Ersteller: Christof Kary

Datum: 06.09.2018

Version: 1.1

Dieses Dokument soll als Leitfaden zum Einstieg in das Projekt EVObot dienen. Es soll die Einarbeitung unabhängig von der gesamten Projektdokumentation erleichtern und damit einen Wissenstransfer zwischen den Projektmitglieder sicherstellen. Sollten die in diesem Dokument genannten Begrifflichkeiten und Funktionen nicht bereits bekannt sein, wird dazu geraten, zu den jeweiligen Punkten in den genannten Links oder darüber hinaus nachzuschlagen.

1 NVIDIA Jetson TX2

Die zentrale Entwicklungsplattform des Projektes stellt das Entwicklerkit *NVIDIA Jetson TX2* dar. Der Computer wurde speziell für KI-Anwendungen, Computer Vision und rechenintensive Grafikanwendungen zur Bildverarbeitung entwickelt. Das Board verfügt über 6 CPU-Kerne und 8 GB internem Arbeitsspeicher und ist damit für hohe Rechenleistungen ausgelegt. Das zentrale Rechenmodul ist auf einer Adapterplatine mit zahlreichen Anschlüssen für umfangreiche Erweiterungen verbaut. Zur hardware- und schnittstellenspezifischen Einarbeitung sei auf den *User Guide* verwiesen. Ausführliche Beschreibungen und Tutorials sind auf der Homepage `elinux.org` zusammengefasst. Umfangreiche benutzerfreundliche Beispiele zum Arbeiten mit dem Jetson-Board finden sich auf `jetsonhacks.com`.

Um das Energie- und Leistungsmanagement zu verwalten, liefert das Jetson-Board das Kommandozeilen-Tool *nvpmodel*. Durch den Befehl

```
sudo nvpmodel -m [mode]
```

lässt sich zwischen 5 verschiedenen Leistungsmodi wählen. Es gilt zu erwähnen, dass der EVObot in der aktuellen Umsetzung dauerhaft im maximalen Leistungsmodus betrieben wird, um alle 6 CPU-Kerne bei maximaler Taktfrequenz von 2 GHz zu nutzen. Alternativ können durch Ausführen des Shell Skriptes

```
sudo ./jetson_clocks.sh
```

sämtliche Energiesparoptionen deaktiviert werden.

2 Linux Ubuntu

Auf der verwendeten Hardware läuft Ubuntu als eine Linux-Distribution in der Version *16.04.4 LTS*. Wurde zuvor nocht nicht mit Linux-Systemen gearbeitet, ist eine Einarbeitung in die Ubuntu-Umgebung dringend zu empfehlen. Es sollten sich zunächst wichtige Befehle angeeignet und der allgemeine Umgang mit der Benutzeroberfläche und dem Arbeiten über kommandozeilenbasierte Terminaleingaben angeeignet werden. Für Einsteiger in das Ubuntu-System sei auf *diese Grundlagenerklärung* verwiesen. Generell eignet sich das *Wiki* der Homepage `ubuntuusers.de` als Nachschlagewerk rund um Linux. Generell ist es von Vorteil, stets eine *Befehlsübersicht* zur Hand zu haben. Ein Auszug wichtiger, projektbezogener Befehle die häufig Verwendung finden, sind:

`cd`:	Wechsel des Arbeitsverzeichnisses
`ls`:	Auflistung von Dateien
`sudo`:	Führt nachfolgenden Befehl mit Root-Rechten aus
`mkdir`:	Erzeugung von Verzeichnissen
`cmake`:	Erstellt und kompiliert Projekte oder Makefiles
`apt-get`:	Zum Installieren, Updaten, Löschen, ... von Paketen

Der primär verwendete Benutzer auf dem *Jetson TX2* ist `nvidia`. Das Passwort für diesen Benutzer lauter ebenfalls `nvidia`. Zwar wurde die Passwortabfrage beim Systemboot deaktiviert, jedoch muss das Passwort zum Ausführen eines *sudo*-Befehls und zum Aktivieren eines *SSH*-Zugriffes verwendet werden.
Ein SSH-Zugriff ist nötig, um einen kabellosen, netzwerkbasierten Terminalzugriff auf dem EVObot zu ermöglichen. Auf Ubuntu ist bereits ein SSH-Zugang vorinstalliert. Möchte man von einem Windows-PC auf das Jetson-Board zugreifen, ist hierzu die Software *PuTTY* oder *KiTTY* nötig. Im Netzwerk *evo-flacht* ist der EVObot mit der IP 10.30.60.171 zu finden. Die IP-Adresse kann durch den Befehl

```
ifconfig
```

abgerufen werden.
Wird auf den EVObot lediglich über eine SSH-Verbindung zugegriffen, wird empfohlen, die grafische Benutzeroberfläche über den Befehl

```
sudo service lightdm stop
```

zu deaktivieren. Hiernach wird die Berechnung der GUI verhindert, um sämtliche Leistungsressourcen für die automatisierte Fahrroutine zur Verfügung zu stellen. Durch Verwendung der Option `start` im oben genannten Befehl oder durch einen Reboot des Systems lässt sich die grafische Benutzeroberfläche wieder aktivieren.

