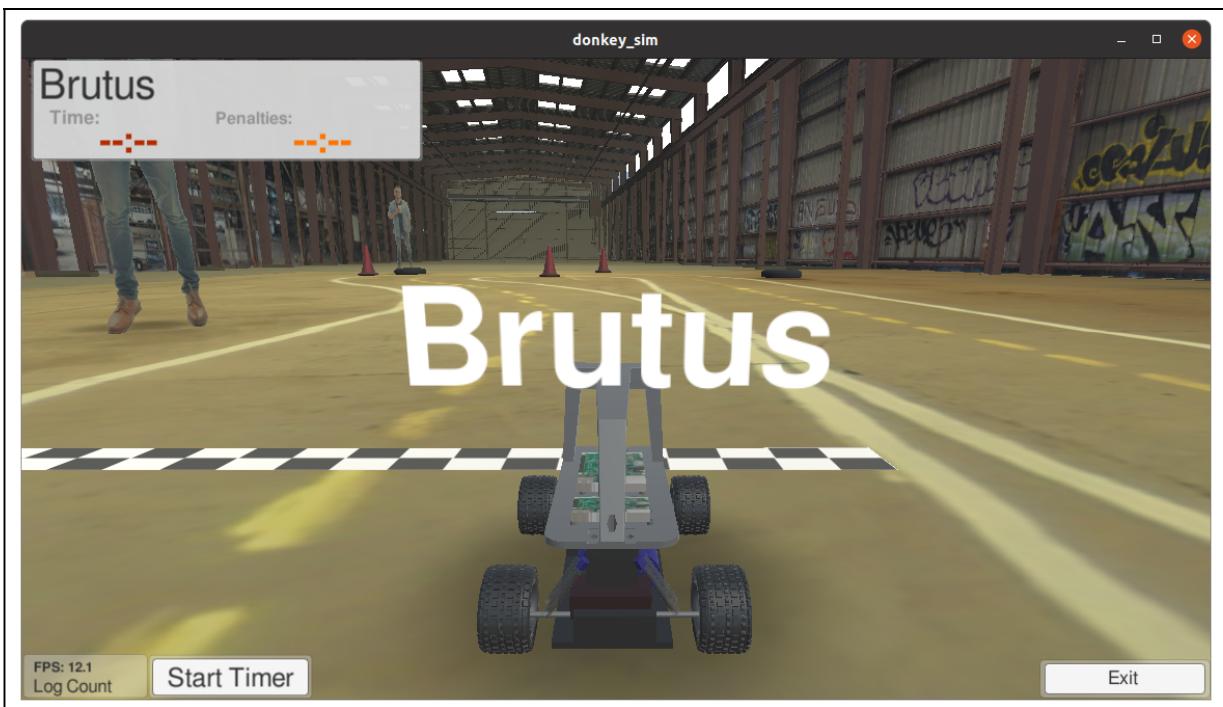


Abbildung 3.1: Donkey Car Simulator (Unity Umgebung)

Sehr viel was Sie im Simulator über das Donkey Car Framework lernen würde später auch Anwendung finden wenn Sie das richtige Donkey Car als physisches Modell aufbauen.

Im Simulator erfassen Sie Trainingsdaten und mit diesen Trainingsdaten trainieren Sie den Autopiloten der das Auto um die Rennstrecke steuert. Sie nutzen dazu genau die gleichen Befehle und Konzepte die Sie später auch im echten Modellauto verwenden würden.

Sehr interessant an diesem Simulator ist die Möglichkeit am Schreibtisch verschiedenes zu testen. Dazu gehören z. B. spezielle Fahrverhalten wie das Überholen dem neuronalen Netz beizubringen oder wenn Sie tiefer einsteigen möchten auch unterschiedliche Designs von neuronalen Netzen zu erproben bevor Sie dieses in der realen Welt auf Ihrem Donkey Car zum Einsatz bringen. Auch lassen sich sehr schön Bildmanipulationen ausprobieren wie z. B. die Helligkeit in den Bildern zu verändern um z. B. zu testen ob dadurch ein neuronales Netz tendenziell stabiler wird und das Auto zuverlässiger um die Rennstrecke steuert.

Bei all den Möglichkeiten im Simulator muss Ihnen gewusst sein, dass die reale Welt um ein vielfaches komplexer ist als die in einem aufgeräumten und sterilen Simulator. Hier hat die Community mit dem

Donkey Car Simulator noch einen weiten Weg vor sich wenn es einmal darum gehen sollte vom Simulator ein neuronales Netz direkt in die reale Welt zu übertragen.

Ich werde Sie jetzt Schritt für Schritt durch die Installation des Donkey Car Simulators unter einer Ubuntu und Windows Umgebung führen. Ich nutze dazu zwei alte Lenovo Laptops. Das eine Modell ist ein T520 und das andere ein T450s die beide noch über keine GPU Unterstützung verfügen. Wie eingangs erwähnt kann der Simulator nicht auf einem Jetson Nano installiert werden da die Unity Engine nur für die X86 Architektur verfügbar ist. Getestet habe ich diese Anleitung für **Ubuntu 18.04**, **Ubuntu 20.04** sowie **Windows 10**. Wobei es unter **Windows 10** kleinere Herausforderungen gab für die ich Ihnen jeweils eine Lösung vorschlagen werde.

Sollten ihr Rechner über eine GPU verfügen werde ich an entsprechender Stelle die notwendigen Befehle aufführen die Sie verwenden müssen um die GPU Version von TensorFlow zu installieren.

Die originale Anleitung zum Donkey Car Simulator finden Sie unter der folgenden Adresse.

URL: <http://docs.donkeycar.com/guide/simulator/>

Im jetzt folgenden Kapitel werde ich Sie Schritt für Schritt durch die Installation des Simulator unter einer **Windows 10** Umgebung führen. Wenn Sie Ubuntu einsetzen dann überspringen Sie bitte das folgende Kapitel.

3.1 Die Windows Host PC Installation

Ich möchte vorab Sie darauf hinweisen, dass es ein kleineres Probleme gibt das ich bei der Installation des Donkey Car Frameworks samt Simulator unter Windows bis jetzt noch nicht auflösen konnte. Die direkte Unterstützung eines Joysticks in der Anaconda Umgebung unter Windows konnte ich nicht aktivieren. Dennoch kann ein Joystick zum Einsatz kommen, wenn das Donkey Car über die Web-Oberfläche ferngesteuert wird.

Somit ist der Donkey Car Simulator unter Windows voll funktionsfähig und mit den im Simulator aufgezeichneten Trainingsdaten können Sie ein funktionierendes neuronales Netz trainieren. Meiner Erfahrung und Einschätzung nach funktioniert das Donkey Car Framework bis jetzt unter Ubuntu ein kleines bisschen besser. Die Community für die Linux Variante des Donkey Cars ist größer und entsprechend schneller bei der Problemlösung. Aber auch bei den Problemen die ich unter Windows auflösen musste um diese Anleitung so gut wie möglich schreiben zu können konnte ich auf die Hilfe der Community zählen.

Jetzt geht es aber los mit der Einrichtung des Donkey Car Frameworks unter Windows. Zuerst beginnen wir mit den notwendigen Tools die Sie erst einmal installieren müssen um überhaupt das Donkey Car Framework samt Simulator Erweiterung unter Windows installieren zu können.

3.1.1 Notwendige Tools:

Damit das Donkey Car Framework unter **Windows 10** installiert werden kann müssen Sie zunächst einmal die nachfolgenden Programme herunterladen und installieren.

Miniconda

Laden Sie die neueste Version von Miniconda3 über die folgende URL herunter.

URL: <https://docs.conda.io/en/latest/miniconda.html>

Bei mir war es die Version **Miniconda3 Windows 64-bit mit Python 3.8**. Diese oder eine neuere installieren Sie jetzt und lassen alle Vorgeschlagenen Einstellungen für die Installation auf Default also so wie vom Installer vorgeschlagen stehen.

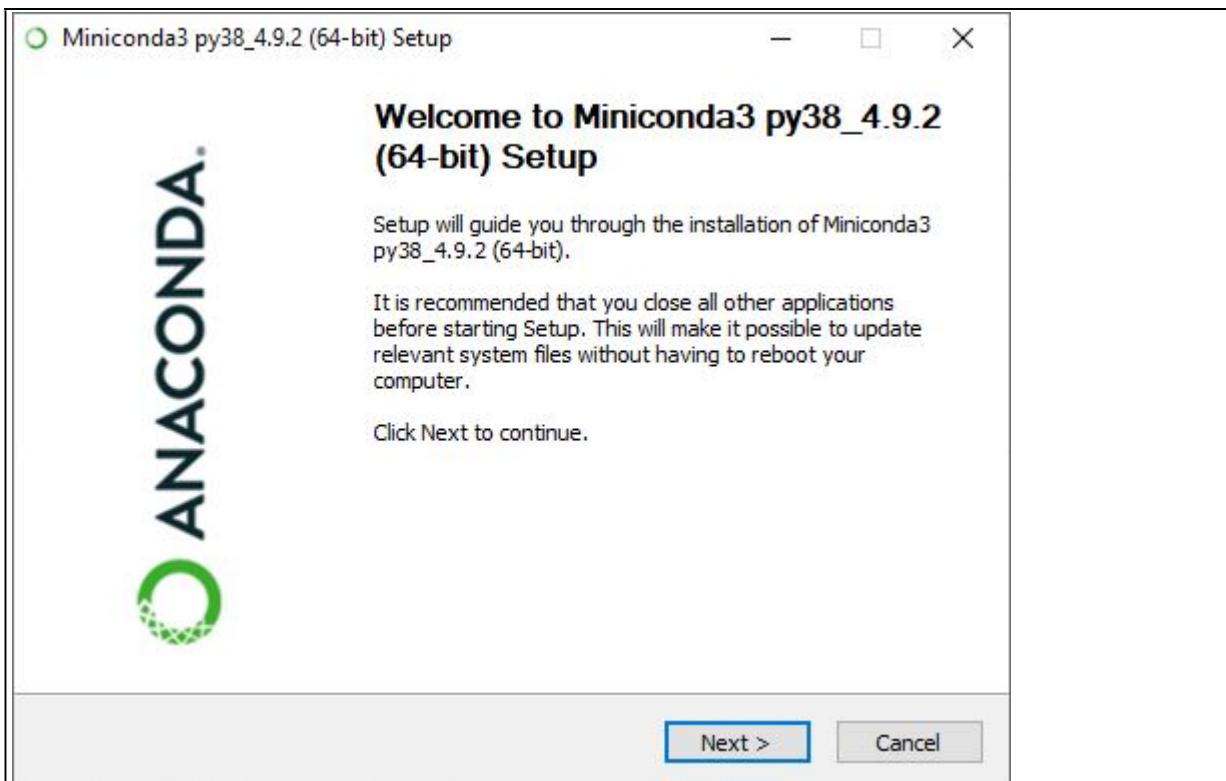


Abbildung 3.2: Miniconda3 Installation

Nach dem Sie ein paar Mal auf Next getippt haben sollte die Installation ohne Fehler abgeschlossen sein. Sie sollten jetzt unter Programme einen Eintrag **Anaconda Promt (miniconda3)** finden. Wenn Sie das Programm starten öffnet sich ein Terminal Fenster bzw. Prompt. In diesem werden Sie später dann arbeiten und das Donkey Car Framework installieren.

Das Fenster bzw. der Prompt sieht wie folgt aus.



Abbildung 3.3: Anaconda Promt (miniconda3)

Git Client installieren

Sie werden im Laufe der nächsten Abschnitte das Donkey Car Framework von GitHub herunter laden. Dazu benötigen Sie den Git Client innerhalb der Anaconda Umgebung auf Ihrem Windows PC. Mit dem folgenden Befehl installieren Sie den GitClient innerhalb der base Anaconda Umgebung:

Befehl: conda install -c anaconda git

Microsoft Visual Studio Code

Für kleinere Anpassungen an Python Programme empfehle ich Ihnen Visual Studio Code von Microsoft zu installieren. Natürlich geht auch ein beliebiger Texteditor aber ich empfehle dennoch Visual Studio Code. Mit Visual Studio Code fallen Ihnen die Anpassungen sicher sehr leicht und Sie habe eine moderne Entwicklungsumgebung auf Ihrem Rechner installiert.

URL: <https://code.visualstudio.com/download>

3.1.2 Installation des Donkey Car Frameworks

Legen Sie jetzt auf Ihrem Laufwerk einen Ordner mit dem Namen **donkeycar_sim** an. Das können Sie ganz einfach mit dem Dateiexplorer von Windows machen.

Starten Sie jetzt die Anaconda-Konsole bzw. den Promot und wechseln Sie in den Ordner **donkeycar_sim**.

Befehl: cd donkeycar_sim

Jetzt müssen Sie als nächstes das Donkey Car Git Repository herunter laden. Dazu führen Sie jetzt im Ordner **donkeycar_sim** den folgenden Befehl aus.

Befehl: git clone <https://github.com/autorope/donkeycar>

Laden Sie jetzt noch zusätzlich die Simulator Umgebung **gym-donkeycar** des Donkey Car Frameworks in den Ordner **donkeycar_sim** herunter.

Befehl: git clone <https://github.com/tawnkramer/gym-donkeycar>

Nach dem Sie jetzt die beiden Frameworks herunter geladen haben wechseln Sie bitte mit dem folgenden Befehl in das Verzeichnis **donkeycar**. In dieses Verzeichnis wurde zuvor das Donkey Car Framework geklont.

Befehl: cd donkeycar

Anschließens checken Sie noch den master Branch aus mit dem nachfolgenden Befehl.

Befehl: git checkout master

Bitte beachten Sie den nachfolgenden Hinweis bevor Sie mit der Einrichtung der Miniconda3 Umgebung weiter fortfahren.

Hinweis: Sollen Sie in Ihrem PC eine NVIDIA Grafikkarte mit Leistungsstarker GPU verbaut haben dann überspringen Sie den folgenden Abschnitt und folgen weiter der Anleitung im Kapitel 3.1.4.

3.1.3 Donkey Car Framework mit CPU Unterstützung einrichten

Jetzt müssen Sie die virtuelle **Miniconda3** Umgebung mit dem Namen **donkey** einrichten.

Hinweis: Bitte öffnen Sie z. B. mit Nano die **ubuntu.yml** oder **pc.yml** Datei im Pfad **/donkeycar_sim/donkeycar/install/envs** und prüfen welche Version von TensorFlow installiert werden soll. Steht bei Ihnen die folgende Zeile in der **ubuntu.yml** oder **pc.yml** **tensorflow=2.2.0** dann ersetzen Sie die Versionsnummer jetzt durch **tensorflow=2.3.0** denn für Anaconda gibt es aktuell keine Version 2.2.0 von TensorFlow.

Nach dem Sie die Änderung der Versionsnummer vorgenommen haben können Sie mit der Installation weiter machen.

Führen Sie jetzt den folgenden Befehl z. B. im **/donkeycar_sim/donkeycar** Ordner aus abhängig davon wo Sie das Framework herunter geladen haben.

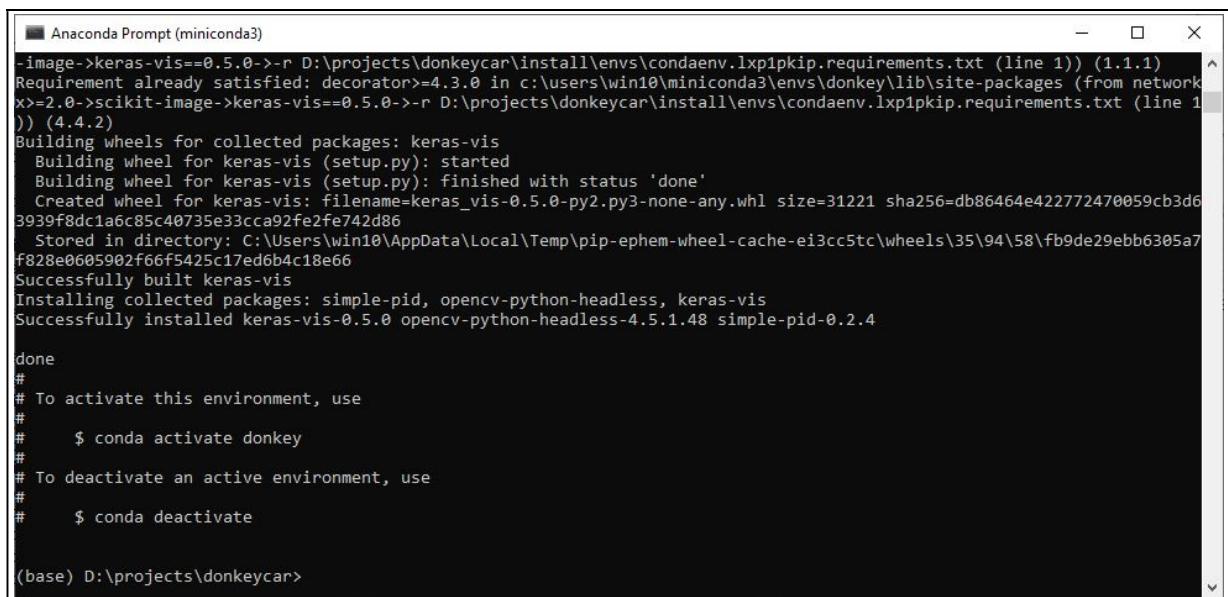
Befehl: conda env create -f install\envs\pc.yml

oder

Befehl: conda env create -f install\envs\ubuntu.yml

Das Einrichten der Conda Umgebung **donkey** dauert je nach Internetanbindung mehrere Minuten bis die Installation abgeschlossen ist.

Nach dem die Installation erfolgreich durchgelaufen ist, sollten Sie folgende Meldung in dem Conda Terminal Fenster angezeigt bekommen.



```
[base] D:\projects\donkeycar>
```

```
-> keras-vis==0.5.0->-r D:\projects\donkeycar\install\envs\condaenv.lxpipkip.requirements.txt (line 1)) (1.1.1)
Requirement already satisfied: decorator>=4.3.0 in c:\users\win10\miniconda3\envs\donkey\lib\site-packages (from networkx>=2.0->scikit-image->keras-vis==0.5.0->-r D:\projects\donkeycar\install\envs\condaenv.lxpipkip.requirements.txt (line 1))
Building wheels for collected packages: keras-vis
  Building wheel for keras-vis (setup.py): started
  Building wheel for keras-vis (setup.py): finished with status 'done'
  Created wheel for keras-vis: filename=keras_vis-0.5.0-py2.py3-none-any.whl size=31221 sha256=db86464e422772470059cb3d63939f8dc1a6c85c40735e33cca92fe2fe742d86
  Stored in directory: C:\Users\win10\AppData\Local\Temp\pip-ephem-wheel-cache-ei3cc5tc\wheels\35\94\58\fb9de29ebb6305a7f828e0605902f66f5425c17ed6b4c18e66
Successfully built keras-vis
Installing collected packages: simple-pid, opencv-python-headless, keras-vis
Successfully installed keras-vis-0.5.0 opencv-python-headless-4.5.1.48 simple-pid-0.2.4

done
#
# To activate this environment, use
#
#     $ conda activate donkey
#
# To deactivate an active environment, use
#
#     $ conda deactivate

(base) D:\projects\donkeycar>
```

Abbildung 3.4: Erfolgreiche Installation der virtuellen Umgebung **donkey**

Aktivieren Sie jetzt die virtuelle Umgebung **donkey** mit dem folgenden Befehl. Dieser sollte Ihnen auch am Ende der zuvor gestarteten Installation angezeigt werden wie Sie dies im Bild vorher sehen.

Befehl: conda activate donkey

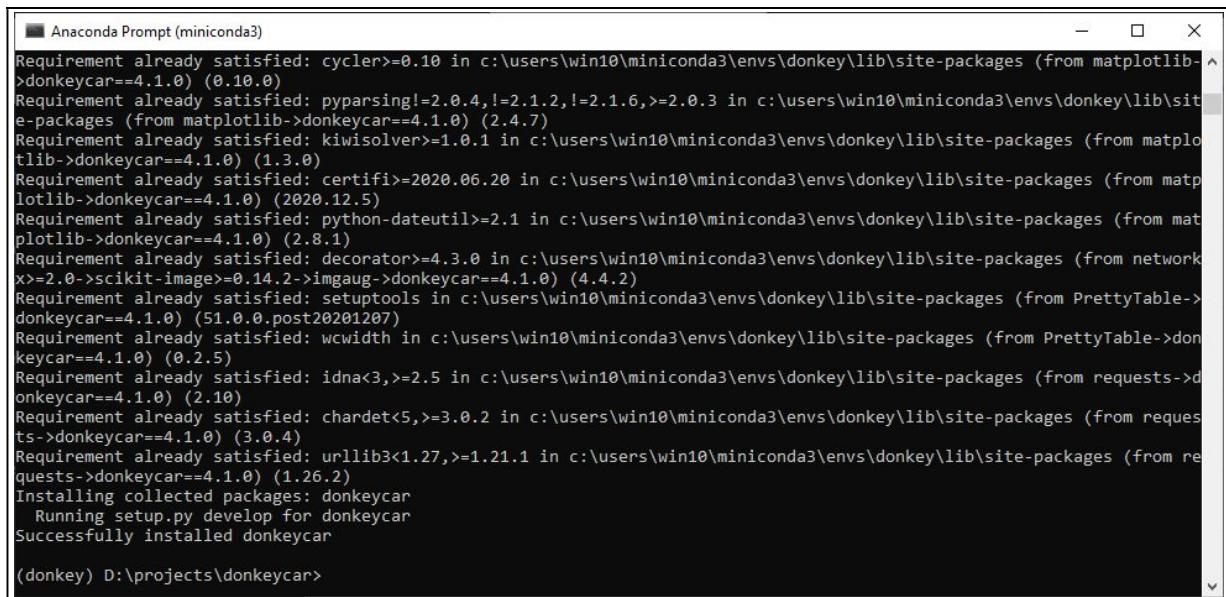
Für den Fall das Sie auch eine virtuelle Umgebung einmal entfernen möchten dann können Sie den folgenden Befehl zum Löschen dieser samt allen installierten Packages verwenden.

Befehl: conda env remove --name myenv

Hat alles ohne einer Fehlermeldung geklappt dann können Sie jetzt das Donkey Car Framework installieren. Die Installation starten Sie mit dem folgenden Befehl innerhalb der aktiven Conda Umgebung **donkey**.

Befehl: pip install -e .[pc]

Das nachfolgende Bild zeigt die Abschlussmeldung nach dem das Donkey Car Framework erfolgreich installiert wurde.



```
Anaconda Prompt (miniconda3)
Requirement already satisfied: cycler>=0.10 in c:\users\win10\miniconda3\envs\donkey\lib\site-packages (from matplotlib->donkeycar==4.1.0) (0.10.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in c:\users\win10\miniconda3\envs\donkey\lib\site-packages (from matplotlib->donkeycar==4.1.0) (2.4.7)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\win10\miniconda3\envs\donkey\lib\site-packages (from matplotlib->donkeycar==4.1.0) (1.3.0)
Requirement already satisfied: certifi>=2020.06.20 in c:\users\win10\miniconda3\envs\donkey\lib\site-packages (from matplotlib->donkeycar==4.1.0) (2020.12.5)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\win10\miniconda3\envs\donkey\lib\site-packages (from matplotlib->donkeycar==4.1.0) (2.8.1)
Requirement already satisfied: decorator>=4.3.0 in c:\users\win10\miniconda3\envs\donkey\lib\site-packages (from networkx>=2.0->scikit-image>0.14.2->imgaug->donkeycar==4.1.0) (4.4.2)
Requirement already satisfied: setuptools in c:\users\win10\miniconda3\envs\donkey\lib\site-packages (from PrettyTable->donkeycar==4.1.0) (51.0.0.post20201207)
Requirement already satisfied: wcwidth in c:\users\win10\miniconda3\envs\donkey\lib\site-packages (from PrettyTable->donkeycar==4.1.0) (0.2.5)
Requirement already satisfied: idna<3,>=2.5 in c:\users\win10\miniconda3\envs\donkey\lib\site-packages (from requests->donkeycar==4.1.0) (2.10)
Requirement already satisfied: chardet<5,>=3.0.2 in c:\users\win10\miniconda3\envs\donkey\lib\site-packages (from requests->donkeycar==4.1.0) (3.0.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\win10\miniconda3\envs\donkey\lib\site-packages (from requests->donkeycar==4.1.0) (1.26.2)
Installing collected packages: donkeycar
  Running setup.py develop for donkeycar
Successfully installed donkeycar
(donkey) D:\projects\donkeycar>
```

Abbildung 3.5: Erfolgreiche Donkey Car Framework Installation

Jetzt ist die Donkey Car Framework Host PC Installation unter Windows erfolgreich abgeschlossen. Im jetzt folgenden Abschnitt werden wir Schritt für Schritt den Simulator auf Ihrem Host PC einrichten.

3.1.4 Donkey Car Framework mit GPU Unterstützung einrichten

Hinweis: Dieser Abschnitt wird bald nachgereicht wenn es meine Zeit zulässt.

3.1.5 Einrichten des Simulators unter Windows

Jetzt brauchen Sie noch die Unity Umgebung für Windows welche die Rennstrecken für die Simulation bereitstellt die Sie mit dem Donkey Car Simulator anschließend abfahren werden. Dazu müssen Sie noch das Paket mit dem Simulator für Windows herunter laden. Alle bis jetzt veröffentlichten Versionen finden Sie unter der folgenden URL.

Laden Sie jetzt die neueste Version des Unity Simulators für Windows herunter.

Download: <https://github.com/tawnkramer/gym-donkeycar/releases>

Anschließend entpacken Sie den Simulator, also die ZIP Datei, an einen Ort Ihrer Wahl z. B. in der Ordner **donkeycar_sim**.

Möchten Sie schon einmal den Simulator starten dann können Sie dafür die exe-Datei **donkey_sim.exe** ausführen.

Jetzt startet der Simulator aber noch ohne dem Donkey Car Framework und Sie sehen nur die Unity Umgebung mit den verschiedenen Rennstrecken.

Hinweis: Sollte eine Meldung kommen das eine EXE-Datei ausgeführt werden die Windows erst einmal nicht vertraut, dann bestätigen Sie bitte in diesem Dialog das Sie der EXE-Datei vertrauen damit Windows den Simulator auch ausführt.

Jetzt folgt die Installation der Donkey Car Simulator Erweiterung. Dazu wechseln Sie in den Ordner **gym-donkeycar** mit dem folgenden Befehl.

Befehl: cd gym-donkeycar

Jetzt müssen Sie noch die Donkey Car Framework Erweiterung für den Simulator installieren und dazu führen Sie den folgenden Befehl aus.

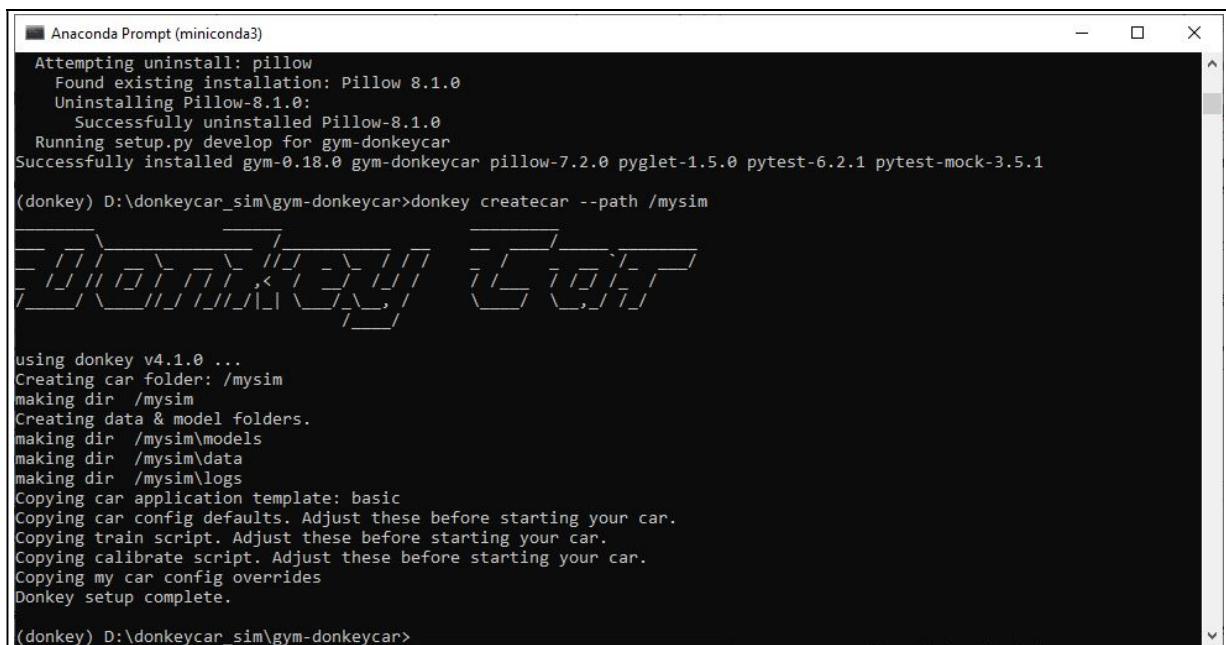
Befehl: pip install -e .[gym-donkeycar]

Die Installation kann mehrere Minuten dauern abhängig davon wie schnell ihr Host PC ist oder aber auch die Internetanbindung.

Nach dem auch das Donkey Car Simulator Framework eingerichtet ist müssen Sie noch eine Instanz von diesem erzeugen mit der Sie später arbeiten werden. Diese Instanz werden Sie auf Ihre Bedürfnisse hin konfigurieren. Führen Sie dazu den folgenden Befehl aus der einen Ordner mit dem Namen **mysim** auf Ihrem Laufwerk anlegt.

Befehl: donkey createcar --path /mysim

Sie sollten jetzt eine Ausgabe wie folgt erhalten und die Donkey Car Simulator Umgebung ist initialisiert. Es wurden verschiedene Ordner und Dateien angelegt wie z. B. die **myconfig.py** Datei. In dieser **myconfig.py** Dateien erfolgen alle Konfigurationen bezüglich dem Donkey Car Framework also Ihre persönliche Individualisierung der Donkey Car Instanz.



```
Anaconda Prompt (miniconda3)
Attempting uninstall: pillow
Found existing installation: Pillow 8.1.0
Uninstalling Pillow-8.1.0:
  Successfully uninstalled Pillow-8.1.0
Running setup.py develop for gym-donkeycar
Successfully installed gym-0.18.0 gym-donkeycar pillow-7.2.0 pyglet-1.5.0 pytest-6.2.1 pytest-mock-3.5.1
(donkey) D:\donkeycar_sim\gym-donkeycar>donkey createcar --path /mysim
using donkey v4.1.0 ...
Creating car folder: /mysim
making dir /mysim
Creating data & model folders.
making dir /mysim\models
making dir /mysim\data
making dir /mysim\logs
Copying car application template: basic
Copying car config defaults. Adjust these before starting your car.
Copying train script. Adjust these before starting your car.
Copying calibrate script. Adjust these before starting your car.
Copying my car config overrides
Donkey setup complete.

(donkey) D:\donkeycar_sim\gym-donkeycar>
```

Abbildung 3.6: Erfolgreiche Installation des Donkey Car Simulator Frameworks

Jetzt haben Sie das Donkey Car Framework und alle Komponenten für die Simulation auf Ihrem Rechner installiert. Als nächstes müssen Sie das Donkey Car Framework noch konfigurieren. Zunächst aber können Sie in der Unity Umgebung einfach einmal herum fahren ohne das Donkey Car Framework zu nutzen. Im Kapitel 0 geht es dann weiter mit der Konfiguration des Donkey Car Frameworks. Hier wird nicht mehr unterschieden ob diese für Windows oder Ubuntu vorgenommen wird. Im Fall von Unterschieden zwischen Windows und Ubuntu werden diese kurz erläutert.

3.2 Die Ubuntu Host PC Installation

In diesem Kapitel geht es darum das Donkey Car Framework samt Simulator unter Ubuntu zu installieren. Meiner Erfahrung nach fällt die Installation unter Ubuntu etwas leichter, bei Schwierigkeiten bekommt man auch eher und vor allem schneller Hilfe als unter Windows. Da sich diese Anleitung auch an Einsteiger in die Linux Welt richtet beginnt dieser Abschnitt erst einmal mit Tools und Tipps die Ihnen die Arbeit unter Linux erleichtern für den Fall, dass Sie sich mit Linux noch nicht so gut auskennen.

3.2.1 Kleinen Helfer und Tipps:

Sind Sie nicht so mit Linux vertraut, dann empfehle ich Ihnen ein paar hilfreiche Tools wie den Text-Editor Nano, den Dateiexplorer Midnight-Commander sowie Screen zu installieren. Diese werden Ihnen dabei helfen, dass Ihnen die Konfiguration des Roboter-Autos etwas leichter von der Hand geht. So reduzieren sich die speziellen Linux Kommandozeilenbefehle auf ein Minimum. All das erkläre ich Ihnen jetzt Schritt für Schritt

Ubuntu Installation aktualisieren

Als erstes bringen Sie aber die lokalen Repository-informationen der Ubuntu Installation ihres Host PC auf den neuesten Stand. Dazu führen Sie den folgenden Befehl aus.

Befehl: sudo apt-get update

Mit dem upgrade Befehl aktualisieren Sie die bereits installierten Programme unter Ubuntu.

Befehl: sudo apt-get upgrade

Wahl des Display Managers

Während der Installation der neuesten Softwarepakete kommt auch eine Frage zu dem Display Manager und die Option den Display-Manager zu wechseln. Der gdm3 Display Manager ist als Default Option vorausgewählt wie das nachfolgende Bild zeigt. Da der gdm3 Displaymanager meiner persönlichen Meinung nach sich sehr einfach nutzen lässt drücken Sie jetzt bitte auf OK um diesen weiter zu verwenden.

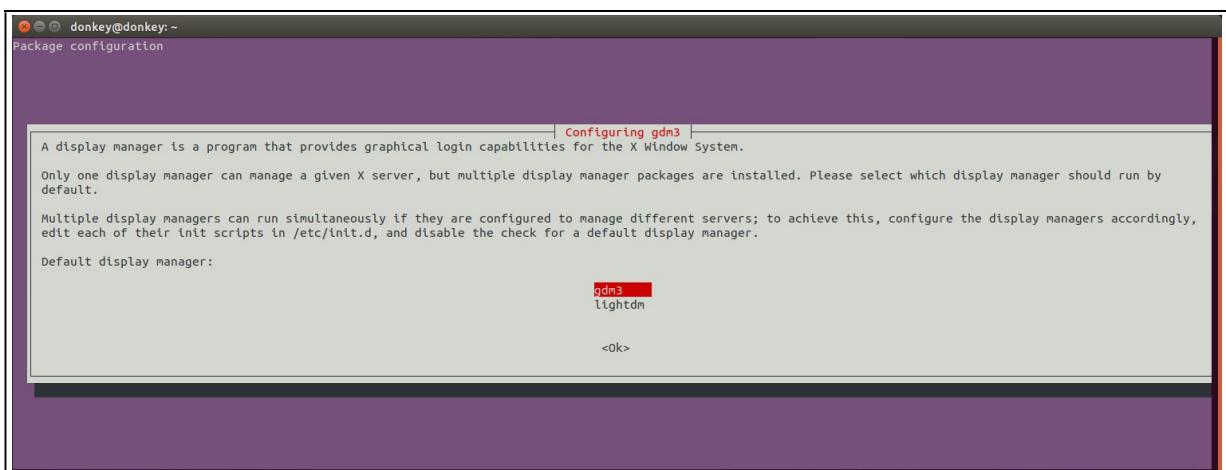


Abbildung 3.7: DGM3 Display Manager

Texteditor Nano

Installieren Sie jetzt den Text-Editor **Nano** zusätzlich zu den bereits vorhandenen Text-Editoren. Nano ist ein recht einfach zu bedienender Text-Editor für das Terminal Fenster. Nano als Text-Editor werden Sie im weiteren Verlaufe immer wieder nutzen um z. B. kleine Anpassungen in Konfigurationsdateien oder Python-Programmen vorzunehmen.

Befehl: sudo apt-get install nano -y

Mit der Tastenkombination **STRG + X** beenden Sie den Text-Editor Nano. Haben Sie eine Datei verändert müssen Sie das Speichern der Änderung noch mit einem Y für Yes durch Drücken der Entertaste bestätigen. Möchten Sie die Änderung verwerfen dann Drücken Sie N für No und die Entertaste.

Im Terminal Fenster geben Sie nano <Datei-Name> ein um den Text-Editor zu starten und gleichzeitig eine bestimmte Datei zu öffnen.

Befehl: nano <Dateiname>

Midnight Commander

Mit dem **Midnight Commander** können Sie im Terminalfenster ganz einfach Dateien editiert oder kopiert, ohne dass Sie die dazu notwendigen Linux-Befehle für das Terminalfenster kennen müssen. Installieren Sie den Midnight-Commander (mc) mit dem folgenden Befehl.

Befehl: sudo apt-get install mc -y

Möchten Sie den **Midnight Commander** starten dann geben Sie den folgenden Befehl im Terminal Fenster ein.

Befehl: mc

Mit den F-Tasten auf Ihrer Tastatur rufen Sie die wichtigsten Befehle des Midnight Commanders auf. Die kleine Übersicht soll Ihnen dabei helfen den Einstieg in die Bedienung des MC leichter zu meistern.

- F4: Öffnet die ausgewählte Datei z. B. im Text Editor Nano.
- F5: Kopiert eine Datei oder Ordner die z. B. im linken Fenster ausgewählt ist in den im rechten Fenster geöffneten Pfad.
- F6: Verschiebt eine Datei oder Ordner die z. B. im linken Fenster ausgewählt ist in den im rechten Fenster geöffneten Pfad.
- F7: Legt einen neuen Ordner an.
- F8: Löscht die Datei oder den Ordner der aktuell ausgewählt ist.
- F10: beendet den Midnight Commander

FFmpeg Videobibliothek (optional)

Mit **FFmpeg** können Sie unter Linux digitales Video- und Audiomaterial aufnehmen, konvertieren und auch streamen. Damit Sie aus den aufgezeichneten Trainingsdaten, also den tausenden Bildern die Sie benötigen um das neuronale Netz zu trainieren, ein Video für die einfache Qualitätssicherung erstellen können empfehle ich Ihnen daher das vielseitige Tool **FFmpeg** zu installieren. Zu einem späteren Zeitpunkt in diesem Buch stelle ich Ihnen noch ein Skript vor mit dem Sie aus Ihren Trainingsbildern automatisch Videos erstellen können. Mit dem nachfolgenden Befehl installieren Sie **FFmpeg**.

Befehl: sudo apt-get install ffmpeg -y

Netzwerk Dateifreigabe

Damit Sie einfach von z. B. Windows aus auf die Dateien ihres Rechners und später auf die Trainingsdaten für Backups, die Qualitätssicherung etc. zugreifen können bietet es sich an **SAMBA** für die Dateifreigabe im Netzwerk zu installieren. Den **SAMBA** Server installieren Sie mit folgenden Befehl.

Befehl: sudo apt-get install samba -y

Nach dem SAMBA installiert ist müssen Sie noch im Ordner /etc/samba die Datei smb.conf anpassen. Dazu öffnen Sie diese im Text-Editor Nano mit dem folgenden Befehl.

Befehl: sudo nano /etc/samba/smb.conf

Ganz an das Ende der Datei fügen Sie die folgenden Zeilen ein. Mit dieser kleinen Anpassung geben Sie den /home/ Ordner ihres Ubuntu Rechners im Netzwerk frei.

[Roboter-Auto]

```
comment = Samba on Ubuntu
path = /home/
read only = no
browsable = yes
```

Damit der Zugriff ohne Probleme klappt muss noch eine kleine Konfiguration am Sabma-Server vorgenommen werden. Der Samba Server verwendet nicht das Passwort des im System angelegten Users. Daher müssen Sie zu Ihrem User den Sie unter Ubuntu angelegt haben noch ein Passwort vergeben für den Samba Server. Nur so können Sie ohne Probleme von z. B. Windows aus auf die Freigabe im Netzwerk zugreifen.

Führen Sie den folgenden Befehl aus um der Samba-Konfiguration zu ihrem bereits im System vorhandenen User ein extra Passwort für den Samba-Service zu hinterlegen.

Befehl: sudo smbpasswd -a username

Jetzt könne Sie mit dem nachfolgenden Befehl den Samba-Service neustarten und z. B. von einem Windows Rechner aus auf die Dateifreigabe unter Ubuntu zugreifen.

Befehl: sudo service smbd restart

Da unter einer typischen Ubuntu Installation der **OpenSSH Server** nicht installiert ist habe ich diesen mit dem folgenden Befehl nachträglich installiert. Dieser wird nur benötigt wenn Sie sich remote via SSH auf dem Ubuntu Host PC anmelden möchten.

Befehl: sudo apt install openssh-server

Jetzt haben Sie Ihre Linux Ubuntu Installation soweit aktualisiert und vielleicht auch die kleinen Helfer in Form von Programmen installiert wie von mir empfohlen. Jetzt geht es dann Schritt für Schritt weiter mit der Installation der Programme die für die Inbetriebnahme des Donkey Car Simulators notwendig sind.

3.2.2 Installation - Miniconda

Jetzt laden Sie zunächst einmal die Miniconda3 Installationsdatei herunter um **Miniconda3** auf dem Host PC zu installieren. Mit Hilfe von Miniconda3 wird eine virtuelle Umgebung bereitgestellt in der das Donkey Car Framework installiert wird.

Mit den beiden folgenden Befehlen laden Sie Miniconda3 in das Homeverzeichnis ihres Users herunter.

Befehl: cd ~

Befehl: wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh

Wenn der Download abgeschlossen ist, starten Sie die Installation von **Miniconda3** mit dem nachfolgenden Befehl.

Befehl: ./Miniconda3-latest-Linux-x86_64.sh

Jetzt kann es passieren, dass eine Meldung nach dem ausführen des Befehls wie nachfolgende gezeigt erscheint und die Installation nicht möglich ist also abbricht.

Fehlermeldung: -bash: ./Miniconda3-latest-Linux-x86_64.sh: Permission denied

Wenn diese Meldung bei Ihnen erscheint, dann ändern Sie die Dateirechte mit folgenden Befehl ab und starten die Installation erneut.

Befehl: sudo chmod 777 Miniconda3-latest-Linux-x86_64.sh

Wenn Sie die Installation starten konnten sollten Sie nach ein paar Sekunden die Nutzungsbedingungen angezeigt bekommen. Drücken Sie solange die ENTER Taste bis Sie aufgefordert werden die Nutzungsbedingungen zu akzeptieren oder abzulehnen. Geben Sie bei der Abfrage bitte „yes“ ein um diese anzunehmen und drücken Sie erneut die Enter Taste zur Bestätigung.

Direkt anschließend kommt die Frage wo Miniconda3 installiert werden soll. Drücken Sie wieder die Enter Taste wenn Miniconda3 in das Homeverzeichnis Ihres Users installiert werden soll. Ich empfehle hier die Default Installation in das Homeverzeichnis ihres Users. Jetzt dauert die Installation mehrere Sekunden bis Minuten.

Nach der Installation kommt die Frage ob Miniconda3 so eingerichtet werden soll, dass initial die Conda base Umgebung gestartet wird wenn Sie sich anmelden. Der Einfachheit halber bestätigen Sie diesen Schritt ebenfalls mit „yes“.

Hinweis: Möchten Sie conda immer manuell von Hand starte dann können Sie statt „yes“ wie vorgeschlagen „no“ eingeben. In diesem Fall tragen Sie dann die folgende Zeile ganz am Ende in die .bashrc Datei in ihrem Homeverzeichnis ein.

```
export PATH=~/miniconda3/bin:$PATH
```

Jetzt laden Sie bitte mit dem folgenden Befehl die .bashrc Datei neu um virtuelle Umgebung **base** zu starten.

Befehl: source ~/.bashrc

Das folgende Bild zeigt Ihnen das Terminalfenster ohne aktiver virtuellen Umgebung base und im unter Teil mit aktiver virtuellen Umgebung base.

<Bild einfügen>

Geben Sie jetzt zur Probe ob Miniconda3 richtig installiert und eingerichtet wurde den Befehl **Conda** im Terminal Fenster ein.

Befehl: conda

Es startet jetzt **Conda** und zeigt Ihnen die Parameter an die Sie nach dem Aufruf des Befehls conda als Parameter eingeben können.

<Screenshot>

Wenn das geklappt hat dann folgt jetzt der nächste Schritt, nämlich das herunter laden des Donkey Car Frameworks.

Installieren Sie vorsichtshalber noch Git auf Ihrem Ubuntu Rechner. Wenn Git bereits installiert ist dann geht der nachfolgende Aufruf ganz schnell und es wird Git nicht noch einmal installiert.

Befehl: sudo apt-get install git

Jetzt ist sichergestellt, dass der Git Client auf Ihrem Rechner installiert ist.

Installation des Donkey Car Frameworks

Das Donkey Car Framework soll in einen Ordner **donkeycar_sim** installiert werden der unterhalb des Homeverzeichnisses Ihres Users liegt. Daher legen Sie jetzt mit dem folgenden Befehl diesen Ordner in Ihrem Homeverzeichnis an.

Befehl: mkdir ~/donkeycar_sim

Anschließend wechseln Sie in den Ordner **donkeycar_sim** mit dem folgenden Befehl.

Befehl: cd ~/donkeycar_sim

Jetzt müssen Sie als nächstes das Donkey Car Framework über die entsprechende Git Repository herunter laden. Dazu führen Sie jetzt im Ordner **donkeycar_sim** den folgenden Befehl aus.

Befehl: git clone https://github.com/autorope/donkeycar

Nach dem Sie das Donkey Car Framework herunter geladen haben benötigen Sie noch die Erweiterung des Frameworks für den Simulator. Auch diese Erweiterung laden Sie wieder herunter via Git mit dem folgenden Befehl in den Ordner **donkeycar_sim**.

Befehl: git clone https://github.com/tawnkramer/gym-donkeycar

Nach dem Sie jetzt die beiden Frameworks herunter geladen haben wechseln Sie bitte mit dem folgenden Befehl in das Verzeichnis **donkeycar**. In dieses Verzeichnis wurde zuvor das Donkey Car Framework geklont.

Befehl: cd donkeycar

Anschließens checken Sie noch den master Branch aus mit dem nachfolgenden Befehl.

Befehl: git checkout master

Bitte beachten Sie den nachfolgenden Hinweis bevor Sie mit der Einrichtung der Miniconda3 Umgebung weiter fortfahren.

Hinweis: Sollen Sie in Ihrem PC eine NVIDIA Grafikkarte mit Leistungsstarker GPU verbaut haben dann überspringen Sie den folgenden Abschnitt und folgen weiter der Anleitung im Kapitel 3.2.4.

3.2.3 Donkey Car Framework mit CPU Unterstützung einrichten

In diesem Abschnitt folgt die Einrichtung der virtuelle **Miniconda3** Umgebung mit dem Namen **donkey** Ohne GPU also NVIDIA Grafikkarten Unterstützung.

Hinweis: Bitte öffnen Sie z. B. mit Nano die **ubuntu.yml** oder **pc.yml** Datei im Pfad **/donkeycar_sim/donkeycar/install/envs** und prüfen welche Version von TensorFlow installiert werden soll. Steht bei Ihnen die folgende Zeile in der **ubuntu.yml** oder **pc.yml tensorflow=2.2.0** dann ersetzen Sie die Versionsnummer jetzt durch **tensorflow=2.3.0** denn für Anaconda gibt es aktuell keine Version 2.2.0 von TensorFlow.

Nach dem Sie die Änderung der Versionsnummer vorgenommen haben können Sie mit der Installation weiter machen.

Führen Sie jetzt den folgenden Befehl z. B. im **/donkeycar_sim/donkeycar** Ordner aus abhängig davon wo Sie das Framework herunter geladen haben.

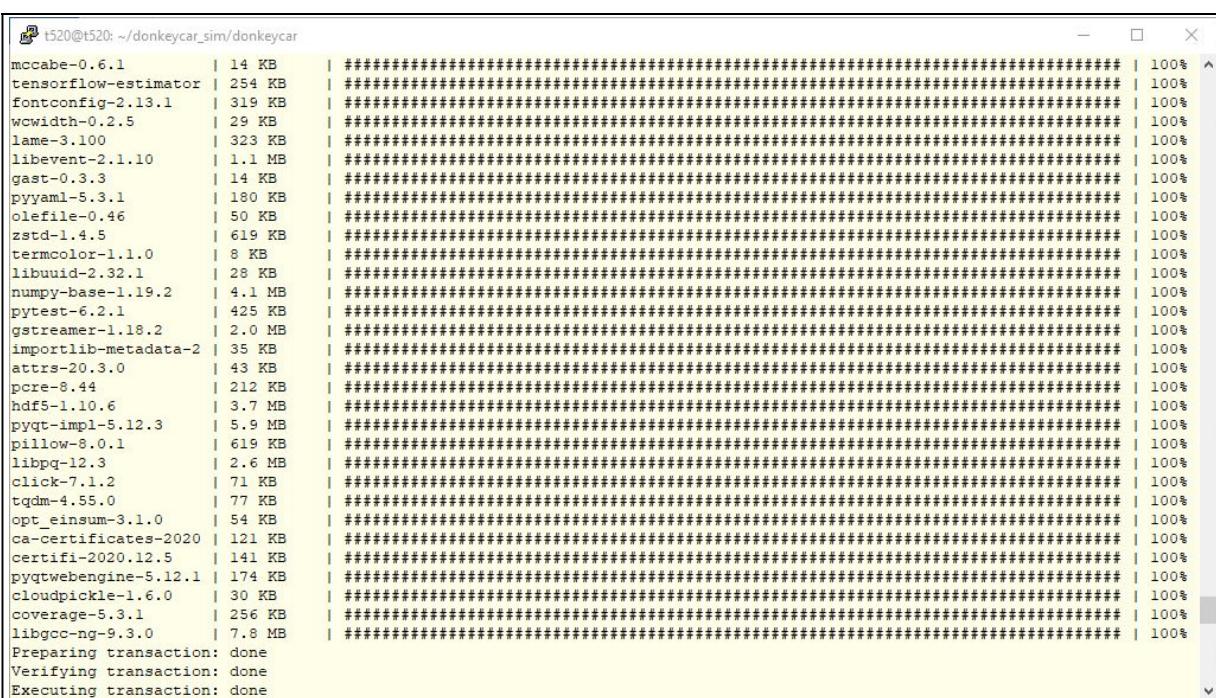
Hinweis: Für Linux und MAC gibt es unterschiedliche Installationsdateien. Daher folgt jetzt die Unterscheidung zwischen Linux und MAC.

Für Linux:

Befehl: conda env create -f install/envs/ubuntu.yml

Für MAC OS:

Befehl: conda env create -f install/envs/mac.yml



The screenshot shows a terminal window with a list of packages being installed. The progress is indicated by a series of '#' characters. The packages listed include mccabe, tensorflow-estimator, fontconfig, wcwidth, lame, libevent, gast, pyyaml, olefile, zstd, termcolor, libuuid, numpy-base, pytest, gstreamer, importlib-metadata, attrs, pcre, hdf5, pyqt-impl, pillow, libpq, click, tqdm, opt_einsum, ca-certificates, certifi, pyqtwebengine, cloudpickle, coverage, and libgcc-ng. The transaction is shown as preparing, verifying, and executing.

```
mccabe-0.6.1          | 14 KB    | #####| 100% ^  
tensorflow-estimator | 254 KB   | #####| 100% ^  
fontconfig-2.13.1    | 319 KB   | #####| 100% ^  
wcwidth-0.2.5        | 29 KB    | #####| 100% ^  
lame-3.100           | 323 KB   | #####| 100% ^  
libevent-2.1.10      | 1.1 MB   | #####| 100% ^  
gast-0.3.3           | 14 KB    | #####| 100% ^  
pyyaml-5.3.1         | 180 KB   | #####| 100% ^  
olefile-0.46         | 50 KB    | #####| 100% ^  
zstd-1.4.5            | 619 KB   | #####| 100% ^  
termcolor-1.1.0       | 8 KB     | #####| 100% ^  
libuuid-2.32.1        | 28 KB    | #####| 100% ^  
numpy-base-1.19.2     | 4.1 MB   | #####| 100% ^  
pytest-6.2.1          | 425 KB   | #####| 100% ^  
gstreamer-1.18.2      | 2.0 MB   | #####| 100% ^  
importlib-metadata-2 | 35 KB    | #####| 100% ^  
attrs-20.3.0           | 43 KB    | #####| 100% ^  
pcre-8.44             | 212 KB   | #####| 100% ^  
hdf5-1.10.6           | 3.7 MB   | #####| 100% ^  
pyqt-impl-5.12.3      | 5.9 MB   | #####| 100% ^  
pillow-8.0.1           | 619 KB   | #####| 100% ^  
libpq-12.3             | 2.6 MB   | #####| 100% ^  
click-7.1.2             | 71 KB    | #####| 100% ^  
tqdm-4.55.0            | 77 KB    | #####| 100% ^  
opt_einsum-3.1.0        | 54 KB    | #####| 100% ^  
ca-certificates-2020  | 121 KB   | #####| 100% ^  
certifi-2020.12.5       | 141 KB   | #####| 100% ^  
pyqtwebengine-5.12.1    | 174 KB   | #####| 100% ^  
cloudpickle-1.6.0        | 30 KB    | #####| 100% ^  
coverage-5.3.1           | 256 KB   | #####| 100% ^  
libgcc-ng-9.3.0          | 7.8 MB   | #####| 100% ^  
Preparing transaction: done  
Verifying transaction: done  
Executing transaction: done
```

Abbildung 3.8: Einrichten der Conda Umgebung

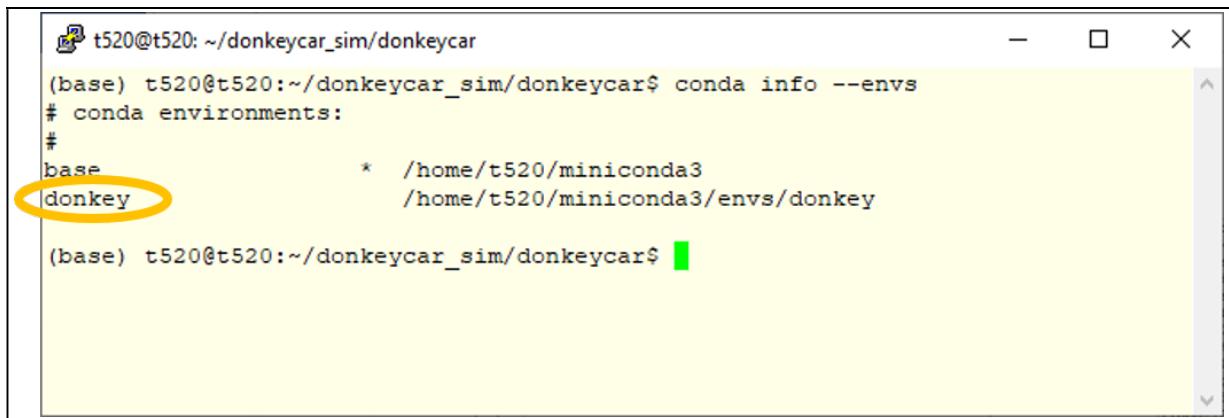
Da ich die Installation nur für Linux testen konnte möchte Ich Ihnen hier dennoch einen Hinweis geben für den Fall das Ihnen viele Fehler und Warnungen angezeigt werden. Bei mir es dann in diesem Fall immer so, dass die virtuelle Umgebung mit dem Namen **donkey** nicht aktiviert werden konnte, da diese nicht angelegt wurde. Die Lösung sollte für alle Betriebssystem die gleiche sein.

Hinweis: Sollten jetzt Fehler erschienen, die besagen, dass ausgewählte Pakete nicht zur Conda Installation passen und Abhängigkeiten Konflikte verursachen dann ist sehr wahrscheinlich die Installationsdatei **ubuntu.yml** veraltet und Sie müssen auf dem Git Hub Repository nach einer aktuellen Version suchen.

Mit dem folgenden Befehl können Sie sich alle Umgebungen auflisten lassen die die conda kennt und verwaltet.

Befehl: conda info --envs

Sie sollten wie im folgenden Bild gezeigt die base und donkey Umgebung sehen.



```
t520@t520: ~/donkeycar_sim/donkeycar
(base) t520@t520:~/donkeycar_sim/donkeycar$ conda info --envs
# conda environments:
#
base                         * /home/t520/miniconda3
donkey                         /home/t520/miniconda3/envs/donkey

(base) t520@t520:~/donkeycar_sim/donkeycar$
```

Abbildung 3.9: Liste der Conda Umgebungen

Wenn die Einrichtung der **donkey** Umgebung unter **Miniconda3** geklappt hat dann aktivieren Sie diese jetzt. Denken Sie immer daran die virtuelle Umgebung mit dem Namen **donkey** zu aktivieren wenn Sie den Rechner z. B. neu gestartet haben.

Befehl: conda activate donkey

Für den Fall das Sie auch eine virtuelle Umgebung einmal entfernen möchten dann können Sie den folgenden Befehl zum Löschen dieser samt allen installierten Packages verwenden.

Befehl: conda env remove --name myenv

Wenn Sie die virtuelle **donkey** Umgebung aktivieren konnten dann installieren Sie jetzt das Donkey Car Framework mit dem folgenden Befehl auf Ihrem PC System.

Befehl: pip install -e .[pc]

Für einen MAC geben Sie bitte folgenden Befehl ein.

Befehl: pip install -e .[pc]

3.2.4 Donkey Car Framework mit GPU Unterstützung einrichten

In diesem Abschnitt folgt die Einrichtung der virtuelle **Miniconda3** Umgebung mit dem Namen **donkey** mit GPU also NVIDIA Grafikkarten Unterstützung. Dieser Abschnitt erklärt die Einrichtung der Donkey Car Umgebung unter Ubuntu. Der Unterschied zu Windows ist in diesem Fall wie die NVIDIA Treiber installiert werden da dies unterschiedlich zu Windows ist.

Damit die Grafikkarte bzw. die GPU auch von TensorFlow genutzt werden kann müssen die passenden CUDA Treiber installiert werden. Dazu öffnen Sie bitte die folgende Seite bei NVIDIA und folgen dort der Installationsanleitung für die CUDA Treiber.

URL: <https://developer.nvidia.com/cuda-11.0-download-archive>

Sehr gute Erfahrungen habe ich gemacht wenn ich die CUDA Treiber als **deb (network)** Installation installiert habe wie im folgenden Bild gezeigt.

Hinweis: Alle jetzt folgenden Installationsschritte wurden in der aktiven conda Umgebung (base) durchgeführt in der später auch das Donkey Car Framework eingerichtet wird.

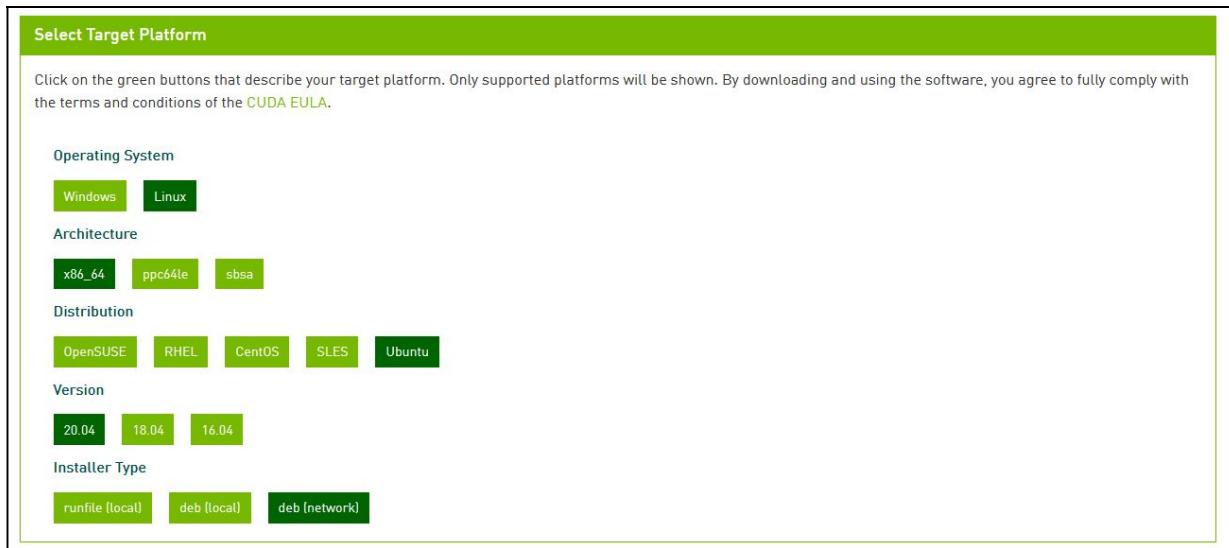


Abbildung 3.10: CUDA Treiber Auswahl

Anschließend bekommen Sie eine Reihe von Befehlen angezeigt die Sie einem nach dem anderen installieren. Zu dem Zeitpunkt als diese Anleitung geschrieben wurde sahen die Befehle wie folgt gezeigt aus.



Abbildung 3.11: NVIDIA CUDA Installations-Befehle

Nach dem die Installation der CUDA Treiber erfolgreich war installieren Sie bitte noch das folgende Paket.

Befehl: reboot

Nach dem Neustart, der vorherigen Installation die CUDA Treiber und der Installation des CUDA Toolkit können Sie sich mit dem folgenden Befehl anzeigen lassen ob ihre Grafikkarte bzw. Grafikkarten auch alle gefunden werden. Wenn diese angezeigt werden sollten die Grafikkarten ab jetzt einwandfrei unter Ubuntu funktionieren.

Für die Anzeige der bei Ihnen im Rechner verbauten Grafikkarten führen Sie den folgenden Befehl aus.

Befehl: watch -n 1 nvidia-smi

Die Anzeige sollte jetzt bei Ihnen ähnlich wie auf dem von mir verwendeten Rechner mit zwei RTX Grafikkarten aussehen.

```
Every 0,5s: nvidia-smi                                         A6000: Sat Jun 26 21:12:56 2021
Sat Jun 26 21:12:56 2021
+-----+
| NVIDIA-SMI 465.19.01      Driver Version: 465.19.01      CUDA Version: 11.3 |
+-----+
| GPU  Name      Persistence-M| Bus-Id      Disp.A | Volatile Uncorr. ECC | |
| Fan  Temp  Perf  Pwr:Usage/Cap| Memory-Usage | GPU-Util  Compute M. |
|                               |             |            | MIG M. |
+-----+
| 0  NVIDIA RTX A6000     On  | 00000000:0A:00.0  On |                    Off | |
| 30%   42C    P5    30W / 300W |      163MiB / 48682MiB |      14%     Default |
|                               |                  |            | N/A      |
+-----+
+
| Processes:
| GPU  GI  CI          PID  Type  Process name          GPU Memory |
|           ID  ID                                     Usage      |
+-----+
| 0    N/A N/A        1271    G    /usr/lib/xorg/Xorg      69MiB |
| 0    N/A N/A        1540    G    /usr/bin/gnome-shell    92MiB |
+-----+
```

Abbildung 3.12: NVIDIA SMI RTX A6000

Sollte Ihre RTX Grafikkarte nicht angezeigt werden dann können Sie versuchen mit dem folgenden Befehl diese zu aktivieren.

Befehl: CUDA_VISIBLE_DEVICES=0

Jetzt haben Sie auf Ihrem Ubuntu System alle notwendigen Treiber installiert und überprüft ob die Grafikkarte auch im System bzw. von CUDA gefunden wird.

Sollten Sie mehrere Grafikkarten in Ihrem System verbaut haben dann sollten diese in der NVIDIA SMI Anzeige einzeln aufgeführt werden mit unterschiedlichen Gerätenummern. Im folgenden Screenshot sind die Geräte IDs 0 und 1 vergeben. In dem Fall das Sie Sie mehrere Grafikkarten verbaut haben um z. B. für eine Schulung die Trainingszeit der neuronalen Netze der Teilnehmer so kurz wie möglich zu halten werden ich später darauf noch einmal eingehen wie Sie die unterschiedlichen Grafikkarten anhand ihrer PCIx Bus Adressen im Donkey Car Framework ansprechen können.

```

Every 1,0s: nvidia-smi                                         ai: Sat Mar 6 05:59:18 2021

Sat Mar 6 05:59:18 2021
+-----+
| NVIDIA-SMI 460.32.03      Driver Version: 460.32.03    CUDA Version: 11.2 |
+-----+
| GPU  Name      Persistence-M| Bus-Id     Disp.A  | Volatile Uncorr. ECC | | |
| Fan  Temp  Perf  Pwr:Usage/Cap| Memory-Usage | GPU-Util  Compute M. |
| |          |          |             | GPU-Util  Compute M. |
| |          |          |             |             MIG M. |
+-----+
| 0  Quadro RTX 8000     On | 00000000:21:00.0 On |           14%   Default |
| 34% 35C     P8    36W / 260W |      301MiB / 48598MiB |           | N/A |
+-----+
| 1  Quadro RTX 8000     On | 00000000:4A:00.0 Off |          0%   Default |
| 33% 28C     P8    12W / 260W |      5MiB / 48601MiB |          | N/A |
+-----+
| Processes:                               GPU Memory |
| GPU  GI  CI   PID  Type  Process name        Usage  |
| ID  ID
+-----+
| 0  N/A N/A  1538   G  /usr/lib/xorg/Xorg      163MiB |
| 0  N/A N/A  1820   G  /usr/bin/gnome-shell    106MiB |
| 0  N/A N/A  2237   G  ...gAAAAAAA --shared-files  28MiB |
| 1  N/A N/A  1538   G  /usr/lib/xorg/Xorg       4MiB |
+-----+

```

Abbildung 3.13: NVIDIA SMI 2 x Quadro RTX 8000

Wenn alles soweit geklappt hat dann können Sie mit der Einrichtung der Mini-Conda Umgebung weiter fortfahren.

Hinweis: Bitte öffnen Sie z. B. mit Nano die `ubuntu.yml` oder `pc.yml` Datei im Pfad `/donkeycar_sim/donkeycar/install/envs` und prüfen welche Version von TensorFlow installiert werden soll. Steht bei Ihnen die folgende Zeile in der `ubuntu.yml` oder `pc.yml` `tensorflow=2.2.0` dann ändern Sie den Namen jetzt in `tensorflow-gpu=2.2.0` ab. So wird TensorFlow mit GPU Unterstützung auf Ihrem System installiert.

Nach dem Sie die Änderung der Versionsnummer vorgenommen haben können Sie mit der Installation weiter machen.

Führen Sie jetzt den folgenden Befehl z. B. im `/donkeycar_sim/donkeycar` Ordner aus abhängig davon wo Sie das Framework herunter geladen haben.

Hinweis: Für Linux und MAC gibt es unterschiedliche Installationsdateien. Daher folgt jetzt die Unterscheidung zwischen Linux und MAC.

Für Linux:

Befehl: `conda env create -f install/envs/ubuntu.yml`

Für MAC OS:

Befehl: `conda env create -f install/envs/mac.yml`

```

a6000@a6000: ~/donkeycar_sim/donkeycar
Requirement already satisfied: imageio>=2.3.0 in /home/a6000/miniconda3/envs/donkey/lib/python3.7/site-packages (from scikit-image->keras-vis==0.5.0->-r /home/a6000/donkeycar_sim/donkeycar/install/envs/condaenv.qk0aaikp.requirements.txt (line 1)) (2.9.0)
Requirement already satisfied: scipy>=0.19.0 in /home/a6000/miniconda3/envs/donkey/lib/python3.7/site-packages (from scikit-image->keras-vis==0.5.0->-r /home/a6000/donkeycar_sim/donkeycar/install/envs/condaenv.qk0aaikp.requirements.txt (line 1)) (1.6.2)
Requirement already satisfied: decorator<5,>=4.3 in /home/a6000/miniconda3/envs/donkey/lib/python3.7/site-packages (from networkx>=2.0->scikit-image->keras-vis==0.5.0->-r /home/a6000/donkeycar_sim/donkeycar/install/envs/condaenv.qk0aaikp.requirements.txt (line 1)) (4.4.2)
Building wheels for collected packages: keras-vis
  Building wheel for keras-vis (setup.py): started
  Building wheel for keras-vis (setup.py): finished with status 'done'
  Created wheel for keras-vis: filename=keras_vis-0.5.0-py2.py3-none-any.whl size=30966 sha256=5dd6d9b5185b0b9d85499b324b969c4081861c43ace09e9558fb3b7aaa709c6
  Stored in directory: /tmp/pip-ephem-wheel-cache-rjzbyr62/wheels/35/94/58/fb9de29ebb6305a7f828e0605902f66f5425c17ed6b4c18e66
Successfully built keras-vis
Installing collected packages: simple-pid, opencv-python-headless, keras-vis
Successfully installed keras-vis-0.5.0 opencv-python-headless-4.5.2.54 simple-pid-1.0.1

done
#
# To activate this environment, use
#
#     $ conda activate donkey
#
# To deactivate an active environment, use
#
#     $ conda deactivate
(base) a6000@a6000: ~/donkeycar_sim/donkeycar$ 

```

Abbildung 3.14: Donkey Car Framework erfolgreich installiert

Da ich die Installation nur für Linux testen konnte möchte Ich Ihnen hier dennoch einen Hinweis geben für den Fall das Ihnen viele Fehler und Warnungen angezeigt werden. Bei mir es dann in diesem Fall immer so, dass die virtuelle Umgebung mit dem Namen donkey nicht aktiviert werden konnte, da diese nicht angelegt wurde. Die Lösung sollte für alle Betriebssystem die gleiche sein.

Hinweis: Sollten jetzt Fehler erschienen, die besagen, dass ausgewählte Pakete nicht zur Conda Installation passen und Abhängigkeiten Konflikte verursachen dann ist sehr wahrscheinlich die Installationsdatei **ubuntu.yml** veraltet und Sie müssen auf dem Git Hub Repository nach einer aktuellen Version suchen.

Mit dem folgenden Befehl können Sie sich alle Umgebungen auflisten lassen die die conda kennt und verwaltet.

Befehl: conda info --envs

Sie sollten wie im folgenden Bild gezeigt die base und donkey Umgebung sehen.

```

a6000@a6000: ~/donkeycar_sim/donkeycar$ conda info --envs
# conda environments:
#
base          * /home/a6000/miniconda3
donkey        /home/a6000/miniconda3/envs/donkey

```

Abbildung 3.15: Liste der Conda Umgebungen

Wenn die Einrichtung der **donkey** Umgebung unter **Miniconda3** geklappt hat dann aktivieren Sie diese jetzt. Denken Sie immer daran die virtuelle Umgebung mit dem Namen **donkey** zu aktivieren wenn Sie den Rechner z. B. neu gestartet haben und mit dem Donkey Car Framework arbeiten möchten.

Befehl: conda activate donkey

Das nachfolgende Bild erklärt wie Sie erkennen können ob Sie in der aktiven Anaconda **donkey** Umgebung sind.

```

a6000@a6000: ~/donkeycar_sim/donkeycar$ conda info --envs
# conda environments:
#
base          * /home/a6000/miniconda3
donkey        /home/a6000/miniconda3/envs/donkey

```



```

a6000@a6000: ~/donkeycar_sim/donkeycar$ conda activate donkey

```

Abbildung 3.16: Erklärung der Aktivierung der Anaconda donkey Umgebung

1. An dem vorgestellten (base) in der Terminalzeile erkennen Sie, dass Sie sich in der aktiven virtuellen Conda Umgebung **base** befinden.
2. Mit dem Befehl „conda activate donkey“ aktivieren Sie die virtuelle Anaconda Umgebung **donkey**.
3. An dem vorgestellten (donkey) erkennen Sie, dass Sie in der aktiven virtuellen Umgebung **donkey** sind.

Für den Fall das Sie auch eine virtuelle Umgebung einmal entfernen möchten dann können Sie den folgenden Befehl zum Löschen dieser samt allen installierten Packages verwenden.

Befehl: conda env remove --name myenv

Wenn Sie die virtuelle **donkey** Umgebung aktivieren konnten dann installieren Sie jetzt das Donkey Car Framework mit dem folgenden Befehl auf Ihrem PC System.

Befehl: pip install -e .[pc]

Für einen MAC geben Sie bitte folgenden Befehl ein.

Befehl: pip install -e .[pc]

Jetzt sollten Sie noch die empfohlenen zusätzlichen Bibliotheken mit dem nachfolgenden Befehl installieren.

Befehl: conda install pytorch torchvision torchaudio cudatoolkit -c pytorch

Mit dem folgenden Befehl können Sie sich jetzt in der aktiven Conda Umgebung donkey die installierte CUDNN, TensorFlow und CUDA Toolkit Versionen anzeigen lassen.

Befehl: conda list | grep -E "cudnn|tensorflow|cudatoolkit"

```
a6000@a6000:~$ conda activate donkey
(base) a6000@a6000:~$ conda list | grep -E "cudnn|tensorflow|cudatoolkit"
cudatoolkit          10.1.243           h6bb024c_0
cudnn                7.6.5              cuda10.1_0
pytorch               1.7.0           py3.7_cuda10.1.243_cudnn7.6.3_0    pytorch
tensorflow            2.2.0           gpu_py37h1a511ff_0
tensorflow-base       2.2.0           gpu_py37h8a81be8_0
tensorflow-estimator  2.5.0           pyh7b7c402_0
tensorflow-gpu        2.2.0           h0d30ee6_0
(base) a6000@a6000:~$
```

Abbildung 3.17: Detailinformationen über die CONDA Umgebung donkey

3.2.5 Einrichten des Simulators unter Ubuntu

Sie haben bereits schon die Donkey Car Framework Erweiterung gym-donkeycar für den Simulator herunter geladen. Daher gehen Sie jetzt wieder in den Ordner **donkeycar_sim** der in Ihrem Homeverzeichnis liegt und wechseln mit dem folgenden Befehl in den Ordner **gym-donkeycar**.

Befehl: cd gym-donkeycar

Hinweis: Falls noch nicht geschehen und um Fehler zu vermeiden aktivieren Sie jetzt mit dem folgenden Befehl die die virtuelle Umgebung **donkey**.

Befehl: conda activate donkey

Wenn Sie in der virtuellen Umgebung „donkey“ sind dann starten Sie die Installation des Donkey Car Simulator Frameworks mit dem folgenden Befehl.

Befehl: pip install -e .[gym-donkeycar]

Anschließend, wenn die Installation durchlaufen ist initialisieren Sie noch eine Donkey Car Instanz des Simulators. Dazu gehen Sie wieder in den Ordner `donkeycar_sim` und führen den folgenden Befehl aus.

Befehl: `donkey createcar --path ~/mysim`

Mit diesem Befehl würde der Ordner `mysim` im Homeverzeichnis Ihres Users angelegt. In diesem Ordner finden Sie verschiedene Dateien wie die `myconfig.py` Datei sowie Ordner.

Unity Umgebung einrichten

Jetzt brauchen Sie noch die **Unity** Umgebung welche die virtuelle Welt mit verschiedenen Rennstrecken bereitstellt in der Sie mit dem Donkey Car Simulator anschließend ihre Runden drehen werden. Dazu müssen Sie noch die Unity Umgebung als ZIP-Datei herunter laden. Möchten Sie sich zunächst einmal informieren welche Version der Unity Simulator Umgebung aktuell veröffentlicht wurden so können Sie das über die folgende URL machen.

URL: <https://github.com/tawnkramer/gym-donkeycar/releases>

Zu dem Zeitpunkt als ich diese Anleitung geschrieben habe war die Version **Race Edition v21.04.15** die aktuelle und von mir für diese Beschreibung verwendete Version.

Laden Sie jetzt mit dem folgenden Befehl die entsprechende ZIP Datei für Ihr Betriebssystem herunter.
Linux

Befehl: wget <https://github.com/tawnkramer/gym-donkeycar/releases/download/v21.04.15/DonkeySimLinux.zip>

MAC:

Befehl: wget <https://github.com/tawnkramer/gym-donkeycar/releases/download/v21.04.15/DonkeySimMac.zip>

Nach dem Sie die *.zip Datei herunter geladen haben entpacken Sie diese im Verzeichnis `donkeycar_sim`. Jetzt haben Sie einen neuen Ordner der den Namen `DonkeySimLinux` für die Linux Installation trägt. Ähnlich wird es an dieser Stelle auch für den MAC sein.

Die ZIP-Datei entpacken Sie im Terminal Fenster mit dem folgenden Befehl.

Befehl: `unzip DonkeySimLinux.zip`

Führen Sie jetzt unter der Grafischen Oberfläche ihres Host PC Betriebssystems den Simulator aus. Das ist einfach nur ein Test, ob alles soweit mit dem Simulator funktioniert. Unter Linux und wohl auch MAC starten Sie diesen mit dem folgenden Befehl. Den Simulator finden Sie in dem Ordner der nach dem Entpacken der `DonkeySimLinux.zip` angelegt wurde.

Befehl: `./donkey_sim.x86_64`

Hinweis: Sollten Sie nicht die Rechte haben die Datei „`donkey_sim.x86_64`“ ausführen zu dürfen dann ändern Sie diese mit dem folgenden Befehl entsprechend ab.

Befehl: `sudo chmod 777 donkey_sim.x86_64`

Nach dem Sie den Befehl zum Ausführen des Simulators erfolgreich ausführen konnten startet der Donkey Car Simulator und zeigt Ihnen die folgende Oberfläche an. Gut zu erkennen sind die verschiedenen Rennstrecken die bereits vorhanden sind.



Abbildung 3.18: Unity Donkey Car Simulator

Jetzt haben Sie das Donkey Car Framework und alle Komponenten für die Simulation auf Ihrem Rechner installiert. Als nächstes müssen Sie das Donkey Car Framework noch konfigurieren. Zunächst aber können Sie in der Unity Umgebung einfach einmal herum fahren ohne das Donkey Car Framework zu nutzen. Im Kapitel 0 geht es dann weiter mit der Konfiguration des Donkey Car Frameworks. Hier wird nicht mehr unterschieden ob diese für Windows oder Ubuntu vorgenommen wird. Im Fall von Unterschieden zwischen Windows und Ubuntu werden diese kurz erläutert.

3.3 Konfiguration des Donkey Car Frameworks (*myconfig.py*)

Die nachfolgende Konfiguration des Donkey Car Frameworks ist für Windows und Ubuntu so gut wie dieselbe. Es gibt lediglich ein paar kleine Unterschiede auf die jeweils im Text eingegangen wird. Damit das Donkey Car Framework, wenn dieses gestartet wird, die Simulator Umgebung lädt muss die ***myconfig.py*** Datei noch angepasst werden. Denn historisch gesehen wurde das Framework entwickelt um auf physischen Modellautos installiert zu werden und wurde nicht mit dem Fokus auf eine Simulation entwickelt.

Die Datei ***myconfig.py*** finden Sie in den Ordner **mysim** in dem Sie zuvor die Donkey Car Umgebung initialisiert hatten.

Windows:

Um die ***myconfig.py*** Datei zu editieren verwenden Sie am besten Visual Studio Code.

Ubuntu:

Befehl: cd ~/mysim

Um die ***myconfig.py*** Datei zu editieren verwenden Sie am besten den Text Editor Nano.

Befehl: nano myconfig.py

Jetzt haben Sie die Datei ***myconfig.py*** ob unter Windows oder Ubuntu geöffnet und können die notwendigen Anpassungen vornehmen.

Wenn Sie mehrere Läufe mit Ihrem Donkey Car im Simulator absolvieren, dann ist es sehr gut für eine spätere Qualitätskontrolle wenn Sie die einzelnen Läufe auch unterscheiden können. Das ist möglich indem Sie den Parameter ***AUTO_CREATE_NEW_TUB*** in der ***myconfig.py*** auf True setzen. Angepasst sollte die Zeile jetzt wie folgt aussehen.

- ***AUTO_CREATE_NEW_TUB* = True**

Hinweis: Auch ist es erst dann möglich von verschiedenen Rechnern oder im Verbund mit Freunden Trainingsdaten zu tauschen wenn diese in einzelnen Ordnern liegen und nicht alle in einem Ordner gesammelt werden.

Als nächstes müssen Sie ziemlich bis an das Ende der ***myconfig.py*** gehen um die nachfolgenden Zeilen zu finden. Diese müssen Sie entsprechend durch entfernen des vorgestellten # aktivieren. Diese Zeilen sorgen dafür, dass die Donkey Car Simulator Umgebung bestehend aus dem Unity Simulator gestartet wird wenn Sie das Donkey Car Framework ausführen über das Terminal Fenster ausführen.

DONKEY_GYM = True

Windows Spezifika:

Wichtig ist, dass Sie den richtigen Pfad zu dem Unity Simulator und der EXE-Datei ***donkey_sim.exe*** hinterlegen. Denn über die ***myconfig.py*** Datei wird der Simulator bzw. die Unity Umgebung gestartet. Den Simulator hatten Sie bereits schon herunter geladen und die ZIP Datei des Simulators entpackt.

- DONKEY_SIM_PATH = "D:\\DonkeySimWin\\donkey_sim.exe"
- **Hinweis:** Achten Sie unbedingt unter Windows darauf, dass Sie bei der Angabe der Pfade die doppelten \\ verwenden.

Ubuntu Spezifika:

Achten Sie darauf, den richtigen Pfad zu Ihrem Unity Simulator bei dem Platzhalter <user name> einzutragen.

DONKEY_SIM_PATH = "/home/<user name>/donkeycar_sim/DonkeySimLinux/donkey_sim.x86_64"

Rennstrecken auswählen

Es stehen verschiedene Rennstrecken zur Verfügung. Sie können so wie ich hier einfach einmal alle einfügen und durch setzen der # aktivieren oder deaktivieren.