Alternativ zu einem SSH-Zugriff kann eine *VNC*-Verbindung zum Jetson-Board aufgebaut werden. Diese wird dazu verwendet, um eine Remote-Desktop Anwendung zu ermöglichen. Auf dem Jeston-Board selbst ist hierzu bereits eine VNC-Applikation aktiviert. Auf dem fernsteuernden Rechner muss zunächst die Software *VNC Viewer* installiert und eine neue Verbindung mit der Korrekte IP hergestellt werden. Sobald die VNC-Verbindung aufgebaut wurde, kann mit dem Terminal-Befehl

```
sudo xrandr --fb "Breite"x"Höhe"
```

die gewünschte Desktopgröße eingestellt werden.
Es wird dringend davon abgeraten, die Fahrroutine des EVObots über eine aktive VNC-Verbindung zu starten, da die Übertragung des Desktops einen erheblichen Anteil der verfügbaren CPU-Leistung beansprucht. Dabei kann es zu einer hohen Latenz in der Algorithmenberechnung kommen, die sich negativ auf das Regelverhalten auswirkt.

3 ROS - Robot Operating System

Auf dem Ubuntu-Betriebssystem arbeitet ROS in der Distributionsversion *Kinetic*. Dies ist eine Open Source Middleware speziell für Robotikanwendungen. Das Framework stellt umfangreiche Programmbibliotheken, Werkzeuge, Gerätetreiber, Visualisierungstools und Möglichkeiten zum Nachrichtenaustausch. Die gesamte Entwicklungsumgebung auf dem EVObot wurde mithilfe von ROS erstellt und liefert die gesamte Architektur der einzelnen Softwarekomponenten zur automatisierten Fahrfunktion. Daher ist eine ausführliche Einarbeitung in ROS essentiell. Einsteigern in das Arbeiten mit ROS wird es empfohlen, sich das *Wiki der offiziellen Homepage* anzueignen und die *Tutorials* inklusive der angegebenen Beispiele

aufmerksam durchzuarbeiten. Um eine noch umfangreichere Einarbeitung zu ermöglichen, kann das Buch *Programming robots with ROS* von Morgan Quigley, Brian Gerkey und William D. Smart herangezogen werden.

Es sollte nachvollzogen werden, dass ROS als Gesamtsystem anders agiert als unabhängig voneinander ausführbare Programme und Funktionen. Es findet innerhalb der ROS-Umgebung eine effiziente Kommunikation der Teilfunktionen durch eine flexible und simple Datenstruktur statt. Betrachtet man die Prozessarchitektur, befindet sich ROS auf der selben Ebene wie die ausführbaren Applikationen in einer Rechnerstruktur und stellt das Interface zwischen Hardware und der IPC (Inter-Process Communication) dar. Besonders die fundamentalen Konzepte von ROS und die Origanisation in *Nodes, Messages, Topics* und *Services* sollte verstanden werden. Ein weiterer wichtiger Aspekt ist das Prinzip des Informationsaustausches zwischen einzelnen Knoten (Nodes) über einen *publisher-* und *subscriber-*Mechanismus und dem damit verbundenen Aufruf von *callback-Funktionen.*
Eine Schnellübersicht der ROS-spezifischen Tools und Befehle bietet dieses *Cheat Sheet*. Darüber hinaus liefert das Framework *rqt* eine komfortable Benutzeroberfläche um Signalwerte während der Programmlaufzeit grafisch darzustellen oder um das Prozessverhalten zu analysieren. Das Paket *rostopic* liefert Debugging-Informationen einzelner Topics.

Der Workspace und sämtliche projektbezogene Dateien befinden sich im Ubuntu-System in dem Verzeichnis `/home/nvidia/EVObot/`. Das Package, das die entsprechenden launch-Files, message-Beschreibungen und ROS-Knoten als ausführbare Skripte enthält, liegt in dem Ordner `autonomous`. Die gesamte Fahrroutine wird über den Kommandozeilenbefehl `roslaunch` mit Angabe des Pakets und dem gewünschten launch-File gestartet. Wird der EVObot über SSH-Befehl ohne eine grafische Benutzeroberfläche bedient, sollte der Befehl

```
roslaunch autonomous lanefollowergray.launch
```

ausgeführt werden. Ist jedoch der Desktop aktiv, kann der Befehl

```
roslaunch autonomous lanefollowergray_visual.launch
```

aufgerufen werden. Hierbei wird dem Benutzer zusätzlich das eingelesene Kamerabild, die detektierten Fahrspuren und die Verkehrszeichenerkennung als Image Stream ausgegeben.

4 OpenCV

OpenCV ist eine freie Programmbibliothek mit diversen Algorithmen spezielle für die Bildverarbeitung und maschinelles Sehen. Die Bibliothek kann ich gängige Programmiersprachen wie C, C++, Python oder Java eingebunden werden. Durch die zahlreichen Funktionen lassen sich unter Anderem auf eine übersichtliche Art Algorithmen zur Gesichts- und Objekterkennung, zur umfangreichen Filterung und Klassifizierung oder zum Maschinellen Lernen umsetzen. Gängige Algorithmen wie beispielsweise eine *Canny Edge Detection* können zur Einarbeitung ebenfalls auf `jetsonhacks.com` nachvollzogen und entsprechende Beispiele umgesetzt werden. ALternativ bietet OpenCV eine sehr umfangreiche *Dokumentation* rund um die Funktionalitäten der Bibliothek.

5 Python

Sämtliche, auf dem EVObot umgesetzten Funktionen wurden in der universellen Programmiersprache Python mit der Version 2.7 verfasst. Als Entwicklungsumgebung kommt *PyCharm* in der quelloffenen Community Version zum Einsatz. Damit ist eine übersichtliche Versionskontrolle und eine automatische Codevervollständigung im Python-Stil möglich.