- #DONKEY_GYM_ENV_NAME = "donkey-generated-track-v0"
- DONKEY_GYM_ENV_NAME = "donkey-warehouse-v0"
- #DONKEY_GYM_ENV_NAME = "donkey-generated-roads-v0"
- #DONKEY_GYM_ENV_NAME = "donkey-avc-sparkfun-v0"
- #DONKEY_GYM_ENV_NAME = "donkey-roboracingleague-track-v0"
- #DONKEY_GYM_ENV_NAME = "donkey-waveshare-v0"

Die angepassten und eingefügten Zeilen sollten jetzt bei Ihnen wie im nachfolgenden Bild für ubuntu gezeigt bis auf den <User Name> identisch sein.

```

a6000@A6000: ~
GNU nano 5.4
/home/a6000/mysim/myconfig.py

# WEB CONTROL
import os
# import os is important for the configuration of the proxy settings below
WEB_CONTROL_PORT = int(os.getenv("WEB_CONTROL_PORT", 8887)) # which port to listen on when making a web controller
WEB_INIT_MODE = "local" # which control mode to start in. one of user|local_angle|local. Setting local will start in ai mode.

# RECORD OPTIONS
# RECORD_DURING_AI = False      #normally we do not record during ai mode. Set this to true to get image and steering records for your AI. Be careful not to use them to train.
AUTO_CREATE_NEW_TUB = True       #create a new tub (tub_VV_MM_DD) directory when recording or append records to data directory directly

# DonkeyGym
# Only on Ubuntu linux, you can use the simulator as a virtual donkey and
# issue the same python manage.py drive command as usual, but have them control a virtual car.
# This enables that, and sets the path to the simulator and the environment.
# You will want to download the simulator binary from: https://github.com/tkmkramer/donkey_gym/releases/download/v18.9/DonkeySimLinux.zip
# Then extract that and modify DONKEY_SIM_PATH.

DONKEY_GYM = True
DONKEY_SIM_PATH = "/home/a6000/donkeycar_sim/DonkeySimLinux/donkey_sim.x86_64" #"/home/tkmkramer/projects/sdsandbox/sdSim/build/DonkeySimLinux/donkey_sim.x86_64" when racing on v1.9
DONKEY_GYM_ENV_NAME = "donkey-warehouse-v0" # ("donkey-generated-track-v0")|("donkey-generated-roads-v0")|("donkey-warehouse-v0")|("donkey-avc-sparkfun-v0")
GYM_CONF = { "body_style" : "donkey", "body_rgb" : (128, 128, 128), "car_name" : "Ingmar", "font_size" : 40 } # body style(donkey|bare|car01) body rgb 0-255
GYM_CONF["country"] = "Germany"
GYM_CONF["bot"] = "race Virtual and real robo cars."
GYM_HOST = "127.0.0.1" # when racing on virtual-race-league use host "trainmydonkey.com"
SIM_ARTIFICIAL_LATENCY = 0 # this is the millisecond latency in controls. Can use useful in emulating the delay when using a remote server. values of 100 to 400 probably work well
# Help          ^O Write Out    ^W Where Is    ^K Cut           ^T Execute      ^C Location     ^U Undo        ^I Set Mark    M-] To Bracket M-Q Previous   ^B Back
# Read File    ^R Read File   ^A Replace      ^P Paste         ^J Justify     ^G Go To Line  M-U Redo      ^O Where Was  M-W Next     ^F Forward
# Exit          ^X Exit

```

Abbildung 3.19: *myconfig.py* Simulator Anpassung

Soll im Simulator das Donkey Car Ihren Namen tragen und farblich etwas angepasst werden dann können Sie noch die folgenden Zeilen direkt unter der Liste der Rennstrecken ergänzen oder anpassen wenn diese bereits vorhanden sind.

- GYM_CONF = { "body_style" : "donkey", "body_rgb" : (128, 128, 128), "car_name" : "Ingmar", "font_size" : 40}
- GYM_CONF["racer_name"] = "Ingmar"
- GYM_CONF["country"] = "Germany"
- GYM_CONF["bio"] = "I race robots."
- **AI_THROTTLE_MULT=1.0**

Übernehmen Sie die Zeilen 1:1 dann trägt ihr Donkey Car den Namen „Brutus“ und ist von der Farbe in einen Art Grünton getaucht wie das folgende Bild aus dem Unity Simulator zeigt.



Abbildung 3.20: individuelle Anpassung im Simulator

Joystick Konfiguration Windows

Bei der Steuerung haben Sie verschiedene Möglichkeiten. Probieren Sie einfach einmal aus mit welche Sie am besten zurechtkommen. Ich verwende auch hier wieder meinen **Sony PS4** Kontroller für die Steuerung des Donkey Cars über das offene Browserfenster.

Im nachfolgenden Bild habe ich den Button mit einem Orange farbigen Kringel hervorgehoben über den Sie unter dem Chrome Browser die Steuerung des Doneky Cars über einen Kontroller wie einem Sony PS4 Kontroller aktivieren können.

Hinweis: Leider habe ich bis jetzt keinen Weg gefunden wie ein PS4 Kontroller direkt für die Steuerung des Donkey Cars über die **Anaconda** Umgebung genutzt werden kann. Der einzige Ausweg ist die Web-Oberfläche zu starten in z. B. Chrome Browser und über diesen dann das Donkey Car mit einem Hardware-Kontroller zu steuern. Hier ist es dann auch wieder möglich den PS4 Kontroller zu verwenden.

Während Sie jetzt ihre Runden drehen werden auch schon die ersten Trainingsdaten aufgezeichnet. Diese finden Sie im Ordner **mysim/data/images**. Mehr zu dem Thema was gute Trainingsdaten ausmacht und wie Sie das neuronale Netz mit diesen Daten trainieren erfahren Sie im Kapitel 3.4.“.

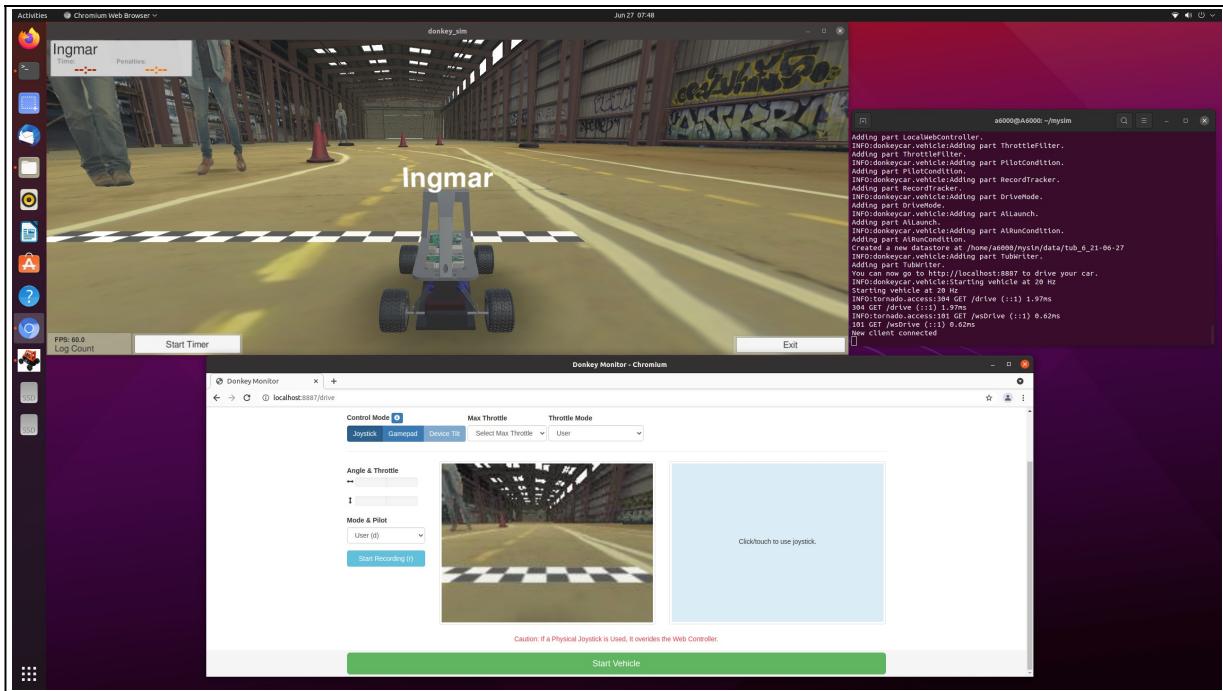


Abbildung 3.21: Steuerung des Donkey Cars über die Web-Oberfläche

Hinweis: Bitte installieren Sie für die Steuerung des Donkey Cars über die Web-Oberfläche als Browser Chromium. Mit diesem hatte ich keine technischen Probleme was die Steuerung mit der Maus also die Eingabe der Steuerbefehle anging.

Joystick Konfiguration Ubuntu

Möchten Sie lieber einen Joystick oder ein Gamepad am Host PC verwenden um das Donkey Car im Simulator zu steuern müssen Sie auch hier wieder die myconfig.py Datei entsprechend anpassen. Wie genau das geht wurde bereits im Kapitel 3.5.1 erklärt.

Das folgende Bild zeigt als kleine Erinnerung die Anpassungen die vorgenommen werden müssen.

```
mc [root@t520]:/home/t520/mysim
GNU nano 4.8          /home/t520/mysim/myconfig.py      Modified
# WEB CONTROL
# WEB_CONTROL_PORT = int(os.getenv("WEB_CONTROL_PORT", 8887)) # which port to listen >
# WEB_INIT_MODE = "user"           # which control mode to start in. one of user|lo>
#
# JOYSTICK
USE_JOYSTICK_AS_DEFAULT = True      #when starting the manage.py, when True, will not >
JOYSTICK_MAX_THROTTLE = 0.8         #this scalar is multiplied with the -1 to 1 thrott>
# JOYSTICK_STEERING_SCALE = 1.0       #some people want a steering that is less sensit>
AUTO_RECORD_ON_THROTTLE = True       #if true, we will record whenever throttle is not >
CONTROLLER_TYPE='ps4'               #(ps3|ps4|xbox|nimbus|wiiu|F710|rc3|MM1|custom) cu>
# USE_NETWORKED_JS = False          #should we listen for remote joystick control ov>
# NETWORK_JS_SERVER_IP = "192.168.0.1" #when listening for network joystick control, wh>
# JOYSTICK_DEADZONE = 0.0            # when non zero, this is the smallest throttle b>

^G Get Help   ^C Write Out  ^W Where Is   ^K Cut Text   ^J Justify   ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell  ^ Go To Line
```

Abbildung 3.22: Anpassung myconfig.py

Jetzt ist alles soweit in der myconfig.py vorbereitet. Speichern Sie bitte die Änderungen ab. Als nächstes können Sie ein paar Runden drehen und alle Einstellungen testen. Dabei werden auch schon die ersten Trainingsdaten aufgezeichnet.

Möchten Sie jetzt ein paar Runden mit Ihrem Donkey Car im Simulator drehen dann führen Sie den folgenden Befehl aus.

Befehl: python manage.py drive

Auch sollte die direkte Unterstützung ihres Kontrollers jetzt funktionieren. Wenn nicht bzw. wenn Tasten ihrer Meinung nach komisch belegt sind, dann erfahren Sie im folgenden Abschnitt mehr zur generellen Konfiguration eines Kontrollers.

Hinweis: An meinem Laptop steht das Bild immer wieder auf dem Kopf wenn ich den manage.py drive Befehl ausführe. Abhilfe schafft hier der folgende Befehl ausgeführt in einem extra Terminal Fenster.

Befehl: xrandr -o normal

3.3.1 Gamepad Konfiguration unter Ubuntu

Viele der Einstellungen die das Gamepad als Kontroller betreffen können über das Donkey Car Framework direkt angepasst werden. Diese Art von Anpassungen werden einmal in der **myconfig.py** Datei sowie die **controller.py** Datei vorgenommen. Die „myconfig.py“ Datei finden Sie im Verzeichnis in dem Sie das Donkey Car Framework instanziert haben. Für den Simulator ist das der Ordner **~/mysim**. Die **controller.py** Datei finden Sie je nach gewählten Pfaden ihrer Installation im Ordner **~/donkeycar_sim/donkeycar/donkeycar/parts**.

Anpassung der Lenkungseingabe

Möchten Sie die Lenkrichtung an Ihrem Kontroller vertauschen (links / rechts) dann öffnen Sie jetzt die **myconfig.py** Datei ihres Projektes und suchen nach der folgenden Zeile.

```
# JOYSTICK_STEERING_SCALE = 1.0
```

Statt der 1.0 schreiben Sie jetzt eine -1.0 und entfernen die Auskommentierung am Beginn der Zeile. Im Ergebnis sieht die angepasste Zeile jetzt wie folgt aus und die Lenkung sollte jetzt Spiegelverkehrt funktionieren.

JOYSTICK_STEERING_SCALE = -1.0

Speichern Sie die **myconfig.py** Datei ab und testen Sie ob Sie Ihr Donkey Car jetzt besser lenken können.

Hinweis: Lenkt Ihnen ihr Donkey Car zu schnell bzw. zu stark, dann können Sie statt 1.0 hier auch eine 0.5 eintragen. Anschließend ist die Lenkung deutlich weicher und nicht mehr so zackig.

Beispiel: JOYSTICK_STEERING_SCALE = 0.5

Anpassung der Beschleunigungseingabe

Die Anpassung bzw. Änderung der Eingabe für die Beschleunigung möchte ich Ihnen nachfolgend näher erklären. Soweit es bei meinem PS4 Gamepad Änderungen bei der Steuerung bedurfte, konnte ich diese via Software vornehmen. Möchten Sie selber die Belegung an Ihrem Gamepad ändern dann müssen Sie die Belegung in der **controller.py** Datei anpassen. Die **controller.py** Datei finden Sie z. B. je nach Installationspfad des Donkey Car Frameworks im Pfad `~/donkeycar_sim/donkeycar/donkeycar/parts/`. Suchen Sie in der **controller.py** Datei für z. B. einen PS4 Gamepad nach der folgenden Klasse „PS4Joystick(Joystick)“ um die Tasten- und Achsenbelegung am PS4 Gamepad zu ändern. In der **controller.py** Datei finden Sie zu Beginn das Mapping auf die Buttons und Achsen z. B. des PS4 Gamepads.

Sind Sie sich nicht sicher welche ID zu welcher Taste oder Achse gehört, dann können Sie dies mit dem folgenden kleinen Python Programm dem sogenannten Joystick Tester in Erfahrung bringen. Das Python Programm habe ich als *.ZIP Datei auf meinem Blog zum Download bereitgestellt.

Download: <https://custom-build-robots.com/jetson-nano-download-de>

Anpassung des Share Buttons (Fahrmodus wechseln)

Ich hatte noch das Problem, dass ich wenn das Donkey Car im Simulator autonom fahren sollte aber auch im echten Modell Roboter-Auto der Schalter um den Fahrmodus zu ändern (user, local angel und local) nicht funktionierte. Ich musste die technische ID des share Buttons in der „controller.py“ Datei ändern so dass ich den User Modi mit der Share Taste wechseln konnte.

Diese Anpassung können Sie in der Klasse class PS4Joystick(Joystick): durchführen. Suchen Sie in der Definition der Klasse nach dem share Button und Sie sollten die folgende Zeile sehen.

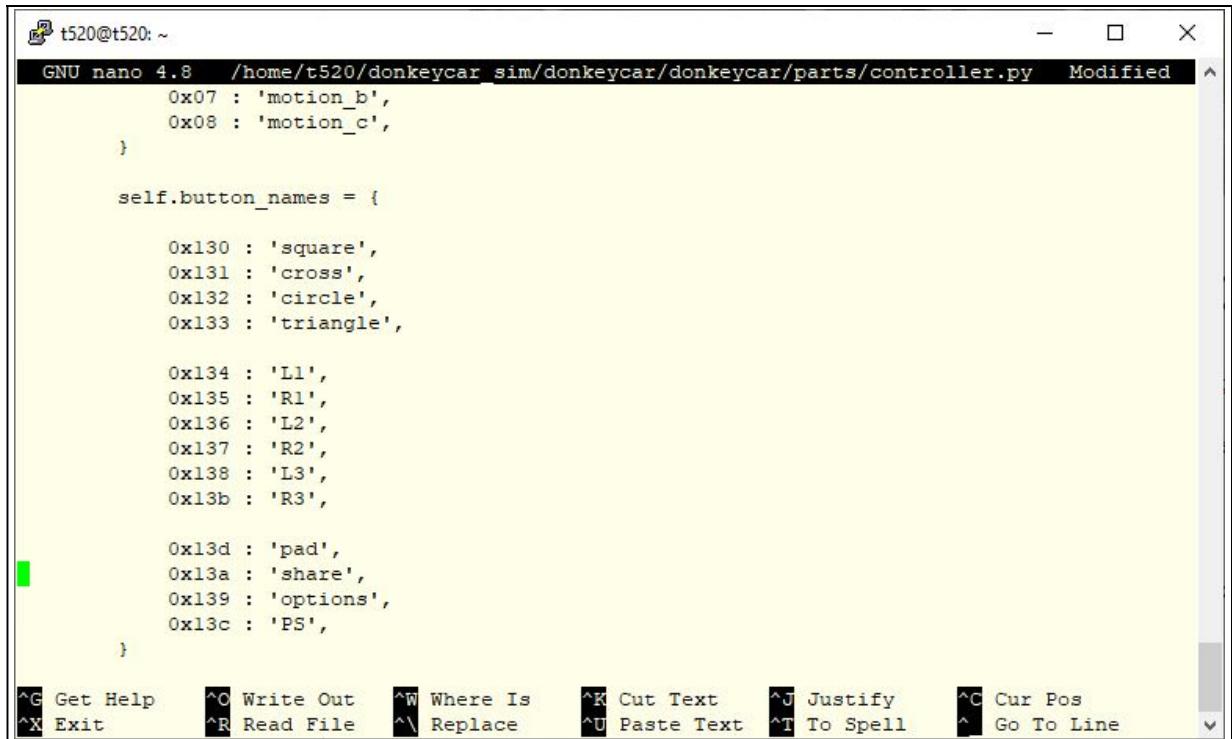
0x138 : 'share',

Ändern Sie diese wie folgt ab:

0x13a : 'share',

Achten Sie darauf, dass Sie ein paar Zeilen oberhalb jetzt eine doppelte Belegung für 0x13a haben. Ändern Sie daher die Belegung für L3 ebenfalls auf die jetzt freie Belegung 0x138 ab.

Das fertige Ergebnis der Anpassungen sollte jetzt wie folgt aussehen.



```

t520@t520: ~
GNU nano 4.8 /home/t520/donkeycar_sim/donkeycar/donkeycar/parts/controller.py Modified
    0x07 : 'motion_b',
    0x08 : 'motion_c',
}

self.button_names = {

    0x130 : 'square',
    0x131 : 'cross',
    0x132 : 'circle',
    0x133 : 'triangle',

    0x134 : 'L1',
    0x135 : 'R1',
    0x136 : 'L2',
    0x137 : 'R2',
    0x138 : 'L3',
    0x13b : 'R3',

    0x13d : 'pad',
    0x13a : 'share',
    0x139 : 'options',
    0x13c : 'PS',
}

```

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
 ^X Exit ^R Read File ^V Replace ^U Paste Text ^T To Spell ^ ^ Go To Line

Abbildung 3.23: Belegung der PS4 Buttons anpassen

Nach dieser Änderung der Joysticksteuerung ist es möglich das Donkey Car mit dem rechten Hebel zu beschleunigen und mit dem Share Button den Modi zu ändern. Den Modi ändern zu können wir etwas später wichtig wenn das Donkey Car im Simulator autonom fahren soll.

Dann ist es jetzt endlich soweit und Sie drehen ein paar Runden mit dem Donkey Car im Simulator und zeichnen Trainingsdaten auf. Ich bin ca. 8 Runden gefahren bzw. habe 16.000 Records aufgezeichnet. Mit diesen Trainingsdaten habe ich anschließend mein neuronales Netz trainiert. Das Ergebnis des fertig trainierten neuronalen Netzes war etwas durchwachsen aber $\frac{1}{2}$ Runde hat es meistens ohne große Fehler geschafft.

Die so aufgezeichneten Trainingsdaten finden Sie im Ordner [~/mysim/data/](#) und dort in verschiedenen tub Ordnern pro Lauf.

Hinweis: Wichtig ist bei der Aufzeichnung der Trainingsdaten, dass diese nur geschrieben werden wenn das Donkey Car auch etwas beschleunigt wird. Das bedeutet wenn Sie in eine Kurve fahren dürfen Sie nicht den rechten Joystick in die neutrale Stellung bringen. Wenn Sie das machen wird während der Kurvenfahrt kein Datensatz geschrieben und es fehlen Ihnen wichtige Trainingsdaten zur Kurvenfahrt.

Achten sie auch auf die Qualität der Trainingsdaten. Trainingsdaten bei denen Sie sich verfahren haben oder die einfach schlecht sind sollten Sie auch umgehend löschen. Dazu entfernen sie am besten den entsprechenden tub* Ordner. Für das Training des neuronalen Netzes sollten im Ordner [~/mysim/data](#) nur Bilder und Json Dateien liegen die Sie auch für das Training des neuronalen Netzes wirklich verwenden möchten.

Im jetzt folgenden Kapitel geht es nur um das Aufzeichnen von guten Trainingsdaten und um das Training des Autopiloten der anschließend im Simulator das Donkey Car lenkt.

3.4 Vom Trainingsdaten erstellen bis zum autonomen Fahren

Der Ablauf ist annähernd immer gleich wie unser Roboter-Auto im Simulator oder in der realen Welt auf seinen Einsatz vorbereitet wird. Erst einmal muss festgelegt wie die Trainingsdaten erzeugt werden. In unserem Fall werden wir diese manuell erstellen durch Behaviroal Clonen. Aber es sind auch Kombinationen möglich z. B. Trainingsdaten die auch teilweise synthetisch erstellt wurden oder Trainingsdaten die verfremdet wurden durch z. B. den Einsatz von Filtern, die die Bilder heller oder dunkler machen. Durch eine Kombination von verschiedenen Techniken kann so am Ende ein deutlich stabileres neuronales Netz angelernt werden. Wir werden jetzt zusammen dem Roboter-Auto bzw. dem neuronalen Netz durch Behaviroal Cloning das autonome Fahren anlernen. Im Prinzip funktioniert der Ablauf wie in folgender Darstellung visualisiert. Innerhalb einer großen Schleife sind drei weitere Schleifen in Gelb, Blau und Grün dargestellt.

- Die gelbe Schleife stellt das Trainingsdaten erzeugen da.
- Die blaue Schleife übernimmt die Aufgabe der Vorbereitungen wie die Definition des Neuronalen Netzes und dessen Training.
- Die grüne Schleife beschreibt wie das fertig trainierte Modell auf das Roboter-Auto übertragen und auf dem Racetrack evaluiert wird wie gut es funktioniert.

Anschließend wiederholt sich der Prozess wieder bis das Roboter-Auto fehlerfrei zu fahren gelernt hat.

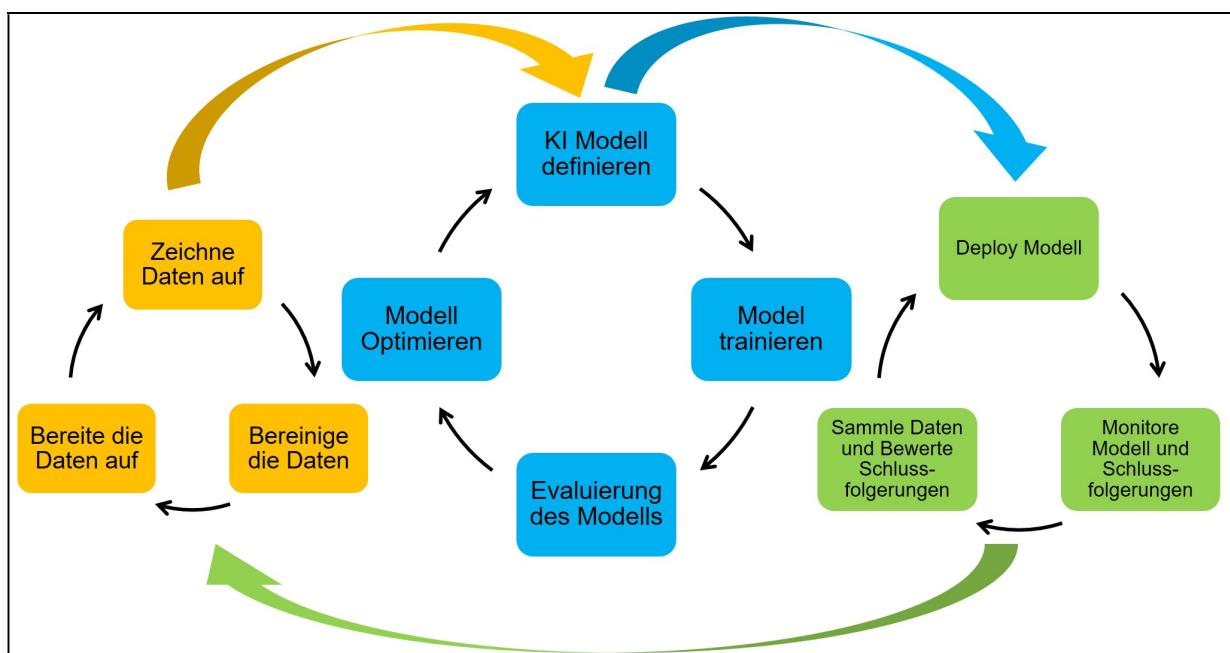


Abbildung 3.24: AI Training- und Anwendungsschleife

Damit Sie einen Eindruck für die Komplexität erhalten habe ich den Kasten „Zeichne Daten auf“ heraus genommen und nachfolgend etwas detaillierter dargestellt. Beim **Behaviroal Cloning** gibt der Mensch vor wie sich das Roboter-Auto verhalten soll. Die Eingabe des Lenkeinschlages und der Geschwindigkeit wird über den Joystick erfasst und zusammen mit einem Bild abgespeichert. Dieser Vorgang wiederholt sich ca. 30x in der Sekunde je nach Konfiguration in der ***myconfig.py***. Die sogenannte Drive-Loop sieht im Detail wie folgt aus.

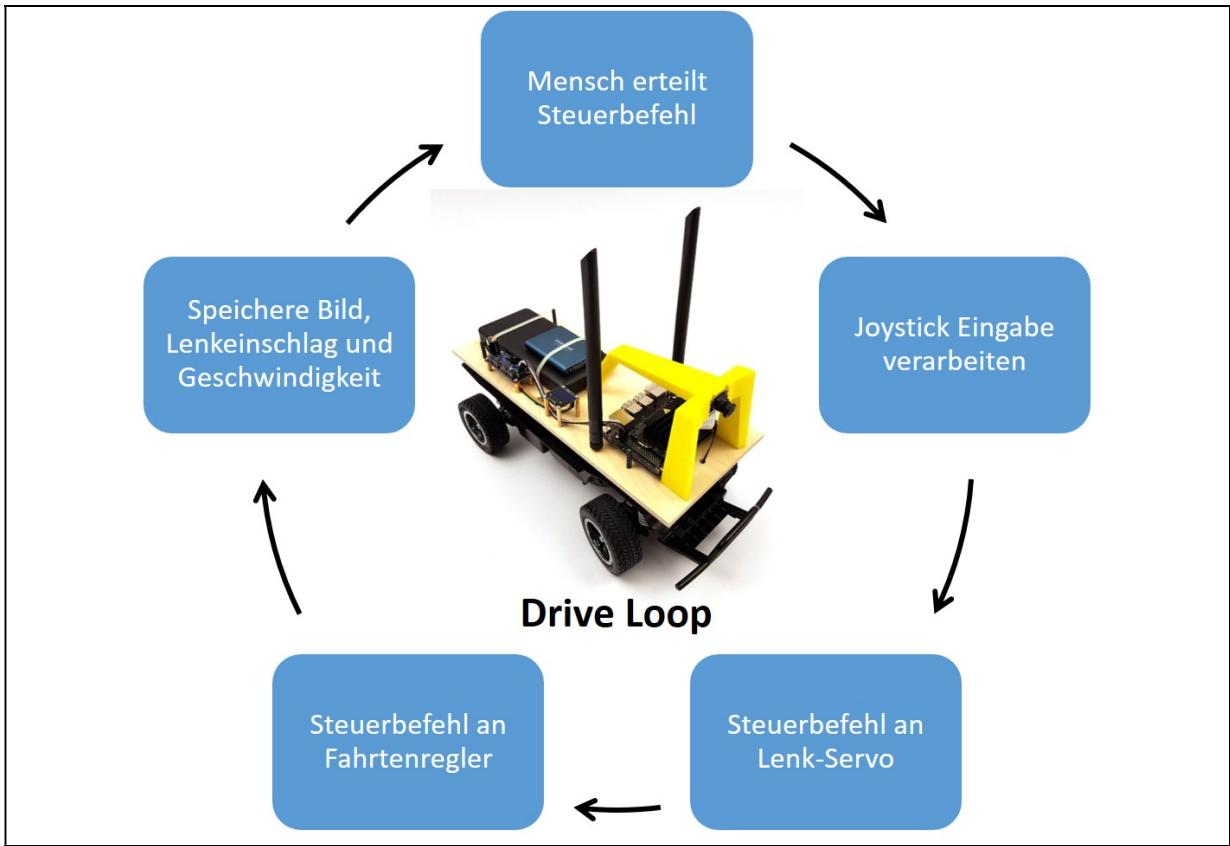


Abbildung 3.25: Drive Loop

Jetzt ist es aber genug mit der grauer Theorie und Sie starten lieber gleich die Trainingsdatenaufzeichnung im Simulator.

3.4.1 Trainingsdaten im Simulator aufzeichnen

Ab jetzt gibt es keine großen Unterschiede mehr zwischen der Linux Welt und der Windows Welt. Daher werde ich für den Fall das es einen Unterschied gibt diesen im Text erkenntlich machen. Um die Aufzeichnung der Trainingsdaten zu starten wechseln Sie in den Ordner `mysim` in dem auch die angepasste `myconfig.py` Datei liegt. Führen Sie jetzt den folgenden Befehl in ihrer aktiven `donkey` Conda Umgebung aus.

Hinweis: Denken Sie daran, falls nicht bereits geschehen jetzt die virtuelle Conda Umgebung `donkey` zu starten.

Befehl: `python manage.py drive`

Der Unity Simulator sollte sich nach ein paar Sekunden öffnen, die in der `myconfig.py` konfigurierte Rennstrecke laden und das Donkey Car vor der Ziellinie platzieren. Nach ein paar Sekunden sollten Sie dann wiederum das Donkey Car mit dem Joystick steuern können. Drehen Sie jetzt die ersten Runden auf der Rennstrecke und erstellen Sie so die Trainingsdaten.



Abbildung 3.26: Donkey Car Simulator live

Jetzt kann es passieren, dass Sie das Donkey Car nicht über den rechten Joystick beschleunigen können und eventuell bei der Lenkung links und rechts vertauscht ist. Möchten Sie Ihr Gamepad individuell anpassen dann können Sie im Kapitel „Gamepad Konfiguration unter Ubuntu“ nachlesen wie das genau im Detail funktioniert.

3.4.2 Das neuronale Netz trainieren Windows

Mit der aktuellen Donkey Car Version werden nicht mehr pro *.jpg Datei auch eine *.json Datei erstellt die die Lenk- und Beschleunigungswerte zu dem einzelnen Bild speichert. Vielmehr wird in einem Ordner oberhalb des `images` Ordner eine `catalog` Datei erstellt die genau diese Informationen pro Bild speichert.

Haben Sie ausreichend Trainingsdaten erstellt um die 8.000 Bilder dann können Sie mit dem folgenden Befehl das Donkey Car Framework im Conda Prompt beenden.

Befehl: **STRG + C**

Machen Sie sich jetzt mit den aufgezeichneten Trainingsdaten vertraut und schauen sich einmal die aufgezeichneten Bilder in Ruhe an. Die zu den Bildern passenden Lenk- und Beschleunigungswerte sind in der `catalog` Datei eine Ebene oberhalb des `images` Ordner gespeichert.

Möchten Sie jetzt das neuronale Netz trainieren dann führen Sie bitte den folgenden Befehl aus wenn alle Bilder in einem Ordner liegen.

Befehl: `python train.py --tubs=data --model models/mypilot.h5`

Haben Sie mehrere Ordner in denen die Bilder für das Training des neuronalen Netztes liegen dann sieht der Befehl wie folgt aus. Gut zu erkennen sind die einzelnen Ordner die zum Training heran gezogen werden sollen.

Befehl: `python train.py --tubs=data/tub_test/,data/tub_2_21-01-16/,data/tub_1_21-01-16/,data/tub_3_21-01-16/ --model models/big.h5`

Das Training sollte jetzt starten und Sie bekommen im Terminal Fenster angezeigt wie **TensorFlow** die Trainingsdaten in ein Set für das Training und in ein Set für die Validierung aufteilt.

```

Anacoda Prompt (miniconda3) - deactivate - python train.py --tubs=data --model models/mypilot.h5

dropout_6 (Dropout)      (None, 50)      0      dense_2[0][0]
n_outputs0 (Dense)       (None, 1)        51      dropout_6[0][0]
n_outputs1 (Dense)       (None, 1)        51      dropout_6[0][0]
=====
Total params: 817,028
Trainable params: 817,028
Non-trainable params: 0

None
Using catalog D:\mysim\data\catalog_0.catalog
Loading tubs from paths ['data']
Records # Training 521
Records # Validation 131
Epoch 1/100
5/5 [=====] - ETA: 0s - loss: 0.4137 - n_outputs0_loss: 0.0833 - n_outputs1_loss: 0.3304
Epoch 00001: val_loss improved from inf to 0.35031, saving model to models/mypilot.h5
5/5 [=====] - 8s 2s/step - loss: 0.4137 - n_outputs0_loss: 0.0833 - n_outputs1_loss: 0.3304 - val_lo
ss: 0.3503 - val_n_outputs0_loss: 0.0579 - val_n_outputs1_loss: 0.2924
Epoch 2/100
5/5 [=====] - ETA: 0s - loss: 0.3544 - n_outputs0_loss: 0.0759 - n_outputs1_loss: 0.2785
Epoch 00002: val_loss improved from 0.35031 to 0.28372, saving model to models/mypilot.h5
5/5 [=====] - 7s 1s/step - loss: 0.3544 - n_outputs0_loss: 0.0759 - n_outputs1_loss: 0.2785 - val_lo
ss: 0.2837 - val_n_outputs0_loss: 0.0573 - val_n_outputs1_loss: 0.2264
Epoch 3/100
5/5 [=====] - ETA: 0s - loss: 0.3211 - n_outputs0_loss: 0.0792 - n_outputs1_loss: 0.2419

```

Abbildung 3.27: Ausgabe TensorFlow training

Das fertig trainierte neuronale Netz wird dann mit dem Namen ***mypilot.h5*** im Ordner **models** abgespeichert wenn das Training abgeschlossen ist.

3.4.3 Das neuronale Netz trainieren Ubuntu

Nachfolgend erkläre ich Ihnen wie Sie das neuronale Netz mit den von Ihnen im Simulator aufgezeichneten Trainingsdaten trainieren. Dazu beenden Sie den Simulator und das laufende Donkey Car Framework im Terminal Fenster mit dem folgenden Befehl.

Befehl: STRG + C

Die aufgezeichneten Trainingsdaten also alle Bilder und dazugehörige catalog Dateien finden Sie in entsprechenden **images** Ordnern pro Lauf im Ordner **~/mysim/data/**. Eventuell sollten Sie eine Qualitätskontrolle durchführen und Ordner mit Trainingsdaten die nicht so gut sind direkt löschen.

Mit dem folgenden Befehl trainieren Sie jetzt ein neuronales Netz. Führen Sie den Befehl im Terminal Fenster auf Ihrem Host PC aus.

Befehl: `python train.py --tubs=data --model models/mypilot.h5`

Haben Sie mehrere Ordner in denen die Bilder für das Training des neuronalen Netzes liegen dann sieht der Befehl wie folgt aus. Gut zu erkennen sind die einzelnen Ordner die zum Training heran gezogen werden sollen.

Befehl: `python train.py --tubs=data/tub_test/,data/tub_2_21-01-16/,data/tub_1_21-01-16/,data/tub_3_21-01-16/ --model models/big.h5`

Je nach Leistung des Host PCs und der Anzahl der Trainingsdaten kann das Training des neuronalen Netzes schnell abgeschlossen sein oder etwas dauern.

Das folgende Bild zeigt die Ausgabe im Terminal während dem Training eines neuronalen Netzes für das Donkey Car.

```
t520@t520: ~  
=====  
=====  
Total params: 817,028  
Trainable params: 817,028  
Non-trainable params: 0  
  
None  
found 0 pickles writing json records and images in tub /home/t520/mysim/data/tub_1_20-12-30  
/home/t520/mysim/data/tub_1_20-12-30  
collating 8074 records ...  
train: 6459, val: 1615  
total records: 8074  
steps_per_epoch 50  
WARNING:tensorflow:From /home/t520/mysim/train.py:588: Model.fit_generator (from tensorflow.py  
thon.keras.engine.training) is deprecated and will be removed in a future version.  
Instructions for updating:  
Please use Model.fit, which supports generators.  
Epoch 1/100  
50/50 [=====] - ETA: 0s - loss: 0.1933 - n_outputs0_loss: 0.1199 - n_  
50/50 [=====] - 102s 2s/step - loss: 0.1933 - n_outputs0_loss: 0.1199  
- n_outputs1_loss: 0.0734 - val_loss: 0.1175 - val_n_outputs0_loss: 0.0937 - val_n_outputs1_1  
oss: 0.0239  
Epoch 2/100  
3/50 [>.....] - ETA: 59s - loss: 0.1450 - n_outputs0_loss: 0.1195 - n_  
outputs1_loss: 0.0255
```

Abbildung 3.28: TensorFlwo Ausgabe während dem Training

Nach dem das Training abgeschlossen ist wird noch angezeigt wie das Training als solches verlaufen ist. Anhand dem Graphen lassen sich Rückschlüsse bzw. vergleiche zwischen verschiedenen Trainingsläufen und Trainingsdaten ziehen. Das Bild zeigt solch einen Graphen und wie gut bzw. schnell das Netz gelernt hat.

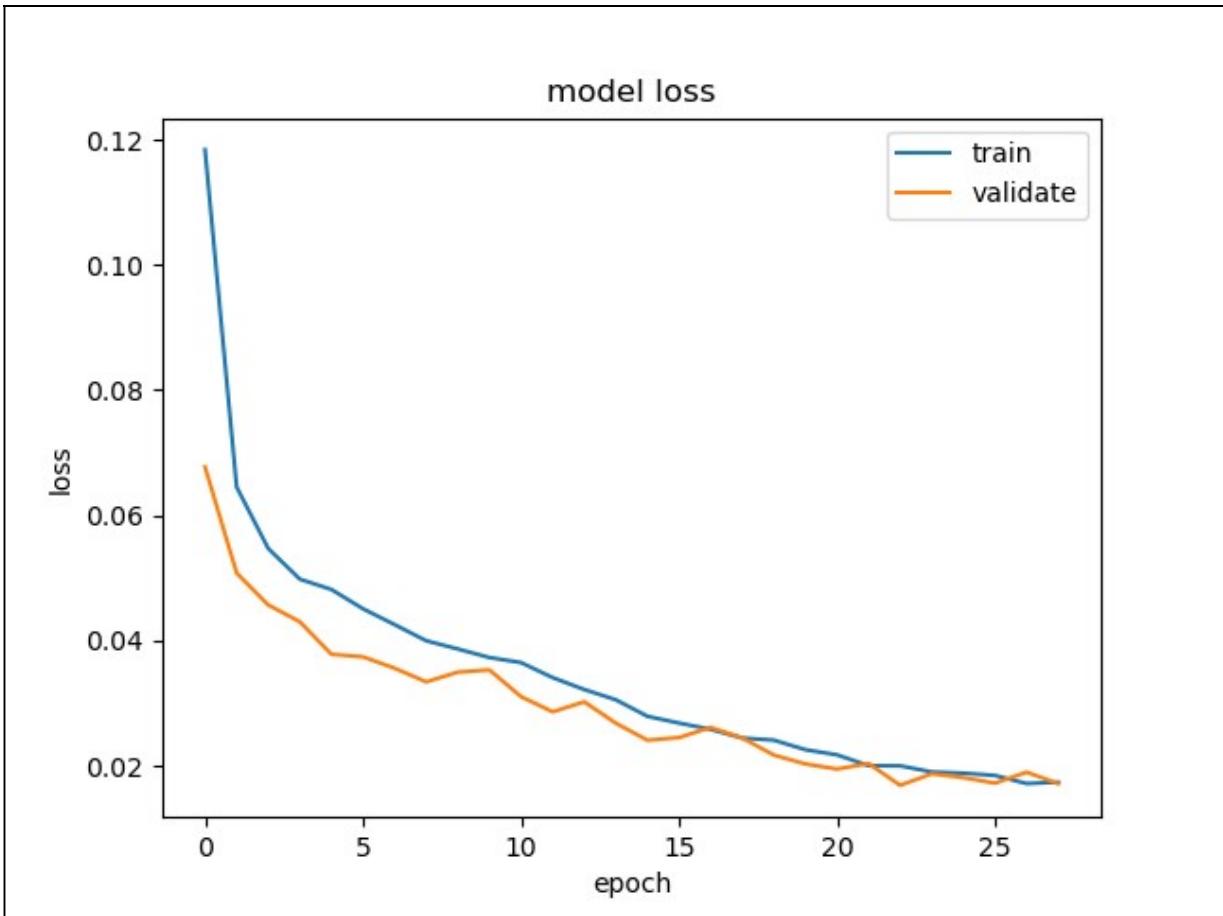


Abbildung 3.29: Darstellung des Trainingsverlaufes

3.4.4 Das fertige neuronale Netz testen

Jetzt ist es an der Zeit zu überprüfen wie gut das fertig trainierte neuronale Netz in der Lage ist im Simulator das Donkey Car um den Kurs zu steuern. Für das Testen des fertigen neuronale Netzes führen Sie den folgenden Befehl im Terminalfenster aus. Das neuronale Netz das in dem nachfolgenden Befehl aufgerufen wird trägt den Namen mypilot.h5. Wenn Sie den Befehl ausführen lädt dieser die Simulator Umgebung und stellt das Donkey Car vor die Ziellinie aber noch fährt das Donkey Car nicht los.

Windows:

Befehl: python manage.py drive --model models/mypilot.h5

Ubuntu / MAC:

Befehl: python ~/mysim/manage.py drive --model models/mypilot.h5

Steuerung unter Windows:

Öffnen Sie wieder einen Browser und melden Sie sich an der Web-Oberfläche des Donkey Car Frameworks an. Über diese Oberfläche hatten Sie bereits die Trainingsdatenaufzeichnung das Donkey Car im Simulator gesteuert. Die URL ist wie folgt aufgebaut:

URL: <IP Adresse ihres Host PC>:8887

Wechseln Sie auf den Modi „Local Angle“ und lassen das neuronale Netz ihr Donkey Car lenken. Die Geschwindigkeit geben Sie selber im rechten Feld mit der Maus vor. In dem Conda Promt in dem Sie das Donkey Car Framework ausgeführt haben, sehen Sie welcher Modi gerade aktiv ist. Wenn jetzt alles gut läuft sollte Ihr Donkey Car ein paar Runden ohne Fehler fahren können.

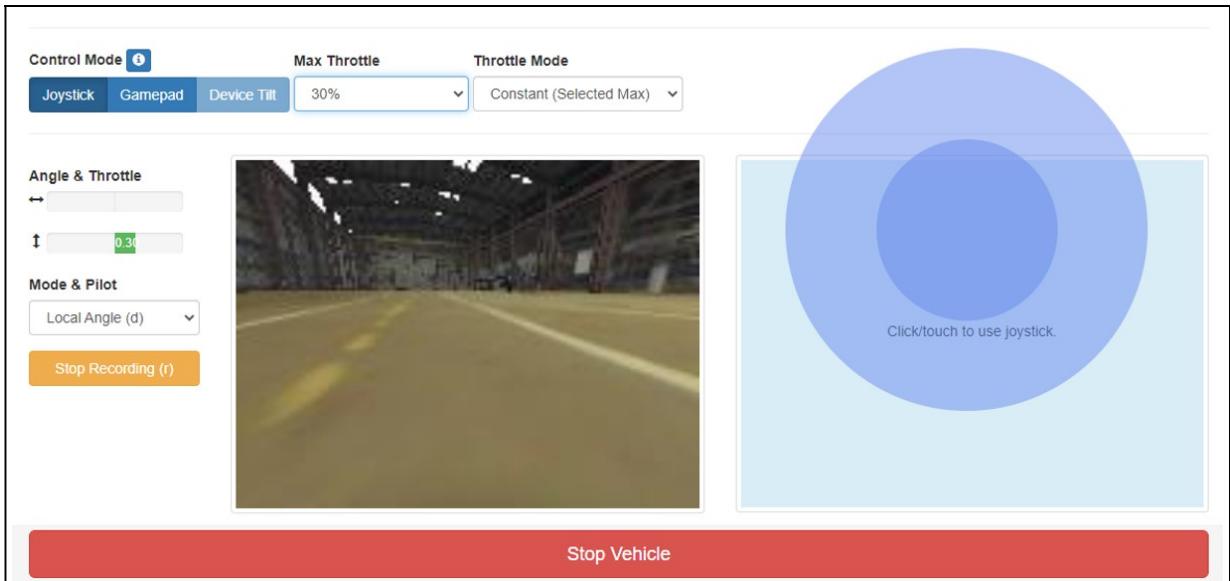


Abbildung 3.30: Donkey Car Framework Web-Steuerung

Als nächstes könnten Sie den Modi auf **Local Pilot** wechseln und das Donkey Car sollte komplett autonom um die Rennstrecke fahren.

Steuerung unter Ubuntu:

Drücken Sie jetzt z. B. die Share Taste 1x an Ihrem PS4 Kontroller um den Fahr-Modi zu wechseln. In der Terminal Ausgabe sehen Sie welchen Modi Sie aktuell aktiv haben. Sie sollten den Modi local_angle auswählen damit ihr Doneky Car im Simulator erst einmal selbstständig lenkt und Sie mit dem Kontroller die Geschwindigkeit vorgeben. Wenn das neuronale Netz in der Lage ist das Donkey Car um den Kurs zu steuern können Sie durch erneutes Drücken der des Share Buttons auf den Modus local wechseln. Jetzt sollte das Donkey Car komplett autonom seine Runden drehen.

Wenn alles bis hier hin funktioniert hat sollte das Donkey Car hoffentlich ohne Fehler seine Runden drehen. Wenn nicht dann zeichnen Sie zusätzliche Trainingsdaten auf und trainieren einen neuen Autopiloten der hoffentlich besser auf der Rennstrecke funktioniert.

3.5 Rennen fahren im Simulator

Es ist auch möglich, im Simulator mehrere Donkey Cars bzw. neuronale Netze gleichzeitig fahren zu lassen. So können z. B. verschiedene Teams ihre neuronalen Netze gegeneinander antreten lassen. Aber nicht nur für Wettbewerbe ist das parallele Ausführen von neuronalen Netzen interessant sondern auch für die Erstellung von speziellen Trainingsdaten. Solche speziellen Trainingsdaten könnten z. B. Daten zum Thema Ausweichen und Überholen beinhalten. Aber es wäre auch möglich Daten zu erfassen um z. B. in einem Rennen andere Autos vom Kurz abzudrängen. Aber Sie werden es schon ahnen, dass das Erstellen solcher speziellen Trainingsdaten gar nicht so einfach ist.

Das folgende Bild zeigt zwei mit unterschiedlichen Trainingsdaten trainierte Autopiloten die in einem Rennen gegeneinander antreten.

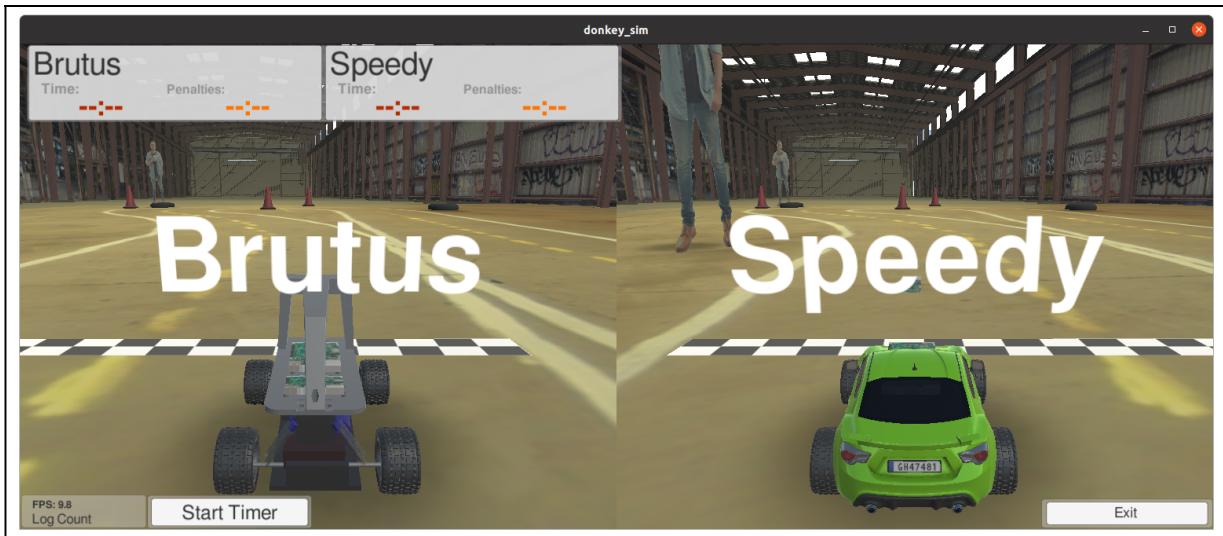


Abbildung 3.31: Rennen fahren im Simulator

Es ist möglich mehrere Autopiloten auf einem Rechner zu starten die sich dann an der Unity Engine anmelden oder eben remote über das Internet. So können z. B. Teilnehmer oder Teams weltweit ihre Netze testen. Eine Herausforderung die dann aber Berücksichtigt werden muss ist sicher die Latenz in der Kommunikation wenn die Rechner weiter voneinander entfernt sind.

Im folgenden Abschnitt gehe ich auf die *myconfig.py* Konfigurationsdatei ein.

3.5.1 Anpassung der myconfig.py

Damit mehrere Autopiloten bzw. neuronale Netze gleichzeitig in einem Simulator geladen werden können müssen ein paar wenige Einstellungen am der *myconfig.py* vorgenommen werden.

WEB CONTROL

Diese Anpassung ist für Sie nur wichtig, wenn Sie mehrere neuronale Netze gleichzeitig auf ein und denselben Rechner starten damit sich diese am Simulator anmelden. Jetzt haben Sie für das Starten der Weboberfläche für die Steuerung des Donkey Cars immer eine URL mit Port aufgerufen. Diese URL hatte den folgenden Aufbau:

URL: <IP Adresse des Host PC>:8887

Starten Sie jetzt aber mehrere Instanzen des Donkey Car Frameworks auf einem Rechner dann muss der Port unterschiedlich sein um die einzelnen Instanzen auseinander halten zu können beim Aufruf der URL.

Damit Sie unterschiedliche Ports pro Instanz vergeben können müssen Sie eine Kopie der *myconfig.py* Datei mit eigenen Namen z. B. *myconfig_port.py* anlegen.

Haben Sie die Kopie *myconfig_port.py* angelegt dann öffnen Sie diese bitte und suchen Sie nach der folgenden Zeile.

#WEB CONTROL

Eine Zeile tiefer passen Sie jetzt den Port an auf z. B. 8889. Die Zeile sollte dann wie folgt aussehen.

```
WEB_CONTROL_PORT = int(os.getenv("WEB_CONTROL_PORT", 8889))
```

Eine Zeile tiefer können Sie noch darauf Einfluss nehmen in welchem Modus die Weboberfläche starten soll bzw. das neuronale Netz starten soll.

Möchten Sie, dass das Donkey Car direkt nach dem Laden und Anmelden am Simulator losfährt dann ändern Sie den Modi von *user* auf *local* um.

Die Zeile sollte dann wie folgt aussehen.

```
WEB_INIT_MODE = "local"
```

DonkeyGym

Jetzt folgen noch zwei Anpassungen zum Aufruf des Simulators bzw. der Unity Engine. Suchen Sie nach der folgenden Zeile die die Unity Engine startet. Diese sieht für Windows z. B. wie folgt aus.

```
DONKEY_SIM_PATH = "D:\\DonkeySimWin\\donkey_sim.exe"
```

Passen Sie diese Zeile wie nachfolgend gezeigt an und ersetzen den Pfad zur Unity Engine durch **remote**. Im Ergebnis sollte die Zeile jetzt wie folgt aussehen.

```
DONKEY_SIM_PATH = "remote"
```

Mit diesen Anpassungen können Sie jetzt mehrere Autopiloten auf einem PC starten und im Unity Simulator fahren lassen.

Simulator von n-Rechner aus starten

Möchten Sie z. B. über das lokale Netzwerk oder Internet mit Ihrem Autopiloten an einem Rennen teilnehmen dann müssen Sie im Abschnitt DonkeyGym die remote Adresse des Rechners eintragen der den Unity Simulator hosted.

Dazu suchen Sie nach der nachfolgenden Zeile und tragen die IP Adresse oder den Servernamen des Zielrechners mit dem Simulator ein.

```
SIM_HOST = "<Adresse des Zielrechners>"
```

Zu Beginn von diesem Abschnitt, war ich kurz auf das Problem mit der Latenzzeit eingegangen wenn die teilnehmenden Rechner weiter entfernt sind vom Host der den Simulator bereitstellt. Um die Probleme der Laufzeit in der Verzögerung der Steuerung etwas auszugleichen kann man den folgenden Parameter setzen. Hier lohnt es sich etwas zu testen und zu spielen wie die Einstellung in Millisekunden am besten wirkt.

```
SIM_ARTIFICIAL_LATENCY = 0
```

Jetzt haben Sie beide Möglichkeiten kennen gelernt wie Sie über die **myconfig.py** ihre Donkey Car Instanz konfigurieren können um z. B. lokal auf einem Rechner mehrere Instanzen zu starten oder remote mit mehreren Donkey Cars in einem Simulator zusammen ein Rennen zu bestreiten.

3.5.2 Überholen trainieren - Trainingsdaten erfassen

Ein spannendes Thema ist die Erfassung von Trainingsdaten um einen Autopiloten das Überholen von anderen Donkey Cars beizubringen. Zunächst einmal brauchen Sie einen Autopiloten der stabil ohne Fehler viele Runden in einer gemütlichen Geschwindigkeit auf der Rennstrecke dreht. Diesen Autopiloten führen Sie jetzt mehrfach aus damit z. B. 2 oder 3 Autos auf dem Rennstrecke gleichzeitig fahren. Starten Sie die Autopiloten so, dass immer ein ausreichender Abstand zwischen den Donkey Cars verbleibt und diese nicht sofort aneinander stoßen.

Jetzt hoffen wir einfach, dass diese 2 bis 3 Donkey Cars keine Unfälle bauen und Sie mit einem weiteren, das die selbst steuern jetzt Trainingsdaten aufzeichnen können. Jetzt versuchen Sie einfach die langsam fahrenden Donkey Cars zu überholen oder hinter diesen her zu fahren ohne gegen diese zu stoßen. Nehmen Sie ca. 20 Überholmanöver auf und trainieren Sie anschließend das Neuronale Netz. Sie sollten wenn Sie jetzt dieses neuronale Netz auf der Rennstrecke testen sehen das es andere Donkey Cars überholen kann bzw. Ansätze sehen das es versucht die anderen Donkey Cars zu überholen.

Mit Sicherheit ist das Erstellen von solchen speziellen Trainingsdaten aufwändig und Datenintensiv. Vielleicht haben Sie ja einen besseren Ansatz als diesen den ich hier skizziert habe.

3.6 Was Sie bis hier erreicht haben

Sie haben die Donkey Car Simulator Umgebung erfolgreich auf Ihrem PC installiert und sind jetzt in der Lage an Ihrem PC Trainingsdaten zu erzeugen. Auch haben Sie Ihr neuronales Netz trainiert und dieses auf der Rennstrecke getestet. Das alles konnten Sie ganz bequem vom Schreibtisch aus ausprobieren. Sie mussten dazu nicht in den Garten oder auf einen großen Parkplatz gehen. Das ist der große Vorteil wenn man über einen Simulator zum Training von neuronalen Netzen verfügt. Sie können

jetzt verschiedenes anhand der aufgezeichneten Trainingsdaten ausprobieren um ihr neuronale Netz stabiler zu machen damit das Donkey Car sicher die Ziellinie erreicht. Die so gewonnenen Erkenntnisse werden sich Teilweise auf die reale Welt, also auf ihr echtes Donkey Car übertragen lassen mit dessen Bau sich die jetzt anschließenden Kapitel beschäftigen.

Ich persönlich finde es immer noch faszinierend wie anders sich ein neuronales Netz unter realen Umweltbedingungen verhält und welche Überraschung hier die künstliche Intelligenz in Form des neuronalen Netzes parat hält. Aber all das können sie selber erleben wenn Ihr echtes Modell Roboter-Auto Fahrt aufnimmt.

Ich freue mich sehr über Ihr Feedback zu diesem Kapitel. Schreiben Sie mir Ihre Gedanken, Fragen und Vorschläge an die folgende E-Mail Adresse: ebook@custom-build-robots.com

4 Baue Dein eigenes Donkey Car - Einführung in die benötigten Komponenten

Sie kennen sich schon auf der Software Seite sehr gut aus. Jetzt tauchen Sie in die Welt der Hardware eines Roboter-Autos ein. Die Wahl des richtigen Fahrwerkes und passenden Komponenten ist für den Erfolg dieses Projektes ausschlaggebend. Daher behandelt das jetzt folgende Kapitel ausführlich die Elektronik und Hardware des echten Donkey Cars.

Sie werden dem Roboter-Auto das wir zusammen aufbauen durch Behavioral Cloning das autonome Fahren beibringen. Das Konzept und das Vorgehen hinter dem Behavioral Cloning haben Sie bereits im Simulator kennen gelernt. Keine Sorge dabei wird der RC Modellbau bei diesem Projekt nicht im Vordergrund stehen. Vielmehr werden Sie sich auf die Software und das Training Ihres Neuronalen Netzes bzw. Autopiloten konzentrieren und hier ihr Wissen aufbauen. Sie werden zusammen mit dem umgebauten Modellauto wertvolle Erfahrungen rund um das Thema Artifical Intelligence (AI) sammeln die Ihnen auch später bei anderen Problemstellungen helfen werden die Komplexität und Herausforderungen abschätzen zu können.

Sie müssen für dieses Projekt nicht programmieren können sondern werden das ausgereifte Donkey Car Frameworks verwenden und dieses lediglich installieren und konfigurieren. Bei der Konfiguration werde ich Ihnen die zu ändernden Programmzeilen jeweils exakt vorgeben.

Das Donkey Car Framework wurde von Will Roscoe und Adam Conway Ende 2016 entworfen und ermöglicht den einfachen Einstieg in die Welt der autonom fahrenden Modellautos. Dank der großen Beliebtheit und der großen Community entwickelt sich dieses Framework rasant weiter und zählt heute zu einem der stabilsten sowie ausgereiften Open Source Lösungen. Das Donkey Car Projekt setzt auf klassische RC-Modellauto Technik die die mechanische Basis des Roboter-Autos bildet. Somit beschäftigen Sie sich nur zu Beginn mit dem Thema Modellbau und konzentrieren sich nach dem Zusammenbau auf das Thema AI und Trainingsdaten sammeln. So werden Sie praktisch lernen was es heißt die Daten sind das neue Öl die uns antreiben und weiter bringen werden.

Der Jetson Nano der die zentrale Recheneinheit des Roboter-Autos darstellt bzw. das neuronale Netz das auf diesem ausgeführt wird muss zuerst angelernt werden. Dazu werden Sie die benötigten Trainingsdaten selber durch manuelles fahren mit dem Modellauto erzeugen. In der Welt der künstlichen Intelligenz nennt man diese Art des Trainings von neuronalen Netzen Behavioral Cloning wie Sie eingangs bereits gelernt haben. Spannend wird es dann wenn Sie in Ihrem selber trainierten Modell ihre persönliche Fahrweise wieder erkennen und somit ein erstes Gefühl für den menschlichen Bias erhalten, den Sie ihrem Modell beiläufig durch die von Ihnen aufgezeichneten Trainingsdaten mitgegeben haben. Unter einem menschlichen Bias versteht man unbewusste Verhaltensmuster die Sie im Laufe Ihres Lebens angenommen haben und der immer wieder genannt wird wenn neuronale Netze z. B. nur von westlich geprägten sowie männlichen Wissenschaftlern aufgebaut und trainiert werden.

4.1 Auswahl der Hardware

Für den Aufbau des Roboter-Autos benötigen Sie verschiedene Hardware- sowie Elektronik-Komponenten die in diesem Kapitel im Detail erläutert werden. Ich habe bei der Wahl der Komponenten darauf geachtet, dass die Kosten für den Bau des Roboter-Autos so niedrig wie möglich ausfallen. Nicht alles muss unbedingt neu gekauft werden. So schlage ich z. B. vor für das Roboter-Auto ein gebrauchtes Tamiya Modellauto Chassis zu verwenden. Hier gibt es im Internet ein vielfältiges Angebot mit teilweise kaum genutzten Modellautos die schon mit Akku und Ladegerät angeboten werden. Ich werde in den einzelnen Abschnitten zu den Komponenten noch tiefer in die Details eintauchen warum welche Komponente genauso von mir gewählt wurde.

Damit Sie aber zunächst einen gesamthaften Überblick über die notwendigen Komponenten für den Bau des Roboter-Autos erhalten habe ich diese für Sie in der nachfolgenden Tabelle zusammengestellt.

Beschreibung:	Preis in €:
Jetson Nano 4GB Ram als zentrale Recheneinheit	109,-
Kamera Modul mit 160° Weitwinkel Linse	28,-
Micro SD-Karte 32 GB	7,-
SSD Festplatte 250 GB (optional)	75
Lüfter 5V für Jetson Nano mit PWM Steuerung	15,-
WIFI - Intel AC8265 Wireless NIC Modul	25,-
Power Bank mit 5V und ca. 5A	64,-
RC Akku mit 7,2V und 4.000 bis 5.000 mAh	30,-
RC Akku Ladegerät	60,-
Externes Netzteil für den stationären Betrieb	30,-
Tamiya Chassis für das Roboter-Auto (gebraucht)	140,-
Grundplatte aus Holz	4,-
3D Druck Überrollbügel	38,-
Messing Abstandshalter	11,-
Adafruit PCA9685 Servo Kontroller	19,-
Gamepad für die Steuerung des Roboter-Autos	26,-
OLED Display	4,-
Diverse Kabel	6,-
RC Akku Spannungsanzeige	8,-

Tabelle 4.1 Komponentenliste

Entsprechend dieser Übersicht mit all ihren Komponenten die Sie für den Bau des Roboter-Autos benötigen belaufen sich die Kosten auf ca. 700,- €. Dafür ist dieses Roboter-Auto dann aber auch im Freien nutzbar, sehr stabil für den Fall, dass das neuronale Netz sich überraschend verhält und die Ersatzteilversorgung im Falle eines Tamiya Chassis ist in Deutschland sehr gut.

Eine Liste der Komponenten mit Links auf entsprechende Web-Shops finden Sie auf meinem Blog unter der folgenden URL.

URL: <https://custom-build-robots.com/jetson-nano-komponenten-de>

Das nachfolgende Bild zeigt die wesentlichen Komponenten die für den Bau und Betrieb des Roboter-Autos benötigt werden. Das gezeigte Modellauto wurde gebraucht im Internet erstanden.



Abbildung 4.1: Komponenten für das Donkey Car

4.1.1 Zentrale Recheneinheit und Zubehör

Für alle Roboter-Autos ob in Echt oder wie hier als Modell ist eine leistungsstarke Rechen- und Sensor-Architektur sowie eine ausgereiftes AI-Modell ein entscheidender Erfolgsfaktor. Die erfassten Sensordaten müssen durch ein neuronales Netz bzw. durch eine Reihe von untereinander verbundenen

neuronalen Netzen in kürzester Zeit bewertet werden. Diese Architektur aus Hardware, Software und neuronalen Netzen trifft in ihrer Gesamtheit anschließend eine Vorhersage wie der Lenkeinschlag zu der aktuellen Geschwindigkeit gewählt werden muss um das Roboter-Auto z. B. erfolgreich durch eine Kurve steuern zu können. Ein weiteres neuronales Netz in diesem Verbund bewertet dann z. B. Vorfahrts-regeln, Verkehrsschilder sowie Ampeln und greift so wiederum auf die Geschwindigkeit ein um z. B. das Roboter-Auto zu stoppen bei einer roten Ampelschaltung. All diese Entscheidungen müssen in kürzester Zeit von der zentralen Recheneinheit getroffen und an die Steuerung des Roboter-Autos weiter gegeben werden.

NVIDIA Jetson Nano

In diesem Projekt wird der NVIDIA Jetson Nano mit 4GB Ram verwendet da er für seine Größe und Preis die beste GPU Rechenleistung bietet. Dank seiner Maxwell™ -Architektur und 128 NVIDIA CUDA® Recheneinheiten bietet er bei dem Ausführen und vor allem bei dem Training von neuronalen Netzen deutlich mehr Performance als der aktuelle Raspberry Pi 4 mit 4 GB RAM. Da das Training des Neuronalen Netzes sehr rechenintensiv ist würden Sie mit einem Raspberry Pi 4 mit 4 GB RAM viele Stunden bis Tage warten müssen bis das neuronale Netz fertig trainiert wäre. Mit dem Jetson Nano klappt dieses Training deutlich schneller und einen funktionierenden Autopiloten der mit ca. 6.000 Trainings-Datensätze trainiert wird ist in ca. 20 Minuten fertig berechnet.

Hinweis: Sollten Sie über einen modernen Rechner verfügen der vielleicht auch eine NVIDIA GPU verbaut hat wird das Training des neuronalen Netzes auf diesem Rechner deutlich schneller fertig abgeschlossen sein als auf dem Jetson Nano. Dennoch möchte man nicht immer seinen Laptop oder Desktop PC mitnehmen und so bietet sich das Training des neuronalen Netzes direkt im Roboter-Auto an.

Kamera Modul

Bei der Wahl der Kamera ist wichtig, dass diese mit dem Jetson Nano kompatibel ist. NVIDIA unterstützt offiziell verschiedene Kamera Modelle die aber alle auf den Sony IMX219 Chip aufsetzen. Für diesen Chip liefert NVIDIA im Image für den Jetson Nano die notwendigen Treiber gleich mit. Eine Liste von kompatiblen Kameras finden Sie auf meinem Blog unter der zu Beginn dieses Kapitels genannten URL.

Ich empfehle immer einer Kamera mit einer Weitwinkellinse mit 150° bis 160° im Roboter-Auto zu verbauen. So ist das Blickfeld nicht zu klein und erfasst sehr gut die Streckenmarkierung links und rechts auch bei einer Fahrt durch eine Kurve. Das Blickfeld der Kamera sollte nicht zu groß sein (> 160°) umso wenige externe Einflussfaktoren abseits der Strecke wie sich bewegenden Menschen oder Möbel zu erfassen.

Speicher – micro SD-Karte

Für den Jetson Nano benötigen Sie eine micro SD-Karte auf der Sie das Betriebssystem und später auch die Software installieren. Daher sollte diese Karte sehr schnell sein beim Lesen sowie Schreiben von Daten. Damit die SD-Karte nicht gleich vollläuft sollten Sie ein 32 GB großes Modell wählen. Aber lesen Sie bitte noch die nachfolgende Empfehlung zum Thema SSD Festplatte als Speicher anstelle der micro SD-Karte des Roboter-Autos durch.

Speicher – SSD Festplatte

Auf den Speicher des Roboter-Autos wird sehr häufig lesend und schreiben zugegriffen. Da auch das neuronale Netz direkt auf dem Jetson Nano berechnet werden soll. Meine Erfahrung hat gezeigt, dass die klassischen micro SD-Karten sehr schnell Defekte zeigen bei einer intensiven Nutzung wie es das Training eines Neuronalen Netzes darstellt. Daher ist meine ganz klare Empfehlung eine SSD Festplatte für das Ausführen des Betriebssystems und auch für das Training des Neuronalen Netzes zu verwenden. Eine SSD Festplatte die am USB 3.0 Anschluss des Jetson Nano angeschlossen wird und 250 GB bis 500 GB groß ist reicht völlig aus.

Hinweis: Möchten Sie erste einmal mit der micro SD-Karte starten dann können Sie später auch noch wenn Sie schon Trainingsdaten aufgezeichnet haben und Ihr Roboter-Auto bereits autonom fährt auf eine SSD Festplatte wechseln und die Daten auf die SSD umziehen.

WLAN für den Jetson Nano

Der Jetson Nano selber verfügt über kein eingebautes WIFI Modul. Daher müssen Sie den Jetson Nano erst um ein WIFI Modul erweitern. Damit Sie die beste Treiberunterstützung, eine große Signal Reichweite und einen sehr guten W-LAN Empfang erhalten sollten sie wie von NVIDIA empfohlen das Intel AC8265 Wireless NIC Modul verbauen. Die Erfahrung auf einer Vielzahl von Veranstaltungen hat gezeigt, dass es mit nicht wesentlich günstigeren USB Modulen immer wieder Probleme mit Verbindungsabbrüchen gibt.

Lüfter für den Jetson Nano

Der Jetson Nano wird beim Training eines Neuronalen Netzes schnell sehr heiß. Daher empfehle ich einen kleinen 5V Lüfter auf dem Kühlkörper zu befestigen. Damit der Lüfter z. B. beim stationären Betrieb auf dem Schreibtisch nicht zu laut ist reicht es auch völlig aus diesen mit 3,3V zu betreiben. Wenn Sie möchten können Sie sich auch einen PWM gesteuerten Lüfter zulegen dessen Drehzahl abhängig von der Temperatur des Jetson Nano geregelt werden kann.

Energieversorgung - Einführung

Die passende Energieversorgung des Roboter-Autos zusammen mit dem Jetson Nano ist noch ein entscheidender Punkt für den Erfolg und Spaß an dem Projekt. Ich habe bereits ca. 8 dieser Roboter-Autos gebaut und mit diesen sehr viele Erfahrungen bei internationalen Trainings und Vorträgen auf der Welt sammeln dürfen. Es gibt zwei Möglichkeiten die Energieversorgung im Roboter-Auto umzusetzen. Die erste ist ein RC Akku der den Jetson Nano und den Antriebsmotor mit Strom versorgt. Die zweite Möglichkeit ist eine getrennte Stromversorgung des Jetson Nano mit einer Power Bank und die Stromversorgung des Antriebsmotors mit einem RC-Akku.

Hinweis: Wenn Sie nur einen RC Akku verwenden um den Jetson Nano und den Antriebsmotor mit Energie zu versorgen so kann es passieren, dass der Jetson Nano abstürzt wenn der Antriebsmotor bei einer Beschleunigung zu viel Energie verbraucht. Daher empfehle ich eine getrennte Energieversorgung des Jetson Nano und des Antriebsmotors.

Energieversorgung – Jetson Nano

Ich empfehle Ihnen den Jetson Nano an einer Power Bank anzuschließen wenn das Roboter-Auto fährt und nicht Stationär z. B. auf dem Schreibtisch steht. Wenn Sie schon eine Power Bank besitzen die 5V und mindestens 2,0A bis 2,5A an einem ihrer Ausgänge liefert dann probieren Sie diese zunächst aus ob der Jetson Nano mit dieser stabil läuft bevor Sie sich eine neu kaufen. Sollten Sie feststellen, dass der Jetson Nano unter Last nicht stabil läuft dann empfehle ich Ihnen eine Power Bank mit ca. 28.000 mAh, der typischen Spannung von 5V aber einem extra starken Strom von ca. 5A. Mit solch einer Power Bank habe ich bis jetzt immer sehr gute Erfahrungen gemacht in einer Vielzahl von Roboter-Autos auf Basis des Jetson Nano. Auf meinem Blog finden Sie ein entsprechendes Modell in der Komponentenliste.

Energieversorgung – Antrieb des Roboter-Autos

Für den Antriebsmotor des Roboter-Autos verwenden Sie am besten einen 4.000 mAh bis 5.000 mAh starken RC Akku. Hier können Sie einen LiPo RC Akku verwenden, wenn Sie mit dieser Technik vertraut sind oder einen klassischen NiMh Modell der eine Unterspannung des Akkus etwas besser verträgt. Für meine Schulungen mit bis zu sechs dieser Donkey Cars habe ich immer NiMh Akkus verwendet um die 14 Akkus auch z. B. im Flugzeug transportieren zu dürfen was mit LiPo Akkus schwierig ist.

Energieversorgung – Stationärer Betrieb mit externen 220V Netzteil

Damit der Jetson Nano im stationären Betrieb ohne Zeiteinschränkung betrieben werden kann ist ein externes Netzteil sehr zu empfehlen. Dieses sollte wieder die bekannten 5V bei ca. 5A liefern können. Wenn Sie später das Neuronale Netz auf dem Jetson Nano direkt trainieren steigt die Leistungsaufnahme des Jetson Nano auf ca. 20W an. Daher sollte das Netzteil auch statt der 4A am besten 5A liefern um etwas Reserven übrig zu haben. Auch wird die höhere Leistung des Jetson Nanos nur abgerufen wenn dieser über die externe Stromversorgung mit Energie versorgt wird.

4.1.2 Auswahl des Chassis

Da es ja nicht um ein Modellbau Projekt gehen soll empfehle ich den Kauf eines Ready-to-Run (RtR) 1:10 Modellautos. Diese RtR Modellautos sind fix und fertig aufgebaut und Sie müssen sich nicht mit Themen des Modellbaus herumschlagen. Genau so macht es auch Waymo die nicht selber ein Auto entwickeln und bauen sondern bestehende Plattformen wie den Chrysler Pacifica für ihre Zwecke mit Sensoren und Recheneinheiten ausstatten und diese so zu autonom fahrenden Roboter-Autos umbauen.

Ich empfehle immer Modelle von Tamiya mit einem CC-01 Fahrwerk für den Bau eines Roboter-Autos. Dieses gibt es mit verschiedenen Aufbauten aber das Fahrwerk ist immer das gleiche wenn es sich um ein CC-01 Fahrwerk handelt. Gut geeignet aber nicht ganz so stabil da deutlich sportlicher ist auch das Tamiya TT-02 Chassis. Die Tamiya Chassis sind auf diversen Plattformen im Internet gebraucht günstig zu erwerben und das CC-01 Chassis hält dank dem stabilen Aufbau sehr viele Unfälle ohne sofortige Beschädigungen aus. Es wird Ihnen ganz sicher passieren, dass Ihr neuronales Netz abhängig davon was es gelernt hat mit Top-Speed plötzlich Vorwärts oder Rückwärts fährt und es zu einem Unfall mit Ihrem Donkey Car kommt. Wichtig ist mir auch bei diesem Projekt die leichte Verfügbarkeit von Ersatzteilen die bei Tamiya Modellen generell gegeben ist.

Ein gebrauchtes Chassis – Was ist wichtig

Achten Sie beim Kauf eines gebrauchten Chassis unbedingt darauf, dass dieses auch Fahrbereit ist. Es sollte ein Lenk-Servo und Fahrtenregler verbaut sein, Räder im Lieferumfang enthalten sein sowie keine großen Beschädigungen aufweisen. Ich hatte selber die negative Erfahrung gemacht, dass ich ein gebrauchtes CC-01 Chassis erworben habe dessen Federung hinten rechts entgegen der Produktbeschreibung des Verkäufers defekt war. Aber mit ein paar Ersatzteilen konnte ich das recht einfach und günstig reparieren. Bei dem Fahrtenregler fragen Sie bitte bei dem Verkäufer nach der Typenbezeichnung damit Sie auch die Anleitung zu diesem im Internet finden können. Die Anleitung ist wichtig damit Sie vor dem Kauf noch herausfinden können ob dieser Fahrtenregler über eine BEC (Battery Eliminator Circuit) Funktion verfügt und wie der Vorwärtsgang, die Motorbremse und der Rückwärtsgang konfiguriert werden kann. Die BEC Funktion ist nicht ganz so wichtig spart Ihnen aber später etwas Aufwand bei der Verkabelung des Servo-Kontrollers und der Stromversorgung des Lenk-Servos. Denn genau die Stromversorgung des Lenk-Servos übernimmt die BEC Funktion des Fahrtenreglers.

Hinweis: Auf der offiziellen Donkey Car Seite werden zu den verschiedenen 3D Druck-Dateien auch kompatible Modelle aufgezählt die aber in Nordamerika vertrieben werden und in Europa meiner Erfahrung nach nur individuell importiert werden können.

Grundplatte

Damit Sie alle Komponenten wie den Überrollbügel, Jetson Nano, Servo-Kontroller etc. des Roboter-Autos auf dem CC-01 Chassis befestigen können benötigen Sie die Grundplatte. Diese Grundplatte wird statt der Karosserie auf Ihrem Chassis befestigt. Ich Empfehle Ihnen eine Multiplex Platte mit einer Stärke von maximal 6mm zu kaufen. Diese Platte sollte in etwa die folgenden Abmessungen haben:

Maße: 360 x 150 x 6 mm

Im Baumarkt können Sie sich die Grundplatte passgenau für wenige Euro zuschneiden lassen.

Messing-Abstandshaltern

Für die Befestigung des Jetson Nano, des OLED Displays und des Servo-Kontrollers empfehle ich Ihnen den Einsatz von Messing-Abstandshaltern. Diese sind Ideal um in der Grundplatte verschraubt zu werden. Anschließend befestigen Sie die Komponenten dann auf diesen Abstandshaltern.

3D Druck - Überrollbügel

Damit der Jetson Nano in Ihrem Roboter-Auto gut geschützt ist, empfehle ich einen Überrollbügel samt Kamerahalterung mit einem 3D Drucker auszudrucken. Die dazu benötigten STL-Dateien stellt das Donkey Car Projekt online zur Verfügung. Zu diesem Überrollbügel gibt es noch weitere STL-Dateien wie eine Grundplatte und einer Adapter-Platte für den Jetson Nano. Diese Platten werden meiner Erfahrung nach nicht benötigt da diese nicht zum CC-01 Chassis von Tamiya kompatibel sind. Daher empfehle ich Ihnen auf diese beiden Platten zu verzichten und sich das Geld zu sparen. Die benötigte STL Datei für den Überrollbügel finden Sie unter der folgenden URL.

URL: <https://www.thingiverse.com/thing:2566276>

4.1.3 Servo Kontroller

Damit der Jetson Nano bzw. die Software die Befehle an die Lenkung und den Antriebsmotor weiter geben kann wird ein Servo-Kontroller benötigt. An diesen schließen Sie direkt den Lenk-Servo und den elektronischen Fahrtenregler (ESC) an. So können Sie später mit einem Gamepad oder auch das neuronale Netz die Kontrolle über das Roboter-Auto übernehmen.

4.1.4 Gamepad

Sie werden die Trainingsdaten für das Training des Neuronalen Netzes selber aufzeichnen. Damit Sie gute Trainingsdaten erhalten und diese nicht zackig und ruckelig in der Lenkbewegung und Beschleunigung sind empfehle ich die Verwendung eines Gamepads. Ein günstiges Modell wie der „EasySMX Controller“ hat sich als völlig ausreichend erwiesen. Sollten aber z. B. von einer Playstation oder X-Box entsprechende Controller vorhanden sein dann können Sie diese zunächst einmal testen ob Sie diese verwenden können. Das Funksignal verschiedener Gamepads die ich testen konnte ließ sich auch nicht durch die offene Elektronik des Roboter-Autos oder durch die unzähligen Smartphones von Trainingsteilnehmern stören. Hier hatte ich teilweise negative Erfahrungen mit Funktastaturen gemacht die z. B. bei angeschlossenem CSI Kabel der Kamera nicht mehr einwandfrei funktioniert haben.

4.1.5 OLED Display

Sehr zu empfehlen ist auch ein kleines OLED Display auf dem Sie die Belegung des Arbeitsspeichers und die CPU Auslastung sich anzeigen lassen können. Aber die meiner Meinung nach wichtigste Funktion des OLED-Displays ist die Anzeige der aktuellen IP-Adresse des Jetson Nano. So ist die IP Adresse ihres Jetson Nano ganz schnell in fremden Netzwerken ersichtlich und ob sich der Jetson Nano mit dem WIFI Verbunden hat oder eben nicht. So spart Ihnen ein OLED Display mit entsprechender Ausgabe sehr viel Frust und die Suche nach einem Monitor, Maus und Tastatur.

4.1.6 Kabel und Adapter

Damit Sie die verschiedenen Komponenten im Roboter-Auto mit ihren unterschiedlichen Schnittstellen miteinander verbinden können benötigen Sie verschiedene Kabel. Der nachfolgende Abschnitt beschreibt welche Kabel Sie genau für was brauchen.

USB-A Verlängerung

Eine USB-A Verlängerungskabel von ca. 15cm oder einen bereits vorhandenen kleinen USB Hub empfehle ich aus Platzgründen zu verwenden. So können Sie den Funkempfänger des Gamepads bequem am Jetson Nano anschließen. Denn je nach Aufbau des Roboter-Autos kann es passieren, dass die USB Anschlüsse des Jetson Nano für lange und große Adapter nicht mehr ohne weiteres zugänglich sind.

Abgeschnittenes USB-A Kabel

Ein extra USB A-Kabel benötigen Sie falls Ihr Fahrtenregler nicht über die Battery Elimination Circuit (BEC) Funktion verfügt. In diesem Fall müssen Sie den Servo-Motor für die Lenkung des Roboter-Autos separat über den Servo-Kontroller mit Strom versorgen. Hier reicht ein altes Kabel z. B. von einer nicht mehr benötigten Computermaus welches Sie abschneiden können.

Female-to-Female Jumper-Kabel

Sie werden den Servo-Kontroller an vier Pins der GPIO Pinleiste des Jetson Nano anschließen. Dazu benötigen Sie vier dieser Female-to-Female Jumper-Kabel. Vier weitere benötigen Sie noch wenn Sie ein OLED Display einbauen und das ebenfalls anschließen möchten.

Buchsenleiste Adapter

Damit es keine Probleme mit Wackelkontakte bei den Steckverbindungen der Female-to-Female Kabel zwischen dem Jetson Nano und Servo-Kontroller bekommen empfehle ich für diese Verbindung die 8 Kontakte fest zu verlöten. Damit Sie die vier Female-to-Female Jumper-Kabel nicht direkt am Jetson Nano und Servo-Kontroller anlöten verwenden Sie bitte zwei 2x6 Buchsenleisten die Sie auf die GPIO Pins des Jetson Nano und auf die sechs Pins des Servo-Kontrollers aufstecken können. An die beiden Buchsenleisten löten Sie die vier Female-to-Female Jumper Kabel direkt an.

4.1.7 RC Akku - Spannungsanzeige

Als letzte Komponente dieser Aufzählung möchte ich noch eine Spannungsanzeige für den RC Akku empfehlen. Mit dieser kleinen Anzeige haben Sie die aktuelle Spannung die am Motor anliegt im Blick. Mit dieser Anzeige fällt es Ihnen leichter eine Tiefentladung des RC-Akkus zu vermeiden.

4.1.8 Komponentenliste - online

Eine immer wieder aktualisierte Liste der empfohlenen Komponenten finden Sie auf meinem Blog unter der folgenden URL.

URL: <https://custom-build-robots.com/jetson-nano-komponenten-de>

Ich freue mich sehr über Ihr Feedback zu diesem Kapitel. Schreiben Sie mir Ihre Gedanken, Fragen und Vorschläge an die folgende E-Mail Adresse: ebook@custom-build-robots.com

5 Aufbau des Roboter-Autos und einrichten des Jetson Nano

Jetzt geht erst einmal los mit dem Basteln und kleineren Lötaufgaben, denn alle Komponenten vom Jetson Nano bis zum Überrollbügel müssen zu einem funktionierenden Roboter-Auto zusammengefügt werden.

Nehmen Sie, falls bei Ihrem RC Auto vorhanden die Karosserie ab und machen Sie sich mit der bereits im Chassis verbauten Elektronik vertraut. Falls der Funkempfänger nicht stört lassen Sie diesen einfach eingebaut. Trennen Sie lediglich die Stromversorgung und ziehen Sie das Kabel des Lenk-Servos und das Kabel des Fahrtenreglers am Empfänger ab. Das Ziel ist es, das Roboter-Auto so aufzubauen, dass Sie mit wenigen Handgriffen zwischen Roboter-Auto und funkfernsteuertem Auto hin und her wechseln können wenn Sie das möchten.

In der jetzt folgenden Beschreibung wird davon ausgegangen, dass Sie das CC-01 Chassis von Tamiya verwenden. Allerdings ist die Anleitung so allgemein beschrieben, dass diese auch für viele andere RC Auto Chassis funktionieren wird da der typische Aufbau von RC Autos sehr ähnlich ist. Eventuell müssen Sie die Grundplatte auf der die Elektronik befestigt wird etwas modifizieren damit diese zu Ihrem Modell und Befestigungspunkten passt.

5.1 Benötigtes Werkzeug

Diese kleine Übersicht soll Ihnen helfen das richtige Werkzeug zur Hand zu haben wenn Sie jetzt loslegen das Roboter-Auto aufzubauen.

- 2mm, 3mm, 5,5mm, 6,5mm und 12mm Holz-Bohrer
- 16 mm Loch-Bohrer
- Bohrmaschine oder idealerweise eine Säulenbohrmaschine
- Lötstation und eine 3. Hand
- Geodreieck und Bleistift
- Schraubenzieher Set

Neben dem Werkzeug brauchen Sie noch ein paar Holzschrauben um z. B. den Überrollbügel auf der Grundplatte festschrauben zu können

- Verschiedene kleine Schrauben wie z. B. M3 mit einer Länge von ca. 25mm

5.2 Vorbereiten der Grundplatte

Das CC-01 und TT-02 Chassis haben hinten und vorne die für Modellautos typischen zwei Karosseriehalterung bzw. Stifte die nach oben stehen. Messen Sie die Position der vier Stifte exakt aus und übertragen Sie die Position auf die Grundplatte die Sie bereits z. B. im Baumarkt haben zuschneiden lassen. Dabei sollten die Löcher so positioniert werden, dass die Grundplatte hinten mit dem Chassis abschließt und von oben betrachtet zentriert auf dem Chassis sitzt.

Die vier Löcher mit einem Durchmesser von 5mm für die Karosseriehalterung können Sie wie in der folgenden Bohrschablone angegeben in die Grundplatte bohren. Prüfen Sie aber vor dem Bohren bitte ob die Maße tatsächlich zu Ihrem Chassis passen.

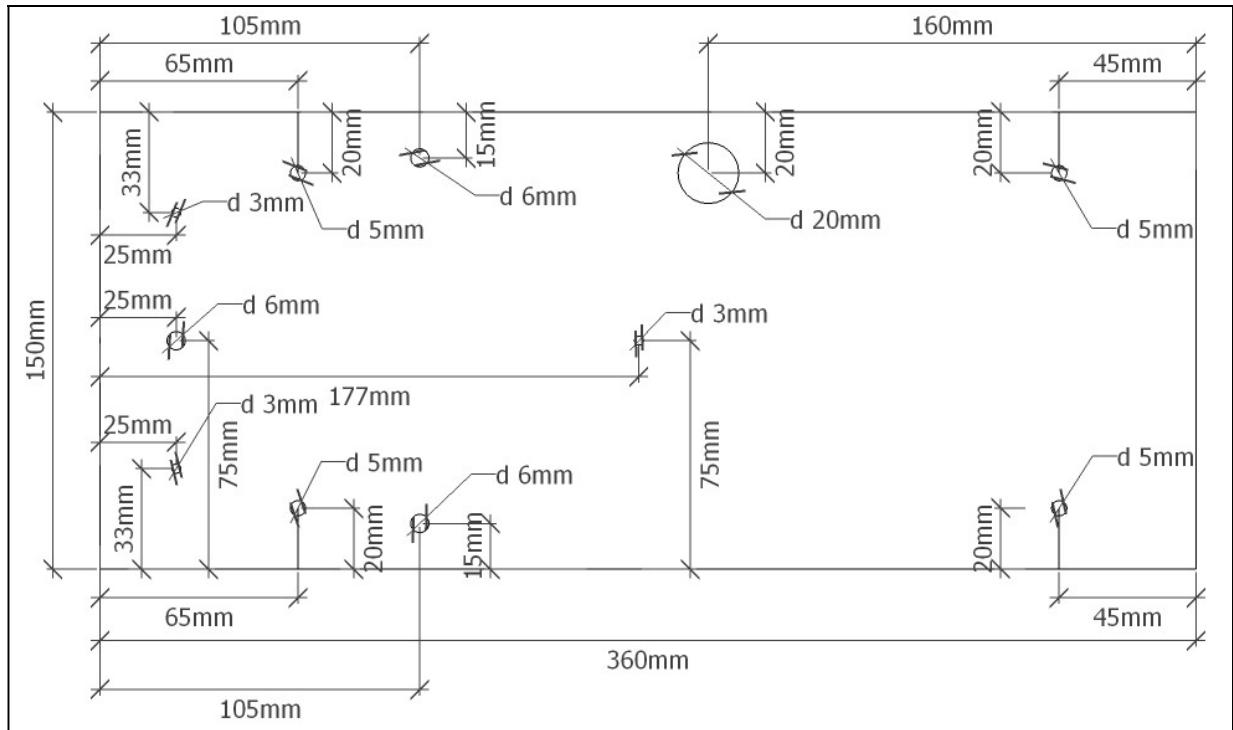


Abbildung 5.1: Grundplatte

Sie haben die Grundplatte jetzt auf Ihrem Chassis aufgesteckt und diese sitzt fest ohne die vier Karosseriehalterungen zu verbiegen.



Abbildung 5.2: Grundplatte auf dem Chassis montiert

Die weiteren Löcher die bereits in der Bohrschablone eingezeichnet sind werden Sie später in die Grundplatte bohren. Bitte lesen Sie daher vor dem Bohren der weiteren Löcher noch den Abschnitt Komponenten befestigen genau durch.

5.3 Komponenten befestigen

Nach dem die Grundplatte mittig und fest auf dem Chassis sitzt werden die weiteren Komponenten auf der Grundplatte befestigt. Der Überrollkäfig gibt hier die Position aller weiteren Komponenten vor. Daher wird jetzt der Überrollkäfig als erstes befestigt.

5.3.1 Überrollkäfig befestigen

Auf der Grundplatte wird jetzt die passende Position für den Überrollkäfig gesucht. Achten Sie dabei darauf, dass die Weitwinkelkamera ca. 3 cm bis 4 cm hinter dem Stoßfänger sitzt damit dieser bzw.

Teile des Chassis nicht im Bild zu sehen sind. Sie können die Kamera jetzt schon einmal für die Positionsbestimmung des Überrollkäfigs in die für die Kamera vorgesehene Halterung vorne am Überrollkäfig einschieben. Von vorne betrachtet sitzt die Kamera bzw. der Überrollkäfig wieder mittig auf der Grundplatte. Messen Sie jetzt die Position der drei Löcher auf der Unterseite des Überrollkäfigs aus und übertragen Sie diese auf die Oberseite der Grundplatte. Bohren Sie jetzt mit einem ca. 3mm Bohrer die drei Löcher in die Grundplatte um von unten den Überrollkäfig festzuschrauben.

Schrauben Sie jetzt den Überrollkäfig von unten mit drei Schrauben auf der Grundplatte fest und prüfen Sie ob dieser fest sitzt. Die Grundplatte mit Überrollkäfig sollte wie im folgenden Bild gezeigt in etwas aussehen.



Abbildung 5.3: Grundplatte mit Überrollkäfig

5.3.2 Die Position für Elektronische Komponenten festlegen

Packen Sie falls noch nicht geschehen den Jetson Nano, den Funkempfänger des Gamepads, den Servo-Kontroller sowie das OLED Display aus.

Ich empfehle den Jetson Nano unter dem Überrollkäfig zu befestigen. Die Anschlüsse wie USB, HDMI, LAN etc. zeigen nach hinten. So ist der Jetson Nano bei Transporten des Roboter-Autos zu z. B. Meetups und bei Unfällen geschützt.

Hintergrund: Bei einem Workshop ist mein Roboter-Auto in voller Fahrt unter einen Heizkörper gefahren und hat, da nicht einmal ein minimaler Schutz vorhanden war, die GPIO-Pinleiste des Jetson Nano abrasiert.

Legen Sie den Jetson Nano an entsprechender Position unter dem Überrollkäfig auf die Grundplatte. Achten Sie darauf, dass die Anschlüsse wie HDMI, Netzwerk und USB frei zugänglich sind. Stecken Sie jetzt das USB A Verlängerungskabel oder den mini USB-Hub ein. Schließen Sie am Verlängerungskabel oder mini-USB Hub den Funkempfänger des Gamepads an. Verwenden Sie ein USB-WIFI Modul anstelle des Intels NIC Moduls um Ihren Jetson Nano W-LAN fähig zu machen dann stecken Sie dieses jetzt auch in einen freien USB Port ein.

Haben Sie die passende Position für den Jetson Nano mit der angeschlossenen Elektronik gefunden, dann markieren Sie jetzt die vier Löcher auf der Grundplatte mit denen die Platine des Jetson Nano festgeschraubt wird.

Jetzt müssen Sie noch ein etwas größeres Loch in die Grundplatte bohren durch welches Sie das Kabel des Servo-Motors und das Kabel des Fahrtenreglers hindurch stecken können. Am besten sitzt das Loch etwas rechts am Rand der Grundplatte, so dass Sie noch die Option haben den Überrollkäfig nach hinten oder vorne versetzen zu können. Für das Loche nehmen Sie falls vorhanden einen 16mm Loch-Bohrer oder bohren zwei Löcher mit etwas Überlappung nebeneinander mit z. B. einem 12mm Holzbohrer.

Stecken Sie die Grundplatte wieder auf das Chassis auf. Verbinden Sie jetzt den Servo-Kontroller mit dem Lenk-Servo und Fahrtenregler. Holen Sie dazu die beiden Kabel des Lenk-Servo und das des Fahrtenreglers nach oben durch das dafür vorgesehene 16mm Loch. Den Fahrtenregler schließen Sie

am Kanal 0 und den Lenk-Servo am Kanal 1 an. Achten Sie bei der Suche nach einer geeigneten Position für den Servo-Kontroller darauf, dass die Kabel nicht unter Spannung stehen. Markieren Sie jetzt die Position der vier Löcher mit denen der Servo-Kontroller befestigt wird ebenfalls auf der Grundplatte.

Jetzt suchen Sie sich z. B. im Heck des Roboter-Autos noch einen Platz für die Power Bank. Diese können Sie z. B. mit zwei Kabelbinder und vier Löchern in der Grundplatte recht einfach auf dieser befestigt. Markieren Sie wieder die vier Löcher wenn Sie die Lösung mit den Kabelbindern nachbauen möchten.

Das folgende Bild zeigt einen Zwischenstand wie in etwas der Aufbau jetzt aussehen sollte. Der Jetson Nano, die Power Bank und der Servo-Kontroller wurden bei dieser Aufnahme bereits befestigt. Gut zu sehen sind auch die Kabel vom Fahrtenregler und Lenk-Servo die durch das 16mm Loch nach oben geführt wurden.

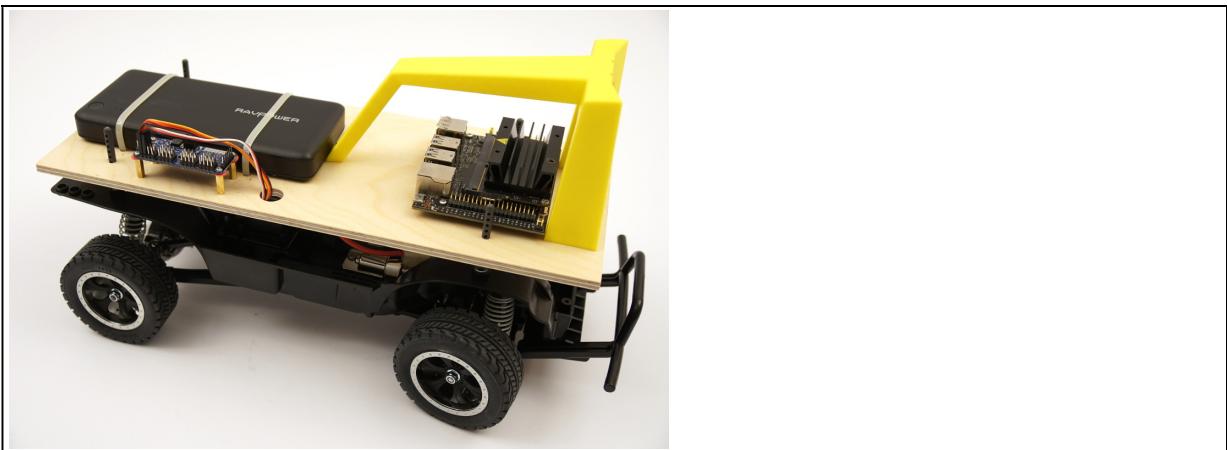


Abbildung 5.4: Roboter Auto aktuelle Montage

Die Position des OLED Displays sollten Sie so wählen, dass Sie dieses noch gut ablesen können und dass es nicht zu weit weg ist vom Servo-Kontroller da es an dessen I²C Bus Ausgang angeschlossen wird. Um keinen Platz zu verlieren befestigen Sie das OLED-Display eventuell direkt rechts neben dem Servo-Kontroller. Dann markieren Sie wieder die vier Haltelöcher des Displays auf der Grundplatte.

Wenn Sie eine SSD Festplatte, wie empfohlen verwenden dann können Sie diese direkt zusammen mit der Power Bank unter einem der beiden Kabelbinder gleich mit befestigen. Die Befestigung mit den Kabelbindern ermöglicht es Ihnen die SSD Festplatte wie auch die Power Bank noch heraus ziehen zu können und auch wieder unter die Kabelbinder zustecken.

Hinweis: Meine praktische Erfahrung bei der Befestigungslösung der Power Bank mit Kabelbindern hat gezeigt, dass Sie die Power Bank im Heck noch mit einer Schraube hinter der Power Bank vor versehentlichem herausrutschen sichern sollten.

5.3.3 Löcher bohren

Eventuell müssen Sie den Überrollbügel wieder abschrauben wenn Sie jetzt die ganzen Löcher bohren die Sie markiert haben. Bohren Sie die Löcher für die drei Platinen Jetson Nano, Servo-Kontroller und OLED-Display mit einem 2mm Bohrer in die Grundplatte. In diese 2mm Löcher sollten Sie jetzt die Messing-Abstandshalter gut und stabil eindrehen können. Die vier Löcher für die Befestigung der Power Bank mit den Kabelbindern bohren Sie mit einem passenden Bohrer der der Breite der Kabelbinder entspricht.

5.3.4 Intel AC8265 Wireless Antennen befestigen

Haben Sie sich für die Verwendung des Intel AC8265 Wireless NIC Moduls entschieden dann montieren Sie dieses jetzt. Dazu lösen Sie die beiden Schrauben mit denen das Jetson Nano Modul auf der Trägerplatine befestigt ist. Klicken Sie das Modul heraus wie früher die SO-Dimm-Speicherriegel. Schließen Sie die beiden Kabel für die Antennen am NIC Modul an, entfernen Sie die

Befestigungsschraube und stecken Sie das NIC Modul in den dafür vorgesehenen Slot unterhalb des Jetson Nano. Befestigen Sie das Modul wieder mit der Schraube, stecken Sie jetzt den Jetson Nano wieder in die Halterung und schrauben Sie diesen mit den beiden Schrauben fest.

Bohren Sie noch zwei Löcher mit einem ca. 6,5mm Bohrer in die Grundplatte um die beiden WIFI-Antennen befestigen zu können. Achten sie unbedingt bei der Wahl der Position der Löcher für die beiden Antennen und ihren SMA Buchsen darauf, dass die beiden Kabel vom NIC Modul zu den Löchern der SMA Buchsen lang genug sind. Ich habe vor dem Jetson Nano mittig in die Grundplatte ein 6mm Loch gebohrt durch welches ich die dünnen Antennenkabel zu den SMA Buchsen geführt habe.

Hinweis: Damit Sie die SMA Buchsen an der 6mm dicken Grundplatte festschrauben können müssen Sie eventuell die beiden Löcher von unten bis zur Mitte der Grundplatte mit einem 12mm Bohrer etwas aufbohren. So schaut das Gewinde etwas weiter durch die Grundplatte hindurch und Sie können die Antennen noch auf der SMA Buchse selber festschrauben.

Im folgenden Bild sehen Sie das Loch vor dem Jetson Nano durch welches die Antennenkabel die vom NIC Modul kommen zu den SMA Buchsen durchgeführt werden. Rechts hinten im Bild sehen Sie eine der beiden Antennen die bereits befestigt ist.

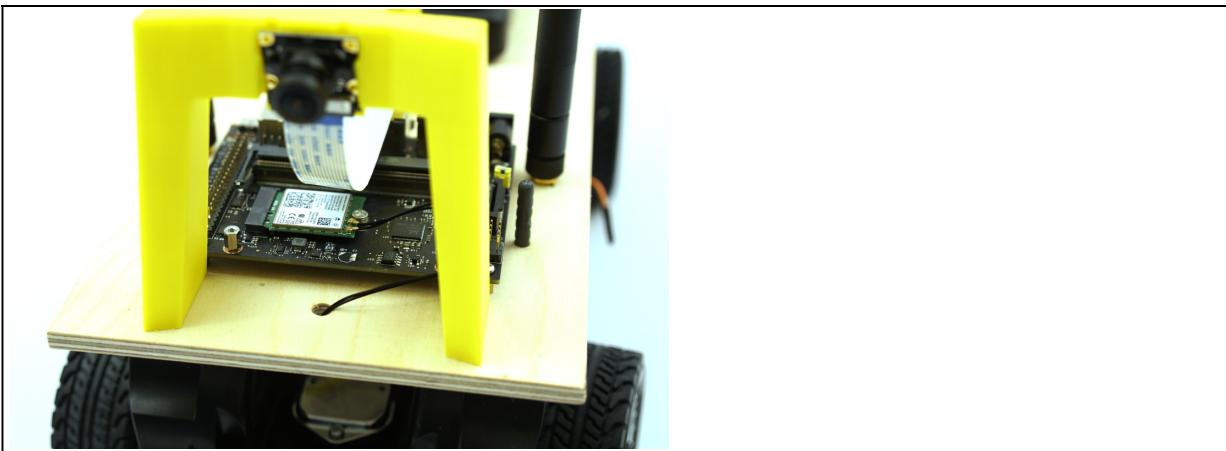


Abbildung 5.5: Grundplatte mit Antennenkabeldurchführung

Jetzt befestigen Sie wieder den Überrollbügel und alle elektronischen Komponenten wie den Jetson Nano, den Servo-Kontroller etc. auf der Grundplatte.

5.4 Verkabelung der Komponenten im Roboter-Auto

In diesem Abschnitt werden Sie die Verkabelung der einzelnen Komponenten im Roboter-Auto vornehmen. Zuerst erkläre ich Ihnen aber was der I²C Bus genau ist an dem der Servo-Kontroller und das OLED Display angeschlossen werden.

5.4.1 I²C Bus eine kurze Einführung

Der Inter-Integrated Circuit Bus in weiteren Verlauf als I²C Bus abgekürzt ist ein serieller Bus der von Philips Semiconductors 1982 entwickelt wurde für die Kommunikation zwischen einem Kontroller und angeschlossenen Komponenten. Der Jetson Nano verfügt über einen I²C Bus der über die beiden Pins 3 (SDA) und Pin 5 (SCL) der GPIO Pinleiste herausgeführt ist. Am I²C Bus werden Sie den Servo Kontroller und das OLED Display in Reihe anschließen.

5.4.2 Verkabelung I²C Bus

Damit die I²C Verbindung zwischen Jetson Nano und Servo-Kontroller auch bei Erschütterungen des Roboter-Autos Wackelkontakt frei ist empfehle ich Ihnen die vier Female-to-Female Jumper Kabel nicht nur zu stecken sondern diese fest an einer 2x6 Buchsenleiste zu verlöten. Mit der Verwendung einer 2x6 Buchsenleisten müssen Sie nicht am Jetson Nano selber die vier Kabel festlöten sondern löten diese an

der Buchsenleiste fest. Diese stecken Sie dann auf die ersten sechs Pins der GPIO Pinleiste des Jetson Nano und erhalten so eine Wackelkontakt freie Verbindung zum Servo-Kontroller.

Im nachfolgenden Bild ist noch einmal die GPIO Pinleiste mit dem I²C Bus Pins hervorgehoben. Gut zu erkennen ist auch der erste Pin mit den 3,3V und der GND Pin gegenüber dem SCL Pin.

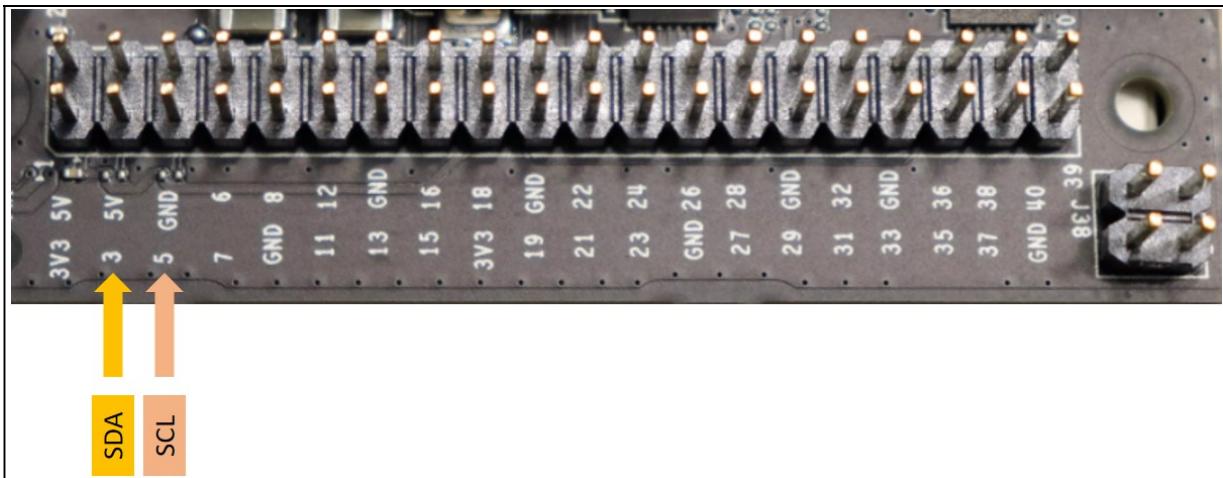


Abbildung 5.6: GPIO Pinout des Jetson Nano (I²C)

Machen Sie sich mit der Pin-Belegung der Buchsenleiste vertraut welchen Pin des Jetson Nano welcher Pin an der Buchsenleiste jetzt verlängert.

Entfernen Sie bei vier der Jumpe Kabel jeweils die Plastikhülsen, so dass Sie das blanke Kabel bzw. die acht freigelegte Buchse des Kabels auf beiden Seiten sehen. Ziehen Sie vor dem anlöten der Kabel an die Buchsenleiste diese wieder am Jetson Nano ab um diesen vor der Hitze des Lötkolbens zu schützen. Schieben Sie ein Kabel nach dem anderen zusammen mit einem kleinen Stück Schrumpfschlauch auf den Pin der Buchsenleiste für z. B. den GND Anschluss auf. Löten Sie jetzt das Kabel an der Buchsenleiste fest und isolieren Sie es mit dem Schrumpfschlauch. Das wiederholen Sie jetzt für die drei weiteren Anschlüsse SDA. SCL und 3,3V.

Die nachfolgende Tabelle hilft Ihnen die richtigen Pins am Jetson Nano für den I²C Bus zu identifizieren. Tragen Sie in die Tabelle die Farben der Kabel des I²C Bus ein die Sie verwendet haben.

Jetson Nano Pin-Bezeichnung	Kabel-Farbe	I ² C Bus Pin-Bezeichnung
3,3V		VCC (3,3V)
3		SDA
5		SCL
GND		GND

Tabelle 5.1 I²C Bus Mapping

Hinweis: Achte Sie bitte darauf, dass der Servo-Kontroller und das OLED Display mit einer Spannung von 3,3V betrieben werden. Andernfalls könnten Sie bei der Verwendung von einer 5V Spannung den Jetson Nano beschädigen.

Die Buchsenleiste die am Jetson Nano aufgesteckt wird sieht fertig gelötet wie auf dem folgenden Bild gezeigt aus. Im Bild markiert wurden die Pins mit ihrer jeweiligen Funktion des I²C Bus sowie der 3,3V und GND Pin.

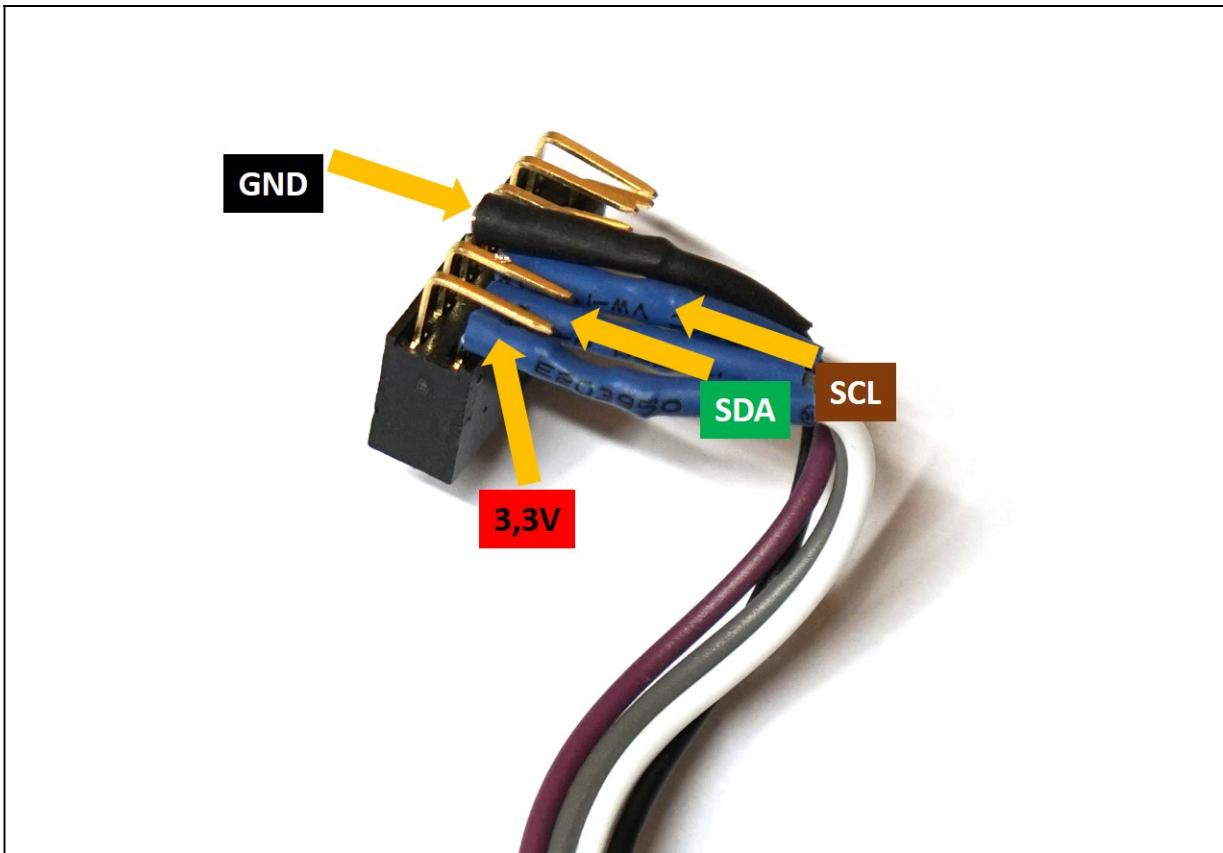


Abbildung 5.7: Buchsenleiste für den Jetson Nano

An den vier anderen Enden des Female-to-Female Kabels denken Sie auch wieder an den Schrumpfschlauch für die Isolierung der Kabel. Löten Sie jetzt die vier Kabel entsprechend der Pin-Belegung des Servo-Kontrollers an der zweiten Buchsenleiste fest.

Auf dem nachfolgenden Bild sehen Sie links die Buchsenleiste die auf den Servo-Kontroller aufgesteckt wurde und rechts die Buchsenleiste die auf dem Jetson Nano steckt. Beide sind über ein vier-Adriges Jumper-Kabel verbunden welches an beiden Buchsenleisten fest verlötet und mit Schrumpfschlauch isoliert wurde. Die I²C Verbindung auf diese Art herzustellen ist deutlich Wackelkontakt freier als nur die Jumperkabel aufzustecken.



Abbildung 5.8: Buchsenleiste rechts und links fest verlötete Jumpe Kabel

Das OLED Display wird ebenfalls am I²C Bus (Ausgang) des Servo-Kontrollers angeschlossen. Wenn dieses z. B. wegen eines Wackelkontakte ausfällt ist das nicht so ärgerlich wie bei dem Servo-Kontroller. Daher können Sie das OLED Display auch einfach nur anstecken und arbeiten hier nicht mit weiteren Buchsenleisten.

Das folgende Bild zeigt wie die Verkabelung des Roboter-Autos bis jetzt aussieht.

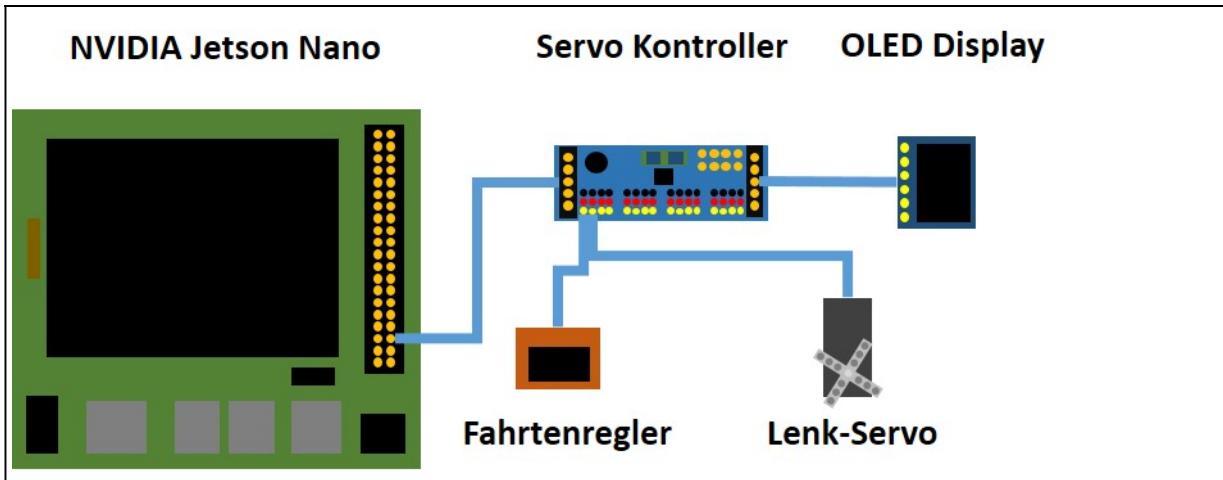


Abbildung 5.9: Logische Verkabelung des Donkey Cars

5.4.3 Fahrtenregler mit aktiver BEC:

Verfügt Ihr Fahrtenregler über eine aktive BEC (Battery Elimination Circuit) Funktion dann versorgt der Fahrtenregler den Servo-Kontroller und somit auch den Lenk-Servo mit einer Spannung von 5V. In diesem Fall müssen Sie den Servo-Kontroller nicht extra mit einer 5V Spannung versorgen. Eine vorhandene BEC Funktion ist eigentlich der Normalfall bei den Fahrtenreglern.

5.4.4 Fahrtenregler ohne BEC:

Verfügt Ihr Fahrtenregler nicht über eine BEC Funktion, dann müssen Sie den Servo-Kontroller mit der Power-Bank verbinden damit der Lenk-Servo mit Strom versorgt wird.

Um die Spannungsversorgung herstellen zu können benötigen Sie jetzt ein abgeschnittenes USB A-Kabel das ca. 30cm lang ist von z. B. einer alten USB-Maus oder USB-Tastatur. Dieses verbinden Sie mit den Schwarzen und Roten Adern des USB Kabels an dem Anschlussterminal des Servo-Kontrollers. Die Weiße und Grüne Ader im USB Kabel sind die Adern für die Datenübertragung und werden nicht benötigt. Diese können Sie ganz kurz abschneiden so dass sie nicht stören. Wenn Sie die beiden Kabel (rot/schwarz) am Servo-Kontroller angeschlossen haben stecken Sie das USB Kabel in die Power-Bank ein und versorgen so den Servo-Kontroller und den angeschlossenen Lenk-Servo mit einer Spannung von 5V.

5.4.5 Kamera anschließen

Sie müssen das CSI Kabel der Kamera am Jetson Nano einstecken. Achten Sie bitte darauf, dass Sie die Verriegelung des CSI Anschlusses am Jetson Nano vorsichtig öffnen. Die beiden Enden des CSI Kabels sind jeweils blau markiert. Die Blaue Seite des CSI Kabels zeigt beim Anschließen am Jetson Nano von diesem Weg.

Am anderen Ende des Kabels schließen Sie jetzt die Kamera an und stecken diese in die Halterung am Überrollkäfig. Beim anschließen des CSI Kabels an der Kamera kann es je nach Modell technische Unterschiede geben ob die blau markierte Seite des CSI Kabels zu oder weg von der Linse zeigt. Das müssen Sie bitte beim Hersteller Ihrer Kamera nachlesen oder einfach ausprobieren.

Fertig aufgebaut sollte Ihr Roboter-Auto jetzt ähnlich dem folgenden Bild aussehen.

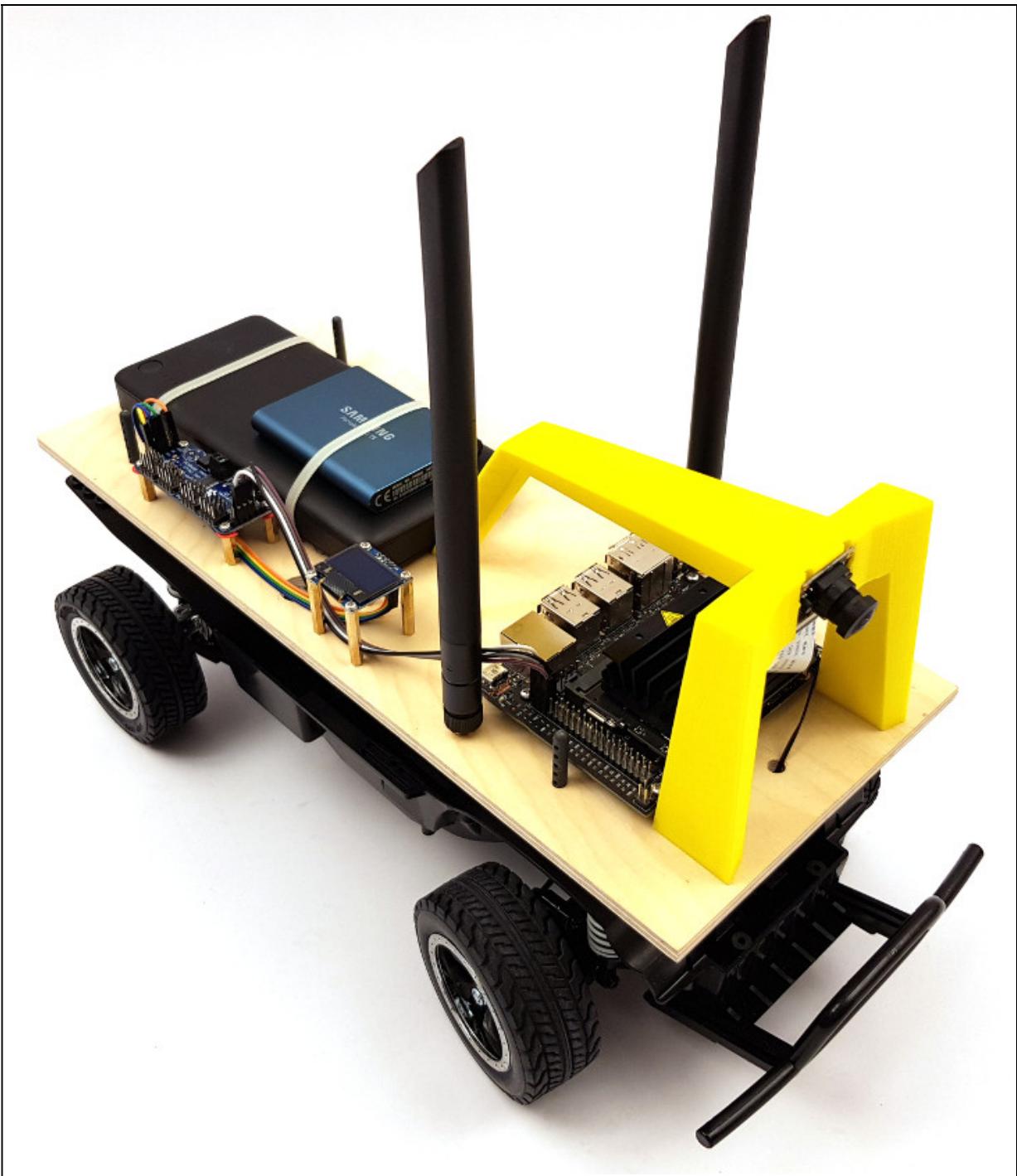


Abbildung 5.10: Das fertig aufgebaute Roboter Auto

Jetzt haben Sie den Aufbau des Roboter-Autos sowie die Verkabelung der elektronischen Komponenten abgeschlossen und erfahren im folgenden Kapitel wie Sie den Jetson Nano in Betrieb nehmen.

Ich freue mich sehr über Ihr Feedback zu diesem Kapitel. Schreiben Sie mir Ihre Gedanken, Fragen und Vorschläge an die folgende E-Mail Adresse: ebook@custom-build-robots.com

6 Einrichten des Betriebssystem des Jetson Nano

Endlich ist es soweit, der Lötkolben ist abgekühlt und die Bohrmaschine verstummt. Der Jetson Nano wird jetzt zum Leben erweckt und Sie stehen kurz davor in die Tiefen der künstlichen Intelligenz und Trainingsdaten Sammelei einzutauchen.

Nach dem Sie Ihr Roboter-Auto soweit aufgebaut haben folgt jetzt der nächste Schritt die Inbetriebnahme des Jetson Nano. Dazu müssen Sie die von NVIDIA bereitgestellt Image Datei auf die micro SD-Karte aufspielen.

Hinweis: Unabhängig davon ob Sie Ihr Roboter-Auto mit einer micro SD-Karte oder SSD Festplatte betreiben werden müssen Sie die folgenden Schritte durchführen.

Laden Sie jetzt die neueste Image Datei von der NVIDIA Web-Seite für den Jetson Nano herunter. Für dieses E-Book wurde von das Image mit dem Namen „jetson-nano-4gb-jp441-sd-card-image.zip“ und der Jet Pack Version 4.4 verwendet.

Die neueste Image Datei finden Sie immer auf der NVIDIA Webseite unter der folgenden URL.

URL: <https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit#write>

Es gibt verschiedene Tools mit denen Sie die Image Datei auf die micro SD-Karte schreiben können. Das Tool von Balena Etcher ist recht weit verbreitet und für diverse Plattformen wie MacOS, Windows und Linux verfügbar. Das Programm finden Sie unter der folgenden Adresse zum Download.

URL: <https://www.balena.io/etcher/>

Schreiben Sie jetzt mit Hilfe des Tools Etcher die Image Datei auf Ihre micro SD-Karte.

Nach dem Sie das Image erfolgreich auf Ihre micro SD-Karte geschrieben haben stecken Sie diese in den Jetson Nano. Schließen den Jeston Nano an einem Monitor, Maus und Tastatur an. Stellen Sie die Stromversorgung des Jetson Nano mit dem externen Netzteil her und schalten Sie diesen ein.

Hinweis: Denken Sie daran den Jumper J48 direkt hinter der DC Buchse zu setzen denn andernfalls startet der Jetson Nano nicht wenn Sie das externe Netzteil anschließen. Haben Sie keinen Jumper zur Hand, dann können sie auch ein Female-to-Female Jumperkabel anstelle eines Jumpers verwenden.

Sie sollten jetzt das Betriebssystem Ubuntu, welches Sie zuvor auf die micro SD-Karte geschrieben haben hochfahren sehen. Folgen Sie dem Installation-Wizard und konfigurieren Sie Ihren Jetson Nano.

Hinweis: Achten Sie bei den Fragen des Installation-Wizard auf die Frage nach dem automatischen Anmelden am eingerichteten WIFI. Diese Funktion sollten Sie bitte aktivieren.

Tipp: Wenn Sie noch nicht so mit Linux und Ubuntu vertraut sind dann hilft ihnen das folgende Kapitel „Kleinen Helfer und Tipps:“ sicher einfacher einen Einstieg zu finden. In diesem beschreibe ich die kleinen Tools unter Linux die ich immer selber direkt installiere. Diese können Ihnen je nach eigener Erfahrung und Wissen die Arbeit unter Linux erheblich erleichtern.

Im jetzt folgenden Abschnitt geht es darum die Ubuntu Installation zu aktualisieren sowie etwas für die Bedürfnisse des Donkey Car Projektes anzupassen.

Daher aktualisieren Sie erst einmal die Repository Informationen der Ubuntu Jetson Nano Installation auf den neuesten Stand.

Befehl: sudo apt-get update

Nach dem die Respository Informationen auf dem neuesten Stand sind aktualisieren Sie mit dem folgenden Befehl alle bereits installierten Programme.

Befehl: sudo apt-get upgrade.

Während die Aktualisierung läuft werden Sie gefragt ob Sie den Display Manager wechseln möchten. Bitte übernehmen Sie einfach die Default Einstellung also den gdm3 Displaymanager. Da Sie später das Donkey Car komplett ohne einem Monitor und grafischer Oberfläche betreiben werden kann hier der

etwas größere und Ressourcen hungriger gdm3 Display Manager installiert werden. In einem späteren Schritt, wenn die Software auf dem Donkey Car komplett funktioniert wird das Betriebssystem so konfiguriert das dieses ohne einen Display Manager zu starten hochfährt.

Nach dem das Update abgeschlossen ist werden Sie jetzt die Voraussetzung schaffen, dass bei den noch folgenden schwergewichtigen Installationsprozessen keine Fehler wegen zu wenig Arbeitsspeicher ausgegeben werden. Daher legen Sie jetzt eine Auslagerungsdatei für den Arbeitsspeicher die sogenannte SWAP Datei an.

6.1.1 SWAP Datei einrichten

Legen wir los mit der Anlage der SWAP Datei. Wenn das Betriebssystem auf eine SWAP Datei zugreifen kann, dann kann das Betriebssystem in diese Teile des Arbeitsspeichers auslagern. Legen Sie jetzt eine SWAP Datei mit einer Größe von ca. 12 GB an, wenn Sie noch ausreichend Platz auf der SD-Karte frei haben. Wenn der Platz nicht ausreicht dann machen Sie die SWAP Datei einfach etwas kleiner aber mindestens 6 GB groß.

Mit dem folgenden Befehl können Sie sich anzeigen lassen wie viel Speicherplatz noch auf der micro SD-Karte verfügbar ist.

Befehl: df -h

Lässt der freie Speicherplatz noch eine 12 GB große SWAP-Datei zu, dann übernehmen Sie den folgenden Befehl. Andernfalls passen Sie die Zahl 12 entsprechend der gewünschten Größe der SWAP-Datei im nachfolgenden Befehl an.

Befehl: sudo fallocate -l 12G /var/swapfile

Jetzt müssen noch die richtigen Rechte gesetzt werden damit auf die SWAP Datei zugegriffen werden kann.

Befehl: sudo chmod 600 /var/swapfile

Anschließend wird die SWAP Datei erzeugt mit der zuvor angegeben Größe.

Befehl: sudo mkswap /var/swapfile

Mit dem folgenden Befehl wird die SWAP Datei aktiviert.

Befehl: sudo swapon /var/swapfile

Damit nach jedem Neustart die SWAP Datei auch aktiviert wird muss der folgende Befehl ausgeführt werden der die SWAP-Datei mit ihrem Pfad in die fstab einträgt.

Befehl: sudo bash -c 'echo "/var/swapfile swap swap defaults 0 0" >> /etc/fstab'

Sollte dieser Befehl eine Fehlermeldung auswerfen, dann starten Sie den Midnight Commander und ergänzen die folgende Zeile in der /etc/fstab Datei.

/var/swapfile swap swap defaults 0 0

Die /etc/fstab Datei sollte jetzt wie folgt aussehen. Der Eintrag mit der SWAP Datei ist gut zu erkennen.

```
# /etc/fstab and tweaking it as you see fit. See fstab(5).
# <file system> <mount point> <type> <options>
/dev/root / ext4 defaults
/var/swapfile swap swap defaults 0 0
```

Abbildung 6.1: fstab SWAP Datei

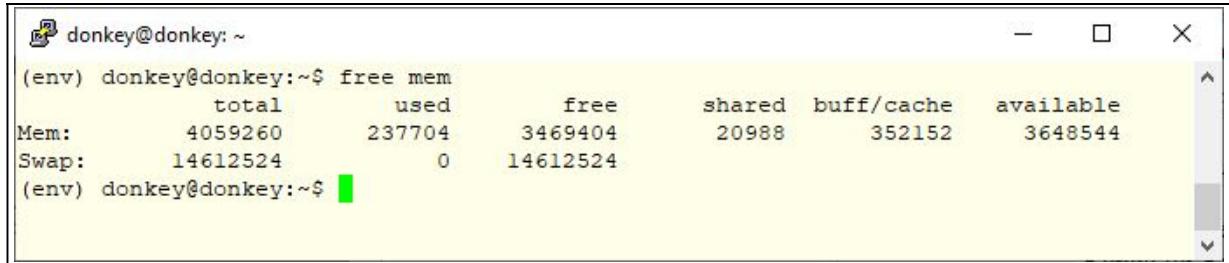
Jetzt ist alles soweit eingerichtet von den kleinen Tools und Optimierungen die Ihnen helfen leichter unter Linux im Terminal Fenster zurecht zu kommen bis hin zur SWAP-Datei. Starten Sie jetzt den Jetson Nano neu

Befehl: sudo reboot

Nach dem Neustart des Jetson Nano können Sie mit dem folgenden Befehl überprüfen, ob die SWAP Datei aktiv eingebunden wurde.

Befehl: free mem

Die Ausgabe die Sie in Ihrem Terminal Fenster angezeigt bekommen sollte wie folgt aussehen. Die ausgegebene Speichergröße kann sich entsprechend Ihrer Hardware und Konfiguration unterscheiden.



	total	used	free	shared	buff/cache	available
Mem:	4059260	237704	3469404	20988	352152	3648544
Swap:	14612524	0	14612524			

Abbildung 6.2: free mem Ausgabe

6.2 SSD Festplatte einrichten – empfohlen (aber optional)

Die Anforderungen des Donkey Car Frameworks an die Hardware des Jetson Nano sind sehr hoch und eine SSD Festplatte bringt Stabilität und Geschwindigkeit in das Roboter-Auto Projekt. Einen großen Unterschied werden Sie merken wenn Sie das Neuronale Netz trainieren da bringt die SSD Festplatte ihre Vorteile bei der Lese- und Schreibgeschwindigkeit voll in das Projekt ein.

Im Internet gibt es die Webseite www.jetsonhacks.com die immer sehr gute Anleitungen samt Skripten zur Jetson Familie von NVIDIA. online stellt. Genau hier gibt es auch eine Anleitung die genau beschreibt wie der Jetson Nano konfiguriert werden kann das dieser von einer SSD Festplatte booted. Dieser Anleitung werden wir folgen ohne auf die Hintergründe einzugehen. Diese sind auf der Seite selber sehr gut ausgeführt.

Zunächst müssen Sie das Projekt rootOnUSB von dem JetsonHacks GitHub Account herunter laden. Wechseln Sie dazu bitte in das Home-Verzeichnis ihres Users auf dem Jetson Nano und führen Sie den folgenden Befehl aus.

Befehl: git clone <https://github.com/JetsonHacksNano/rootOnUSB>

Wechseln Sie in das Verzeichnis rootOnUSB

Befehl: cd rootOnUSB

Als nächsten Schritt muss das initramfs mit USB Unterstützung gebaut werden. Nur so ist es möglich sehr früh im Boot-Prozess auf die USB Schnittstelle des Jetson Nano und somit auf die angeschlossene SSD Festplatte zuzugreifen. Das Skript addUSBTToInitramfs.sh übernimmt genau diese Aufgabe. Führen Sie das Skript mit dem folgenden Befehl aus.

Befehl: ./addUSBTToInitramfs.sh

Nach dem Sie den Befehl ausgeführt haben, werden mehrere Wahrungen angezeigt, dass verschiedene Dateien nicht gefunden werden konnten. Die Warnungen sollten Sie nicht beunruhigen und die USB Unterstützung für das Booten von der SSD Festplatte wird trotz dieser Meldungen funktionieren.

Die Ausgabe und die angezeigten Wahrungen sollten wie folgt aussehen.

```

donkey@donkey: ~/rootOnUSB
donkey@donkey:~/rootOnUSB$ ./addUSBToInitramfs.sh
Adding USB to initramfs
[sudo] password for donkey:
Warning: couldn't identify filesystem type for fsck hook, ignoring.
I: The initramfs will attempt to resume from /dev/zram3
I: (UUID=cf251400-52da-4d26-a4e1-4feb4135011d)
I: Set the RESUME variable to override this.
/sbin/ldconfig.real: Warning: ignoring configuration file that cannot be opened: /etc/ld.so.conf.d/aarch64-linux-gnu_EGL.conf: No such file or directory
/sbin/ldconfig.real: Warning: ignoring configuration file that cannot be opened: /etc/ld.so.conf.d/aarch64-linux-gnu_GL.conf: No such file or directory
donkey@donkey:~/rootOnUSB$ 

```

Abbildung 6.3: rootOnUSB Wahrnungen

Jetzt müssen Sie die angeschlossene SSD Festplatte noch vorbereiten. Legen Sie dazu eine große Partition die ext4 formatiert ist auf dieser an. Bei Ubuntu ist das Programm Disks bereits installiert das Sie jetzt brauchen. Wenn Sie jetzt eine Partition mit Disks auf Ihrer SSD Festplatte anlegen dann nennen Sie diese Partition z. B. SSD. Den Volumen-Namen der Festplatte benötigen wir im nachfolgenden Schritt wenn alle Daten von der micro SD-Karte auf die Festplatte kopiert werden.

Die Festplatte sollte jetzt eingerichtet in Disks wie folgt aussehen. Sie sehen in dem Bild, dass diese den Volumen-Namen SSD erhalten hat. Auch ist zu sehen, dass die Festplatte als Gerät /dev/sda erkannt wurde.

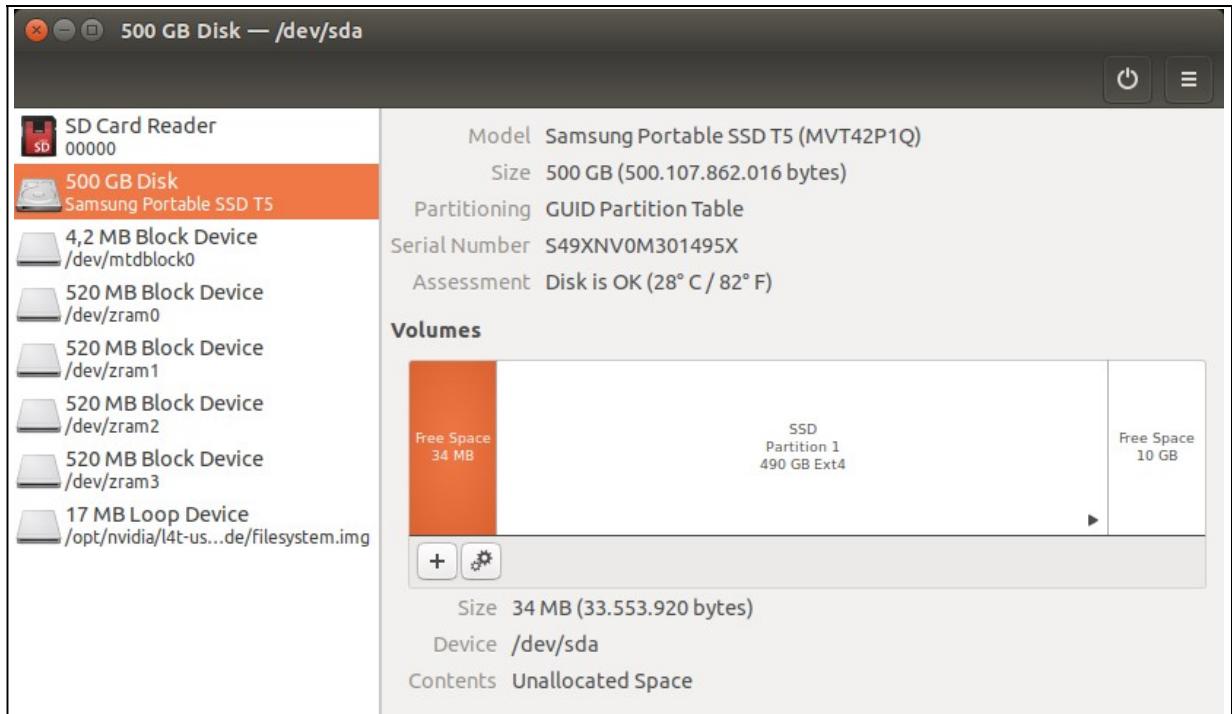


Abbildung 6.4: Programm Disks unter Ubuntu

Anschließend kopieren Sie alle Daten von der micro SD-Karte auf die Festplatte mit dem Volumen-Namen SSD.

Hinweis: Bitte achten Sie vor dem Ausführen des folgenden Befehls darauf, dass die Festplatte bzw. die Partition mit dem Namen SSD auch eingebunden bzw. gemounted ist. Um die Partition zu mounten können Sie links im Menü von Ubuntu die Partition SSD durch Anklicken ihres Icons diese so einbinden bzw. mounten.

Führen Sie jetzt den folgenden Befehl aus um den Kopiervorgang zu starten.

Befehl: ./copyRootToUSB.sh -v SSD

Der Kopiervorgang dauert mehrere Kaffee lang.

6.2.1 Erster Eintrag – LABEL primary

Nach dem der Kopiervorgang abgeschlossen ist, werden Sie noch eine letzte Konfiguration vornehmen. Mit dieser legen Sie fest, dass der Jetson Nano während dem Bootvorgang von der micro SD-Karte auf die SSD Festplatte wechselt und von dort das Betriebssystem hochfährt. Öffnen Sie dazu mit dem Text-Editor nano die extlinux.conf Datei.

Befehl: sudo nano /boot/extlinux/extlinux.conf

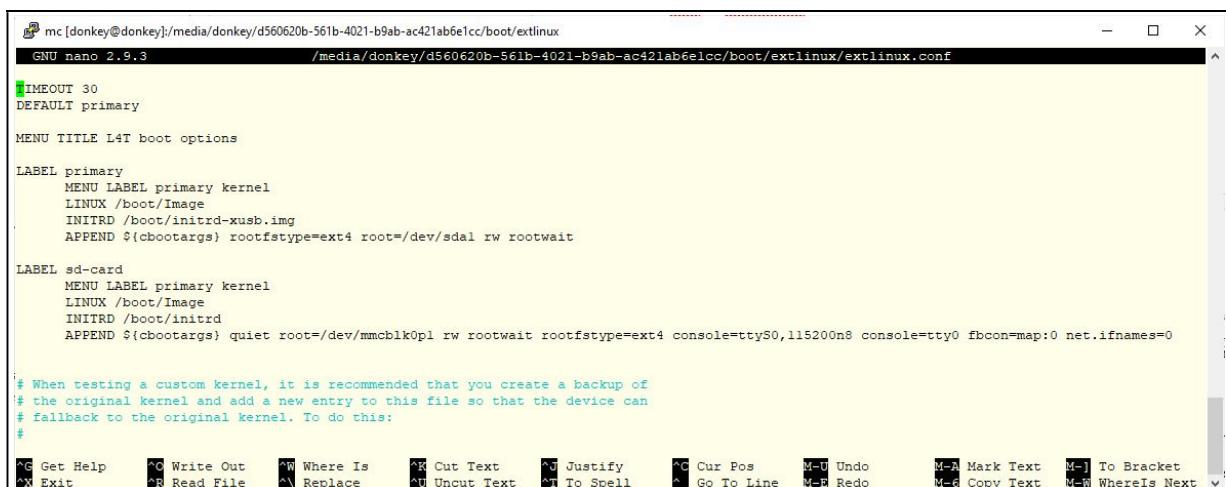
Kopieren Sie das komplette Label primary mit seinen fünf Zeilen einmal und passen Sie jetzt die Konfiguration wie nachfolgend beschrieben an.

Der Label Eintrag „primary“ wird nur insofern angepasst, dass der Pfad der micro SD-Karte durch den Pfad der SSD Festplatte ersetzt wird. Mit dem bereits bekannten Befehl df -h können Sie sich noch einmal vergewissern wie die Partition genau heißt die Sie angelegt haben. Diese sollte z. B. sda1 heißen und über den Pfad /dev/sda1 ansprechbar sein.

6.2.2 Zweiter Eintrag – LABEL sd-card

Da Sie den ersten Eintrag ja kopiert haben müssen Sie hier nur den Namen des Labels anpassen auf z. B. „sd-card“. Passen Sie ebenfalls die verbleibenden Zeilen wie beschrieben an.

Fertig konfiguriert sieht die Datei extlinux.conf dann wie folgt aus.



```
mc [donkey@donkey]:/media/donkey/d560620b-561b-4021-b9ab-ac421ab6e1cc/boot/extlinux
GNU nano 2.9.3                                     /media/donkey/d560620b-561b-4021-b9ab-ac421ab6e1cc/boot/extlinux/extlinux.conf

TIMEOUT 30
DEFAULT primary

MENU TITLE L4T boot options

LABEL primary
    MENU LABEL primary kernel
    LINUX /boot/Image
    INITRD /boot/initrd-xusb.img
    APPEND ${cbootargs} rootfstype=ext4 root=/dev/sda1 rw rootwait

LABEL sd-card
    MENU LABEL primary kernel
    LINUX /boot/Image
    INITRD /boot/initrd
    APPEND ${cbootargs} quiet root=/dev/mmcblk0p1 rw rootwait rootfstype=ext4 console=ttyS0,115200n8 console=tty0 fbcon=map:0 net.ifnames=0

# When testing a custom kernel, it is recommended that you create a backup of
# the original kernel and add a new entry to this file so that the device can
# fallback to the original kernel. To do this:
#



^G Get Help   ^W Write Out   ^W Where Is   ^K Cut Text   ^J Justify   ^C Cux Pos   M-U Undo
^X Exit      ^R Read File   ^\ Replace   ^U Uncut Text  ^I To Spell   ^A Go To Line  M-E Redo
                                         M-A Mark Text   M-J To Bracket
                                         M-C Copy Text  M-W WhereIs Next
```

Abbildung 6.5: extlinux.conf Anpassung

Hier die Anpassung noch einmal in Textform. Die wichtigen Stellen auf die unbedingt geachtet werden muss sind fett hervorgehoben.

```
TIMEOUT 30
DEFAULT primary

MENU TITLE L4T boot options

LABEL primary
    MENU LABEL primary kernel
    LINUX /boot/Image
    INITRD /boot/initrd-xusb.img
    APPEND ${cbootargs} rootfstype=ext4 root=/dev/sda1 rw rootwait
```

```
LABEL sd-card
MENU LABEL primary kernel
LINUX /boot/Image
INITRD /boot/initrd
APPEND ${cbootargs} quiet root=/dev/mmcblk0p1 rw rootwait rootfstype=ext4
console=ttyS0,115200n8 console=tty0 fbcon=map:0 net.ifnames=0
```

Speichern Sie jetzt die Änderungen in extlinux.conf ab und starten Sie den Jetson Nano neu.

Wenn Sie sich anschließend wieder auf der grafischen Oberfläche anmelden sollten Sie links unten das Symbol für eine eingesteckte micro SD-Karte sehen. Ist dem so dann hat Ihr Jetson Nano Fehlerfrei von der SSD-Festplatte das Betriebssystem gestartet. Wenn nicht dann ist eventuell etwas schief gelaufen in der extlinux.conf Datei. Prüfen Sie hier noch einmal die Anpassungen die Sie vorgenommen haben.

6.3 Was Sie bis hier erreicht habe.

Sie haben alle Komponenten vom Servo Kontroller bis hin zu der Kamera angeschlossen und die micro SD-Karte mit dem Jetson Nano Image vorbereitet. Auch haben Sie den Jetson Nano eingeschalten und die SWAP-Datei eingerichtet. Ihr Jetson Nano startet fehlerfrei von der micro SD-Karte oder eventuell auch von der SSD Festplatte wenn Sie eine für dieses Projekt verwenden.

Ich freue mich sehr über Ihr Feedback zu diesem Kapitel. Schreiben Sie mir Ihre Gedanken, Fragen und Vorschläge an die folgende E-Mail Adresse: ebook@custom-build-robots.com

7 Installation des Donkey Car Frameworks und Kalibrierung des Roboter-Auto

Schön dass Sie sich weiter voran arbeiten und nicht rechts heraus gefahren sind. In diesem Kapitel geht es endlich darum das Roboter-Auto zum Leben zu erwecken. Von der Installation des Donkey Car Frameworks über das Kalibrieren bis zum ersten Fahren.

Das Donkey Car Framework bildet die Softwarebasis des Roboter-Autos. Mit der Installation der in diesem Kapitel aufgeführten Softwarepakete zusammen mit dem Donkey Car Framework wird aus dem ehemaligen RC Auto ein autonom fahrendes Modellauto auf Jetson Nano Basis. Das Donkey Car Framework ist seit der Version 3.x auch für den Jetson Nano verfügbar und kann auf diesem ohne Probleme installiert werden. Sie sollten sich für die Installation etwas Zeit nehmen. Sie müssen ja nicht immer vor dem Rechner sitzen und zuschauen wie die Softwareinstallation voranschreitet.

Mit den ersten beiden Befehlen wird der Jetson Nano noch einmal auf den neuesten Softwarestand gebracht da vielleicht seit der letzten Aktualisierung schon etwas Zeit verstrichen ist. Daher sollten Sie diese beiden Befehle immer ausführen bevor Sie wie gewohnt die restlichen Befehle einen nach dem anderen eingeben und die Software Schritt für Schritt installieren.

Befehl: sudo apt-get update

Befehl: sudo apt-get upgrade

Bevor Sie jetzt die weiteren Bibliotheken und Tools für das Donkey Car Framework installieren prüfen Sie bitte, dass auch die SWAP-Datei aktiv ist. Das können Sie ganz einfach mit dem folgenden Befehl machen.

Befehl: free mem

Wenn Sie die SWAP-Datei nicht sehen dann prüfen Sie noch einmal nach ob Sie diese noch konfigurieren oder lediglich aktivieren müssen. Das genaue Vorgehen wurde im Kapitel 6.2 bereits beschrieben.

Hinweis: Aktuell wird das Donkey Car Framework sehr aktiv weiter entwickelt und es kommt immer wieder zu Änderungen. Dieses E-Book wird immer wieder aktualisiert und daher prüfen Sie unter der folgenden URL ob eine neue Version vorliegt.

URL: <bitte ergänzen>

Fahren Sie jetzt mit der Installation fort und installieren Sie die weiteren Pakete auf Ihrem Jetson Nano. Das ist jetzt eine ziemlich lange Liste die Sie abarbeiten müssen.

Befehl: sudo apt-get install -y libhdf5-serial-dev hdf5-tools libhdf5-dev zlib1g-dev zip libjpeg8-dev liblapack-dev libblas-dev gfortran

Befehl: sudo apt-get install -y python3-dev python3-pip

Befehl: sudo apt-get install -y libxslt1-dev libxml2-dev libffi-dev libcurl4-openssl-dev libssl-dev libpng-dev libopenblas-dev

Befehl: sudo apt-get install -y git

Befehl: sudo apt-get install -y openmpi-doc openmpi-bin libopenmpi-dev libopenblas-dev

Befehl: sudo -H pip3 install -U pip testresources setuptools

Befehl: sudo -H pip3 install -U futures==3.1.1 protobuf==3.12.2 pybind11==2.5.0

Befehl: sudo -H pip3 install -U cython==0.29.21

Befehl: sudo -H pip3 install -U numpy==1.19.0

Befehl: sudo -H pip3 install -U future==0.18.2 mock==4.0.2 h5py==2.10.0 keras_preprocessing==1.1.2 keras_applications==1.0.8 gast==0.3.3

Befehl: sudo -H pip3 install -U grpcio==1.30.0 absl-py==0.9.0 py-cpuinfo==7.0.0 psutil==5.7.2 portpicker==1.3.1 six requests==2.24.0 astor==0.8.1 termcolor==1.1.0 wrapt==1.12.1 google-pasta==0.2.0

Befehl: sudo -H pip3 install -U scipy==1.4.1

Befehl: sudo -H pip3 install -U pandas==1.0.5

Befehl: sudo -H pip3 install -U gdown

Mit dem folgenden Befehl wird TensorFlow in der Version 2.2.0 auf Ihrem Jetson Nano installiert.

Befehl: sudo -H pip3 install --pre --extra-index-url <https://developer.download.nvidia.com/compute/redist/jp/v44/tensorflow==2.2.0+nv20.6>

Mit den beiden folgenden Befehlen wird PyTorch v1.7 installiert.

Befehl: wget <https://nvidia.box.com/shared/static/wa34qwrwtk9njtyarwt5nvo6imenfy26.whl> -O torch-1.7.0-cp36-cp36m-linux_aarch64.whl

Befehl: sudo -H pip3 install ./torch-1.7.0-cp36-cp36m-linux_aarch64.whl

Als nächstes wird PyTorch Vision installiert vorab werden aber noch ein paar Pakete benötigt..

Befehl: sudo apt-get install libjpeg-dev zlib1g-dev libpython3-dev libavcodec-dev libavformat-dev libswscale-dev

Jetzt laden Sie torchvision v0.8.1 in den Ordner projects herunter und installieren torchvision.

Befehl: mkdir -p ~/projects; cd ~/projects

Befehl: git clone --branch v0.8.1 <https://github.com/pytorch/vision/torchvision>

Befehl: cd torchvision

Befehl: export BUILD_VERSION=0.8.1

Befehl: sudo python3 setup.py install

Befehl: echo "export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1" >> ~/.bashrc

Jetzt sollten Sie den Jeston Nano neu starten bevor Sie mit der Software Installation weiter fortfahren.

Befehl: sudo reboot

Mit dem folgenden Befehl können Sie prüfen welche Tensorflow Version bei Ihnen jetzt installiert ist.

Befehl: python -c 'import tensorflow as tf; print(tf.__version__)'

7.1 Die virtuelle Umgebung einrichten

Das Donkey Car Framework wird in einer virtuellen Umgebung installiert. So sind Sie später flexibler wenn Sie z. B. in einer zweiten virtuellen Umgebung die Software für eine Gesichtserkennung installieren möchten da sich die beiden Installationen nicht gegenseitig beeinflussen. Bei Problemen können Sie z. B. in einer dritten virtuellen Umgebung erneut die Software für das Roboter-Auto installieren und können so zwischen den Installationen wechseln.

Befehl: pip3 install virtualenv

Mit dem folgenden Befehl wird eine virtuelle Umgebung mit dem Namen env angelegt.

Befehl: python3 -m virtualenv -p python3 env --system-site-packages

Für deutlich mehr Komfort wird mit dem folgenden Befehl der Eintrag *source env/bin/activate* in die .bashrc Datei im Home-Verzeichnis am Ende dieser eingetragen. Dieser Eintrag sorgt dafür, dass immer wenn Sie sich am Jetson Nano anmelden die virtuelle Umgebung gestartet wird und Sie sich in dieser anschließend befinden.

Befehl: echo "source env/bin/activate" >> ~/.bashrc

Starten Sie jetzt von Hand das erste Mal die virtuelle Umgebung mit dem Namen env mit dem folgenden Befehl.

Befehl: source ~/bashrc

Sie erkennen im Terminal Fenster in der Eingabe Zeile ganz am Anfang anhand dem „env“ das Sie sich in der virtuellen Umgebung mit dem Namen „env“ befinden. Im folgenden Bild ist die Stelle im Terminal Fenster mit einem orangen Kreis hervorgehoben.

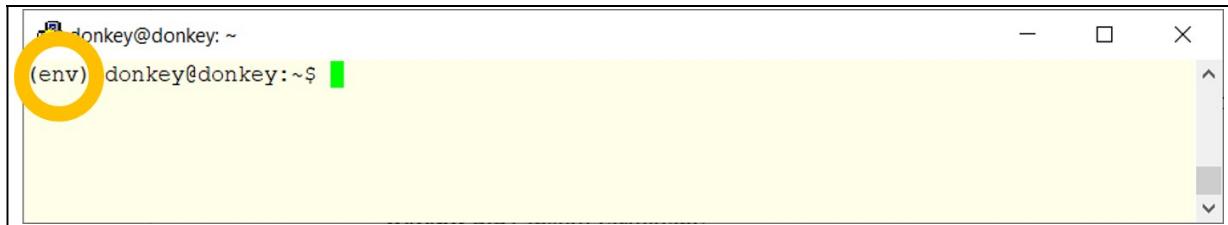


Abbildung 7.1: Aktive virtuelle Umgebung

Möchte man die virtuelle Umgebung wieder deaktivieren bzw. verlassen dann können Sie das mit dem folgenden Befehl machen.

Befehl: deactivate

Nach dem jetzt auch die virtuelle Umgebung soweit eingerichtet ist kann die Installation weiter gehen aber befolgen Sie vielleicht noch den nachfolgenden Rat an dieser Stelle.

Die Installation des Jetson Nano hat bei mir bis jetzt ca. 3 bis 4 Stunden in Anspruch genommen. Daher wäre jetzt der richtige Zeitpunkt ein Abbild, also ein Image, der micro SD-Karte als Backup zu erstellen.

Beispiel Befehl: dd if=/dev/sdc of=~/image_nano.img status=progress

Fahren Sie den Jetson Nano mit dem folgenden Befehl herunter.

Befehl: sudo shutdown -h now

7.2 Donkey Car Framework installieren

Jetzt sind alle Vorbereitungen soweit getroffen und das Donkey Car Framework für den Jetson Nano kann von GitHub herunter geladen werden. Am besten speichern Sie das Framework in einen Ordner mit dem Namen „projects“ in Ihrem Home-Verzeichnis.

Hinweis: Bitte achten Sie darauf, dass Sie die folgenden Schritte der Softwareinstallation in der aktiven virtuellen Umgebung „env“ durchführen.

Mit dem folgenden Befehl legen Sie den Ordner mit dem Namen projects an.

Befehl: mkdir ~/projects

Jetzt wechseln Sie in den Ordner projects.

Befehl: cd ~/projects

Von GitHub laden Sie jetzt das aktuelle Donkey Car Framework herunter. Das klappt mit den folgenden Befehlen.

Befehl: git clone <https://github.com/autorope/donkeycar>

Bitte wechseln Sie jetzt in den Ordner „donkeycar“ unter dem Ordner „projects“.

Befehl: cd donkeycar

Jetzt müssen Sie noch das Projekt auschecken.

Befehl: git checkout master

Nach so vielen Vorbereitungen folgt jetzt die Installation des Donkey Car Frameworks. Das Framework gibt es auch für den Raspberry Pi und wird im Default für diesen installiert. Daher müssen Sie den Parameter [nano] bei dem folgenden Installationsbefehl mit an den Aufruf anhängen.

Befehl: pip install -e .[nano]

Hinweis: Falls eine Fehlermeldung kommt, dass der „pip install command“ nicht gefunden wurde dann starten Sie Ihren Jetson Nano neu. Nach dem Neustart sollte der Befehl ausführbar sein.

Die Installation des Donkey Car Frameworks dauert jetzt nur noch ein paar Minuten.

Jetzt folgt das Anlegen der eigenen Donkey Car Instanz auf Ihrem Jetson Nano. Dazu legen Sie einen Ordner mit dem Namen mycar in Ihrem Homeverzeichnis an und wechseln anschließend in diesen Ordner.

Befehl: mkdir ~/mycar

Befehl: cd ~/mycar

Mit dem folgenden Befehl wird eine Donkey Car Instanz im Ordner mycar angelegt.

Befehl: donkey createcar --path ~/mycar

Sollte eine Fehlermeldung erscheinen wie die nachfolgende, dann stimmen die Dateirechte des Ordners nicht.

Fehlermeldung: PermissionError: [Errno 13] Permission denied: '/home/robo-Auto/mycar/models'

Mit dem nachfolgenden Befehl werden die Dateirechte entsprechend gesetzt und die Fehlermeldung wird nicht mehr erscheinen. Anschließend wiederholen Sie den createcar Befehl für das Anlegen der Donkey Car Instanz.

Befehl: sudo chmod 777 ~/mycar

Jetzt ist alles an Software auf dem Jetson Nano installiert die für ein funktionierendes Roboter-Auto benötigt wird. Der nächste Schritt ist die Konfiguration der gerade frisch angelegten Donkey Car Instanz mit den technischen Parametern der von Ihnen verwendeten Roboter-Auto Hardware wie Chassis, Servo-Motor und Fahrtenregler. Nur wenn die Parameter konfiguriert sind ist es später möglich das Roboter-Auto zu beschleunigen, zu lenken und die Kamera am Jetson Nano zu nutzen.

7.2.1 I2C Bus und Servo Kontroller PCA9685 Anpassung

Es kann passieren, dass der Servo Kontroller PCA9685 der am I2C Bus hängt nicht gefunden wird da wir mit unserem User keinen Zugriff auf den I2C Bus des Jetson Nano haben.

Ob der User den Sie verwenden auf den I2C Bus zugreifen kann testen Sie mit dem folgenden Befehl.

Befehl: sudo i2cdetect -r -y 1

Nach dem Sie den Befehl ausgeführt haben wird mit den I2C-Adressen 40 und 70 der PCA9685 Servo-Controller gefunden. Mit der I2C Adresse 3c das bereits angeschlossene OLED Display. Die Anzeige im Terminal Fenster sieht dann wie folgt aus.

```
donkey@donkey: ~$ sudo i2cdetect -r -y 1
 0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:      --- --- --- --- --- --- --- --- --- ---
10:      --- --- --- --- --- --- --- --- --- ---
20:      --- --- --- --- --- --- --- --- --- ---
30:      --- --- --- --- --- 3c --- --- ---
40: 40 --- --- --- --- --- --- --- --- ---
50:      --- --- --- --- --- --- --- --- ---
60:      --- --- --- --- --- --- --- --- ---
70: 70 --- --- --- --- --- --- --- --- ---
```

Abbildung 7.2: I2C Bus Anzeige aller angeschlossenen Geräte

Kommt eine Fehlermeldung bzw. es wird der Servo Kontroller nicht erkannt, dann kann das mit sehr großer Wahrscheinlichkeit daran liegen, dass der I2C Bus nicht mit Ihrem User gelesen werden darf. Um das zu ändern führen Sie bitte den folgenden Befehl aus um die Gruppenzuordnung des I2C Bus zu ändern.

Befehl: sudo usermod -aG i2c \$USER

Anschließend starten Sie den Jetson Nano neu und testen erneut ob Sie den Servo Kontroller der am I2C Bus angeschlossen ist mit dem i2cdetect Befehl finden können.

7.3 Konfiguration des Roboter-Auto

Jetzt geht es los mit der Konfiguration des Roboter-Autos. Sie haben die Hardware soweit aufgebaut und auf dem Jetson Nano alles an Software wie beschrieben installiert. Das Ziel von diesem Abschnitt ist die Lenkung und den Fahrtenregler so zu konfigurieren, dass Sie das Roboter-Auto mit dem Gamepad ohne Probleme steuern können. Der Geradeauslauf sollte stimmen sowie der Lenkeinschlag nach links und rechts im gleichen Maße möglich sein. Die Konfiguration des Fahrtenreglers ist wichtig, damit das Roboter-Auto richtig beschleunigt und bremst wie es typisch für ein Auto ist.

Hinweis: Da Sie jetzt nicht nur den Jetson Nano einschalten werden sondern auch den Fahrtenregler mit der Batterie verbinden stellen sie das Roboter-Auto auf z. B. einen Karton, so dass die Räder frei in der Luft hängen. Mir ist es schon passiert, dass eines meiner Roboter-Autos unerwartet losgefahren ist. Einmal ist eines vom Tisch gefallen und ein anderes Mal hat ein Auto meinen Monitor ordentlich verkratzt.

7.3.1 Voraussetzungen für die Kalibrierung

Für die weitere Kalibrierung sind Sie via SSH mit dem Donkey Car verbunden und haben die Kabel wie Monitor, Tastatur und Maus abgezogen. Die Räder des Roboter-Autos sind in der Luft und der Jetson Nano bekommt seine Energie über die Power Bank. Der RC Akku für den Antriebsmotor ist voll aufgeladen und ebenfalls angeschlossen.

Anschließend schalten Sie, falls vorhanden, mit dem kleinen Schalter auf der Unterseite des Tamiya Chassis die Stromversorgung vom Fahrtenregler zum Servo-Kontroller ein (BEC Funktion).

7.3.2 Lenkung Kalibrieren:

Falls Sie den Lenk-Servo noch nicht mit dem Servo-Kontroller verbunden haben so müssen sie diesen jetzt am Kanal 1 anschließen. Über das Terminal Fenster wechseln Sie in den Ordner mycar.

Befehl: cd ~/mycar

Jetzt starten Sie das Donkey Car Framework für die Kalibrierung der Lenkung mit dem folgenden Befehl.

Befehl: donkey calibrate --channel 1 --bus=1

Anschließend werden Sie aufgefordert einen PWM Wert einzugeben. Starten Sie mit einem Wert um ca. 400 der in etwa der Geradeauslauf sein sollte. Anschließend suchen Sie sich jeweils den maximalen Wert für den linken und rechten Lenkeinschlag. Bei einem meiner Modelle liegen die Werte im folgenden Bereich. Den maximalen Wert finden Sie in dem Sie den Servo-Motor soweit die Lenkungen einschlagen lassen das dieser mechanisch an einen Widerstand stößt und brummt. Jetzt ändern Sie den Wert solange ab bis kein Brummen mehr zu hören ist.

	Beispiel Werte:	Ihre Werte:
Volleinschlag links:	250	
Geradeauslauf:	400	
Volleinschlag rechts:	550	

Haben Sie die passenden Werte für die Lenkung Ihres Roboter-Autos gefunden, dann notieren Sie sich diese Werte in der obigen Tabelle.

7.3.3 Fahrtenregler Kalibrieren:

Falls Sie den Fahrtenregler noch nicht mit dem Servo-Kontroller verbunden haben so müssen sie diesen jetzt am Kanal 0 anschließen.

Wie schon bei der Kalibrierung der Lenkung müssen Sie jetzt den Wert finden bei dem Ihr Motor still steht. Dazu starten Sie wieder den folgenden Befehl aber jetzt mit dem Kanal 0 für den Fahrtenregler.

Hinweis: Die Reifen des Robote-Autos sind in der Luft.

Befehl: donkey calibrate --channel 0 --bus=1

Wenn Sie den Wert für Stopp des Motors an Ihrem Fahrtenregler durch ausprobieren gefunden haben dann erhöhen Sie diesen Wert in ca. 20er Schritten. So können Sie feststellen ab wann das Roboter-Auto anfängt vorwärts zu fahren. Erhöhen Sie den PWM Wert weiter bis sich die Räder mit Top-Speed drehen.

Wenn Sie den Wert für den Rückwärtsgang suchen dann müssen Sie immer mit dem Wert für Motor-Stopp anfangen und dann diesen Wert in ca. 20er Schritten verkleinern bis Ihr Modell Auto schnell Rückwärts fährt.

Bei meinem Modell liegen die Werte im folgenden Bereich.

	Beispiel Werte:	Ihre Werte:
Positive Beschleunigung:	500	
Motor-Stopp:	400	
Negative Beschleunigung:	300	

Haben Sie die passenden Werte für den Fahrtenregler Ihres Roboter-Autos gefunden, dann notieren Sie sich diese in der obigen Tabelle.

Hinweis: Es kann sein, dass der Wert für die positive Beschleunigung kleiner ist als der Wert für die negative Beschleunigung. Ist dem so dann ist später am rechten Joystick des Gamepads die Richtung vertauscht mit der Sie Ihr Roboter-Auto beschleunigen oder abbremsen.

Ich werde Ihnen im folgenden Abschnitt erklären wie Sie via Konfiguration in der Software die Fahrtrichtung vertauschen können ohne den Antriebsmotor eventuell neu zu verkabeln.

7.3.4 VERTAUSCHTE RICHTUNG BEI DER BESCHLEUNIGUNG

Sollte Ihr Roboter-Auto statt vorwärts leider rückwärtsfahren und umgedreht dann müssen Sie noch eine kleine Anpassung im Part actuator.py vornehmen.

Öffnen Sie die Datei actuator.py im Ordner ~/projects/donkeycar/parts wieder mit dem Text-Editor Ihrer Wahl und suchen Sie nach der Klasse „PWMThrottle“. In dieser müssen Sie für die beiden Variablen MIN_THROTTLE und MAX_THROTTLE lediglich das Vorzeichen des übergebenen Wertes ändern.

Die Klasse PWMThrottle ändern Sie wie folgt ab:

class PWMThrottle:

- ...
- MIN_THROTTLE = 1
- MAX_THROTTLE = -1

Ab jetzt sollte Ihr Roboter-Auto vorwärts fahren wenn Sie den rechten Joystick nach vorne bewegen und rückwärts wenn Sie den Joystick auf sich zu ziehen.

7.3.5 KONFIGURATIONSDATEI MYCONFIG.PY

Die zentrale Konfigurationsdatei Ihres Roboter-Autos ist die myconfig.py. Diese Datei wird auch bei einem Upgrade des Donkey Car Frameworks nicht überschrieben und liegt im Ordner ~/mycar.

Die wesentlichen Anpassungen werden Sie jetzt in der myconfig.py vornehmen. Dazu führe ich Sie Schritt für Schritt durch die Konfiguration. Öffnen Sie die Datei z. B. über eine SSH Verbindung im Text Editor Nano oder einem Editor Ihrer Wahl.

Befehl: nano ~/mycar/myconfig.py

Die erste Änderung wird im Punkt #PATHS vorgenommen. Hier werden die Pfade festgelegt in welche die Trainingsdaten und später das fertig trainierte Neuronale Netz bzw. der Autopilot abgespeichert werden.

- #PATHS
- CAR_PATH = PACKAGE_PATH = os.path.dirname(os.path.realpath(__file__))

- DATA_PATH = os.path.join(CAR_PATH, 'data')
- MODELS_PATH = os.path.join(CAR_PATH, 'models')

Anschließend folgt die Einstellung für die Drive-Loop. Die Drive Loop legt fest wie schnell die Schleife durchlaufen wird die ein aktuelles Bild der Kamera + Lenkeinschlag + Geschwindigkeit jeweils als *.jpg und dazu gehöriger *.JSON Datei abspeichert.

- #VEHICLE
- DRIVE_LOOP_HZ = 20
- MAX_LOOPS = None

Die jetzt folgenden Einstellungen für die Kamera sind wichtig. Hier muss für den Jetson Nano der Eintrag von PICAM auf CSIC geändert werden. Abhängig von Ihrem verwendeten Kameramodell und wie die Kamera montiert ist könnte es sein, dass das Bild auf dem Kopf stehend angezeigt wird. Sie können das Bild noch rotieren und richtig stellen wenn Sie das möchten. Mit dem Parameter „CSIC_CAM_GSTREAMER_FLIP_PARM“ können Sie das Kamerabild drehen. Auch wenn das Bild z. B. auf dem Kopf steht stört dies das Neuronale Netz später nicht. Nur für uns als Menschen ist dann das Bild etwas schwerer zu deuten.

- #CAMERA
- CAMERA_TYPE = "CSIC"
- IMAGE_W = 160
- IMAGE_H = 120
- IMAGE_DEPTH = 3
- CAMERA_FRAMERATE = DRIVE_LOOP_HZ
- CSIC_CAM_GSTREAMER_FLIP_PARM = 0

Der Jetson Nano erwartet den I²C Bus auf der Bus-Nummer 1 statt 0. Daher muss die Zeile PCA9685_I2C_BUSNUM entsprechend angepasst werden.

- #9865, over rides only if needed, ie. TX2..
- ...
- PCA9685_I2C_BUSNUM = 1

Da das Donkey Car Framework verschiedene Antriebsarten und Lenkungen unterstützt müssen Sie noch festlegen welche der verschiedenen Möglichkeiten Sie nutzen. Wenn Sie einen klassischen Modellbau Fahrtenregler (english: ESC electronic speed controller) und Lenk-Servo verbaut haben dann müssen Sie jetzt „SERVO_ESC“ auswählen.

- #DRIVETRAIN
-
- DRIVE_TRAIN_TYPE = "SERVO_ESC"

Jetzt schlagen Sie bitte Ihre Notizen ein paar Seiten vorher auf und übertragen die PWM-Werte für die Lenkung und den Fahrtenregler nachfolgend in die myconfig.py ein.

- #STEERING
- STEERING_CHANNEL = 1
- STEERING_LEFT_PWM = 460
- STEERING_RIGHT_PWM = 290
- ...
- #THROTTLE
- THROTTLE_CHANNEL = 0
- THROTTLE_FORWARD_PWM = 500
- THROTTLE_STOPPED_PWM = 370

- THROTTLE_REVERSE_PWM = 220

Da Sie später das Neuronale Netz auf dem Jetson Nano trainieren werden müssen Sie auch für diesen Teil der myconfig.py Datei die Anpassungen vornehmen. Ich trainiere meine Modelle immer mit den default Einstellungen wie folgt.

- #TRAINING
- DEFAULT_MODEL_TYPE = 'linear'
- BATCH_SIZE = 128
- TRAIN_TEST_SPLIT = 0.8
- MAX_EPOCHS = 100
- SHOW_PLOT = True
- VEBLOSE_TRAIN = True
- USE_EARLY_STOP = True
- EARLY_STOP_PATIENCE = 5
- MIN_DELTA = .0005
- PRINT_MODEL_SUMMARY = True
- OPTIMIZER = None
- LEARNING_RATE = 0.001
- LEARNING_RATE_DECAY = 0.0

Zwei Parameter aus dem Abschnitt TRAINING der Konfiguration sollten Sie noch etwas besser verstehen damit Sie, wenn Sie das neuronale Nets später trainieren die Ausgabe während dem Training besser verstehen.

MAX_EPOCHS: Während einer Trainingsepoke wird einmal der gesamte Trainingsdatenbestand dem neuronalen Nets während dem Training präsentiert und das Modell angepasst. Damit legt die Variable MAX_EPOCHS = 100 fest wie viele Trainingsepochen durchlaufen werden bis das Training abgeschlossen ist.

EARLY_STOP_PATIENCE: Während dem Training des neuronalen Netzes wird es häufig vorkommen, dass nach ca. 25 Trainingsepochen die Meldung Early Stopping ausgegeben wird und dass Trainings des neuronalen Netzes abbricht. Die Variable EARLY_STOP_PATIENCE = 5 legt fest, dass das neuronale Nets wenn es fünf Trainingsepochen hintereinander nicht besser wurde das Trainings frühzeitig zu stoppen.

Da ich zu Beginn empfohlen habe die Trainingsdaten immer mit einem Gamepad zu erstellen muss an der jetzt folgenden Stelle in der myconfig.py das Gamepad auch permanent aktiviert werden.

Hinweis: Wenn Sie die Einstellung auf „True“ für die Verwendung des Gamepads setzen wird die Web-Oberfläche des Donkey Car Frameworks nicht mehr gestartet. Möchten Sie die Web-Oberfläche starten und auf diese zugreifen dann müssen Sie die Einstellung wieder auf „False“ setzen. Dann können Sie aber den Joystick bzw. das Gamepad nicht verwenden.

- #JOYSTICK
- USE_JOYSTICK_AS_DEFAULT = True

Es folgen dann noch rund um die Steuerung des Roboter-Autos mit dem Gamepad drei Einstellungen die Sie wie folgt anpassen können. Ich verwende einen PS4 Kontroller mit Sony USB Empfänger und habe entsprechend als Typ ps4 in der Konfiguration ausgewählt. Abhängig von Ihrem Kontroller können Sie hier eine Vielzahl an Typen auswählen (siehe inline Dokumentation). Der EasySMX Controller funktioniert als Typ ps4 ebenfalls sehr gut.

- JOYSTICK_MAX_THROTTLE = 0.8

- AUTO_RECORD_ON_THROTTLE = True
- CONTROLLER_TYPE='ps4'

Speichern Sie jetzt die Änderungen der myconfig.py Datei ab.

Jetzt haben Sie alle Konfigurationen vorgenommen und können das erste Mal das Donkey Car Framework im drive Modus starten. Im aktiven drive Modus steuern Sie Ihr Roboter-Auto manuell mit dem Gamepad und zeichnen die ersten Trainingsdaten auf. Die aufgezeichneten Bilder und Json Dateien finden Sie im Ordner ~/mycar/data und dort in einzelnen Unterordnern die immer mit tub_* im Namen anfangen.

Den drive Modus starten Sie mit dem folgenden Befehl.

Befehl: python ~/mycar/manage.py drive

7.4 ACHTUNG: Fehler im Part camera.py

Es kann passieren, dass das Donkey Car Framework jetzt beim Laden abstürzt. Die Fehlermeldung deutet z. B. darauf hin, dass im Part camery.py etwas nicht richtig funktioniert mit dem Video-Stream der Kamera. Bei der Suche nach der Ursache für den Fehler habe ich festgestellt, dass die initial eingestellte Auflösung für den GStreamer der das Video-Bild verarbeitet physisch zu hoch ist für die von mir verbaute Weitwinkelkamera.

Die von mir gefundene Lösung war die deutlich Reduzierung der Auflösung in der Datei "camera.py" und dort in der Klasse "class CSICamera(BaseCamera):".

Das folgende Bild zeigt die Fehlermeldung die in meinem Fall ausgegeben wurde.

```
GST ARGUS: Cleaning up
CONSUMER: Done Success
GST ARGUS: Done Success
[ WARN:0] global /home/nvidia/host/build_opencv/modules/videoio/src/cap_gstreamer.cpp (886) open OpenCV | GStreamer warning: unable to start pipeline
[ WARN:0] global /home/nvidia/host/build_opencv/modules/videoio/src/cap_gstreamer.cpp (480) isPipelinePlaying OpenCV | GStreamer warning: GStreamer: pipeline have not been created
Exception in thread Thread-1:
Traceback (most recent call last):
  File "/usr/lib/python3.6/threading.py", line 916, in _bootstrap_inner
    self.run()
  File "/usr/lib/python3.6/threading.py", line 864, in run
    self._target(*self._args, **self._kwargs)
  File "/home/donkey/projects/donkeycar/donkeycar/parts/camera.py", line 174, in update
    self.init_camera()
  File "/home/donkey/projects/donkeycar/donkeycar/parts/camera.py", line 169, in init_camera
    self.poll_camera()
  File "/home/donkey/projects/donkeycar/donkeycar/parts/camera.py", line 181, in poll_camera
    self.frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
cv2.error: OpenCV(4.1.1) /home/nvidia/host/build_opencv/modules/imgproc/src/color.cpp:182: error: (-215:Assertion failed) !_src.empty() in function 'cvtColor'
```

Abbildung 7.3: Fehlermeldung camery.px Datei

Die Datei camera.py finden Sie im Ordner ~/projects/donkeycar/donkeycar/parts. In diesen Ordner hatten Sie das GitHub Projekt zuvor herunter geladen.

Öffnen Sie jetzt die Datei camery.py und suchen Sie nach der Klasse "class CSICamera(BaseCamera)": Dort finden Sie die folgende Zeile die Sie bei den Parametern „capture_width“ und „capture_height“ abändern müssen.

- Original Zeile:
- def __init__(self, image_w=160, image_h=120, image_d=3, capture_width=3280, capture_height=2464, framerate=60, gstreamer_flip=0):

- Angepasste Zeile:
- def __init__(self, image_w=160, image_h=120, image_d=3, capture_width=160, capture_height=120, framerate=60, gstreamer_flip=0):

Nach Sie diese Anpassung vorgenommen haben speichern sie die camery.py Datei ab und starten das Donkey Car Framework erneut. Es sollte jetzt der Fehler nicht mehr auftreten.

Den drive Modus starten Sie mit dem folgenden Befehl erneut und sollten jetzt in der Lage sein das Donkey Car zu steuern.

Befehl: python ~/mycar/manage.py drive

Hinweis: Es kann sein, dass Sie zwar mit dem PS4 Controller das Donkey Car lenken können aber Sie es nicht beschleunigen können. Dann lesen Sie bitte im Kapitel „Gamepad Konfiguration“ nach welche Einstellungen Sie anpassen können damit Ihr PS4 Controller oder ähnliches Modell funktioniert. Hier müssen Sie je nach Modell sich etwas einlesen und testen.

Hat soweit alles funktioniert also Sie können mit dem Gamepad lenken und beschleunigen dann stellen Sie jetzt das Donkey Car auf den Boden. Drehen Sie jetzt ein paar Runden und machen Sie sich mit der Steuerung des Roboter-Autos über das Gamepad vertraut.

7.5 Kamera mit fehlerhafter Farbdarstellung

Es kann passieren, dass die verwendete Kamera nicht exakt von dem Betriebssystem und den dort vorhandenen Treibern bzw. Konfigurationen unterstützt wird. So verwende ich in meinen Donkey Car Modellen eine Weitwinkelkamera bei der die Ränder des Bildes links und rechts rot eingefärbt sind.

Es gibt von den verschiedenen Kameraherstellern auch immer wieder entsprechende Konfigurationsdateien, die die Parameter für das Betriebssystem so setzen, dass das Bild deutlich besser aussieht.

Das nachfolgende Bild zeigt das Problem ganz gut. Auf der linken Seite ist ein Bild der Kamera zu sehen mit den falsch dargestellten Farben. Auf der rechten Seite des Bildes ist das Kamera Bild zu sehen nachdem eine für die Kamera speziell erstellte *.isp Datei eingespielt wurde.

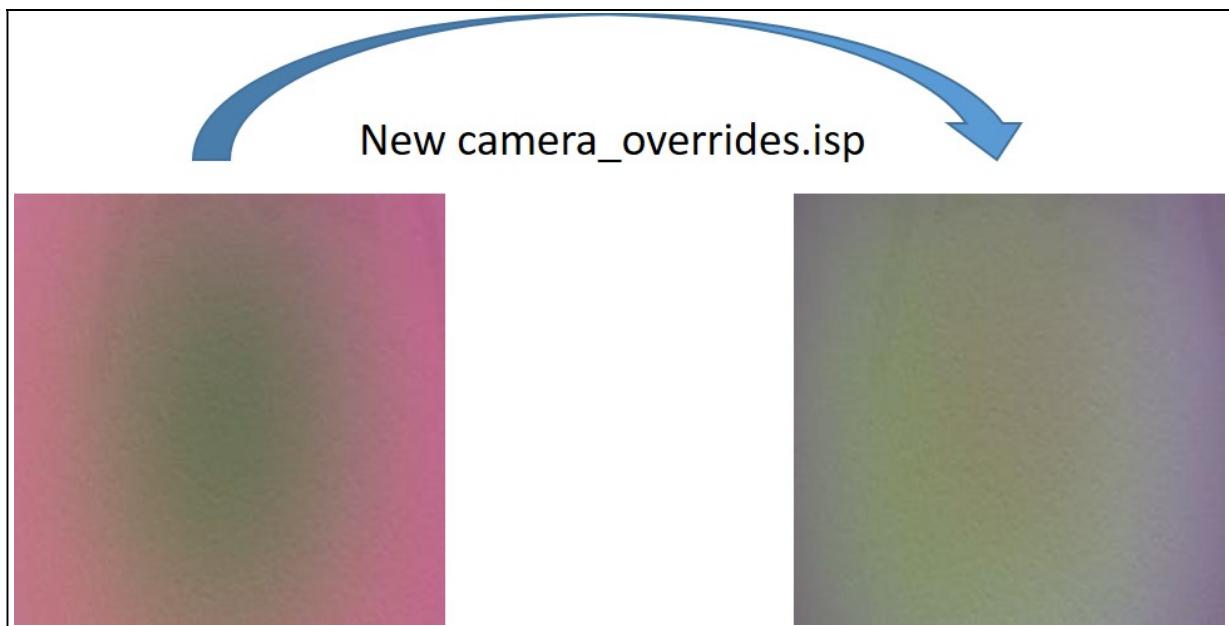


Abbildung 7.4: camery_overrides.isp Konfiguration

Die von mir verwendete camera_overrides.isp Datei gibt es auf meinem Blog unter der folgenden URL zum Download.

Download: <https://custom-build-robots.com/jetson-nano-download-de>

Nachdem Sie die *.zip Datei herunter geladen und entpackt haben geht es weiter. Erst einmal müssen vor der Verwendung der neuen camera_overrides.isp Datei zwei Dateien gelöscht werden. Andernfalls klappt es nicht mit der richtigen Darstellung der Farben.

Hinweis: Denken Sie daran vor dem Löschen oder überschreiben der Dateien jeweils eine Sicherung in z. B. einem Backup Ordner anzulegen.

Führen Sie dazu die beiden folgenden Befehle aus:

Befehl: sudo rm /var/nvidia/nvcam/settings/nvcam_cache_*

Befehl: sudo rm /var/nvidia/nvcam/settings/serial_no_*

Jetzt wechseln Sie in den Ordner im Terminal Fenster in dem Sie die zuvor heruntergeladene und entpackte camera_overrides.isp Datei kopiert haben.

Anschließend führen Sie den folgenden Befehl aus der die camera_overrides.isp Datei in den Ordner „/var/nvidia/nvcam/settings/“ kopiert.

Befehl: sudo cp camera_overrides.isp /var/nvidia/nvcam/settings/

Nachdem Sie die Datei „camera_overrides.isp“ kopiert haben müssen noch die Dateirechte angepasst werden. Das machen Sie mit dem folgenden Befehl:

Befehl: sudo chmod 664 /var/nvidia/nvcam/settings/camera_overrides.isp

Anschließend muss die Datei camera_overrides.isp dem user root zugeordnet werden damit dieser beim Starten des Jetson Nano auf die Datei zugreifen kann. Daher führen Sie jetzt den nachfolgenden Befehl aus.

Befehl: sudo chown root:root /var/nvidia/nvcam/settings/camera_overrides.isp

Dem neuronalen Netz wird die nicht ganz passende Farbe in den Bildern egal sein und genauso gut funktionieren wie mit der korrigierten Farbdarstellung. Aber wenn Sie als Mensch später einmal Videos aus den tausenden Bilddateien für Qualitätsprüfungen der Trainingsdaten erstellen ist es schon schön die richtigen Farben zu sehen.

7.6 Fahren üben

Üben Sie jetzt das Fahren mit Ihrem Roboter-Auto zusammen mit dem Gamepad. Eventuell fällt Ihnen auf, dass Ihr Fahrwerk keinen sauberen Geradeauslauf hat und immer in eine Richtung zieht. Dann korrigieren Sie das durch nachbessern und ausprobieren bei den PWM-Werten für die Lenkung in der myconfig.py Datei.

Hinweis: Für den Fall das bei der Lenkung links und rechts vertauscht sein sollten ist die einfachste Lösung in der myconfig.py Datei den Lenkeinschlag zu vertauschen. Mehr dazu im Kapitel „Führen Sie ihren Autopiloten das erste Mal aus“.

Für das Aufzeichnen fehlerfreier Trainingsdaten für das spätere Training des Neuronalen Netzes ist es sehr wichtig das Sie mit der Lenkung, der Geschwindigkeit kurz gesagt mit der Steuerung ihres Roboter-Autos vertraut sind.

7.7 OLED Display aktivieren

Vielleicht haben Sie, wie von mir empfohlen, ein OLED Display an Ihrem Roboter-Auto verbaut. Dann möchte ich Ihnen in diesem Abschnitt erklären wie Sie die notwendige Software installieren um das OLED Display in Betrieb nehmen zu können. Adafruit bietet eine sehr gute Bibliothek an die zusammen mit dem Jetson Nano und dem I²C Bus genutzt werden kann.

Zuerst müssen Sie von GitHub die Software herunter laden. Dazu führen Sie die beiden folgenden Befehle aus.

Befehl: cd ~

Befehl: git clone https://github.com/adafruit/Adafruit_Python_SSD1306.git

Jetzt wechseln Sie in den Ordner „Adafruit_Python_SSD1306“.

Befehl: cd Adafruit_Python_SSD1306

Anschließend installieren Sie die Software auf Ihrem Jetson Nano mit dem nachfolgenden Befehl.

Befehl: sudo python3 setup.py install

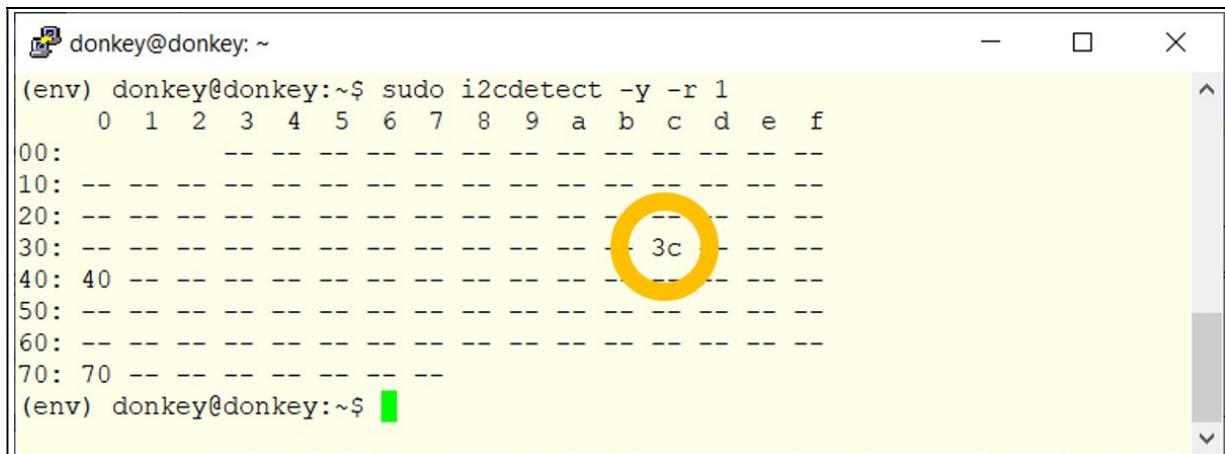
Es könnte sein, dass noch die image Bibliothek fehlt die benötigt wird damit die Anzeige auf dem OLED Display auch wirklich funktioniert. Diese installieren Sie mit dem folgenden Befehl.

Befehl: pip install image

Starten Sie jetzt den Jetson Nano neu und führen Sie anschließend den folgenden Befehl aus, um alle angeschlossenen Geräte am I²C Bus angezeigt zu bekommen.

Befehl: sudo i2cdetect -y -r 1

Es sollte der Servo-Kontroller mit den Adressen 40/70 und das OLED Display mit der Adresse 3c zu sehen sein. Bei mir sieht die Ausgabe wie folgt aus und in Orange hervorgehoben ist das OLED Display mit seiner I²C Adresse zu sehen.



```
donkey@donkey: ~
(env) donkey@donkey:~$ sudo i2cdetect -y -r 1
      0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:   --
10:   --
20:   --
30:   --
40: 40  --
50:   --
60:   --
70: 70  --
(env) donkey@donkey:~$
```

Abbildung 7.5: I²C Bus Anzeige

Da die kleinen Beispielprogramme die Adafruit mitliefert für den Jetson Nano noch nicht angepasst sind stelle ich Ihnen das angepasste stats_nano.py Programm zur Verfügung ohne auf die Details der Anpassung hier einzugehen. Im Quellcode des Python Programmes stats_nano.py können Sie die kleinen Anpassungen bei Interesse im Vergleich zum Original stats.py Programm von Adafruit nachvollziehen.

Laden Sie die entsprechende *.zip Datei über den folgenden Link herunter und entpacken Sie diese auf Ihrem Jetson Nano.

URL: <https://custom-build-robots.com/jetson-nano-download-de>

Die beiden Programm „start-oled.sh“ und „stats.py“ aus der zuvor herunter geladenen OLED_Display_Software.zip Datei legen Sie jeweils im Ordner ~/projects ab in dem Sie zuvor bereits das Donkey Car Framework herunter geladen haben.

Damit das Python Programm für die Displayanzeige immer automatisch gestartet wird müssen Sie noch das Skript mit dem Dateinamen startoled.sh in der /etc/crontab hinterlegen. Die von mir verwendete crontab liegt als Beispiel der „OLED_Display_Software.zip“ Datei bei.

Hinweis: Achten Sie unbedingt darauf, dass die verwendeten Pfade im Skript startoled.sh und nachfolgende in der crontab zu Ihrer Installation passen. Andernfalls wird es zu Fehlern kommen.

Nach dem Sie das Skript im projects Ordner gespeichert haben müssen Sie noch die Dateirechte ändern damit dieses von der /etc/crontab ausgeführt werden kann.

Befehl: sudo chmod 777 startoled.sh

Das Skript startoled.sh selber ist sehr kurz gehalten und startet das OLED Anzeigeprogramm ebenfalls in einer virtuellen Session. Nachfolgend ist der Aufbau der startoled.sh Skriptdatei zu sehen:

```
#!/bin/bash
source ~/env/bin/activate
python /home/donkey/projects/stats.py >> /home/donkey/projects/oled-sh.log 2>&1 &
```

7.7.1 Das Skript startoled.sh selber anlegen

Möchten Sie das Skript startoled.sh selber anlegen dann führen Sie z. B. den folgenden Befehl aus um das Skript mit dem Text-Editor nano anzulegen.

Befehl: nano start-oled.sh

Nach dem Sie das Skript im home-Ordner gespeichert haben müssen Sie noch die Dateirechte ändern damit dieses von der /etc/crontab aufgeführt werden kann.

Befehl: sudo chmod 777 start-oled.sh

Sollte es innerhalb dem Skript start-oled.sh beim Ausführen dessen zu einem Fehler kommen dann schreibt dieses eine Log-Datei mit dem Namen *oled-sh.log* in den Ordner /home/donkey/ bzw. wenn Sie den Pfad angepasst haben in den von Ihnen angegebenen Ordner.

Jetzt müssen Sie noch die folgende Zeile in der /etc/crontab eintragen um das Skript start-oled.sh bei jedem Neustart automatisch auszuführen. Auch hier bitte wieder die Pfade prüfen.

```
python /home/donkey/projects/stats.py >> /home/donkey/projects/oled-sh.log 2>&1 &
```

Wenn Sie jetzt den Jetson Nano neu starten, dann sollten Sie auf dem OLED Display eine Anzeige erhalten. Sollte nichts zu sehen sein dann können Sie im Ordner projects die Datei *oled-sh.log* bzw. *oled-crontab.log* sich anschauen. Diese beiden Log-Dateien werden immer geschrieben wenn das Skript über de crontab ausgeführt wird.

Die /etc/crontab sollte jetzt bei Ihnen wie folgt aussehen.

```
donkey@donkey: /etc
GNU nano 2.9.3          crontab

# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab`
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

@reboot donkey /home/donkey/projects/start-oled.sh >> /home/donkey/projects/oled-crontab.log 2>&1 &

# m h dom mon dow user  command
17 *      * * *    root    cd / && run-parts --report /etc/cron.hourly
25 6      * * *    root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6      * * 7    root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6      1 * *    root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
```

Abbildung 7.6: OLED-Display Anpassung /etc/crontab

7.8 Was Sie bis hier erreicht habe.

Sie haben alle notwendigen Programme und Bibliotheken installiert, die Lenkung und den Fahrtenregler konfiguriert und Sie sind die ersten Runden bereits gefahren. Das OLED Display zeigt die IP-Adresse und weitere Informationen an und Sie sind jetzt gut gerüstet für den nächsten Schritt das Trainieren des Neuronalen Netzes mit Ihren eigenen Trainingsdaten.

Ich freue mich sehr über Ihr Feedback zu diesem Kapitel. Schreiben Sie mir Ihre Gedanken, Fragen und Vorschläge an die folgende E-Mail Adresse: ebook@custom-build-robots.com

8 Vorbereitung für die Trainingsdatenaufzeichnung und Trainings des Autopiloten

Sie haben die ersten Runden mit Ihrem Roboter-Auto gedreht und brennen darauf zu erfahren wie Sie das Neuronale Netz trainieren und autonom fahren können. Dann los hinein in das Vergnügen der autonom fahrenden Modell-Autos.

Im vorherigen Kapitel haben Sie das Roboter-Auto aufgebaut, die Software installiert und sind bereits ein paar Runden gefahren zum Testen ob alles funktioniert. Sie sind mit der Steuerung und dem Fahrverhalten ihres Roboter-Autos vertraut und werden jetzt lernen wie Sie Trainingsdaten aufzeichnen. Aber bevor es jetzt wirklich losgeht kommt noch ein kurzer Abschnitt mit Empfehlungen zum Streckenbau sowie dem weiteren Ablauf bis zum fertig trainierten Neuronalen Netz.

8.1 Strecken für das Roboter-Auto bauen

Die Strecken für die Aufzeichnung der Trainingsdaten können ganz individuell aufgebaut werden. Diese müsse nicht genormt sein solange man nicht für einen Wettbewerb trainiert bei dem es um jede Sekunde geht. Recht einfach und schnell können Sie eine Strecke mit günstigen Klebeband aus dem Malerbedarf auf den Boden zaubern.

Hinweis: Aber passen Sie auf, dass es Ihnen nicht wie einem Kollegen von mir ergeht der voller Begeisterung am Wochenende daheim auf dem frisch verlegten Parkett ein Klebeband aufgeklebt hatte. Am Abend als er das Klebeband vom Boden abriß entfernte er damit nicht nur das Klebeband sondern auch Teile der Parket Versiegelung gleich mit. Der große Vorteil dieser wirklich dumm gelaufenen Aktion war, dass die Fahrbahn damit permanent zur Verfügung stand bis zum nächsten Schleifen und Lackieren.

Der Nachteil mit dem Klebeband sind die Kosten und die große Menge Abfall die jedes Mal anfällt wenn der Track wieder entfernt wird. Das hat mich dazu bewegt etwas nachhaltiger zu denken um die Umwelt zu schonen und die Kosten. Auch hatte ich keine Lust mehr auf den Knie Klebeband verkleben zu müssen.

Wenn Sie mit einem Band oder dicken Seil arbeiten reicht eine einfache Linie aus die sie aufkleben oder eben zwei Bahnen im Abstand von ca. 50cm so dass Ihr Roboter-Auto gut zwischen den Kleebahnen fahren kann. Sie sollten den Radius der Kurven nicht zu eng machen damit Sie gut durch die Kurven kommen.

Ein Streckenverlauf den ich von seinem Prinzip immer wieder verwende sieht wie folgt aus.

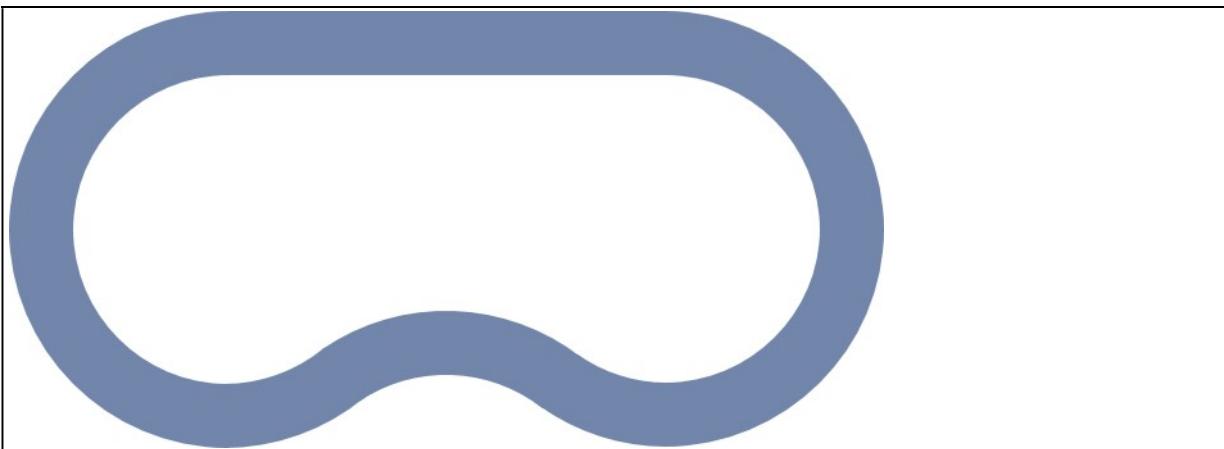


Abbildung 8.1: Idee einer Rennstrecke

Dieser Bahnverlauf vermeidet die Grundlegenden Fehler wie z. B. dass nur immer ein Lenkeinschlag bei einer Fahrrichtung überwiegt.

Ich habe mir aus Gründen der Kosten und des Abfalles für den Streckenbau 100m Nahtband als B-Ware mit einer Breite von ca. 3 cm bis 5 cm im Internet gekauft. Wobei ich die 3cm Variante empfehle da diese weniger Falten wirft bei der Modellierung von Kurven. Auf dieses Nahtband habe ich wiederum Beleiband mit dem eigentlich Vorhänge beschwert werden aufgenäht. So kann ich schöne Strecke mit Kurven und Geraden sehr schnell im Stehen verlegen durch Abwickeln des Bandes von einer großen Rolle. Durch das Bleiband liegt das Nahtband schön am Boden auf ohne große Falten zu werfen bei der 3cm Variante. Wenn das Roboter-Auto über das Band fährt verzieht es sich nicht sondern bleibt liegen. Im freien kann es durch den Wind auch nicht verweht werden. Eine automatische Mechanik zum Aufwickeln mit kleinem Elektromotor habe ich mir ebenfalls gebaut um meinen Rücken zu schonen.



Abbildung 8.2: Nähen der Rennstreckenmarkierung

Eine Strecke kann aber auch ganz einfach z. B. im Freien mit Blättern beschwert mit Kastanien, Post-IT Papiere im Büro oder Kaffeebecher gebaut werden. Wichtig ist das sich der Streckenverlauf gut von der Umgebung abhebt damit Sie als Mensch das Roboter-Auto ohne Fahrfehler um die Strecke steuern können und so fehlerfreie Trainingsdaten erhalten. Später wird der Autopilot nach dem er trainiert wurde dem Streckenverlauf folgen können. Dabei muss es nicht sein das sich das fertig trainierte neuronale Netz auch wirklich an dem Nahtband oder Kaffeebechern orientiert. Denn auch ein neuronales Netz ist „faul“ oder optimiert und sucht sich dank Deep Learning eigene Wegmarken an denen es sich orientiert.

In einem Großraumbüro mit Deckenbeleuchtung ist es auch möglich die Kamera des Roboter-Autos senkrecht nach oben auszurichten. Wichtig ist, dass die Decke bei diesem Aufbau ein Muster aufweist das nicht zu symmetrisch ist sondern etwas Abwechslung bietet. Die Streckenmarkierung am Boden ist dann nur für den Menschen wichtig der das Roboter-Auto auf Kurs hält bei der Aufzeichnung der Trainingsdaten. Anschließend fährt das Roboter-Auto die Strecke anhand der Position der Lampen und Muster in der Decke ab. Die Streckenmarkierung am Boden spielt dann keine Rolle für den Autopiloten. Schalten Sie die Lampen aus wird das Neuronale Netz nicht mehr das Auto steuern können da die Markierung an der Decke sich verändert hat bzw. komplett fehlt.

Generell spielen Veränderungen im Umfeld der Strecke eine große Rolle für den Erfolg oder Misserfolg. So spielt eine wechselnde Beleuchtung durch z. B. die Sonne die in den Raum mit der Strecke einfällt, also Veränderungen in der direkten Umwelt der Strecke, eine entscheidende Rolle. So empfehle ich immer die Trainingsdaten auch bei unterschiedlichen Lichtverhältnissen aufzuzeichnen um den später trainierten Autopiloten so robust wie möglich gegen sich verändernde Umweltfaktoren zu machen. Wenn Sie z. B. zu Meetups mit Ihrem Roboter-Auto gehen oder auf einen Wettbewerb denken Sie auch daran, dass es dann viele Zuschauer gibt die das Setup rund um die Strecke dynamisch verändern.

Die Strecke sollte nicht nur ein Kreis sein sondern Kurven, Geraden und Schwünge enthalten. Denn fahren Sie nur im Kreis so lernt das Neuronale Netz nicht richtig zu lenken da Sie sehr wahrscheinlich immer mit annähernd dem gleichen Lenkeinschlag die Strecke abfahren werden.

In einem späteren Kapitel werde ich Ihnen ein Skript an die Hand geben welches die Bilder z. B. in ihrer Helligkeit verändert und zusammen mit den *.json Dateien abspeichert. So erhalten Sie schneller mehr Trainingsdaten um den Autopiloten so robust wie möglich zu machen.

Ich hatte einmal mit meinem Nahtband eine Strecke im Garten auf der Wiese aufgebaut. Als ich das Nahtband am Abend wieder aufgewickelt hatte startete ich noch einmal den autonomen Modus von meinem Roboter-Auto. Es folgte immer noch dem eigentlich nicht mehr ersichtlichen Streckenverlauf. Aus dieser Erfahrung schließe ich, dass der trainierte Autopilot z. B. das Muster und die Schattierungen des Rasens als Bezug für den Lenkeinschlag und die Geschwindigkeit herangezogen hatte.

Sie können auch durch das Aufstellen von z. B. Kaffeebechern dem Autopiloten das Ausweichen vor Hindernissen beibringen. Dazu verändern Sie die Position der Kaffeebecher auf der Strecke immer wieder und zeichnen erneut Trainingsdaten auf. Machen Sie das ca. 10x pro Kaffeebecher und zeichnen Sie jeweils drei Runden auf. Trainieren Sie mit diesen Daten das Neuronale Netz dann hat es sehr wahrscheinlich gelernt Kaffeebechern an beliebiger Stelle auf der Rennstrecke auszuweichen. Sie sehen schon bei so einer kleinen Aufgabe wie dem Ausweichen von Kaffeebechern müssen Sie einen sehr großen Aufwand bei der Erzeugung der Trainingsdaten treiben. Daher gilt das strukturierte und gelabelte Daten das Öl unserer zukünftigen Wirtschaft sind.

8.2 Gute Trainingsdaten aufzeichnen

Es ist schwer zu sagen was gute und was schlechte Trainingsdaten sind. Das hängt davon ab was Sie genau vorhaben. Eines wird aber sicher passieren, dass Sie Ihren Trainingsdaten Ihren menschlichen BIAs mitgeben also Ihr individuelles Verhalten. Fahren Sie sehr konservativ um die Rennstrecke wird es das Neuronale Netz später auch so machen. Fahren Sie bei allen aufgezeichneten Trainingsdaten mit einer konstanten Geschwindigkeit und beschleunigen oder bremsen nicht dann kann es passieren, dass Ihr Autopilot nicht gelernt hat zu beschleunigen und beim Start nicht losfährt. Schieben Sie das Auto in solch einem Fall mit z. B. dem Fuß kurz an, dann fährt es doch plötzlich autonom ganz schön seine Runde mit konstanter sich nicht verändernder Geschwindigkeit.

Daher empfehle ich immer so viel Abwechslung in die aufgezeichneten Trainingsdaten wie möglich zu bringen. Beschleunigen und bremsen Sie immer wieder während Sie die Trainingsdaten erstellen. Fahren Sie über die Streckenmarkierung hinaus und gleich wieder zurück damit das Neuronale Netz lernt was es in solch einem Fall machen soll. Fahren sie vor allem die Strecke nicht nur in eine Richtung ab sondern auch in die andere. Fahren sie immer nur z. B. im Uhrzeigersinn den Kurs ab dann kann der Pilot später nicht autonom gegen den Uhrzeigersinn die Strecke abfahren. Aber probieren Sie das doch einfach mal selber aus.

Mit diesen Tipps aus meiner Erfahrung heraus sollten Sie für das Training Ihres Autopiloten gute Trainingsdaten erzeugen können mit denen Ihr Neuronales Netz bzw. der Autopilot das autonome Fahren lernt.

8.3 Gamepad Tastenbelegung

Verwenden Sie wie empfohlen ein Gamepad um das Roboter-Auto zu steuern, dann müssen Sie noch herausfinden wie die Tastenbelegung genau für Ihr Gamepad konfiguriert ist. Sie haben bereits die Konfiguration in der ~/mycar/myconfig.py Datei kennen gelernt und dort die verschiedenen unterstützten Gamepads gesehen.

Interessant ist mit welchen Tasten Sie den Betriebsmodus im autonomen Betrieb ihres Roboter-Autos wechseln können und welche Funktionen es daneben noch gibt die Ihnen die eine oder andere Erleichterung bei der Bedienung der Donkey Car Frameworks bieten. Sie können selber die Tastenbelegung in der controller.py Datei nachsehen und auch dort für Ihren Kontroller verändern wenn Sie möchten.

Eine Genaue Beschreibung wie das funktioniert finden Sie im Kapitel 3.3.1.

8.4 Alternative: Die Web-Steuerung

Verwenden Sie kein Gamepad sondern möchten das Roboter-Auto erste einmal über die Web-Oberfläche steuern und bedienen dann müssen Sie wieder das Donkey Car Framework mit dem drive Parameter ausführen.

Hinweis: Beachten Sie aber, damit das Web-Interface gestartet wird müssen Sie in der ~/mycar/myconfig.py die Einstellung USE_JOYSTICK_AS_DEFAULT für den Joystick auf „False“ setzen.

Starten Sie jetzt das Donkey Car Framework mit dem folgenden Befehl.

Befehl: python ~/mycar/manage.py drive

Nachdem das Framework gestartet ist rufen Sie die folgende URL im Browser auf.

URL: <IP-Adresse Roboter-Auto>:8887

Das Webinterface das ein live Videobild aus dem Roboter-Auto heraus zeigt sieht wie im nachfolgenden Bild gezeigt aus.

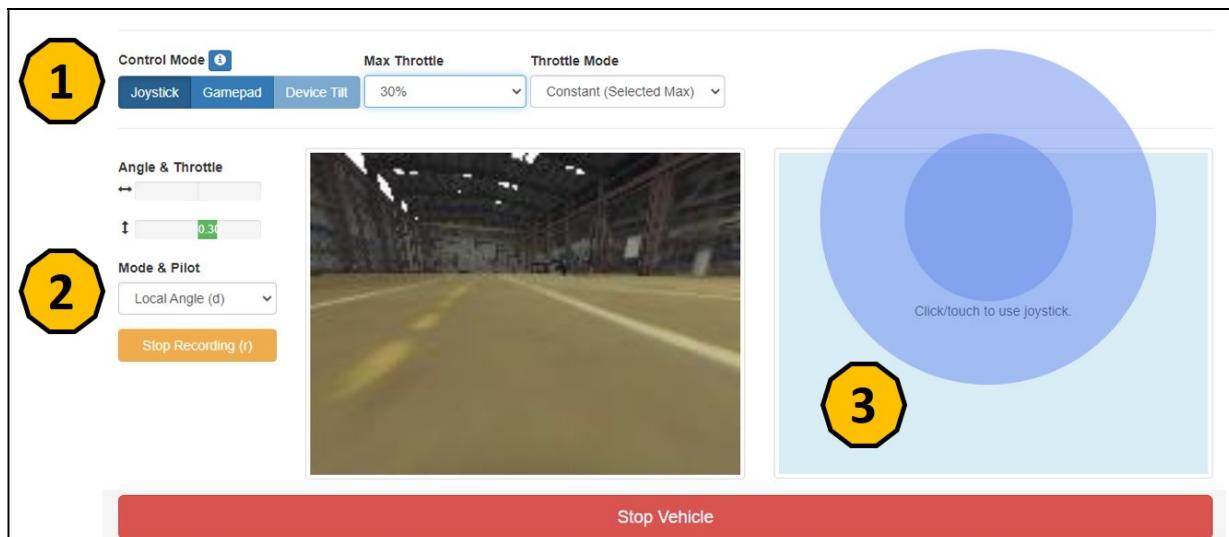


Abbildung 8.3: Donkey Car Web-Interface

Auf der Abbildung sind die einzelnen Funktionen gut zusehen wie die verschiedenen Control Modi mit dem Joystick, Gamepad oder durch kippen z. B. eines Smartphones (Device Tilt).

Die drei wichtigsten Funktionen wurden im Bild nummeriert.

1. Control Mode

Hier können Sie den Modus wechseln wie Sie ihr Roboter-Auto steuern möchten. Verwenden Sie einen aktuellen Chrome Browser auf Ihrem PC und haben einen Joystick oder ein Gamepad an Ihrem PC angeschlossen so können Sie mit diesem jetzt das Roboter-Auto über den Browser fernsteuern. Mit der „Device Tilt“ Funktion können Sie mit Ihrem Smartphone und dessen eingebauten Lagesensor das Roboter-Auto durch entsprechendes kippen des Smartphones steuern.

2. Mode & Pilot

Mit der Dropdown Box „Select Mode“ können Sie zwischen den drei Modi User, Local Pilot und Local Angle wechseln. Später werde ich auf die Bedeutung der einzelnen Modi noch eingehen.

3. Click/touch to use joystick.

Rechts sehen Sie ein Feld innerhalb dem Sie mit gedrückter linken Maustaste am PC das Auto steuern können. Diese Funktion funktioniert auf einem Smartphone oder Tablet ebenfalls mit dem Finger ganz gut. Eventuell ist das auch eher ein Generationen Thema.

Weiter haben Sie noch auf der Web-Oberfläche die Funktion Max Throttle mit einer dazu gehörigen Dropdown Box. Hier können Sie die Geschwindigkeit die das Roboter-Auto maximal fahren soll festlegen. Mit dem Dropdown Box Throttle Mode legen Sie fest ob Sie selber (im User Mode) die Geschwindigkeit steuern oder ob das Roboter-Auto mit der unter Max Throttle gewählten Geschwindigkeit konstant schnell fahren soll.

8.5 Trainingsdaten aufzeichnen

Jetzt haben Sie viel Theorie gelesen und brennen sicher darauf die ersten Trainingsdaten aufzuzeichnen. Bauen Sie Ihre Strecke auf und drehen Sie ein paar Übungsrunden. Zuvor löschen Sie alle Test und Spieldaten im Verzeichnis `~/mycar/data` die dort aufgezeichnet wurden als Sie noch das Fahren geübt habe.

Der Befehl um den drive Modus des Roboter-Autos zu starten ist wie folgt.

Befehl: `python ~/mycar/manage.py drive`

Fahren Sie jetzt ca. 7 Runden im Uhrzeigersinn und 7 Runden gegen den Uhrzeigersinn auf Ihrem Racetrack ab. Wichtig ist, dass Sie ca. 8.000 Datensätze erzeugt haben. Ein Datensatz besteht immer aus einem Bild und der dazugehörigen JSON Datei. Also in Summe sollten Sie dann ca. 16.000 Dateien mit einer `meta.json` Datei vorliegen haben.

Die aufgezeichneten Trainingsdaten bestehend aus Bildern und mindestens einer zu dem Lauf dazugehörigen Katalog Datei.

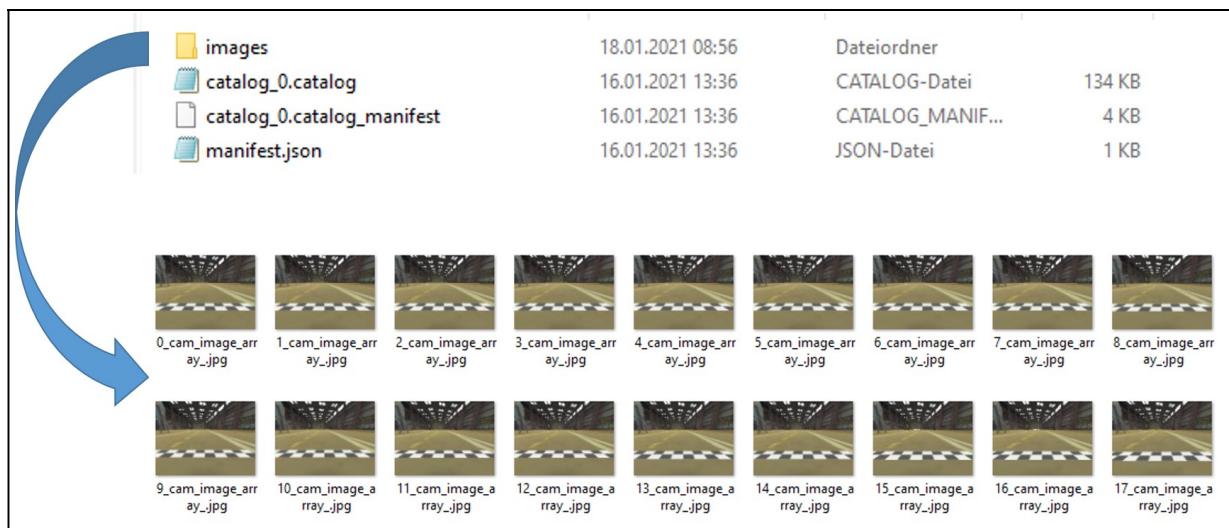


Abbildung 8.4: Donkey Car Trainingsdaten

Wenn Sie keine groben Fahrfehler gemacht haben bei der Erzeugung der Trainingsdaten und nur hin und wieder über die Streckenmarkierung hinaus gefahren sind dann müssen Sie nach meinen Erfahrungen keine Daten bereinigen. Dennoch möchte ich Ihnen im folgenden Abschnitt ein kleines Skript an die Hand geben mit dem Sie Ihre Trainingsdaten einfacher qualitätssichern können.

8.5.1 Daten Qualitätssicherung

Mit dem Skript `auto_ffmpeg.sh` können Sie aus den einzelnen Bildern in den Tub-Ordnern ein Video zur Qualitätssicherung (QS) erstellen. Mit dem Video können Sie schneller eine QS Ihrer aufgezeichneten Trainingsdaten durchführen als wenn Sie versuchen tausende Bilder einzeln zu betrachten.

Das Skript erstellt aus allen Unterordnern im Verzeichnis `~/mycar/data/` und den dort gespeicherten Bildern ein Video mit dem Dateinamen des jeweiligen Ordners. Die Videos werden im Verzeichnis `~/mycar/data/` abgelegt.

Das Skript erhalten Sie auf der folgenden Seite zum Download. Legen Sie dieses im Ordner ~/mycar/ ab.

Download: <https://custom-build-robots.com/jetson-nano-download-de>

Eventuell müssen Sie die Dateirechte des Skriptes ändern damit Sie dieses ausführen können.

Befehl: sudo chmod 777 ~/mycar/ auto_ffmpeg.sh

Das Skript um die Videos zu erzeugen starten Sie mit dem folgenden Befehl:

Befehl: sh auto_ffmpeg.sh

Nach dem das Skript durchgelaufen ist sollten Sie die Videos z. B. mit dem VLC Player abspielen können. Übertragen Sie dazu die Videos am besten auf Ihren PC mit z. B. WinSCP einer freien SFTP Client-Software wenn Sie Windows Nutzer sind.

Trainingsdaten die grobe Fahrfehler aufweisen wie Unfälle oder langes fahren abseits der Streckenmarkierung löschen Sie. Sind Sie mit Ihren verbleibenden Trainingsdaten zufrieden und haben Sie ca. 8.000 Datensätze guter Trainingsdaten im Ordner ~/mycar/data/ dann können Sie jetzt das Neuronale Netz trainieren. Wie genau das funktioniert erfahren Sie im nachfolgenden Kapitel.

8.6 Trainieren Sie ihren Autopiloten

Sie haben jetzt im Ordner ~/mycar/data/ mindestens einen Tub-Ordner der Ihre Trainingsdaten enthält. Alle Ordner mit Trainingsdaten oder Teile davon die Ihre Qualitätssicherung nicht bestanden haben wurden von Ihnen gelöscht.

Hinweis: Denken Sie bitte daran jetzt den Jetson Nano mit dem externen Netzteil zu betreiben damit die maximale GPU Leistung für das Training des Neuronalen Netzes zur Verfügung steht. Andernfalls kann das Training deutlich länger dauern.

Der Jetson Nano ist nicht ideal für das Training von neuronalen Netzen. Hier wäre ein PC mit entsprechender GPU oder starken CPU und einem Arbeitsspeicher von größer 12GB deutlich besser geeignet. Erschwerend dazu kommt noch das nicht alle Bibliotheken für den Jetson Nano zur Verfügung stehen die das Donkey Car Framework bzw. die Community aktuell nutzt.

Daher müssen Sie noch von meiner Webseite einmal die train_nano.py Datei herunter laden und die training_nano.py Datei. Auf der folgenden Seite finden Sie die entsprechende **Nano_training.zip** Datei.

URL: <https://custom-build-robots.com/donkey-car/donkey-car-framework-download-seite/13982>

Entpacken Sie die ZIP Datei z. B. im Home Ordner ihres Users.

train_nano.py

Die Datei **train_nano.py** legen Sie in den Ordner ~/mycar ab.

training_nano.py

Die Datei **training_nano.py** legen Sie in den Ordner ~/projects/donkeycar/donkeycar/pipeline ab.

Wenn Sie beide Dateien entsprechend abgelegt haben dann sollte es mit der für den Jetson Nano angepassten Training-Pipeline klappen das eigene neuronale Netz zu trainieren.

Sie starten mit dem folgenden Befehl das Training des Autopiloten. Für das Training werden mit diesem Befehl alle Tub-Ordner im Verzeichnis ~/mycar/data/ herangezogen. Der am Ende des Trainings erstellte Autopilot erhält den Namen mypilot.h5 und liegt im Pfad ~/mycar/models/.

Befehl: python train_nano.py --tubs=data/tub_2_21-01-16/,data/tub_1_21-01-16/,data/tub_3_21-01-16/ --model models/mypilot.h5

Sie sollten jetzt folgende Ausgabe in der Konsole sehen wenn das Training gestartet ist.

```
(env) donkey@donkey:~$ cd mycar/
(env) donkey@donkey:~/mycar$ python train_nano.py --tubs=data/tub_test/,data/tub_2_21-01-16/,data/tub_1_21-01-16/,data/tub_3_21-01-16/ --model models/big.h5

using donkey v1.0.0-dev ...
2021-01-18 07:41:56.695689: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library libcudart.so.10.2
loading config file: /home/donkey/mycar/config.py
loading personal config over-rides from myconfig.py
"get_model_by_type" model Type is: linear
Created KerasLinear
2021-01-18 07:42:03.829779: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library libcuda.so.1
2021-01-18 07:42:03.849460: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:948] ARM64 does not support NUMA - returning NUMA node zero
2021-01-18 07:42:03.849548: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1561] Found device 0 with properties:
pcibusID: 0000:00:00:00 name: NVIDIA Tegra X1 computeCapability: 5.3
coreClock: 0.9216GHz coreCount: 1 deviceMemorySize: 3.86GB deviceMemoryBandwidth: 194.55MiB/s
2021-01-18 07:42:03.849622: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library libcudart.so.10.2
2021-01-18 07:42:03.865919: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library libcUBLAS.so.10
2021-01-18 07:42:03.881960: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library libcuFFT.so.10
2021-01-18 07:42:03.888081: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library libcurand.so.10
2021-01-18 07:42:03.904888: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library libcusolver.so.10
2021-01-18 07:42:03.915645: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library libcusparse.so.10
2021-01-18 07:42:03.917765: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library libcudnn.so.
```

Abbildung 8.5: Jetson Nano TensorFlow Training

Verbessert sich der Autopilot während fünf Trainingsepochen in Folge nicht dann wird das Training automatisch abgebrochen und eine Meldung „early stopping“ wird ausgegeben. Sie haben den Parameter `EARLY_STOP_PATIENCE` = 5 bereits kennen gelernt und können diesen z. B. auf 7 hoch setzen und ausprobieren ob Sie so zu besseren Ergebnissen ihres Autopiloten kommen. Meiner Erfahrung nach spielen aber die Trainingsdaten und hier wiederum die Anzahl der Datensätze sowie die Vielfältigkeit in den Daten eine entscheidende Rolle einen besseren Autopiloten zu erhalten.

Am Ende des Trainings sollten Sie eine Ausgabe wie in etwa diese angezeigt bekommen.

```
donkey@donkey:~  
38/38 [=====] - 30s 800ms/step - loss: 0.0559 - n_outputs0_loss: 0.0525 - n_outputs1_loss: 0.0034 -  
val_loss: 0.0543 - val_n_outputs0_loss: 0.0521 - val_n_outputs1_loss: 0.0021  
Epoch 19/100  
9/9 [=====] - 4s 403ms/step - loss: 0.0491 - n_outputs0_loss: 0.0468 - n_outputs1_loss: 0.0023  
  
38/38 [=====] - 33s 858ms/step - loss: 0.0553 - n_outputs0_loss: 0.0520 - n_outputs1_loss: 0.0033 -  
val_loss: 0.0491 - val_n_outputs0_loss: 0.0468 - val_n_outputs1_loss: 0.0023  
Epoch 20/100  
9/9 [=====] - 5s 509ms/step - loss: 0.0548 - n_outputs0_loss: 0.0526 - n_outputs1_loss: 0.0022  
  
38/38 [=====] - 33s 873ms/step - loss: 0.0532 - n_outputs0_loss: 0.0499 - n_outputs1_loss: 0.0033 -  
val_loss: 0.0548 - val_n_outputs0_loss: 0.0526 - val_n_outputs1_loss: 0.0022  
Epoch 21/100  
9/9 [=====] - 2s 262ms/step - loss: 0.0569 - n_outputs0_loss: 0.0545 - n_outputs1_loss: 0.0024  
  
38/38 [=====] - 29s 760ms/step - loss: 0.0523 - n_outputs0_loss: 0.0492 - n_outputs1_loss: 0.0030 -  
val_loss: 0.0569 - val_n_outputs0_loss: 0.0545 - val_n_outputs1_loss: 0.0024  
Epoch 0021: early stopping  
Training completed in 0:11:54.  
  
----- Best Eval Loss :0.048293 -----  
not saving loss graph because matplotlib not set up.  
(env) donkey@donkey:~$
```

Abbildung 8.6: Autopilot erfolgreich trainiert

8.7 Führen Sie ihren Autopiloten das erste Mal aus

Nach dem Sie jetzt erfolgreich Ihren ersten Autopiloten trainiert haben führen Sie diesen jetzt aus. Wichtig ist, dass Sie in der myconfig.py Datei die Verwendung eines Gamepads aktiviert haben.

Mit dem folgenden Befehl starten Sie jetzt den drive Modus mit dem von Ihnen trainierten Autopiloten der hier den Namen mypilot.h5 trägt.

Befehl: python manage.py drive --model ~/mycar/models/mypilot.h5

Jetzt wird erst einmal nichts passieren und das Auto bleibt still stehen. Der drive Modus kennt drei Modi zwischen denen Sie mit der „Share“ Taste an Ihrem PS4 Controller oder mit der „Back“ Taste des „EasySMX Controllers“ wechseln können.

Der drive Modus kennt die folgenden drei Modi

User: Sie können manuell das Roboter-Auto steuern.

Local Angle: Der Autopilot lenkt und Sie kontrollieren die Geschwindigkeit.

Local Pilot: Der Autopilot lenkt und steuert die Geschwindigkeit autonom.

Wechseln Sie jetzt auf den Modus „Local“ und lassen Sie Ihr Roboter-Auto autonom fahren. Im Terminal Fenster bekommen Sie immer den aktuell gewählten Modus angezeigt. Sollte es während dem autonomen Fahren zu einem Fahrfehler des Autopiloten kommen, so können Sie durch Drücken der Share-Taste oder Back-Taste auf den User Modi wechseln und so die Kontrolle über das Roboter-Auto wieder übernehmen.

Alternative Web-Steuerung: Verwenden Sie die Web-Steuerung dann müssen Sie in der Dropdown Box „Mode & Pilot“ den Modus auf „Local Pilot“ wechseln damit Ihr Roboter-Auto autonom seine Runden dreht.

Sind Sie nicht mit den Fahreigenschaften des Autopiloten zufrieden dann zeichnen Sie wieder Trainingsdaten auf und trainieren Sie Ihr Model erneut. Eventuell müssen Sie Teile der bereits vorhandenen Trainingsdaten löschen.

Hinweis: Viele Daten helfen nicht immer viel. Daher starten Sie mit 8.000 bis maximal 10.000 Datensätze um erst einmal ein Gefühl dafür zu bekommen wie ein Neuronales Netz lernt.

8.8 Hilfe mein Roboter-Auto fährt nicht automatisch los

Bei verschiedenen Roboter-Autos die ich bereits aufgebaut hatte stellte ich das Phänomen fest, dass im „Local“ Modus das Roboter-Auto nicht losgefahren ist. Bei diesen Fahrzeugen kam das Donkey Car Framework in der Version 2.5.x zum Einsatz. Habe ich das Roboter-Auto kurz angeschoben konnte es von sich aus weiter fahren. Tritt bei Ihnen ebenfalls dieses Problem auf, dann können Sie mit z. B. der Taste „L1“ eine konstante Geschwindigkeit aktivieren und erhöhen und mit der Taste „R1“ diese wieder reduzieren und Ihr Roboter-Auto sollte damit anfahren können.

Ganz genau konnte ich das Problem noch nicht eingrenzen aber mit der neuesten Donkey Car Version 3.x habe ich dieses Problem nicht mehr beobachten können.

8.9 Was Sie bis hier erreicht haben

Sie haben jetzt sehr viel über das Donkey Car Framework gelernt und wie Sie z. B. eine Strecke für das Roboter-Auto aufbauen. Welche Fehler es zu vermeiden gilt beim Erzeugen von guten Trainingsdaten und wie das Training des Autopiloten funktioniert. Ihr Roboter-Auto hat die ersten autonomen Fahrversuche unternommen. Weiter haben Sie eventuell mit den verschiedenen Leistungseinstellungen des Jetson Nano den einen oder anderen Test unternommen um einen stabilen Betrieb des Roboter-Autos zu gewährleisten.

Ich freue mich sehr über Ihr Feedback zu diesem Kapitel. Schreiben Sie mir Ihre Gedanken, Fragen und Vorschläge an die folgende E-Mail Adresse: ebook@custom-build-robots.com

9 Tipps, Tricks und weiterführende Informationen zum Roboter-Auto

Ihr Roboter-Auto fährt bereits autonom und Sie möchten noch ein paar kleine Tipps hören. Dann sind Sie hier genau richtig und ich verrate Ihnen wie ich mein Roboter-Auto optimiert habe und wie ich mir Trainings Daten erzeuge ohne diese wirklich Runde für Runde aufzuzeichnen.

9.1 Zusätzlichen Arbeitsspeicher freigeben

Der Jetson Nano mit seinen 4GB Ram arbeitet sehr schnell bezüglich der Speicherauslastung am Limit. Da Sie wohl wie ich die meiste Zeit das Roboter-Auto über eine offene SSH Session mit Befehlen administrieren benötigen Sie die grafische Oberfläche eigentlich nicht. Daher können Sie diese deaktivieren und nur im Text-Modus das Betriebssystem hochfahren und haben so ein paar hundert MB Ram mehr frei und erreichen so mit wenig Aufwand eine etwas bessere Performance ihres Roboter-Autos. Wie genau das geht beschreibe ich nachfolgenden.

Damit der grafische Modus deaktiviert wird muss der nachfolgende Befehl im Terminal Fenster ausgeführt werden.

Befehl: sudo systemctl set-default multi-user.target

Nach einem Neustart des Jetson Nano sollte Sie, wenn dieser am HDMI Ausgang an einem Monitor angeschlossen ist den Jetson Nano im Text-Modus hochfahren sehen.

Hinweis: Ich hatte meinen Jetson Nano über den Display-Port Anschluss an einem Monitor angeschlossen und nach dem ausführen dieses Befehls kein Bild mehr gesehen. Auch klappte die SSH Anmeldung nicht. Daraufhin habe ich den Jetson Nano neu installiert. Beim nächsten Versuche stellte ich dann fest, dass das Bild über dem HDMI Ausgang verfügbar gewesen wäre.

Mit dem folgendne Befehl können Sie jetzt prüfen wie viel Arbeitsspeicher frei ist.

Befehl: free mem

Jetzt sollten jetzt ca. 3,42 GB RAM im Arbeitsspeicher frei sein. Im Vergleich mit aktiver grafischer Oberfläche waren nur 2,73 GB RAM im Arbeitsspeicher frei.

Das folgende Bild zeigt den freien Arbeitsspeicher ohne aktiver grafischer Oberfläche direkt nach dem Systemstart.

	total	used	free	shared	buff/cache	available
Mem:	4059260	237704	3469404	20988	352152	3648544
Swap:	14612524	0	14612524			

Abbildung 9.1: Anzeige des freien Arbeitsspeichers

Möchten Sie den grafischen –Modus wieder aktivieren dann führen Sie bitte folgenden Befehl aus.

Befehl: sudo systemctl set-default graphical.target

Haben Sie ihren Jetson Nano im Text-Modus hochgefahren möchten aber von Hand die grafische Oberfläche starten dann können Sie diese mit folgenden Befehl wieder starten.

Befehl: sudo systemctl start gdm3.service

Jetzt haben Sie mit dieser kleinen Anpassung etwas mehr Arbeitsspeicher zur Verfügung und so vielleicht eine etwas bessere Performance erreicht.

9.2 Begrenzung der Leistungsaufnahme des Jetson Nano

Sie können die Leistungsaufnahme des Jetson Nano auf 5W oder 10W begrenzen wenn Sie den Jetson Nano über die micro USB-Schnittstelle mit Strom versorgen. Schließen Sie die Stromversorgung über den Poweranschluss an so läuft der Jetson Nano mit seiner vollen Leistung und benötigt dann ca. 20W also 5V bei 4A. Je nachdem wie tief Sie in das Thema selbst fahrende Modellautos einsteigen und wie viel Leistung Sie im mobilen Betrieb benötigen versorgen Sie den Jetson Nano über einen der beiden Stromanschlüsse mit Strom. Haben Sie die von mir empfohlene Power Bank in Verwendung so können Sie sich ein USB-Kabel abschneiden und einen Stecker für die DC-Buchse des Jetson Nano anlöten und so die maximal Leistung für z. B. schnelle autonome Fahrten des Roboter-Autos vom Jetson Nano abrufen.

Beim Training des Autopiloten auf dem Jetson Nano sollten Sie immer auf die volle Leistung zurückgreifen und den Jetson Nano über das extra Netzteil und die DC-Buchse mit Strom versorgen. Denken Sie daran den Jumper J48 zu setzen.

Im mobilen Betrieb gehe ich jetzt davon aus, dass Sie den Jetson Nano über die micro USB Schnittstelle an der Power-Bank angeschlossen haben. Um den Energieverbrauch auf 5W zu begrenzen führen Sie den folgenden Befehl in der Konsole aus. Mit diesem aktivieren Sie das POWER_MODEL mit der ID=1.

Befehl: sudo nvpmodel -m1

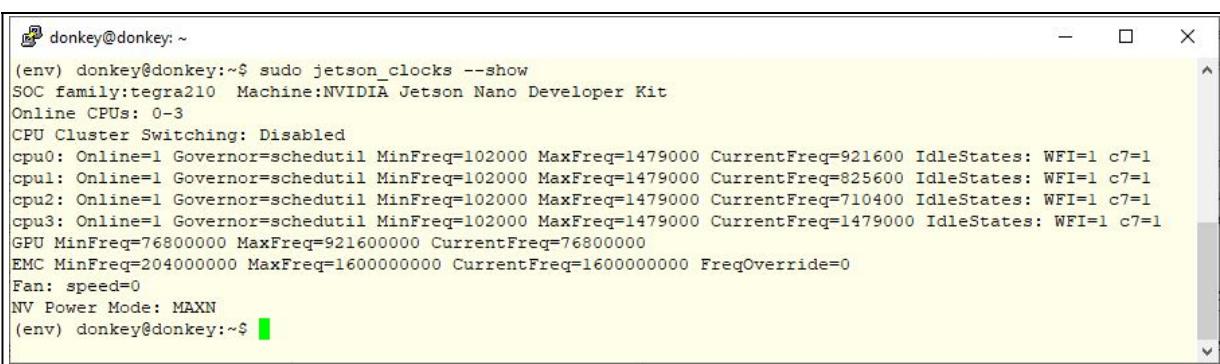
Möchten Sie die maximale Leistung zur Verfügung haben die Sie dem micro USB-Anschluss abverlangen können, dann können Sie den Energieverbrauch wieder auf die 10W hochsetzen. Führen Sie dazu den folgenden Befehl aus der das POWER_MODEL mit der ID=0 aktiviert.

Befehl: sudo nvpmodel -m0

In der Datei /etc/nvpmodel/nvpmodel_t210_jetson-nano.conf finden Sie die detaillierten Energie-Einstellungen des Jetson Nano und können den Unterschied zwischen den beiden Modi m1 und m0 also POWER_MODEL 1 bzw. 0 nachlesen. Der wesentliche Unterschied zwischen den Modi macht die Anzahl der verwendeten CPUs und die maximal erlaubte Frequenz der CPUs aus. Genau hier können Sie auch Ihren eigenen Modus definieren wenn Sie sich auskennen und dies einmal ausprobieren möchten.

Möchten Sie sehen mit welchen Leistungs-Einstellungen der Jetson Nano aktuell arbeitet können sie mit dem folgenden Befehl sich diese anzeigen lassen.

Befehl: sudo jetson_clocks --show



```
donkey@donkey: ~
(env) donkey@donkey:~$ sudo jetson_clocks --show
SOC family:tegra210 Machine:NVIDIA Jetson Nano Developer Kit
Online CPUs: 0-3
CPU Cluster Switching: Disabled
cpu0: Online=1 Governor=schedutil MinFreq=102000 MaxFreq=1479000 CurrentFreq=921600 IdleStates: WFI=1 c7=1
cpu1: Online=1 Governor=schedutil MinFreq=102000 MaxFreq=1479000 CurrentFreq=825600 IdleStates: WFI=1 c7=1
cpu2: Online=1 Governor=schedutil MinFreq=102000 MaxFreq=1479000 CurrentFreq=710400 IdleStates: WFI=1 c7=1
cpu3: Online=1 Governor=schedutil MinFreq=102000 MaxFreq=1479000 CurrentFreq=1479000 IdleStates: WFI=1 c7=1
GPU MinFreq=76800000 MaxFreq=921600000 CurrentFreq=76800000
EMC MinFreq=204000000 MaxFreq=1600000000 CurrentFreq=1600000000 FreqOverride=0
Fan: speed=0
NV Power Mode: MAXN
(env) donkey@donkey:~$
```

Abbildung 9.2: Jetson Nano clock mode

9.3 Screen der Terminalmultiplexer

Eventuell kennen Sie Screen schon und nutzen diese bereits. Wenn nicht, dann möchte ich Ihnen kurz die Vorteile von Screen bei der Verwendung im Roboter-Auto erklären. Screen ist ein Terminalmultiplexer der es Ihnen ermöglicht eine Terminalsession zu starten die weiter läuft auch wenn Sie die SSH Verbindung über die Sie am Roboter-Auto angemeldet sind unterbrochen wurde. Sie

können immer wieder zu dieser im Programm Screen gestarteten Session zurückkehren. Das hat den entscheidenden Vorteil, dass bei einem Netzwerkproblem und Unterbrechung der SSH Verbindung das Roboter-Auto nicht stehen bleibt da die Session nicht so leicht abstürzt sondern weiter läuft und mit Ihr alle gestarteten Programme. Screen installieren Sie mit dem folgenden Befehl auf Ihrem Jetson Nano.

Befehl: sudo apt-get install -y screen

Wenn Sie via SSH auf Ihrem Roboter-Auto angemeldet sind dann starten Sie mit dem folgenden Befehl eine Screen Session der Sie dem Namen „sitzung1“ mitgeben.

Befehl: screen -S sitzung1

Nach dem Sie diesen Befehl ausgeführt haben befinden Sie sich in der Screen-Session mit dem Namen „sitzung1“.

Diese Session können Sie mit dem folgenden Befehl verlassen ohne die Sitzung selber und die in ihr ausgeführten Programme zu beenden.

Befehl: Strg + A + D

Möchten Sie sich alle aktiven Screen-Session anzeigen lassen dann führen Sie den folgenden Befehl aus.

Befehl: screen -ls

Sie erhalten eine Liste mit den aktiven Screen-Sessions. Möchten Sie jetzt in eine bestimmte Session zurückkehren wie z. B. in die Session „sitzung1“ dann können Sie das mit dem folgenden Befehl machen.

Befehl: screen -r sitzung1

So hilft Ihnen screen bei eventuellen Verbindungsproblemen dabei nicht ständig das Donkey-Car Framework neu starten zu müssen.

9.4 Trainingsbilder manipulieren

Sie haben bei hellem Licht die Trainingsdaten aufgenommen und jetzt müssen Sie nach dem der Autopilot fertig trainiert ist feststellen, dass sich die Lichtverhältnisse verändert haben. Das ist ein typisches Problem das mir immer wieder passiert ist mit dem Ergebnis, dass der Autopilot das Roboter-Auto nicht mehr fehlerfrei um den den Racetrack steuern konnte. Damit Sie nicht verschiedene Lichtverhältnisse abwarten bzw. erzeugen müssen beim Trainingsdaten erstellen habe ich ein Skript geschrieben das die bereits aufgezeichneten Bilder in ihrer Helligkeit und Farbsättigung verändert. Trainiere ich mit den original Bildern und zusätzlich mit den veränderten Bildern den Autopiloten dann fährt mein Roboter-Auto sehr stabil auch bei sich wechselnden Lichtverhältnissen den Racetrack ab.

In dem kleinen Skript convert_images.sh müssen Sie den Ordner angeben in dem Ihre tub-Ordner mit den Trainingsbildern liegen. Das Skript kopiert pro gefundenen tub-Ordner die *.json Dateien in einen neuen Tub-Ordner und speichert passend zu den *.json Dateien die manipulierten Bilder in diesen mit ab.

Wenn Sie sich das Skript anschauen dann werden Sie sehen, dass es ganz einfach aufgebaut ist und es lediglich das Programm „ImageMagick convert“ nutzt und unterschiedlich parametrisiert aufruft. Durch manuelles anpassen wie löschen, ändern oder duplizieren des Codes können Sie weitere Trainingsbilder mit von Ihnen gewünschten Eigenschaften erstellen.

Das Skript convert_images.sh steht Ihnen unter der folgenden URL zum Download zur Verfügung.

URL: <https://custom-build-robots.com/jetson-nano-download-de>

Eine gute Beschreibung der möglichen Optionen mit denen Sie convert aufrufen können ist unter der nachfolgenden URL zu finden.

URL: <https://imagemagick.org/script/convert.php>

9.5 Hilfe mein Lüfter läuft nicht an

Für den Fall das Sie einen Lüfter auf Ihrem Jetson Nano verbaut haben und dieser startet nicht bzw. dreht zu schnell und damit zu laut etc. dann können Sie die Drehgeschwindigkeit des Lüfters mit dem folgenden Befehl beeinflussen.

Führen Sie den nachfolgenden Befehl im Terminal Fenster aus dreht der Lüfter bei maximaler Geschwindigkeit. Die Zahl 255 in diesem Befehl legt die Geschwindigkeit fest und kann zwischen 0 und 255 liegen.

Befehl: sudo sh -c 'echo 255 > /sys/devices/pwm-fan/target_pwm'

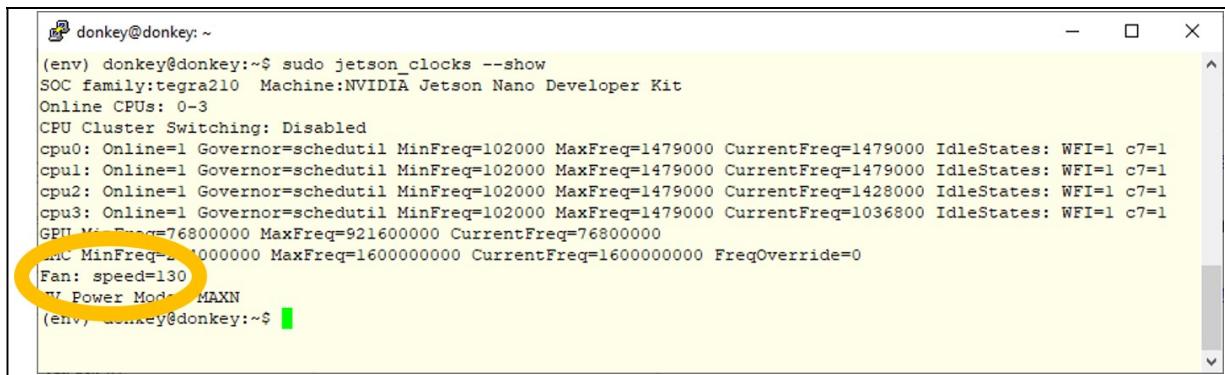
Mit dem nachfolgenden Befehl halten Sie den Lüfter an da die Drehgeschwindigkeit auf 0 gesetzt wurde.

Befehl: sudo sh -c 'echo 0 > /sys/devices/pwm-fan/target_pwm'

Führen Sie den Befehl nachfolgenden Befehl aus dann ist der Lüfter leicht am drehen und kaum zu hören. Mit dieser Einstellung habe ich auch beim Training der neuronalen Netze auf dem Jetson Nano gute Erfahrungen gemacht.

Befehl: sudo sh -c 'echo 130 > /sys/devices/pwm-fan/target_pwm'

Im nachfolgenden Screenshot ist die Geschwindigkeit von 130 die zuvor gesetzt wurde orange hervorgehoben.



```
donkey@donkey: ~
(env) donkey@donkey:~$ sudo jetson_clocks --show
SOC family:tegra210 Machine:NVIDIA Jetson Nano Developer Kit
Online CPUs: 0-3
CPU Cluster Switching: Disabled
cpu0: Online=1 Governor=schedutil MinFreq=102000 MaxFreq=1479000 CurrentFreq=1479000 IdleStates: WFI=1 c7=1
cpu1: Online=1 Governor=schedutil MinFreq=102000 MaxFreq=1479000 CurrentFreq=1479000 IdleStates: WFI=1 c7=1
cpu2: Online=1 Governor=schedutil MinFreq=102000 MaxFreq=1479000 CurrentFreq=1428000 IdleStates: WFI=1 c7=1
cpu3: Online=1 Governor=schedutil MinFreq=102000 MaxFreq=1479000 CurrentFreq=1036800 IdleStates: WFI=1 c7=1
GPU: MinFreq=76800000 MaxFreq=921600000 CurrentFreq=76800000
MC MinFreq=1000000 MaxFreq=16000000000 CurrentFreq=16000000000 FreqOverride=0
Fan: speed=130
V. Power Mode: MAXN
(env) donkey@donkey:~$
```

Abbildung 9.3: Lüfter Geschwindigkeit

9.6 Roboter-Auto mit WIFI-Accesspoint

Vielelleicht waren Sie bereits mit dem Roboter-Auto auch im Freien unterwegs und hatten ebenfalls das Problem z. B. das kein WIFI zur Verfügung stand. In diesen Fällen können Sie am eigenen Smartphone einen Accesspoint starten oder einen eigenen Router mitnehmen. Sie können aber auch ihren eigenen Accesspoint auf Ihrem Roboter-Auto starten und so die Verbindung zwischen Laptop und Roboter-Auto herstellen.

Wie Sie einen Accesspoint mit dem im Betriebssystem bereits vorhandenen NetworkManager auf Ihrem Roboter-Auto starten erkläre ich Ihnen nachfolgend. Es ist eigentlich ganz einfach und wenn Sie den Accesspoint im Roboter-Auto beenden dann wählt sich der Jetson Nano auch direkt wieder in ein Verfügbares und ihm bekanntes Netzwerk ein.

Mit den folgenden Befehlen richten Sie einen Accesspoint mit dem Namen bzw. der SSID „Roboter-Auto“ ein. Das Passwort für diesen Accesspoint ist „robot_123“ und dieses sollten Sie anpassen für Ihr Roboter-Auto. Vielmehr müssen Sie jetzt gar nicht beachten wenn Sie die Befehle einen nach dem anderen eingeben.

Befehl: sudo nmcli con add type wifi ifname wlan0 mode ap con-name Roboter-Auto ssid Roboter-Auto

Befehl: sudo nmcli con modify Roboter-Auto 802-11-wireless.band bg

Befehl: sudo nmcli con modify Roboter-Auto 802-11-wireless.channel 1

Befehl: sudo nmcli con modify Roboter-Auto 802-11-wireless-security.key-mgmt wpa-psk

Befehl: sudo nmcli con modify Roboter-Auto 802-11-wireless-security.psk robot_123

Befehl: sudo nmcli con modify Roboter-Auto ipv4.method shared

Mit dem letzten Befehl dieser kleinen Liste starten Sie den Accesspoint auf Ihrem Roboter-Auto. Anschließend müssen Sie sich mit diesem verbinden. Im OLED Display sollten Sie die IP Adresse 10.42.0.1 sehen. Unter dieser erreichen Sie Ihr Roboter-Auto von z. B. dem Smartphone oder Laptop aus.

Befehl: sudo nmcli con up Roboter-Auto

Möchten Sie sehen welche Accesspoints alle auf Ihrem Jetson Nano schon einmal eingerichtet wurden dann können Sie sich eine kleine Übersicht mit dem folgenden Befehl anzeigen lassen.

Befehl: nmcli con show

Möchten Sie den Accesspoint von Hand starten dann verwenden Sie den nachfolgenden Befehl. Wurde einmal des Accesspoint gestartet und Sie starten den Jetson Nano neu dann wird auch der Accesspoint wieder automatisch hochgefahren.

Befehl: sudo nmcli con up Roboter-Auto

Haben Sie den Accesspoint einmal manuell ausgeschalten und starten den Jetson Nano neu dann wird der Accesspoint auch nicht mehr gestartet. Der Jetson Nano wählt sich dann in das nächste ihm bekannte und erreichbare W-Lan wieder ein. Mit dem folgenden Befehl deaktivieren Sie den Accesspoint auf Ihrem Roboter-Auto

Befehl: sudo nmcli con down Roboter-Auto

Möchten Sie den von Ihnen konfigurierten Accesspoint wieder löschen dann können Sie dies mit dem folgenden Befehl machen.

Befehl: sudo nmcli con delete Roboter-Auto

9.7 Was Sie bis hier erreicht haben

Jetzt haben Sie noch ein paar kleinere Tipps erhalten die Ihnen hoffentlich helfen ein paar Probleme zu umschiffen die mir immer wieder Schwierigkeiten gemacht haben. Mit dem Skript zur Erzeugung von zusätzlichen Trainingsdaten aus bestehenden sollten Sie in der Lage sein einen wesentlich stabileren Autopiloten zu trainieren ohne zusätzliche Trainingsdaten aufzzeichnen zu müssen. Mit der Konfiguration des Accesspoints auf dem Roboter-Auto sind Sie im Freien auf z. B. großen Parkplätzen frei von einem Router bzw. WIFI das Sie mitbringen müssten.

Ich freue mich sehr über Ihr Feedback zu diesem Kapitel. Schreiben Sie mir Ihre Gedanken, Fragen und Vorschläge an die folgende E-Mail Adresse: ebook@custom-build-robots.com